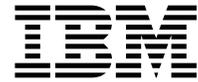
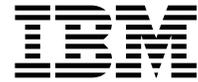


SMARTdata UTILITIES



VSAM Application Programming Interface Reference

SMARTdata UTILITIES



VSAM Application Programming Interface Reference

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

First Edition (April 1997)

This edition applies to all platforms supported by SMARTdata UTILITIES Version 2 Release 1, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Publications are *not* stocked at the address below. Requests for IBM publications should be made to your IBM representative or the IBM branch office serving your locality.

You can order by calling IBM Software Manufacturing Solutions at 1-800-879-2755.

A form for reader comments is provided at the back of this publication. If the form has been removed, address your comments to:

International Business Machines Corporation
RCF Processing Department
G26/050
5600 Cottle Road
SAN JOSE, CA 95193-0001
U.S.A.

IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1993, 1997. All rights reserved.**
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Programming Interface Information	xi
Trademarks and service marks	xii
About This Book	xiii
Who Should Read This Publication	xiii
What You Should Know Before Reading This Publication	xiii
Bibliography	xv
Using This Reference	xvii
Notation Conventions	xvii
Function Descriptions	xvii
DDMExample (Example)	xviii
Syntax	xviii
Parameters	xviii
Returns	xviii
Remarks	xviii
Examples	xix
Chapter 1. Introduction to VSAM as a DDM Implementation	1
Distributed Data Management Overview	1
DDM Record Types	4
RECORD formats	4
RECINA formats	4
Record Attribute Lists (RECALs)	5
Extended Attributes	5
Record Files	6
Record File and Record Length Classes	7
Sequential Files	8
Direct Files	10
Keyed Files	13
Alternate Index File	16
File Naming Conventions	17
Performance Considerations	17
Sequential and Direct Files	18
Keyed and Alternate Index Files	18
Access Methods	18
Promoting Access Methods	20
DDM Cursor	21
DDM Lock Management	22
Concurrency Protection	22
File Locking	23
Record Locking (Implementation is Dependent on the Server)	25
Promoting Locks (Implementation is Dependent on the Server)	27

DDM Architecture Promotions and Exceptions	28
Technical Considerations	29
Chapter 2. Function Lists	31
VSAM Function Descriptions	31
Parameters Used in Function Descriptions	34
Access Functions Applicable to Each File Class	34
Cursor-Positioning Functions Applicable to Each File Class	35
Record File Attributes by File Class	37
Modifiable Record File Attributes by File Class	38
Private File Attributes by File Class	39
Access Functions Applicable to Each Access Method	40
Access Functions Applicable to Each Access Method Continued	42
Chapter 3. VSAM API Functions	43
DDMClose (Close File)	44
DDMCopyFile (Copy File)	46
DDMCreateAltIndex (Create Alternate Index File)	50
DDMCreateRecFile (Create Record File)	57
DDMDelete (Delete File)	64
DDMDeleteRec (Delete Record)	66
DDMForceBuffer (Commit a File's Cached Information)	70
DDMGetRec (Get Record)	72
DDMGetReplyMessage (Get Reply Message)	81
DDMInsertRecEOF (Insert Records at EOF)	83
DDMInsertRecKey (Insert Records by Key Value)	93
DDMInsertRecNum (Insert by Record Number)	98
DDMLoadFileFirst (Load Records into File)	106
DDMLoadFileNext (Load Records into File)	115
DDMModifyRec (Modify Record)	122
DDMOpen (Open File)	127
DDMQueryFileInfo (Get a File's Information)	133
DDMQueryPathInfo (Get File or Subdirectory Information)	135
DDMRename (Rename File)	138
DDMSetBOF (Set Cursor to Beginning of File)	141
DDMSetEOF (Set Cursor to End of File)	144
DDMSetFileInfo (Set File Information)	146
DDMSetFirst (Set Cursor to First Record)	148
DDMSetKey (Set Cursor by Key)	159
DDMSetKeyFirst (Set Cursor to First Record in Key Sequence)	177
DDMSetKeyLast (Set Cursor to Last Record in Key Sequence)	186
DDMSetKeyLimits (Set Key Limits)	195
DDMSetKeyNext (Set Cursor to Next Record in Key Sequence)	202
DDMSetKeyPrevious (Set Cursor to Previous Record in Key Sequence)	220
DDMSetLast (Set Cursor to Last Record)	233
DDMSetMinus (Set Cursor Minus)	243
DDMSetNextKeyEqual (Set Cursor to Next Record with Equal Key)	253
DDMSetNextRec (Set Cursor to Next Record)	269

DDMSetPathInfo (Set File or Directory Information)	288
DDMSetPlus (Set Cursor Plus)	291
DDMSetPrevious (Set Cursor to Previous Record)	301
DDMSetRecNum (Set Cursor to Record Number)	314
DDMSetUpdateKey (Set Update Intent by Key Value)	321
DDMSetUpdateNum (Set Update Intent by Record Number)	330
DDMTruncFile (Move EOF to Current Cursor Position)	337
DDMUnLoadFileFirst (Unload Records from File)	339
DDMUnLoadFileNext (Unload Records from File)	349
DDMUnLockRec (Unlock Implicit Record Lock)	358
Chapter 4. VSAM API Common Parameters	361
ACCINTLS (Access Intent List)	361
ACCMTHCL (Access Method Class)	362
ACCMTHLS (Access Method List)	362
ALCINISZ (Allocate Initial Extent)—DFM Only	363
ALTINDLS (Alternate Index List)	364
BASFILNM (Base File)	365
BASMGNM (Base Management Class Name)	365
BASSTGNM (Base Storage Class Name)	366
CODPNT (Code Point Attribute)	366
CSRPOSST (Cursor Position Status)	366
DATE (Date and Time)	367
DELCP (Record Deletion Capability)	369
DFTREC (Default Record)	369
DTACLSNM (Data Class Name)	370
DTALCKST (Data Lock Status)	371
EOFNBR (End of File Record Number)	372
ERRFILNM (Error File Name)	372
FILBYTCN (File Byte Count)	373
FILCHGDT (File Change Date)—DFM Only	373
FILCLS (File Class)	374
FILCRTDT (File Creation Date)	374
FILHDD (File Hidden)	375
FILINISZ (Initial File Size)	375
FILNAM (File Name)	376
FILPRT (File Protected)	377
FILSIZ (File Size)	377
FILSYS (System File)	378
GETCP (File Get Capability)	378
INSCP (File Insert Capability)	379
KEYDEF (Key Definition)	379
KEYDEFCD (Key Definition Error Code)	380
KEYDUPCP (Duplicate Keys Capability)	381
KEYFLDDF (Key Field Definition)	382
KEYVAL (Key Value)	383
LSTACCDT (Last Access Date)—DFM Only	383
LSTARCDT (Last Archived Date)—DFM Only	384

MAXARNB (Maximum Active Record Number)	384
MAXOPN (Maximum Number of Files Opened)	384
MGMCLSNM (Management Class Name)	385
MODCP (File Modify Capability)	385
NEWFILNM (New File Name)	386
RECAL (Record Attribute List)	386
RECCNT (Record Count)	388
RECINA (Inactive Record)	389
RECLEN (Record Length)	389
RECLENCL (Record Length Class)	390
RECNBR (Record Number)	391
RECORD (Record)	391
RTNCLS (File Retention Class)	392
SRVDGN (Server Diagnostic Information)	392
STGCLSNM (Storage Class Name)	393
SVRCOD (Severity Code)	393
SYNERRCD (Syntax Error Code)	396
TITLE (A Brief Description)	397
Chapter 5. VSAM API Flags	399
AccessFlags (Access Flags)	399
DDM_HLDUPD (Hold Update Intent)	400
DDM_UPDCSR (Update Cursor)	400
DDM_INHMODKY (Inhibit Modified Keys)	400
DDM_ALWINA (Allow Cursor to Be Set to Inactive Record)	401
DDM_HLDCSR (Hold Cursor Position)	401
DDM_BYPDGM (Bypass Damaged Records)	401
DDM_NODATA (No Record Data Returned)	402
DDM_ALLREC (All Records, Active and Inactive)	402
DDM_RTNINA (Return Inactive Record)	402
DDM_KEYVALFB (Key Value Feedback)	402
DDM_RECNRFB (Record Number Feedback)	403
DDM_UPDINT (Update Intent)	403
CopyFlags (Copy Flags)	404
DDM_BYPINA (Bypass Inactive Records)	405
DDM_BYPDGM (Bypass Damaged Records)	405
DDM_ACCORD (Access Order)	405
CreateFlags (Create Flags)	405
DDM_FILPRT (Protected File)	406
DDM_FILSYS (System File)	406
DDM_FILHDD (Hidden File)	407
DDM_MODCP (Allow Modify Record Capability)	407
DDM_INSCP (Allow Insert Record Capability)	407
DDM_GETCP (Allow Get Record Capability)	407
DDM_INIEX (Inhibit Initial Extent)	408
DDM_DELC (Allow Record Deletion)	408
DDM_TMPFIL (Temporary File)	408
DDM_ALDUPKEY (Allow Duplicate Keys)	409

Chapter 6. VSAM API Reply Messages	411
Reply Message Interface	411
Reply Message Structure	411
Reply Messages	412
ACCATHRM (Not Authorized to Use Access Method)	415
ACCINTRM (Access Intent List Error)	415
ACCMTHRM (Invalid Access Method)	416
ADDRRM (Address Error)	417
AGNPRMRM (Permanent Agent Error)	418
BASNAMRM (Invalid Base File Name)	418
CLSDMGRM (File Closed with Damage)	419
CMDCHKRM (Command Check)	419
COMMRM (Communications Error)	421
CSRNSARM (Cursor Not Selecting a Record Position)	424
CVTNFNRM (Conversion Table Not Found)	425
DDFNFNRM (Data Description File Not Found)	425
DFTRECRM (Default Record Error)	426
DRCATHRM (Not Authorized to Directory)	426
DRCFULRM (Directory Full)	427
DTARECRM (Invalid Data Record)	427
DUPFILRM (Duplicate File Name)	429
DUPKDIRM (Duplicate Key Different Index)	429
DUPKSIRM (Duplicate Key Same Index)	430
DUPRNBRM (Duplicate Record Number)	432
ENDFILRM (End of File)	433
EXSCNDRM (Existing Condition)	435
FILATHRM (Not Authorized to File)	435
FILDMGRM (File Damaged)	436
FILFULRM (File Is Full)	438
FILIUSRM (File in Use)	439
FILNAMRM (Invalid File Name)	440
FILNFNRM (File Not Found)	440
FILSNARM (File Space Not Available)	441
FILTNARM (File Temporarily Not Available)	442
FUNATHRM (Not Authorized to Function)	442
FUNNSPRM (Function Not Supported)	443
HDLNFNRM (File Handle Not Found)	443
INTATHRM (Not Authorized to Open Intent for Named File)	444
INVFLGRM (Invalid Flag)	444
INVRQSRM (Invalid Request)	445
KEYDEFRM (Invalid Key Definition)	446
KEYLENRM (Invalid Key Length)	447
KEYUDIRM (Key Update Not Allowed by Different Index)	448
KEYUSIRM (Key Update Not Allowed by Same Index)	449
KEYVALRM (Invalid Key Value)	450
LENGTHRM (Field Length Error)	451
NEWNAMRM (Invalid New File Name)	452
OBJNSPRM (Object Not Supported)	452

OPNMAXRM (Concurrent Opens Exceeds Maximum)	453
PRCCNVRM (Conversational Protocol Error)	454
PRMNSPRM (Parameter Not Supported)	455
RECDMGRM (Record Damaged)	455
RECINARM (Record Inactive)	457
RECIUSRM (Record in Use)	457
RECLNRM (Record Length Mismatch)	458
RECNAVRM (Record Not Available)	459
RECNBRRM (Record Number Out of Bounds)	460
RECNFNRM (Record Not Found)	461
RSCLMTRM (Resource Limits Reached on Target System)	462
SRCLMTRM (Resource Limit Reached in Source System)	463
SYNTAXRM (Data Stream Syntax Error)	463
TRGNSPRM (Parameter Not Supported on Target System)	464
UPDCSRRM (Update Cursor Error)	465
UPDINTRM (No Update Intent on Record)	466
VALNSPRM (Parameter Value Not Supported)	466
XLATERM (Translation Error)	467
Appendix A. Programming Extended Attributes in VSAM APIs	469
Glossary	481
Index	483

Figures

1.	Overview of DDM Processing	3
2.	Local VSAM File Component Parts	8
3.	Sequential File with Variable-Length Records	9
4.	Quasi Byte Stream Record File	10
5.	Direct File with Inactive Fixed-Length Records	12
6.	Direct / Sequential File Format	13
7.	Keyed File	14
8.	Keyed File of Fixed-Length Records	16
9.	Lost Update Concurrency Problem	23
10.	DDMDeleteRec Function	69
11.	DDMInsertRecEOF	85
12.	DDMInsertRecEOF	86
13.	DDMInsertRecEOF Function	90
14.	DDMInsertRecEOF Function with DDM_UPDCSR	91
15.	DDMInsertRecKey Function with DDM_UPDCSR	97
16.	DDMInsertRecNum Function	104
17.	DDMInsertRecNum Function with Multiple Records	105
18.	DDMLoadFileFirst Function to a New File	112
19.	DDMLoadFileFirst Function to Append to a File	113
20.	DDMLoadFileFirst Function to Random Load a Direct File	114
21.	DDMLoadFileNext Function to Append to a File	121
22.	DDMModifyRec Function	126
23.	DDMSetBOF Function	143
24.	DDMSetEOF Function	145
25.	DDMSetFirst Function with DDM_ALLREC Set	153
26.	DDMSetFirst Function with DDM_ALLREC Not Set	154
27.	DDMSetKey Function with RelOpr Set to KEYEQ	165
28.	DDMSetKey Function with RelOpr Set to KEYAE	166
29.	DDMSetKey Function with RelOpr Set to KEYAF	167
30.	DDMSetKey Function with RelOpr Set to KEYBE	168
31.	DDMSetKey Function with RelOpr Set to KEYAE	169
32.	DDMSetKey Function with RelOpr Set to KEYBE	170
33.	DDMSetKey Function with RelOpr Set to KEYBE	171
34.	DDMSetKey Function with RelOpr Set to KEYBF	172
35.	DDMSetKeyFirst Function for Ascending Sequence	180
36.	DDMSetKeyFirst Function for Descending Sequence	181
37.	DDMSetKeyLast Function for Ascending Sequence	189
38.	DDMSetKeyLast Function for Descending Sequence	190
39.	DDMSetKeyLimits Function	199
40.	DDMSetKeyNext Function with Key Limits Set	200
41.	Resetting Limits with DDMSetKey Function	201
42.	DDMSetKeyNext Function with Duplicate Key Values	207
43.	DDMSetKeyNext Function for Ascending Sequence	208
44.	DDMSetKeyNext Function for Descending Sequence	209
45.	DDMSetKeyNext Function with Key Limits Set	210

46.	DDMSetKeyNext Function with Hold Cursor Initially On	211
47.	DDMSetKeyNext Function with Hold Cursor Initially On	212
48.	DDMSetKeyNext Function with Hold Cursor Initially Off	213
49.	DDMSetKeyPrevious Function with Duplicate Key Values	224
50.	DDMSetKeyPrevious Function for Ascending Sequence	225
51.	DDMSetKeyPrevious Function for Descending Sequence	226
52.	DDMSetLast DDM_ALLREC Set Off for Sequential File	237
53.	DDMSetLast DDM_ALLREC Set On for Sequential File	238
54.	DDMSetMinus Function	247
55.	DDMSetNextKeyEqual to Access First Duplicate Key	257
56.	DDMSetNextKeyEqual to Access the Next Duplicate Key	258
57.	DDMSetNextKeyEqual to Access Past the Last Duplicate Key	259
58.	DDMSetNextKeyEqual Function with Key Limits Set	260
59.	DDMSetNextKeyEqual Function with Hold Cursor Initially On	261
60.	DDMSetNextKeyEqual function with Hold Cursor Initially On	262
61.	DDMSetNextKeyEqual function with Hold Cursor Initially Off	263
62.	DDMSetNextRec Function with DDM_ALLREC Set	276
63.	DDMSetNextRec Function with DDM_ALLREC Not Set	277
64.	DDMSetNextRec Function with Hold Cursor Initially On	278
65.	DDMSetNextRec Function with Hold Cursor Initially On	279
66.	DDMSetNextRec Function with Hold Cursor Initially On	280
67.	DDMSetNextRec Function with Hold Cursor Initially Off	281
68.	DDMSetPlus Function	295
69.	DDMSetPrevious Function with DDM_ALLREC Set to On	306
70.	DDMSetPrevious Function with DDM_ALLREC Not Set	307
71.	DDMSetRecNum Function	318
72.	DDMSetUpdateKey Function	325
73.	DDMSetUpdateNum Function	334
74.	DDMTruncFile Function	338
75.	DDMUnloadFileFirst Function When Returning Active or Inactive Records	345
76.	DDMUnloadFileFirst Function Skipping Inactive Records	346
77.	DDMUnloadFileFirst Function Skipping Damaged Records	347
78.	DDMUnloadFileFirst Function Unloading in Key Order	348
79.	DDMUnloadFileNext Function	354
80.	DDMUnloadFileNext Function Skipping Inactive Records	355
81.	DDMUnloadFileNext Function Skipping Damaged Records	356
82.	DDMUnloadFileNext Function Unloading in Key Order	357
83.	Record Length Class Promotion	390
84.	DDMSetNextRec ENDFILRM	434
85.	DDMSetKeyNext ENDFILRM	434
86.	Example of C Program using Extended Attributes	469

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to :

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. M13
5600 Cottle Road
San Jose, CA 95193

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Programming Interface Information

This publication documents General-use Programming Interface provided by SMARTdata UTILITIES.

General-use programming interfaces allow the customer to write programs that obtain the services of SMARTdata UTILITIES.

General-use Programming Interface is identified where it occurs with an introductory statement to a section.

Trademarks and service marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
DFSMSdfp
DFSMS/MVS
IBM
MVS/ESA
Operating System/2
Operating System/400
OS/2
OS/400
VisualAge
VM/ESA

The following terms are trademarks of other companies:

Windows	Microsoft Corp.
Windows NT	Microsoft Corp.
Windows 95	Microsoft Corp.

About This Book

This publication introduces the VSAM Application Programming Interfaces (APIs) for the C programmer developing applications in a distributed environment. This publication discusses the capabilities of the VSAM APIs and how they are used to access remote and local data organized in various file types. This access conforms to a set of protocols defined in the Distributed Data Management (DDM) architecture.

As you will learn, VSAM is an implementation of the DDM Architecture. The first chapter describes the subset of the DDM architecture supported by the VSAM APIs. This chapter describes local and remote file access, and the supported record files and file types.

This book describes the VSAM APIs (also referred to as functions) in detail. It tells you how to code the APIs, gives information about API parameters, and information about the API flags. The last chapter describes the API reply messages.

For information on how to use the Distributed FileManager (DFM) for remote record access, see the appropriate platform *User's Guide*.

Who Should Read This Publication

This book is for you if you are a C application programmer who wants to write applications that open, access, modify, and close record files on local or remote systems.

What You Should Know Before Reading This Publication

You should have an understanding of Distributed Data Management (DDM) architecture level 4.0 and C programming language. A good starting point for learning about DDM architecture is *IBM Distributed Data Management: General Information*. Some higher level programming language products, such as COBOL and PL/I, might provide transparent VSAM API support through their runtime libraries.

Bibliography

You can order books by calling IBM Software Manufacturing Solutions at 1-800-879-2755.

<i>Table 1. SMARTdata UTILITIES for Windows Publications</i>	
Publication Title	Order Number
SMARTdata UTILITIES for Windows Set	SBOF-6135
SMARTdata UTILITIES for Windows Distributed FileManager User's Guide	SC26-7134
SMARTdata UTILITIES Data Description and Conversion	SC26-7091
SMARTdata UTILITIES VSAM Application Programming Interface Reference	SC26-7133
SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion	SC26-7092

<i>Table 2. SMARTdata UTILITIES for OS/2 Publications</i>	
Publication Title	Order Number
SMARTdata UTILITIES for OS/2 Set	SBOF-6131
SMARTdata UTILITIES for OS/2 VSAM in a Distributed Environment	SC26-7063
SMARTdata UTILITIES SMARTsort for OS/2 and AIX	SC26-7099
SMARTdata UTILITIES Data Description and Conversion	SC26-7091
SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion	SC26-7092

<i>Table 3. SMARTdata UTILITIES for AIX Publications</i>	
Publication Title	Order Number
SMARTdata UTILITIES for AIX Set	SBOF-6132
SMARTdata UTILITIES for AIX: VSAM in a Distributed Environment	SC26-7064
SMARTdata UTILITIES SMARTsort for OS/2 and AIX	SC26-7099
SMARTdata UTILITIES Data Description and Conversion	SC26-7091
SMARTdata UTILITIES A Data Language Reference for Data Description and Conversion	SC26-7092

<i>Table 4. Other Publications</i>	
Publication Title	Order Number
IBM Distributed Data Management: General Information	GC21-9527
DDM Architecture: Specifications for ADL	SC21-8286
Character Data Representation Architecture, Level 2	SC09-1390
IBM Systems Journal: Volume 31, No. 3, 1992	G321-5483
IBM Dictionary of Computing	SC20-1699
Compilers—Principles, Techniques, and Tools: by the Addison-Wesley Publishing Company	
IEEE Standard for Binary Floating-Point Arithmetic: ANSI/IEEE STANDARD	754-1985
INTEL 387™ DX User	
IBM Distributed Data Management: General Information	GC21-9527
IBM Distributed Data Management: Reference Guide	SC21-9526
Using Distributed Data Management for the IBM Personal Computer	SC21-9643
AS/400 Communications: Distributed Data Management Guide	SC21-9600
CICS/Distributed Data Management: User's Guide	SC33-0695
IBM 4680 Store Systems: Distributed Data Management: User's Guide	SC30-4915
DFSMS/MVS Version 1 Release 2 Distributed FileManager/MVS Guide and Reference	SC26-4915
DFSMS/MVS Version 1 Release 3 DFSMSdftp Diagnosis Reference	LY27-9606
AIX SNA Server/6000: Configuration Reference	SC31-7002
AIX SNA Server/6000: User's Guide	SC31-7002
AIX DCE Administration Guide	SC23-2475
Encina Server Administration: System Administrator's Guide and Reference for AIX	SC23-2461
SFS Administrator's Guide and Reference Version 2 Release 2	SC33-1609
SFS Programmer's Guide and Reference Version 2 Release 2	SC33-1610
VM/ESA R2.2 SFS and CRR Plan, Administration and Operations	SC24-5649
Distributed Data Management Architecture, Architecture Reference	SC21-9526
OS/2 WARP (Version 3) Control Program Programming Reference	G25H-7102

Using This Reference

Before you begin using this reference, read the following sections to understand the format and access functions for VSAM APIs.

Notation Conventions

The function descriptions and examples are shown in C language. Lengths, code points, bit flag masks, and other values are shown in the following hexadecimal notation:

X'hex value'

with the hexadecimal value enclosed in single quotes following a capital X.

Bit constants appear in the following format:

B'bit value'

with the bit value enclosed in single quotes following a capital B.

Severity codes are shown in decimal and hexadecimal notation.

Note: See the file DUBCODPT.H in the installation directory for an example of the code point notation.

Function Descriptions

The functions in this book follow a pseudo-C high-level language format. The following example outlines the template used for VSAM API descriptions.

DDMExample

DDMExample (Example)

This is the purpose of the function.

Syntax

This is the invocation format (Call Interface) for the function and describes all the parameters of the function.

```
#include dub.h
```

```
APIRET DDMExample (ULONG          Parm 1,  
                  PULONG         Parm 2,  
                  );
```

Parameters

This section contains the parameters that apply to the function.

Parm1

The first parameter (ULONG) of the function.

Parm2

The second parameter (PULONG) of the function.

There are four types of parameters:

Function specific	Used only by the function, such as FileName.
Common	Described in Chapter 4, "VSAM API Common Parameters" on page 361
AccessFlags	Described in "AccessFlags (Access Flags)" on page 399
CreateFlags	Described in "CreateFlags (Create Flags)" on page 405

Returns

This section lists possible reply messages that can be returned in response to invoking this API.

In addition, each function returns a return-code value of the type APIRET.

Remarks

This section contains general comments about the function.

Effect on Cursor Position

This section describes the effect the function has on the position of the cursor.

Locking (for Local VSAM File System Only)

This section describes the kind of locking that occurs for each function on the local VSAM file system.

Exceptions

This section contains tables that list the reply messages you will normally receive and provide detailed information about what causes the reply messages.

DDMExample

Examples

This section contains examples to illustrate what changes may be caused by the function invocation, such as cursor movement and limit resetting.

DDMExample

Chapter 1. Introduction to VSAM as a DDM Implementation

This chapter describes the subset of the Distributed Data Management (DDM) architecture supported by the VSAM APIs. It discusses the API parameters, flags, and messages.

This chapter describes:

- Distributed Data Management Concepts
- Record types and attributes
- Access Methods
- Record file types
- VSAM cursor
- Lock management

Distributed Data Management Overview

SMARTdata UTILITIES implements two components that manage access to files: the local VSAM file system and Distributed FileManager (DFM). The local VSAM file system provides record-type access on the workstation. The Distributed FileManager provides client remote record access to other DDM server implementations. The availability of these SMARTdata UTILITIES components is platform dependent. See the appropriate SMARTdata UTILITIES publication for your platform.

The Distributed Data Management architecture is a methodology used to store, organize, and access data. The architecture defines the protocol for data connectivity between computer systems, regardless of their individual application programs, user applications, hardware, or software.

Using the VSAM APIs, C application programmers can retrieve, add, update, and delete data records from files that reside on the same system or other systems,

The DDM architecture is based on a client/server model. The system that initiates a request for access to data is called the **source** system, or **client**. The system that contains the requested data is called the **target** system, or **server**.

Note: In conformance with this model, the local VSAM file system behaves like a server, though the data is local.

The following terms are used in describing how DDM works.

Local File If data is requested from a file that is located on the system that initiated the request, that file is called a local file.

Remote File If data is requested from a file that is not located on the system that initiated the request, that file is called a remote file.

Note: The definition of local or remote is always from the point of view of the system requesting the data.

Source System The system that initiates requests for access to data is called the source system. The source system can request data from its own local files or from the remote files of another system. A component of the source system is the DDM client. It translates the source system's request for data from a remote system into a standardized DDM request. The DDM client routes the request to the network access software of the source system, which sends the request to the corresponding network access software of the system that contains the requested data.

Target System The system that contains the requested data is called the target system. A component of the target system is the DDM server, which receives the DDM client's request and translates it into a data management request that the target system understands. Once the target system has processed the request, it returns the results of the request to the DDM server. The DDM server routes the results of the request to the network access software which sends the results to the source system.

The Distributed Data Management architecture is represented in Figure 1. The text that follows describes the steps involved in record file access.

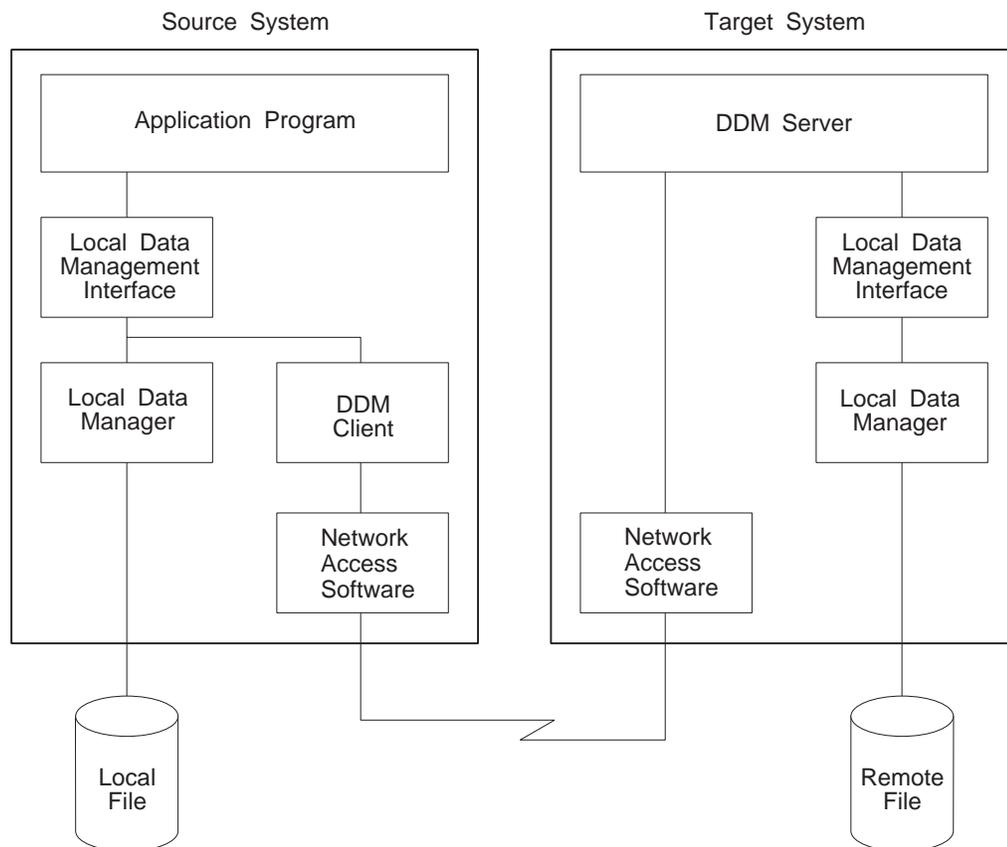


Figure 1. Overview of DDM Processing

- An **Application Program** initiates processing by requesting data.
- The **Local Data Management Interface (LDMI)** determines whether the data requested by the application is on a local (source) or a remote (target) system. If the data is on the local system, the **Local Data Manager (LDM)** of the source system retrieves the requested data from storage. If the data is on the remote system, LDMI invokes the DDM Client.
- The **Distributed Data Management Client** translates the local command into one or more Distributed Data Management commands.
- The **network access software** on the **Source System** transmits the commands to the network access software on the **Target System**.
- The **network access software** on the **Target System** directs the Distributed Data Management command to the Distributed Data Management Server, which handles the request.
- The **Distributed Data Management Server** interprets the Distributed Data Management commands and builds the calls for LDMI on the Target system. The Dis-

tributed Data Management Server builds a data stream with the retrieved data. It then inserts a reply into the data stream and transmits it back to the source system.

Starting with the following section, the rest of this chapter discusses how the DDM architecture is implemented in SMARTdata UTILITIES and supported by the VSAM APIs.

DDM Record Types

Every record-oriented file consists of a set of records. Records are the basic unit of data stored in record-oriented files and are transferred between requesters and files. The record length can be either fixed or variable. The *record number* indicates the record's position in the file in which it is stored. The first position for a record in a file has a record number of one.

The VSAM APIs support two DDM record formats: RECORD and RECINA.

RECORD formats

These are active records and can have fixed or variable lengths. When you create a file, specify a RECLLEN (Record Length) attribute as either the length of the fixed records or the maximum length of the variable records. See 389. for a description of the record length parameter.

Fixed-length record (RECFIX)

A record whose length is specified as an attribute (RECLLEN) of the file in which it is stored and cannot be changed.

Variable-length record (RECVAR)

A record whose length can be changed after it has been written to a file. The length of individual records in the file varies from record to record, but it cannot exceed the maximum length specified by the file's RECLLEN attribute.

Initially-variable-length record (RECIVL)

A record whose length is specified the first time it is written to a file. Once a file position in a file has been assigned a record length, the length of the record position is fixed and cannot be changed. The length of individual records in the file varies from record to record, but it cannot exceed the maximum length specified by the file's RECLLEN attribute.

RECINA formats

These are inactive records used to represent record positions where a record has never been inserted or where a previously active record has been deleted. The RECINA parameter specifies the required length of any record to be inserted at that record position.

Record Attribute Lists (RECALs)

A record attribute list (RECAL) is used to transmit more than one attribute of a record as a single unit. For example, the record number or key value and the record itself can be returned in a RECAL. A RECAL can also return duplicate records using the RECCNT parameter and DATA fields. The record is returned as DATA and the number of duplicate records is returned in RECCNT.

See “RECAL (Record Attribute List)” on page 386 for a description of the RECAL parameter.

Extended Attributes

The VSAM APIs support Extended Attributes (EAs) to associate DDM attributes with a file. The set of VSAM API file attributes is a superset of the standard set of file attributes. This allows programs using the VSAM APIs to access both DDM and operating system dependent attributes without opening the file.

The DDM file attributes supported by the local VSAM file system are listed in Table 13 on page 37 and Table 14 on page 38. Table 13 on page 37 lists the EAs that can only be viewed, and Table 14 on page 38 lists the EAs that can be modified.

The VSAM APIs assume each DDM file attribute is described in a DDM format. These formats are described in Chapter 4, “VSAM API Common Parameters” on page 361.

The EAs reflecting DDM file attributes are coded in C with a prefix of “.DDM_.” The VSAM APIs use the OS/2 DOS-like “EAOP2” structures to read and write EA lists.

The following example is an overview of how to request two EAs (.DDM_DELCP and .DDM_FILCLS) when issuing DDMQueryFileInfo for a sequential, delete-capable file in the current directory. For examples of C code to set up the “GEA2List” and “FEA2List,” see Appendix A, “Programming Extended Attributes in VSAM APIs” on page 469 .

```

DDMQueryFileInfo("\SAMPLE.SEQ",
                1L,
                pointer to an EAOP2 structure,
                size of EAOP2 structure);

Input Data Structures
struct _EAOP2 {
    (4)pointer to GEA2List structure
    (4)pointer to FEA2List structure
    (4)offset to error if any
};
/* end of EAOP2 structure */
struct _GEA2List {
    ---- (4)length of structure = 25
    | (4)nextentry offset = 10 /* each entry must be on a 4 byte boundary */
    | (1)length of name 1 = A
    ---- (B)name 1 = .DDM_DELCP
    ---- (4)next entry offset = 0 /* no entry after this one */
    | (1)length of name 2 = B
    ---- (C)name 2 = .DDM_FILCLS
};
/* end of GEA2List structure */

Structure _FEA2List {
    (4)Length of structure = 3C /* total length of data expected */
    /* each entry is on an 4 byte boundary */
};
/* end of FEA2List structure */

Output Data Structures

Structure _FEA2List {
    ----(4)length of structure = 3C
    | (4)next entry offset = 1C /*note: each entry must be on a 4 byte
    | boundary */
    | (1)flag byte = 0
    | (1)length of name 1 = A
    | (2)length of value for name 1 = 7
    | (B)name 1 = .DDM_DELCP
    | (7)value 1 = 00000007 /* length of value */
    | 111B /* DDM code point for DELCP */
    ---- F1 /* DDM Value for TRUE */
    (2) /* 2 bytes of padding to force */
    /* next entry to a 4 byte */
    /* boundary */
    ----(4)next entry offset = 0 /* there is no next entry */
    | (1)flag byte = 0
    | (1)length of name 2 = B
    | (2)length of value for name 2 = 8
    | (C)name 2 = .DDM_FILCLS
    | (8)value 2 = 00000008 /* length of value */
    | 1110 /* code point for FILCLS */
    ---- 143B /* sequential file */
    /* end FEA2List structure */
};

```

Record Files

A *record file* is a file in which data is stored as a set of discretely addressable structures called records. A record file class describes a method of organizing, accessing, and managing a set of records. The VSAM APIs support sequential, direct, keyed, and alternate index file classes.

All files have the following major components:

- File attributes that are stored as Extended Attributes (EAs), such as record length and file class.
- File record extents that store the record data.
- Special objects:
 - The index of a keyed file is stored in a separate file that is given an internal VSAM name, .DDMEA (AIX local VSAM file system only).
 - Alternate index files related to a base key file.

The length of the records of a file can be either fixed or variable. Once a variable-length record is inserted into a record position of the file, the length of the record at that position remains fixed if the record class is initially variable. It remains variable if the record class is variable.

A file is created with either *delete-capable* or *non-delete-capable* status. If a file is delete-capable, you can issue the DDMDelRec function to delete records from that file. If a file is non-delete-capable, the DDMDelRec function is rejected when issued for the file. You specify delete status when creating the file.

An *access method* is used to process records in a record file. The VSAM APIs support methods that access records by number and by key value. When the DDMOpen (Open File) function opens the file, the access method is bound to a file and remains bound to the file until the DDMClose (Close File) function closes the file or the function is terminated. The access method maintains a cursor for each file to which it is bound. The cursor is set to the beginning of the file when the access method is used to open a file. Access methods are described in “Access Methods” on page 18. The DDM cursor and cursor movement is described in “DDM Cursor” on page 21.

Records can be inserted into a file when it is created, or the application can insert the records later. In order to update or delete a record in a file, you must place an update intent on the record by using the appropriate VSAM API.

Important Note

The local VSAM file system cannot prevent non-DDM access to local VSAM managed files. If these files are processed by non-DDM functions (such as other APIs or user functions), information about the files can be lost and the local VSAM file system will not be able to process the files. Therefore, users MUST NOT access local VSAM-managed files using non-DDM functions.

Record File and Record Length Classes

The VSAM APIs support the following record classes:

- Sequential
- Direct

- Keyed
- Alternate index

The VSAM APIs support three logical record length classes:

- Fixed
- Variable
- Initially Variable

For the local VSAM file system, each of the file classes supported is implemented as a meta-file on top of a standard file. Each file (see Figure 2) consists of two parts.

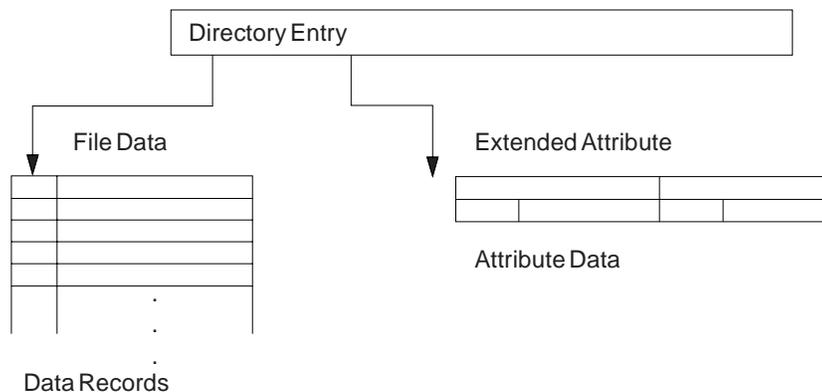


Figure 2. Local VSAM File Component Parts. This figure illustrates the two component parts of a Record File within a Byte Stream file: the Data Records and the Attribute Data.

1. Data and Control Structures (Records)

This is the user file data along with an architected set of control data structures. These structures are defined in a way that allows the DDM file model semantics to be implemented on top of a standard file. From the file system perspective, this is simply the data portion of the file.

2. Attribute Data

The Attribute Data is additional descriptive information required to describe a record-oriented file. This information is called the DDM Attributes. For the AIX local VSAM file system, all of the DDM Attributes are kept in .DDMEA files.

Sequential Files

A sequential file contains records that are arranged in exactly the same order they were placed in the file.

After the initial loading of records, additional records can be added at End-of File (EOF) or inserted into existing inactive record positions. There is no relationship between the contents of a record and its record number.

When a sequential file is created, its allocated record positions can be either:

- initialized to a specified default value,
- initialized as inactive records, or
- uninitialized.

When a file is opened, the cursor is positioned at the Beginning-of-File (BOF). The BOF position for a sequential file is always the position before any record position. The first record position of a sequential file is always the first record in the file, whether the record is active or inactive. The EOF position for a sequential file is one position past the last record position at which an active or inactive record exists. The last record position of a sequential file is always the last active or inactive record in the file.

Figure 3 gives a logical view of a sequential file with variable-length records.

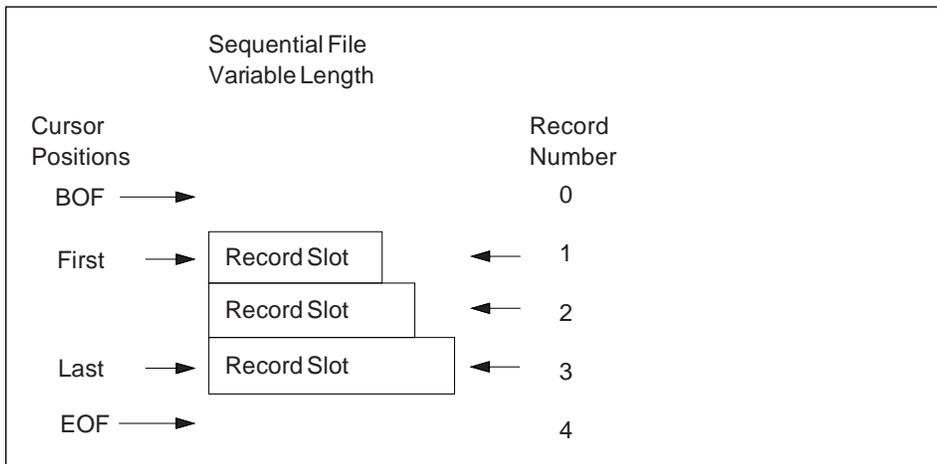


Figure 3. Sequential File with Variable-Length Records

Quasi Byte Stream Files

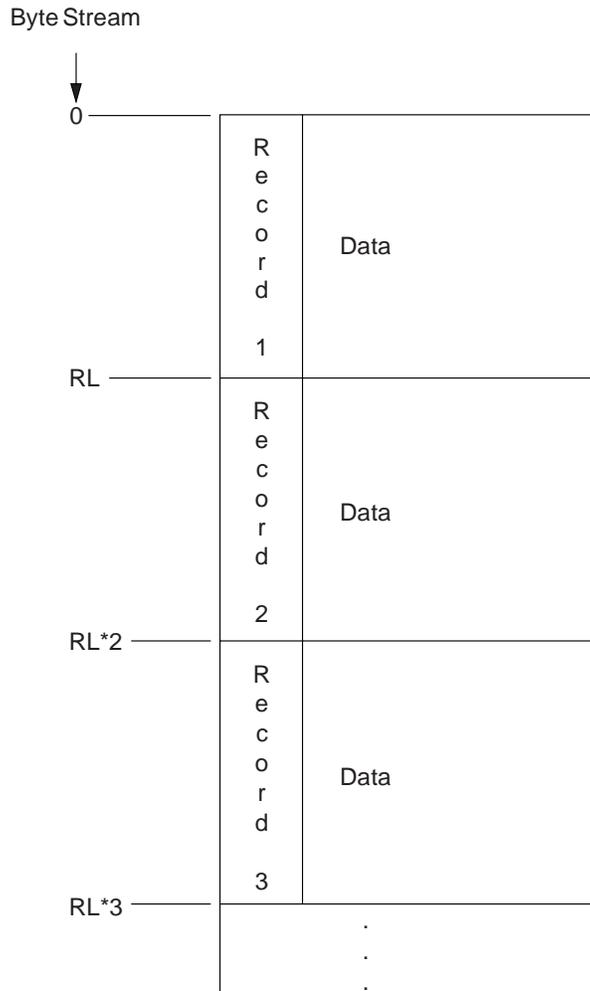
There is a special requirement that a certain type of local VSAM file also look like a byte stream file. A quasi byte stream file is a sequential record file that is created with non-delete-capable status and with fixed-length records. It does not have any record headers or separators. A quasi byte stream file can be read as a pure byte stream file through local byte stream I/O with no change to byte stream applications. There can be no inactive records in a quasi byte stream record file.

Since the file has the same format as a byte stream file, byte stream applications are able to do byte stream read operations on this type of sequential record file.

Important Note

Non-VSAM API applications can read, but not modify, local quasi-byte stream files. If these files are modified by non-VSAM functions, such as user functions, the file attributes will not be updated and information about the files can be lost.

The format of quasi byte stream record files is shown in Figure 4 on page 10



NOTE: RL = Max Record Length (4K default)

Figure 4. Quasi Byte Stream Record File

Direct Files

A direct file contains records that have a relationship between the record contents and the position at which the record is stored. An application program inserting a record into a direct file uses the record number to find the place to insert the record. The application uses the value of one of the record fields as the record number, or calculates a record number value.

When you open the file, the cursor points to the BOF position. For direct files:

The BOF position is always one position before the first record position.

The first record position is the first active record position of the file.

Do not confuse this with *record number one*, which can contain an active record, but not necessarily so.

The last record position is always the last active record in the file.

The EOF position is one position past the last active record position.

You can insert a record at EOF or past EOF in a direct file. If you insert a record past EOF, VSAM will insert inactive records (if they don't already exist) starting at EOF up to the record position where the desired record is to be inserted. For direct files with delete-capable status, you can move the EOF position toward the beginning of the file by deleting the last active record in the file.

The physical boundary for a direct file is defined by the requester when the file is created.

When you create a direct file, you can specify allocated positions as either:

- Initialized to a specified default active record. If you initialize a file with default records, all allocated record positions are active.
- Initialized as inactive records. If you initialize a direct file with inactive records, each record position in the file is inactive until a record is inserted into it. Records can be inserted at any inactive record position within the physical boundaries of the file as long as space is available in the file.
- Uninitialized and treated as inactive records because they are beyond the EOF.

See Figure 5 for a logical view of the BOF, EOF, first record position, and last record position.

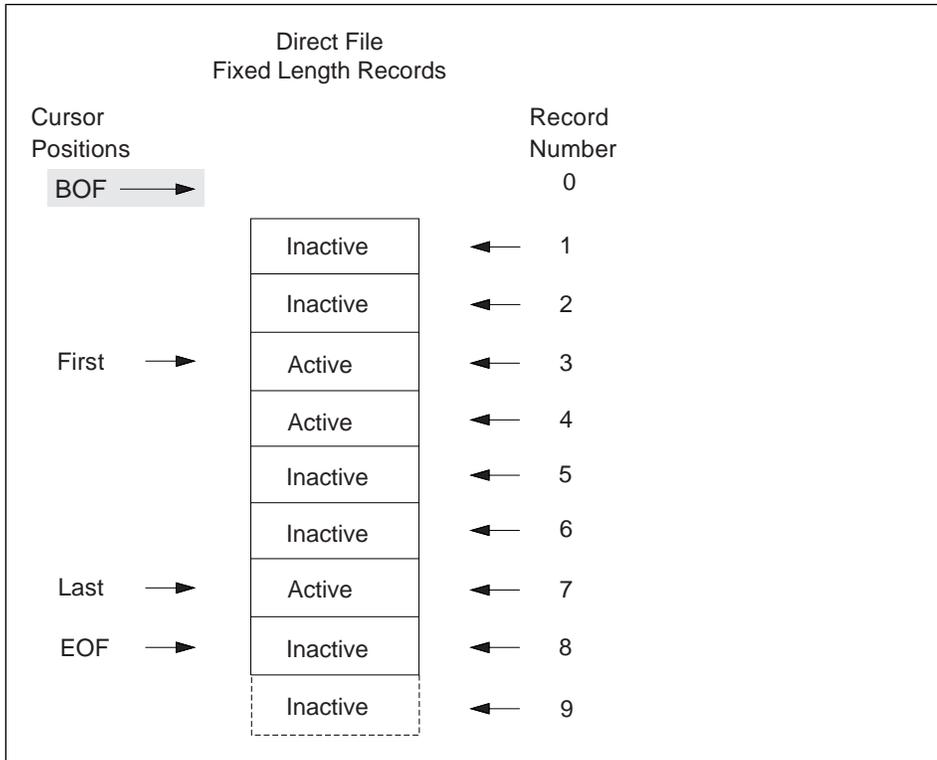


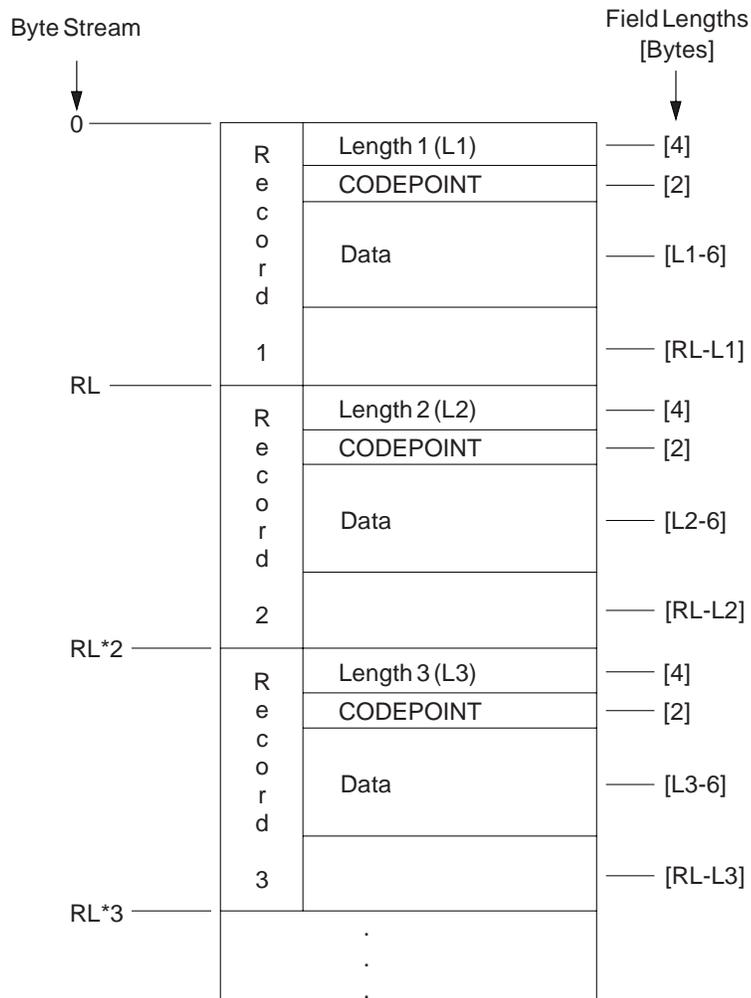
Figure 5. Direct File with Inactive Fixed-Length Records. Note that the EOF is one position past the last active record, even though there is another inactive record in the file.

Media Formats for Direct and Sequential Files

The media formats, or data and control structures, for direct and sequential files are identical. However, a number of semantic differences between them are found in the descriptions of the access functions. Some differences are:

- The EOF positioning is different when you delete records from the end of the file: EOF for direct files retreats, while EOF for sequential files does not.
- In direct files, records can be inserted beyond EOF, and EOF gets moved after the last active inserted record. For sequential files, records can be inserted at, but not beyond, EOF.
- Cursor positioning differs: in DDMSetFirst, DDM_ALLREC must always be False for direct files.

Direct and sequential files have the format shown in Figure 6 on page 13.



NOTE: RL = Max Record Length

Figure 6. Direct / Sequential File Format. This figure shows the format of a direct or sequential file superimposed on a byte stream file. The same format applies to fixed-, variable-, and initially-variable-length records.

Keyed Files

A keyed file is implemented as two files: a DDM sequential file (called the base file or keyed file) and an index file that maps keys to record numbers.

When a keyed file is created, the file name specified in the function is used for the base file. A name is generated by the file system for the index file. The index file is always placed in the same subdirectory as the base file. The name (not including path) of the

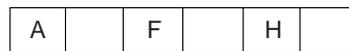
base file is placed in the attribute information of the index file and vice versa for the name of the index file.

When the base file is later opened, the name and path information given in DDOpen is used to locate the base file. The attribute information in the base file is used to get the index file name. Then the same path specified in the function is used to locate the index file. The base and its index must always be in the same subdirectory.

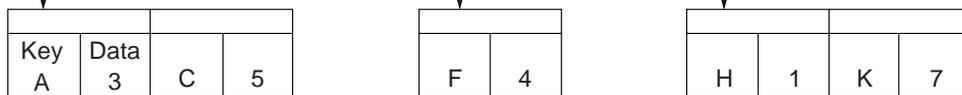
Keyed sequential files contain an overlying B-Tree Indexing structure. Figure 7 illustrates the basic keyed file concepts. Note that the data field of the index file contains a base-file record number.

KEYED SEQUENTIAL
(index file)

Pointer Records



Data Records



KEYED SEQUENTIAL
(base file)

File Records



Figure 7. Keyed File. Example of the structure of a local VSAM file system keyed file. Note that a "keyed file" really consists of two files: an index file and a base file.

A keyed file supports keyed access to the records in the file. Each keyed file has a *file index* that contains an entry for each active record in the file. The index allows an application to process records by referring to the key of the record.

The key, also called the *key field* or *record key* is the portion of the record containing information that identifies the record. Index entries identify a record by the value of its key and the position of the record in the file. The index is ordered as specified by the file attribute, KEYDEF, which you defined when the file is created.

A keyed file has a primary index and can have multiple alternate index files. Any update to a keyed file causes automatic updates to all alternate index files built on that file.

A variable-length record in a keyed file must be large enough for all the key field values in the file index and any alternate index files that use the keyed file as a base file.

When they are created, keyed files can be either:

- initialized with inactive records,
- initialized with active records that have a specified default value, or
- uninitialized.

The BOF for a keyed file is the position before any record positions. When the file is opened, the cursor is positioned at the BOF. The first record position is the first active or inactive record position of the file. This may not be the first record in key sequence.

The EOF position for a keyed file is one past the last record position at which an active or inactive record exists. The last record position is the last active or inactive position of the file. This may not be the last record in key sequence.

Cursor Positioning Functions: Different VSAM APIs have different cursor positioning characteristics:

1. DDMSetNextRec and DDMSetPrevious, set the cursor position relative to **record positions** in the file.
2. DDMSetKeyNext, DDMSetKeyPrevious, and DDMSetNextKeyEqual, set the cursor position relative to **key sequence**.

In Figure 8 on page 16, for example, if the cursor is initially positioned at EOF, DDMSetKeyPrevious moves the cursor to the record whose key is BBB, the last key by key sequence.

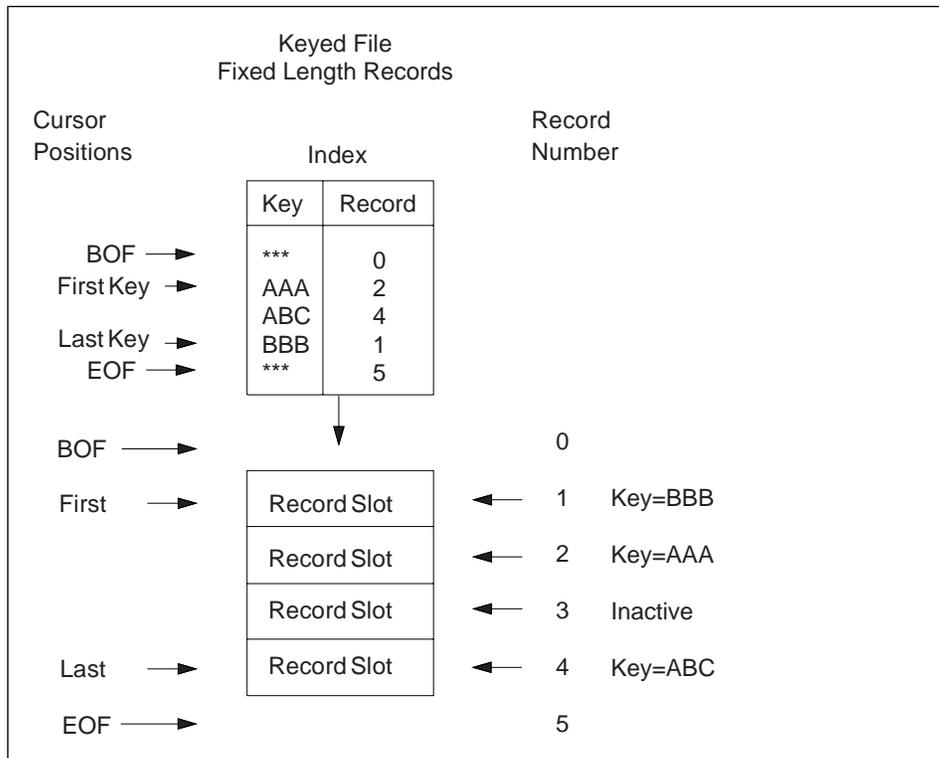


Figure 8. Keyed File of Fixed-Length Records

Alternate Index File

Physically, an alternate index file is identical to the index portion of a keyed file. An alternate index file allows the user to view the *base file* from a different perspective. Typically, an alternate index file will key off a different portion of the base records, thus allowing the user to retrieve records in a different sequence from that provided by the normal keyed file processing. VSAM APIs only support alternate indexes for keyed files. The index file is built from the base portion of the keyed file.

A *base file* is an existing keyed file upon which an alternate index is built. Base file records are the same as the alternate index file records, however the record contents of the base file do not appear in the alternate index file. The base key file has one primary index, and can have multiple alternate index files. Each alternate index file contains an entry for each active record in the file. Updates to a base file result in automatic updates to all of its alternate index files. Every alternate index file has a separate set of attributes.

The BOF and EOF positions in an alternate index file are the same as those of its base file. When you open the file the cursor is positioned at BOF.

If the file has variable record lengths, the lengths must be large enough to include all of the key field values for the alternate index file.

The key, which is also called the *key field* or *record key*, is the portion of the record containing information that identifies the record. Index entries use the value of a record key and the position of the record in the file to identify the record. You use the KEYDEF attribute when creating the file to specify the ordering of the records.

Figure 7 on page 14 illustrates index files.

Fixed-, Variable-, and Initially-Variable-Length Records

The media formats for fixed-, variable-, and initially-variable-length records are identical. However, there are a number of semantic differences between them, found in the descriptions of the access functions.

Some semantic differences between the three classes of record lengths are:

- Fixed-length records must all be the same length.
- Variable-length records can be overwritten with either smaller or larger records as long as the maximum record size is not exceeded.
- Initially-variable-length records can be any size up to the maximum record length when first inserted. Only a record of the same size can overwrite the original record at that location.

File Naming Conventions

The VSAM APIs do not enforce any specific file naming syntax. A file name provided by the application must conform to the naming syntax of the local installed byte stream file system (such as Fat or HPFS) or the target remote DDM system. However, conversion of mixed-case file names to upper-case file names can occur. Thus, any reply messages that contain a file name may not reflect the case that was used as input to the API.

The local VSAM system on OS/2 supports the double backslash naming convention for files located on remote nodes of a Local Area Network (LAN). Note that this convention (known as UNC, for Universal Naming Convention) is only supported for LANs administered by the OS/2 LAN Server product. UNC is used to represent remote file names that were never qualified with a drive letter, for example: DosOpen (\\servername\dir1\a.dat).

Performance Considerations

The following sections recommend which access method to use to optimize performance.

Sequential and Direct Files

For sequential or direct files use the following access methods.

- Specify RELRNDAM on DDMOpen if the predominant order of reading records will be sequential.
- Specify RNDRNDAM on DDMOpen if the predominant order of reading records will be random.
- Specify CMBRNAM if you do not expect a sequential or random access bias.

Keyed and Alternate Index Files

For keyed and alternate files use the following access methods.

- Specify RELKEYAM on DDMOpen if:
 1. the predominant order of reading records will be in key sequence,
 2. the file was loaded in key sequence,
 3. you expect new records to be added in key sequence, and
 4. the file was created without delete capability (DDM_DELCP).
- Specify RNDKEYAN on DDMOpen If the predominant order of reading records will be random.
- Specify CMBKEYAM on DDMOpen
 1. if you do not expect a key sequence or random access bias, or
 2. if the predominant order of reading records will be in key sequence but the file is or has become "disorganized" because it was not loaded sequentially, or because it was created with delete capability (DDM_DELCP).

If a keyed file becomes disorganized (less sequential) after many delete and insert operations, you may be able to improve performance by reorganizing the file using DDMUnLoadFile UnloadOrder=KEYORD and DDMLoadFile.

Access Methods

The VSAM APIs have a series of access methods that provide consistent ways to access the records in a file. To understand how your choice of access method can also affect performance, see Performance Considerations.

For all access methods, the file indexes are updated when keys are updated or when records are inserted or deleted. The following list describes the access methods that the VSAM APIs use when opening files with DDMOpen:

- RELRNBAM (Relative by Record Number Access Method)

Use this access method to process records according to the current cursor position in the record number sequence. The record number is not specified to identify the record; all positioning is relative to the current cursor position. For keyed and alternate index files, records are processed as though the file were sequential. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

You can use this access method with sequential, direct, keyed, or alternate index files.

- RNDRNBAM (Random by Record Number Access Method)

Use this access method to process records in a random sequence as determined by the requester. Record numbers (the positions of records in the file) are used to identify the records. For keyed and alternate index files, records are processed as though the file were sequential. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

You can use this access method for sequential, direct, keyed, or alternate index files.

- CMBRNBAM (Combined Record Number Access Method)

This access method combines the functional capabilities of the RELRNBAM and the RNDRNBAM access methods. The cursor can be set to point to any record by specifying its record number. Relative requests for neighboring records can then be made without specifying record numbers. For keyed and alternate index files, records are processed as though the file were sequential. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

You can use this access method for sequential, direct, keyed, or alternate index files.

- RELKEYAM (Relative by Key Access Method)

Use this access method to process records of keyed or alternate index files in key value sequence. Records can be accessed by moving forward or backward from the current record according to the key sequence. If duplicate keys are present in the file, they are processed in First-In-First-Out (FIFO) order. If a record's key value is modified, its record number is not changed. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

You can use this access method for keyed or alternate index files only.

- RNDKEYAM (Random by Key Access Method)

Use this access method to process records in keyed or alternate index files in a random sequence as determined by the requester. Records are selected by their key values, not by their relative positions. If a record's key value is modified, its record number is not changed. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

You can use this access method for keyed or alternate index files only.

- CMBKEYAM (Combined Key Access Method)

This access method combines the functional capabilities of the RELKEYAM and the RNDKEYAM access methods. The cursor can be set to point to any record by specifying its key. Relative requests for neighboring records can then be made without specifying keys. If duplicate keys are present in the file, they are processed in FIFO order. If a record's key value is modified, its record number is not changed. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

This access method is valid for keyed or alternate index files only.

- CMBACCAM (Combined Access Method)

This access method combines the functional capabilities of the CMBKEYAM and the CMBRNBAM access methods. The cursor can be set to a record with a key or to a record number. Then, from that position, the cursor can be set relatively by key value or by record number. If duplicate keys are present in the file, they are processed in FIFO order. If a record's key value is modified, its record number is not changed. The indexes over the file are maintained when keys are updated or when records are inserted or deleted.

Table 5 shows the access methods you can use with each file class.

Access Method	Access Description	SF	DF	KF	AIF
RELRNBAM	Relative by record number	X	X	X	X
RNDRNBAM	Random by record number	X	X	X	X
CMBRNBAM	Combined by record number	X	X	X	X
RELKEYAM	Relative by key			X	X
RNDKEYAM	Random by key			X	X
CMBKEYAM	Combined by key			X	X
CMBACCAM	Combined access	X	X	X	X
X	The access method supports the file class.				
Blank	The access method does not support the file class.				
SF	Sequential file.				
DF	Direct file.				
KF	Keyed File				
AIF	Alternate Index File				

Promoting Access Methods

The DDM architecture permits the promotion of user-specified access methods. For remote data access, see your DDM server implementation documentation. The following promotions and exceptions pertain to the local VSAM file system.

To open a file, an application program issues the DDMOpen (Open File) function. The local VSAM file system verifies whether the type of file specified by the function can be opened by DDMOpen and notifies the application. If the file can be opened, then:

1. The specified access method is promoted to the appropriate CMBxxxAM.
2. The file is opened under that access method.

3. The access method is bound to the file. The access method remains bound to the file until an application program issues a DDMClose function or the application program is terminated.

If the access method cannot be applied to the file class, the attempt to open the file is rejected with the INVRQSRM reply message. The local VSAM file system also issues the INVRQSRM reply message when a keyed file class function is issued for a non-keyed file.

Each access method defines the VSAM APIs it supports under its instance commands list. These instance commands are also called the *access method commands*. For more information on the commands, see Chapter 2, “Function Lists” on page 31, “Access Functions Applicable to Each File Class” on page 34, and “Cursor-Positioning Functions Applicable to Each File Class” on page 35.

Access method commands are processed by the local VSAM file system and applied against the access method to which the file is bound. If a command is issued and is not supported by the file class, unpredictable results may occur.

The local VSAM file system uses the following promotion rules:

- Promote RELRNBAM and RNDRNBAM access methods to CMBRNBAM to allow any direct or sequential file to be accessed by any of the record number cursor positioning functions.
- Promote RELKEYAM, RNDKEYAM, and CMBKEYAM access methods to CMBACCAM to allow any keyed file to be accessed by any of the cursor positioning functions.

DDM Cursor

Each open file in the DDM architecture has a logical structure associated with it called a cursor. The cursor points to a particular position within the file and also maintains certain information about the file. The DDM cursor has the following logical elements:

- The current position in the file. This can be BOF, an individual record number in the file, or EOF. When the file is opened, the cursor is initially set to BOF.
- The access intent specified for the file when it was opened.
- The level of file sharing specified when the file was opened.
- A hold cursor indicator that specifies if hold cursor position has been requested or not. This indicator is set (or remains set) if the DDM_HLDCSR bit in the AccessFlags parameter of the DDMSetxxx functions is true and is reset if the DDM_HLDCSR bit is false.
- The most recent update intent placed on a record in the file. The update intent is set by the DDMSetUpdatexxx functions. It may also be set by the DDMGetRec function and by most of the DDMSetxxx functions by setting Bit 0 in the AccessFlags parameter.

Note that the update intent can only be specified for a single record.

- The position of the record with this update intent. This record position can be different from the current record if a DDMSetUpdatexxx function was issued or a DDMSetUpdatexxx function is issued with the DDM_HLDUPD bit of the AccessFlags parameter set.
- A locked record indicator that specifies whether the update intent record is locked.
- The high key limit for the file that is set with the DDMSetKeyLimits function.

The cursor position can be adjusted explicitly by issuing the appropriate DDMSetxxx function. The effect each function has on the cursor position is described for each function in the “Effect on Cursor Position” section.

The hold cursor indicator is checked by the DDMSetNextRec, DDMSetKeyNext, and DDMSetNextKeyEqual functions to determine if the cursor should remain at its current position. If the hold cursor indicator has been set on by a previous function and the DDM_HLDCSR bit in the AccessFlags parameter of the current function is false, the cursor remains at its current position when:

- The function is DDMSetNextRec and one of the following conditions is true.
 - The record is active.
 - The record is inactive and the DDM_ALLREC bit in the AccessFlags parameter of this function is true.
- The function is DDMSetKeyNext and the record is active.
- The function is DDMSetNextKeyEqual, the specified key is equal to the key of the current record, and the record is active.

In all other cases, the cursor position is updated.

In the case of errors, the cursor position can be determined from the CSRPOSST (Cursor Position Status) parameter returned in the reply message. (The value of CSRPOSST is always X'F1'.) The CSRPOSST (Cursor Position Status) parameter is described in Chapter 4, “VSAM API Common Parameters” on page 361.

DDM Lock Management

DDM lock management supervises the file and record locks of one or more users on a set of files. The responsibilities of lock management are to:

- Accept lock requests and determine whether the lock request can be granted.
- Keep track of all the file locks held by each user.
- Update the correct cursor to track the granting and releasing of record locks.

Concurrency Protection

File and record locks provide concurrency protection in a multi-user, shared data environment. An example of a typical concurrency problem occurs when an update to a record is lost because of simultaneous updating of the file by two or more users. Figure 9 on page 23 illustrates this problem.

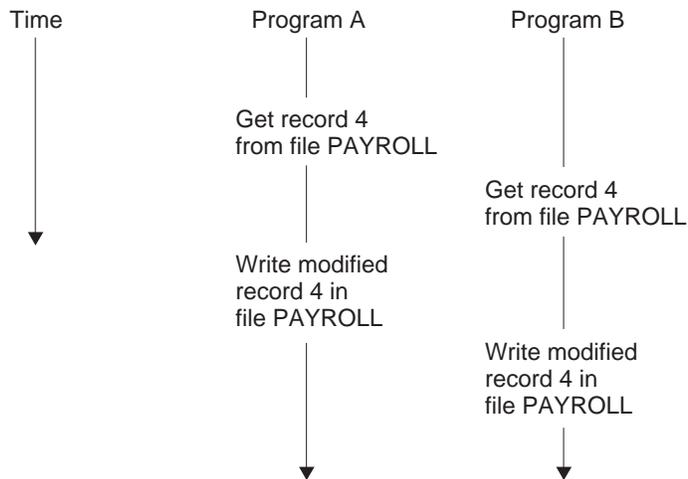


Figure 9. Lost Update Concurrency Problem

Another concurrency problem occurs when a user does not have exclusive rights to a file after it has been accessed. This means that a user cannot read and retrieve the same data from a file accessed before because another user has modified it in the interim. This is called a repeatable read problem.

To avoid these and other concurrency problems, lock protection is needed for files and records. DDM provides file and record locking functions.

The following pages describe the requesting and granting of file and record locks and the level of protection available with locks. The responsibilities of lock management are also summarized.

File Locking

DDM file locks require a requester to obtain an appropriate level of access to a file before allowing any operations to be performed on any record in the file. A requester obtains the appropriate level of access by acquiring a lock that indicates the requester's processing intentions for the file and the degree to which the requester is willing to share the file with concurrent users.

DDM allows the requester to declare processing intentions as follows (the DDM abbreviation for the processing intent is given in parentheses):

- Reference Only (**GET**)

The requester intends to read or use the data in the specified file, but does not intend to modify, delete, or insert any data in the file.

- Change (**MOD**)

The requester intends to update the file by modifying, deleting, or inserting data.

The requester can declare the tolerable level of file sharing with concurrent users. The possible sharing levels are as follows (the DDM abbreviation for the sharing level is given in parentheses):

- No Sharing (**NON**)
The requester wants exclusive control of the file and is not willing to share the file with any concurrent users.
- Reference Only (**GET**)
The requester is only willing to share the file with concurrent users that have Reference Only (GET) intention.
- Change (**MOD**)
The requester is willing to share the file with concurrent users that intend to get, modify, delete, or insert data in the file.

Concurrent users are defined as threads of the same process or threads from different processes.

These processing intentions and file sharing levels produce the combinations listed in Table 6. These combinations are the basis for the different types of DDM-specified file locks.

<i>Table 6. File Locking Combinations</i>			
Processing Intent	Sharing Level		
	NON	GET	MOD
GET	GETNONLK	GETGETLK	GETMODLK
MOD	MODNONLK	MODGETLK	MODMODLK

A requester can acquire many locks on a single file as long as there are no lock conflicts. A lock conflict is a request by any process to obtain a file lock for a file that is already locked exclusively by another process. The locks can all be of the same lock type or different types. The operating system defines the maximum number of file locks a single requester can have on a single file. If a file lock is requested for a file that already has the maximum number of file locks on it, the RSCLMTRM (Resource Limit Error) reply message is returned.

If the file to be locked is an alternate index file, both the base file and the alternate index file are locked.

Requesting and Releasing File Locks

File locks are requested and released implicitly by the following functions:

Function	Action
DDMOpen	Open file
DDMCreateAltIndex	Create alternate index file
DDMDelete*	Delete file
DDMLoadFileFirst	Load records into file
DDMLoadFileNext	Load next record into the file
DDMUnLoadFileFirst	Unload records from file
DDMUnLoadFileNext	Unload next record from the file
DDMRename	Rename file
* DDMDelete does <i>not</i> implicitly release a file lock. The file no longer exists after a DDMDelete.	

Record Locking (Implementation is Dependent on the Server)

The local VSAM file system supports record locking only for files on the client OS/2 system. This section describes this support.

The local VSAM file system supports record locks so that a requester can perform intended operations on a record without interference from concurrent users. Record locks are used only when the requester opens a file with an intent to update the file and specifies that the file is to be shared with another updater. This is called *opened for multiple updaters*.

The local VSAM file system obtains only exclusive record locks. This means that only the requester can update the record. Concurrent users are unable to read the record. Record locks requested for an alternate index file are obtained on the records of the base file. Each process can lock one record in a file. Thus, multiple records in a file can be locked if the file was opened for multiple updaters.

The local VSAM file system does not prevent more than one process from updating a record concurrently; it does not prevent multiple threads within a process from accessing and updating the same record. When threads from the same process are accessing a file using the same file handle, they should use a semaphore to provide mutual exclusion on the file.

Requesting and Releasing Record Locks

Record locks can be implicitly obtained by the following functions:

Function	Action
DDMGetRec	Get record function
DDMDeleteRec	Delete record function
DDMInsertRecxxx	Any insert record function
DDMModifyRec	Modify record function
DDMSetxxx	Any set function

Record locks can be explicitly obtained by the following functions:

Function	Action
DDMSetUpdateNum	Set update intent by record number
DDMSetUpdateKey	Set update intent by key value

The DDMClose function implicitly releases all record locks. Table 7 summarizes which functions lock records and when these record locks are released. DDMUnlockRec explicitly removes a record lock.

Table 7. Releasing Record Locks

Implicit Lock Commands	Release Lock When				
	Function Completed	Record Updated	Cursor Moved	File Closed	See Note
DDMGetRec		X	X	X	X
DDMSetxxx		X	X	X	X
DDMSetUpdateKey		X	X	X	X
DDMSetUpdateNum		X	X	X	X
DDMModifyRec	X				
DDMDeleteRec	X				
DDMInsertRecxxx	X				

Note: DDMUnlockRec, or any function that references a record other than the one currently pointed to by the cursor.

Promoting Locks (Implementation is Dependent on the Server)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation.

The local VSAM file system uses the following locking rules:

- Only exclusive record locks are obtained. This means that only the requester can update the record. Concurrent users are unable to read the record. For more information, see “Record Locking (Implementation is Dependent on the Server)” on page 25.
- DDMLoadFileFirst/Next file locks are promoted to MODNONLK.
- DDMCopyFile promotes “copy-from file” parameter to GETNONLK and the “copy-to file” parameter to MODNONLK.
- Only one exclusive file lock can be held on a file.

A requester can request a GETMODLK, MODGETLK, or MODMODLK lock on a file that is on a redirected drive of a LAN server. To prevent an application from reading a file that another application (on a different system) is modifying, the local VSAM file system promotes the lock as follows:

- GETMODLK to GETGETLK
- MODGETLK to MODNONLK
- MODMODLK to MODNONLK

Granting File and Record Locks

All requests for a lock are made to the operating system by the local VSAM file system. For file locks, the operating system examines all of the file locks held by concurrent users on a file, determines whether a conflict would occur, and decides whether the requested lock can be granted.

Table 8 is a summary of the rules for granting file locks. The left column lists the requested lock types with the strongest lock at the top. Across the top of the table are all of the concurrent user-held locks, from the strongest to the weakest. To read the table, locate the requested lock type in the left column. Then, locate the strongest of the locks held by concurrent users across the top of the table. The intersection of the selected row and column indicates whether the lock request can be granted or whether a lock conflict occurs.

Table 8 (Page 1 of 2). Table for Granting File Locks

Requested Lock	Concurrent User Held File Lock						
	MODNONLK	GETNONLK	MODGETLK	MODMODLK	GETGETLK	GETMODLK	None
MODNONLK	*	*	*	*	*	*	GT
GETNONLK	*	*	*	*	*	*	GT
MODGETLK	*	*	*	*	*	GT	GT
MODMODLK	*	*	*	GT	*	GT	GT
GETGETLK	*	*	*	*	GT	GT	GT

Table 8 (Page 2 of 2). Table for Granting File Locks							
Requested Lock	Concurrent User Held File Lock						
	MODNONLK	GETNONLK	MODGETLK	MODMODLK	GETGETLK	GETMODLK	None
GETMODLK	*	*	GT	GT	GT	GT	GT
Notes:							
GT	Lock request is granted.						
*	Lock conflict occurs.						

The local VSAM file system only attempts to get the lock once and then the lock request is refused with one of the following reply messages:

- File in use reply message (FILIUSRM) if the request is for a file lock.
- Record in use reply message (RECIUSRM) if the request is for a record lock.

DDM Architecture Promotions and Exceptions

All promotions and exceptions described below are allowed by the DDM architecture and by the SAA subset definitions.

The following promotions and exceptions are made by the local VSAM file system:

- Promote the RELRNBAM and RNDRNBAM access methods to the CMBRNBAM access method.

This allows any direct or sequential file to be accessed by any of the record number cursor positioning commands.

- Promote the RELKEYAM, RNDKEYAM, and CMBKEYAM access methods to the CMBACCAM access method.

This allows any keyed file to be accessed by any of the cursor positioning commands.

- This item is for OS/2 local VSAM files on the client OS/2 workstation only:

The local VSAM file system obtains only exclusive record locks. This means that only the requester can update the record. Concurrent users are unable to read the record.

- DDMLoadFileFirst/Next file locks are promoted to MODNONLK.
- DDMCopyFile promotes "copy from file" parameter to GETNONLK and the "copy to file" parameter to MODNONLK.
- Only one exclusive file lock can be held on a file.
- DDMLoadFileFirst returns FILIUSRM instead of INVRQSRM when a file has already been opened by DDMOpen, DDMLoadFileFirst (DDM_CHAIN flag on), or DDMUnLoadFileFirst (More Data flag on).
- DDMUnLoadFileFirst returns FILIUSRM instead of INVRQSRM when a file has already been opened by DDMOpen, DDMLoadFileFirst (DDM_CHAIN flag on).

- DDMDDelete and DDMRename returns FILIUSRM instead of INVRQSRM when a file has already been opened by DDMSOpen, DDMSLoadFileFirst (DDM_CHAIN flag on), or DDMSUnloadFileFirst (More Data flag on).

Technical Considerations

This section contains a list of implementation considerations:

- As part of its internal processing, when the local VSAM file system is instructed to open (DDMSOpen) a member of a keyed file set, all members are opened, using the same access and file share values as specified for the explicitly opened file. If a subsequent DDMSOpen function is issued for a different member of that file set using different access and file share specifications, a conflict will occur.

For example, assuming base file X.BAS and alternate index file X.ALT. If X.BAS is opened for Insert Access Intent with FileShare NONE, VSAM issues a DosOpen for X.BAS and X.ALT, using the same access and share specification. Any subsequent attempts to open X.ALT with Get Access Intent will fail because X.ALT was already opened with FileShare NONE.

- Attempting to issue a name-based VSAM API for a file not belonging to the local VSAM file system will result in the function being rejected with the FILATHRM reply message and a server diagnostic code of 1 (for local VSAM file systems only).
- When processing multiple records, it is faster to request multiple records with DDMSSetNextRec than to request a single record multiple times. The same applies for DDMSSetPrevious and the key file equivalents.
- The local VSAM file system can control access to the same file from multiple processes. It does not control access to the same file from multiple threads in the same process. The process is responsible for synchronization of its threads.
- Exercise caution in defining the record length for a variable-length file. Variable-record-length files are implemented as fixed-record-length files with each record being the maximum length variable record allowed. If small records are used in a variable-length file with a large record length, there can be an excessive amount of unused space within the file. (For local VSAM file systems only.)

Chapter 2. Function Lists

The tables in this chapter group the VSAM APIs according to their capabilities. The VSAM APIs are called VSAM API functions, VSAM functions, or simply functions. The tables describe:

1. VSAM Function Parameters
2. VSAM Functions
3. Access Functions Applicable to Each File Class
4. Cursor–Positioning Functions Applicable to Each File Class
5. Record File Attributes by File Class
6. Modifiable Record File Attributes by File Class
7. Access Functions Applicable to Each Access Method

The server support of these APIs is implementation specific. In general, the details in this chapter is specific to the local VSAM file system.

VSAM Function Descriptions

Table 9 lists and briefly describes each VSAM function.

<i>Table 9 (Page 1 of 3). VSAM Functions</i>	
Function	Description of Function
DDMClose	Terminates the logical connection that DDMOpen establishes between the requester and a file.
DDMCopyFile	Copies a record-oriented file to the target system.
DDMCreateAltIndex	Creates an alternate index file on the target system. The alternate index file provides a key-field access sequence to the records in an existing base target system file. In VSAM, the base file must be a keyed file.
DDMCreateRecFile	Creates a record file on the target system.
DDMDelete	Deletes a file from the target system, releases all locks held on the file, and releases the space it occupied.
DDMDeleteRec	Deletes the record that has an update intent placed on it.
DDMForceBuffer	Commits a file's cached information to non-volatile storage.
DDMGetRec	Gets and returns the record indicated by the current cursor position.
DDMGetReplyMessage	Gets and returns a reply message issued from the previously requested function.
DDMInsertRecEOF	Inserts a record at the end of the file.
DDMInsertRecKey	Inserts one or more records, according to their key values, wherever there is available space in the file.
DDMInsertRecNum	Inserts one or more records at the position specified by the record number parameter.
DDMLoadFileFirst	Loads one or more records into a file.
DDMLoadFileNext	Continues the load of one or more records into a file. Issue DDMLoadFileFirst before DDMLoadFileNext.

Table 9 (Page 2 of 3). VSAM Functions

Function	Description of Function
DDMModifyRec	Modifies the record that has an update intent placed upon it.
DDMOpen	Establishes a logical connection between the using program on the source system and the file on the target system.
DDMQueryFileInfo	Returns the information for a specific file.
DDMQueryPathInfo	Returns information for a specific file or subdirectory.
DDMRename	Renames an existing file.
DDMSetBOF	Sets the cursor to the beginning of file (that is, to the position ahead of the first record on the file).
DDMSetEOF	Sets the cursor to the end of file (that is, to the position following the last record of the file).
DDMSetFileInfo	Specifies information for a file or a directory.
DDMSetFirst	Sets the cursor to the first record of the file.
DDMSetKey	Positions the cursor based on the key value supplied and the relational operator specified for the relational operator parameter.
DDMSetKeyFirst	Sets the cursor to the first record of the file in key sequence.
DDMSetKeyLast	Sets the cursor to the last record of the file in key sequence order.
DDMSetKeyLimits	Sets the limits of the key values for subsequent DDMSetKeyNext or DDMSetNextKeyEqual functions.
DDMSetKeyNext	Sets the cursor to the next record of the file in the key sequence order that follows the record currently indicated by the cursor.
DDMSetKeyPrevious	Sets the cursor to the previous record of the file in the key sequence order that precedes the record currently indicated by the cursor.
DDMSetLast	Sets the cursor to the last record of the file.
DDMSetMinus	Sets the cursor to the record number of the file indicated by the cursor minus the number of record positions specified by the CsrDisp parameter.
DDMSetNextKeyEqual	Sets the cursor to the next record in the key sequence if the key field of that record has a value specified in the KeyValBuf parameter.
DDMSetNextRec	Sets the cursor to the next record of the file that has a record number one greater than the current record position.
DDMSetPathInfo	Specifies information for a file or a directory.
DDMSetPlus	Sets the cursor to the record number of the file indicated by the cursor plus the integer number of records specified by the CsrDisp parameter.
DDMSetPrevious	Sets the cursor to the record of the file that has a record number one less than the current cursor position.
DDMSetRecNum	Sets the cursor to the record of the file indicated by the RecordNumber parameter.
DDMSetUpdateKey	Places an update intent on the record that has a key value equal to the KeyValBuf parameter. The cursor position is not changed.
DDMSetUpdateNum	Places an update intent on the record at the position specified by the RecordNumber parameter. The cursor position is not changed.

<i>Table 9 (Page 3 of 3). VSAM Functions</i>	
Function	Description of Function
DDMTruncFile	Moves EOF to current cursor position.
DDMUnloadFileFirst	Transfers one or more records of a source file to a target system.
DDMUnloadFileNext	Continues the transfer of one or more source file records to a target system. Issue DDMUnloadFileFirst before DDMUnloadFileNext.
DDMUnlockRec	Releases all implicit record locks on records.

Parameters Used in Function Descriptions

Table 10 lists and describes the parameters used with VSAM functions.

<i>Table 10. Parameters Used with VSAM Functions</i>	
Parameter Data Type	Description
USHORT	2 bytes.
ULONG	4 bytes. This is the natural word size of the system. It may be passed by value or reference as a parameter.
PBYTE	Pointer to a byte.
PULONG	Pointer to a ULONG.
szNAME	Null (0) terminated ASCII character string. This parameter is passed only by reference.
PSZ	Pointer to a null-terminated string.
HDDMLOAD	4 bytes. Contains a handle to a DDM file being loaded with DDMLoadFileNext.
PHDDMLOAD	Pointer to a handle to a DDM file being loaded with DDMLoadFileNext.
HDDMFILE	4 bytes. Contains a handle to a DDM file.
PHDDMFILE	Pointer to a handle to a DDM file.
PDDMRECORD	Pointer to a DDM record structure.
PDDMOBJECT	Pointer to a DDM object structure.
PKEYDEFBUF	Pointer to a DDM key buffer structure.
PDDMDFTREC	Pointer to a DDM default record initialization buffer.
PRECNUM	Pointer to a DDM record number structure.
RECNUM	DDM record number structure.
CODEPOINT	2 bytes
PEAOP2	Pointer to an EAOP2 structure.
PRESULTSCODES	Pointer to a structure used in DosExecPgm.
OTHER	Any other structure. This parameter is passed only by reference.

Access Functions Applicable to Each File Class

Table 11 on page 35 lists the functions that can be used with each file class.

<i>Table 11. Access Functions Applicable to Each File Class</i>					
Functions	Description	SF	DF	KF	AIF
DDMClose	Close file	X	X	X	X
DDMCopyFile	Copy file	X	X	X	
DDMCreateAltIndex	Create alternate index file				X
DDMCreateRecFile	Create record file	X	X	X	
DDMDelete	Delete file	X	X	X	X
DDMDeleteRec	Delete record	X	X	X	X
DDMGetRec	Get record	X	X	X	X
DDMInsertRecEOF	Insert record at EOF	X	X	X	X
DDMInsertRecKey	Insert record by key			X	X
DDMInsertRecNum	Insert record by number	X	X	X	X
DDMLoadFileFirst	Load first file	X	X	X	
DDMLoadFileNext	Load next file	X	X	X	
DDMModifyRec	Modify record	X	X	X	X
DDMOpen	Open file	X	X	X	X
DDMRename	Rename file	X	X	X	X
DDMSetUpdateKey	Set update intent by key			X	X
DDMSetUpdateNum	Set update intent by record number	X	X	X	X
DDMTruncFile	Move EOF to current cursor position	X			
DDMUnLoadFileFirst	Unload first file	X	X	X	X
DDMUnLoadFileNext	Unload next file	X	X	X	X
DDMUnLockRec	Unload implicit record lock	X	X	X	X
X	The function is supported by the file class.				
Blank	The function is not supported by the file class and may cause unpredictable results.				
SF	Sequential file.				
DF	Direct file.				
KF	Keyed file.				
AIF	Alternate index file.				

Cursor-Positioning Functions Applicable to Each File Class

Table 12 on page 36 lists the cursor-positioning functions that can be used with each file class.

Table 12. Cursor Positioning Functions Applicable to Each File Class

Functions	Description	SF	DF	KF	AIF
DDMSetBOF	Set cursor to BOF	X	X	X	X
DDMSetEOF	Set cursor to EOF	X	X	X	X
DDMSetFirst	Set cursor to first record	X	X	X	X
DDMSetKey	Set cursor by key			X	X
DDMSetKeyFirst	Set cursor to first record by key			X	X
DDMSetKeyLast	Set cursor to last record by key			X	X
DDMSetKeyLimits	Set key limits			X	X
DDMSetKeyNext	Set cursor to next record by key			X	X
DDMSetKeyPrevious	Set cursor to previous record by key			X	X
DDMSetLast	Set cursor to last record	X	X	X	X
DDMSetMinus	Set cursor minus	X	X	X	X
DDMSetNextKeyEqual	Set cursor to next record with equal key			X	X
DDMSetNextRec	Set cursor to next record	X	X	X	X
DDMSetPlus	Set cursor plus	X	X	X	X
DDMSetPrevious	Set cursor to previous record	X	X	X	X
DDMSetRecNum	Set cursor to record number	X	X	X	X
<p>X The function is supported by the file class.</p> <p>Blank The function is not supported by the file class and unpredictable results may occur.</p> <p>SF Sequential file.</p> <p>DF Direct file.</p> <p>KF Keyed file.</p> <p>AIF Alternate index file.</p>					

Record File Attributes by File Class

These EAs can be viewed by using DDMQueryPathInfo or DDMQueryFileInfo functions on the local VSAM file system.

Table 13 (Page 1 of 2). Record File Attributes by File Class

Record File Attributes Name	EA Description	File Class			
		Sequential File	Direct File	Keyed File	Alternate Index File
ACCMTHLS	Access Method List	X	X	X	X
ALTINDLS	Alternate Index File List			X	
BASFILNM	Base File Name				X
DELCP	Record Deletion Capability	X	X	X	X
DFTREC	Default Record	X	X	X	X
DTACLSNM	Data Class Name	X	X	X	X
EOFNBR	End of File Record Number	X	X	X	X
FILBYTCN	File Byte Count Number	X	X	X	X
FILCLS	File Class	X	X	X	X
FILCRTDT	File Creation Date	X	X	X	X
FILHDD	Hidden File	X	X	X	X
FILINISZ	Initial File Size	X	X	X	X
FILPRT	File Is protected.	X	X	X	X
FILSIZ	Number of active and inactive record positions. Not applicable to files with variable-length records.	X	X	X	X
FILSYS	System File	X	X	X	X
GETCP	Record Get Capability	X	X	X	X
INSCP	Record Insert Capability	X	X	X	X
KEYDEF	Key Definition			X	X
KEYDUPCP	Duplicate Keys Capability			X	X
MAXARNB	Maximum Active Record Number	X	X	X	X
MGMCLSNM	Management Class Name	X	X	X	X
MODCP	Record Modify Capability	X	X	X	X
RECLEN	Record Length	X	X	X	X
RECLENCL	Record Length Class	X	X	X	X
RTNCLS	Retention Class	X	X	X	X
STGCLSNM	Storage Class Name	X	X	X	X
TITLE	Title	X	X	X	X

Table 13 (Page 2 of 2). Record File Attributes by File Class

Record File Attributes Name	EA Description	File Class			
		Sequential File	Direct File	Keyed File	Alternate Index File
X	The file EA is supported by the file class.				
Blank	The file EA is not supported by the file class.				

Modifiable Record File Attributes by File Class

These EAs can be modified using DDMSetPathInfo or DDMSetFileInfo functions on the local VSAM file system.

Table 14. Modifiable Record File Attributes by File Class

Record File Attributes Name	EA Description	File Class			
		Sequential File	Direct File	Keyed File	Alternate Index File
DELCP	Record Deletion Capability	X (see note)	X	X	X
FILHDD	Hidden File	X	X	X	X
FILINISZ	Initial File Size	X	X	X	X
FILPRT	File Is Protected	X	X	X	X
FILSYS	System File	X	X	X	X
GETCP	Record Get Capability	X	X	X	X
INSCP	Record Insert Capability	X	X	X	X
MGMCLSNM	Management Class Name	X	X	X	X
MODCP	Record Modify Capability	X	X	X	X
STGCLSNM	Storage Class Name	X	X	X	X
TITLE	Title	X	X	X	X
X	The file EA is supported by the file class.				
Note	The file EA is not modifiable if the record length class is "fixed."				

Private File Attributes by File Class

These EAs are private to the local VSAM file system and cannot be viewed or modified with VSAM functions and should not be changed by the native workstation commands.

Table 15. Private File Attributes by Class

Record File Attributes Name	EA Description	File Class			
		Sequential File	Direct File	Keyed File	Alternate Index File
MINARNB	Minimum Active Record Number	X	X	X	X
VERSION	RLIO version that created this file	X	X	X	X
PRMINDEX	Name of Primary Index File			X	
BASCHGDT	Base file change date				X
BASEFILE	Name of Base File (see note)			X	
MAXFILESIZE	Maximum File Size	X	X	X	X
FILCHGDT	File Change Date	X	X	X	X
LSTACCDT	Last Access Date	X	X	X	X
PHYEOF	Physical End of File	X	X	X	X
X	The file EA is supported by the file class.				
Blank	The file EA is not supported by the file class.				
Note	Used for Primary Index File Only.				

Access Functions Applicable to Each Access Method

<i>Table 16 (Page 1 of 2). Access Functions Applicable to Each Access Method</i>			
Functions	RELNRBAM	RNDRNBAM	CMBRNBAM
DDMClose	1	1	X
DDMDeleteRec	1	1	X
DDMGetRec	1	1	X
DDMInsertRecEOF	1	1	X
DDMInsertRecKey			
DDMInsertRecNum	2	1	X
DDMModifyRec	1	1	X
DDMOpen	1	1	X
DDMSetBOF	1	1	X
DDMSetEOF	1	1	X
DDMSetFirst	1	1	X
DDMSetKey			
DDMSetKeyFirst			
DDMSetKeyLast			
DDMSetKeyLimits			
DDMSetKeyNext			
DDMSetKeyPrevious			
DDMSetLast	1	1	X
DDMSetMinus	1	2	X
DDMSetNextKeyEqual			
DDMSetNextRec	1	2	X
DDMSetPlus	1	2	X
DDMSetPrevious	1	2	X
DDMSetRecNum	2	1	X
DDMSetUpdateKey			
DDMSetUpdateNum	2	1	X
DDMTruncFile	1	1	X
DDMUnLockRec	1	1	X

Table 16 (Page 2 of 2). Access Functions Applicable to Each Access Method

Functions	RELRNBAM	RNDRNBAM	CMBRNBAM
X	The function is supported by the CMBRNBAM access method.		
1	RELRNBAM and RNDRNBAM are promoted to CMBRNBAM.		
2	The function is processed without regard to the restrictions associated with this access method because the access method is promoted to CMBRNBAM access method (local VSAM file system only).		
Blank	The function is not supported by the access method and may cause unpredictable results.		

Access Functions Applicable to Each Access Method Continued

<i>Table 17. Access Functions Applicable to Each Access Method Continued</i>				
Functions	RELKEYAM	RNDKEYAM	CMBKEYAM	CMBACCAM
DDMClose	1	1	1	X
DDMDeleteRec	1	1	1	X
DDMGetRec	1	1	1	X
DDMInsertRecEOF	2	2	2	X
DDMInsertRecKey	1	1	1	X
DDMInsertRecNum	2	2	2	X
DDMModifyRec	1	1	1	X
DDMOpen	1	1	1	X
DDMSetBOF	1	1	1	X
DDMSetEOF	1	1	1	X
DDMSetFirst	2	2	2	X
DDMSetKey	2	1	1	X
DDMSetKeyFirst	1	1	1	X
DDMSetKeyLast	1	1	1	X
DDMSetKeyLimits	1	2	1	X
DDMSetKeyNext	1	2	1	X
DDMSetKeyPrevious	1	1	1	X
DDMSetLast	2	2	2	X
DDMSetMinus	2	2	2	X
DDMSetNextKeyEqual	1	2	1	X
DDMSetNextRec	2	2	2	X
DDMSetPlus	2	2	2	X
DDMSetPrevious	2	2	2	X
DDMSetRecNum	2	2	2	X
DDMSetUpdateKey	2	1	1	X
DDMSetUpdateNum	2	2	2	X
DDMTruncFile	2	2	2	X
DDMUnLockRec	1	1	1	X

X The function is supported by the CMBACCAM access method.

1 RELKEYAM, RNDKEYAM, and CMBKEYAM are promoted to CMBACCAM.

2 The function is processed without regard to the restrictions associated with this access method because the access method is promoted to CMBACCAM access method (local VSAM file system only).

Chapter 3. VSAM API Functions

This chapter describes the VSAM API functions, their formats and characteristics.

The functions are in alphabetical order and are presented in the structure described in “DDMExample (Example)” on page xviii.

This section describes the General-Use Programming Interface you can use to obtain the services of SMARTdata UTILITIES.

DDMClose

DDMClose (Close File)

This function ends the logical connection that DDMOpen establishes between the requester and a file.

Syntax

```
#include dub.h /* Required for all platforms */  
  
APIRET DDMClose (HDDMFILE      FileHandle  
                 );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

Returns

Message ID	Code Point	Message Title
CLSDMGRM	X'125E'	File Closed with Damage
HDLNFNRM	X'1257'	File Handle Not Found

Remarks

The DDMClose function considers the file closed unless the reply message indicates that an error was detected before starting the DDMClose function. For example, if you receive a SYNTAXRM, PRCCNVRM, or FUNNSPRM reply message. This is true even if the reply message has a severity code greater than 4.

The DDMClose function also works on byte stream files.

In order to reflect changes in file attributes from open-file activities, DDMClose updates the following EAs if the file was opened with other than just GETAI. Examples of changes in file attributes from open-file activities are: update, insert, delete, or truncate.

- EOFNBR
- FILSIZ
- MAXARNB

These EAs are updated not only for a base file, but for all associated index files when DDMClose is issued.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

This function destroys the cursor. there is no cursor position.

Error Termination (SVRCOD of 8)

The cursor position is the same as it was before the function was called. If the error termination occurs after starting DDMClose, this function destroys the cursor. Therefore, there is no cursor position. The value of the CSRPOSST

DDMClose

(Cursor Position Status) parameter on the reply message indicates the state of the cursor.

Severe Termination (SVRCOD of 16 or higher)

CSRPOSST on the reply message. indicates the cursor position. If the severe termination occurs after starting DDMClose, this function destroys the cursor. Therefore, there is no cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

This function releases all locks that are held by the VSAM file system on records in the file. All file locks that were acquired implicitly by the DDMOpen function are released.

If DDMClose ends with a reply message that has a severity code value of ERROR or higher, then:

- For error termination (SVRCOD of 8): Record locks and file locks are the same as before DDMClose was issued. If the error termination occurs after starting DDMClose, this function releases all record locks. The file lock that is obtained by DDMOpen is released.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record locks. If the severe termination occurs after starting the DDMClose function, this function releases all record locks. The file lock that is obtained by DDMOpen is released.

Even if an error occurs after starting DDMClose, this function releases all record locks, and the file lock that is obtained by DDMOpen is released.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file is not open (the file handle is not valid).	HDLNFRM
The file is closed, but it is not possible to complete all operations on the file.	CLSDMGRM

DDMCopyFile

DDMCopyFile (Copy File)

This function copies a file to the target system. (You cannot use DDMCopyFile to copy an alternate index file.)

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMCopyFile (PSZ          FromFileName,
                   PSZ          ToFileName,
                   ULONG         CopyFlags,
                   PBYTE         SubsetDefBuf,
                   CODEPOINT     ToFileOld,
                   CODEPOINT     ToFileNew
                   );
```

Parameters

FromFileName

The pointer (PSZ) to the name of the record-oriented file to be copied. This file is the source of the DDMCopyFile function.

ToFileName

The pointer (PSZ) to the name of the record-oriented file to copy to. This file is the target of the DDMCopyFile function.

CopyFlags

CopyFlags must be set to 0. The bit flags are:

Bit	Meaning
0–31	Reserved flags.

SubsetDefBuf

The pointer (PBYTE) to the subset definition buffer. This pointer must be set to null.

ToFileOld

The code point (CODEPOINT) that specifies the action to take if the file name that is pointed to by ToFileName already exists. The only valid value is:

CPYERR Return Duplicate File Name (X'1483').

The function is rejected with DUFFILRM, and the option returns an error condition (SVRCOD=X'0008'). You must specify CPYERR.

ToFileNew

The code point (CODEPOINT) that specifies the action to take if the file name that is pointed to by ToFileName does not exist. The only valid value is:

DDMCopyFile

CPYDTA Copy with Data Option (X'1466').

You should create a new file and copy the data to it. The new file is created with the same file attribute values as the copy-from file for the following file EAs:

DELCP	File Deletion Capability
DFTREC	Default Record
DTACLSNM	Data Class Name
FILCLS	File Class
FILINISZ	Initial File Size
FILPRT	File Protected
GETCP	File Get Capability
INSCP	File Insert Capability
KEYDEF	Key Definition
KEYDUPCP	Duplicate Keys Capability
MODCP	File Modify Capability
MGMCLSNM	Management Class Name
RECLEN	Record Length
RECLENCL	Record Length Class
RTNCLS	File Retention Class
STGCLSNM	Storage Class Name
TITLE	A Brief Description

For the definition of these EAs, see Chapter 4, "VSAM API Common Parameters" on page 361.

The other file attributes are set as appropriate for a newly created file. The data content of the Fromfile is copied to the Tofile. CPYDTA must be specified.

Returns

Message ID	Code Point	Message Title
DUPFILRM	X'1207'	Duplicate File Name
FILDMGRM	X'125A'	File Damaged
FILIUSRM	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RSCLMTRM	X'1233'	Resource Limits Reached on Target System
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

For remote files, Distributed FileManager requires that the path information for both the FromFile name and the ToFile name must be specified, and it must be the same (for OS/2 only).

DDMCopyFile

Since alternate index files cannot be copied:

- An alternate index file cannot be specified as the FromFile name.
- An alternate index file cannot be copied as an indirect result of copying the base file.

DDMCopyFile does not return the count of the number of records that are copied.

When the FromFile contains damaged records, DDMCopyFile ends without creating a new ToFile copy.

When the FromFile contains inactive records, the inactive records are copied to the ToFile.

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

DDMCopyFile does the following:

If the FromFile exists:

1. Attempts to obtain a GETNONLK on the FromFile.

If the GETNONLK lock is obtained, the function is processed (successfully or unsuccessfully). If the GETNONLK lock is not obtained, the function is rejected with a FILIUSRM reply message.

2. The function releases the GETNONLK lock it obtained on the file.

If DDMCopyFile ends with a reply message that has a severity code value of ERROR or higher:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

DDMCopyFile

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The ToFile exists and CPYERR is specified. The FromFile name is the same as the ToFile name.	DUPFILRM
The EAs described for ToFileNew are required, but cannot be found in the FromFile EA buffer when creating the ToFile.	FILDMGRM
The FromFile is open.	FILIUSRM
The ToFile name is invalid.	FILNAMRM
CopyFlags contains a value other than zero.	INVFLGRM
The FromFile is an alternate index file. The file class is invalid or is not found.	INVRQSRM
SubsetDefBuf contains a value other than null. ToFileOld contains a value other than CPYERR (X'1483'). ToFileNew contains a value other than CPYDTA (X'1466').	PRMNSPRM

This Causes a Reply Message to be Generated with SRVCOD = X'04' for each out-of-sync file in the file object and the Function Continues	With This Reply Message
For the FromFile, if the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file. DDMCopyFile does not re-synchronize the file-change date and time of the FromFile.	FILDMGRM

DDMCreateAltIndex

DDMCreateAltIndex (Create Alternate Index File)

This function creates an alternate index file on the target system.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMCreateAltIndex (PSZ      FileName,
                          PSZ      BaseFileName,
                          ULONG     CreateFlags,
                          PKEYDEFBUF KeyDefBuf,
                          CODEPOINT DupFileOpt,
                          PEAOP2    EABuf
                          );
```

Parameters

FileName

The pointer (PSZ) to the name of the file to be created. This file must be in the same directory as the base file. If a path is not specified, the current path of the base file will be used.

BaseFileName

The pointer (PSZ) to the name of the record-oriented file on which the created file is to be based.

CreateFlags

The CreateFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
10–31	Reserved flags
9	DDM_FILPRT (Protected File)
2–8	Reserved flags
1	DDM_TMPFIL (Temporary File)
0	DDM_ALDUPKEY (Allow Duplicate Keys)

For detailed information on the create flags, see Chapter 5, “VSAM API Flags” on page 399.

KeyDefBuf

The pointer to the key definition buffer (PKEYDEFBUF). The format of the key definition buffer when the function is called:

LL	X'1114'	X'0..10'	X'140F'	KeySeq	X'0044'
KeyLen	KeyDisp				

DDMCreateAltIndex

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the last Key Displacement field.
X'1114'	The value (CODEPOINT) indicating the following field is a key definition.
X'0000010'	The length (ULONG) of the key definition. This length includes the length field and the Key Displacement field.
X'140F'	The value (CODEPOINT) indicating the following data is a key field definition.
KeySeq	Either X'1420' for Ascending Key Sequence field or X'1421' for Descending Key Sequence field.
X'0044'	The value (CODEPOINT) indicating the key field is a byte string.
KeyLen	The length (USHORT) of the key field.
KeyDisp	The offset (ULONG) from the beginning of the key field in the record. If multiple Key Field Definitions are provided, the fields are concatenated to form a combined key. The maximum length of the key is 255 bytes.

Multiple key field definitions are allowed in the Key Definition Buffer. The following example shows two key definitions:

X'0..26'	X'1114'	X'0..10'	X'140F'	X'1420'	X'0044'
...	X'0013'	X'0000010'	...		
X'0..12'	X'140F'	X'1420'	X'0044'	X'0003'	X'0..04'

The following structures define the key definition buffer:

DDMCreateAltIndex

```
/* Define the following key definition buffer structure,          */
/* modeling it after the _DDMOBJECT structure defined in DUBDEFS.H */

typedef struct _MYKEYDEFBUF
{
    ULONG      cbKeyDefBuf;
    CODEPOINT  cpKeyDefBuf;
    KEYFLDDEF  KeyFldDef[1];
} MYKEYDEFBUF, *PMYKEYDEFBUF;

/* Use the following structure to map each key field definition. */
/* It is defined in DUBDEFS.H.                                  */

typedef struct _KEYFLDDEF
{
    ULONG      cbKeyFldDef;
    CODEPOINT  cpKeyFldDef;
    CODEPOINT  cpSequence;
    CODEPOINT  cpKeyClass;
    USHORT     cbKeyField;
    ULONG      oKeyField;
} KEYFLDDEF, *PKEYFLDDEF;
```

where:

cbKeyDefBuf	The length (ULONG) of the key definition buffer from the beginning of cbKeyDefBuf to the end of oKeyField in the last key field definition.
cpKeyDefBuf	The code point value (KEYDEF) indicating that this is a key definition buffer object.
KeyFldDef	One or more contiguous key field definition structures (KEYFLDDEF). Specify an index value that indicates the number of key fields to be defined.
cbKeyFldDef	The length (ULONG) of the key field definition structure from the beginning of cbKeyFldDef to the end of oKeyField.
cpKeyFldDef	The code point value (KEYFLDDF) indicating that this is a key field definition object.
cpSequence	The code point value that indicates the key order: SEQASC Ascending key sequence field SEQDSC Descending key sequence field
cpKeyClass	The code point value (BYTSTRDR) indicating that the key field is a byte string.
cbKeyField	The length (USHORT) of the key field.

DDMCreateAltIndex

oKeyField The offset (ULONG) from the beginning of the key field in the record. If multiple key field definitions are provided, the fields are concatenated to form a combined key. The maximum length of the key is 255 bytes.

DupFilOpt

Indicates the value (CODEPOINT) for the action to be taken if a file with the same name already exists. The valid values are:

DUPFILDO Return Duplicate File Name (X'1459').

The function is rejected with DUFILRM, and this option returns an error condition (SVRCOD=X'0008').

EXSCNDDO Return Existing Condition (X'145A'). The function is rejected with EXSCNDRM, and this option returns a warning condition (SVRCOD=X'0004').

EABuf

The pointer (PEAOP2) to the address of the file's EA data to be set by DDMCreateAltIndex. This is NULL if no additional DDM file attributes are to be set at create time. Refer to "Extended Attributes" on page 5 for more information on the format of this buffer.

Only the following DDM file attributes can be specified in the EA Buffer that is pointed to by this parameter:

TITLE
MGMCLSNM
DTACLNSM
STGCLSNM

For the definition of these EAs, see Chapter 4, "VSAM API Common Parameters" on page 361.

If any other file attributes are specified in this buffer, the function is rejected and a PRMNSPRM reply message is given.

The MGMCLSNM or STGCLSNM file attributes for an alternate index can be specified as different from the base file. However, the target system may not support the difference.

Returns

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Access Method
ADDRRM	X'F212'	Address Error
BASNAMRM	X'1234'	Invalid Base File Name
DRCATHRM	X'1237'	Not Authorized to Directory
DRCFULRM	X'1258'	Directory Full
DUPFILRM	X'1207'	Duplicate File Name
EXSCNDRM	X'123A'	Existing Condition

DDMCreateAltIndex

Message ID	Code Point	Message Title
FILDMGRM	X'125A'	File Damaged
FILNFNRM	X'120E'	File Not Found
FILSNARM	X'120F'	File Space Not Available
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYDEFRM	X'123D'	Invalid Key Definition
KEYVALRM	X'1240'	Invalid Key Value
LENGTHRM	X'F211'	Field Length Error
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRMNSPRM	X'1251'	Parameter Not Supported
RECIUSRM	X'124A'	Record in Use
RSCLMTRM	X'1233'	Target Resource Limits Reached
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

The alternate index file provides an alternate key field access to the records in an existing keyed file.

This function does not require the base file to have any access capabilities. If, however, the base file is created without any access capabilities, `DDMSetPathInfo` must be used to set the access capabilities that are required for further processing.

This function requires exclusive access to the keyed file, which must be closed.

Certain attributes are derived from the base file EAs when creating an alternate index. Derived EAs have the same values as the base file. The following EAs are derived from the base file for the alternate index file:

- DELCP
- EOFNBR
- FILINISZ
- GETCP
- INSCP
- MAXARNB
- MODCP
- RECLEN
- RECLENCL

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

No locks are obtained on the alternate index file as the result of this function.

`DDMCreateAltIndex` does the following:

1. Attempts to obtain a `MODGETLK` lock on the base file.

DDMCreateAltIndex

If the MODGETLK lock is obtained, DDMCreateAltIndex is processed (successfully or unsuccessfully). If the MODGETLK lock is not obtained, DDMCreateAltIndex is rejected with the FILIUSRM reply message.

- Releases the MODGETLK lock it obtained on the base file.

If DDMCreateAltIndex ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The base file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the base file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file specified by BaseFileName is the name of a direct or sequential file.	BASNAMRM
The new file cannot be entered into the directory because the directory is full.	DRCFULRM
The FileName is equal to the BaseFileName, regardless of whether the file exists or the specification of the DupFilOpt parameter. Note: If a file exists with the same name, the DupFilOpt parameter specifies the action to take for this condition.	DUPFILRM
The alternate index files that exist for the specified base file have a last-change date/time for that base file that is different than the current system last-change date/time (System Object Attribute).	FILDMGRM
The base file is open (regardless of the sharing mode specified).	FILIUSRM
The file specified by BaseFileName is an alternate index file. The FileName specified has a path qualifier that is different than the path qualifier specified for BaseFileName.	INVRQSRM
The KeyDefBuf parameter specifies a key length of zero or a value greater than 255. The KeyDefBuf parameter specifies a key that does not fall within the boundaries of the record.	KEYDEFRM
Invalid file attributes specified in the EA buffer.	PRMNSPRM

DDMCreateAltIndex

This Causes a Reply Message to be Generated with SRVCOD = X'04' and the Function Continues	With This Reply Message
For the base file only, if the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file. DDMCreateAltIndex re-synchronizes the file-change date and time of all files in the keyed file object, unless a higher severity condition prevents it from doing so.	FILDMGRM

DDMCreateRecFile (Create Record File)

This function creates a record-oriented file on the target system.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMCreateRecFile (PSZ      FileName,
                        ULONG      CreateFlags,
                        ULONG      RecLen,
                        CODEPOINT  RecLenCls,
                        PKEYDEFBUF  KeyDefBuf,
                        ULONG      InitFileSiz,
                        LONG       MaxFileSiz,
                        CODEPOINT  DupFileOpt,
                        CODEPOINT  DftRecOp,
                        ULONG      RecCnt,
                        PEAOP2     EABuf,
                        CODEPOINT  FileClass,
                        PDDMDFTRC  DftRecBuf
                        );
```

Parameters

FileName

A pointer (PSZ) to the name of the record-oriented file to be created.

CreateFlags

The CreateFlags (ULONG) specify the action to be taken depending on whether the flag bit is set. The bit flags are:

Bit	Meaning
10–31	Reserved flags
9	DDM_FILPRT (Protected File)
8	DDM_FILSYS (System File)
7	DDM_FILHDD (Hidden File)
6	DDM_MODCP (Allow Modify Record Capability)
5	DDM_INSCP (Allow Insert Record Capability)
4	DDM_GETCP (Allow Get Record Capability)
3	DDM_INIEX (Inhibit Initial Extent)
2	DDM_DELCP (Allow Record Deletion Capability)
1	DDM_TMPFIL (Temporary File)
0	DDM_ALDUPKEY (Allow Duplicate Keys)

For detailed information on the create flags, see “CreateFlags (Create Flags)” on page 405.

DDMCreateRecFile

RecLen

Specifies the maximum length (ULONG) of the user data in the DDM record object. For information on the maximum and minimum record lengths, see “RECLEN (Record Length)” on page 389.

RecLenCls

Indicates the value (CODEPOINT) for the type of record length that the record on the file can have. Valid values are:

RECFIX	Fixed Length Record (X'142E')
RECIVL	Initially-Variable-Length Record (X'142F')
RECVAR	Variable-Length Record (X'1431')

KeyDefBuf

The pointer to the key definition buffer (PKEYDEFBUF) or NULL. This parameter is ignored if the create is not being done for a keyed file. When the function is called, the format of the key definition buffer is:

LL	X'1114'	X'0..10'	X'140F'	KeySeq	X'0044'
----	---------	----------	---------	--------	---------

KeyLen	KeyDisp	...
--------	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the last Key Displacement field.
X'1114'	The value (CODEPOINT) indicating the following field is a key definition.
X'0..10'	The length (ULONG) of the key definition. This length includes length field through the Key Displacement field.
X'140F'	The value (CODEPOINT) indicating the following data is a key field definition.
KeySeq	The value (CODEPOINT) to indicate ascending or descending key sequence field: X'1420' Ascending Key Sequence field X'1421' Descending Key Sequence field Key Sequence always assumes the sorting order of the target system.
X'0044'	The value (CODEPOINT) indicating the key field is a byte string.
KeyLen	The length (USHORT) of the key field.

DDMCreateRecFile

KeyDisp The offset (ULONG) from the beginning of the key field in the record. If multiple Key Field Definitions are provided, the fields are concatenated to form a combined key. The maximum length of the key is 255 bytes.

Multiple key field definitions are allowed in the Key Definition Buffer. The following example shows two key definitions:

X'0..26'	X'1114'	X'0..10'	X'140F'	X'1420'	X'0044'
X'0..13'	X'0..12'	X'0..10'	X'140F'	X'1420'	X'0044'
X'0..08'	X'0..04'				

The following structures define the key definition buffer:

```
/* Define the following key definition buffer structure,          */
/* modeling it after the _DDMOBJECT structure defined in DUBDEFS.H */
```

```
typedef struct _MYKEYDEFBUF
{
    ULONG      cbKeyDefBuf;
    CODEPOINT  cpKeyDefBuf;
    KEYFLDDEF  KeyFldDef[1];
} MYKEYDEFBUF, *PMYKEYDEFBUF;
```

```
/* Use the following structure to map each key field definition. */
/* It is defined in DUBDEFS.H.                                   */
```

```
typedef struct _KEYFLDDEF
{
    ULONG      cbKeyFldDef;
    CODEPOINT  cpKeyFldDef;
    CODEPOINT  cpSequence;
    CODEPOINT  cpKeyClass;
    USHORT     cbKeyField;
    ULONG      oKeyField;
} KEYFLDDEF, *PKEYFLDDEF;
```

where:

cbKeyDefBuf The length (ULONG) of the key definition buffer from the beginning of cbKeyDefBuf to the end of oKeyField in the last key field definition.

cpKeyDefBuf The code point value (KEYDEF) indicating that this is a key definition buffer object.

DDMCreateRecFile

KeyFldDef	One or more contiguous key field definition structures (KEYFLDDEF). Specify an index value that indicates the number of key fields that are defined.
cbKeyFldDef	The length (ULONG) of the key field definition structure from the beginning of cbKeyFldDef to the end of oKeyField.
cpKeyFldDef	The code point value (KEYFLDDF) indicating that this is a key field definition object.
cpSequence	The code point value that indicates the key order: SEQASC Ascending key sequence field SEQDSC Descending key sequence field
cpKeyClass	The code point value (BYTSTRDR) indicating that the key field is a byte string.
cbKeyField	The length (USHORT) of the key field.
oKeyField	The offset (ULONG) from the beginning of the key field in the record. If multiple key field definitions are provided, the fields are concatenated to form a combined key. The maximum length of the key is 255 bytes.

InitFileSiz

The first time space is allocated for records, specifies the initial number (ULONG) of records to allocate. A value of 0 indicates that the file exists but has no allocated space. A nonzero value for this parameter causes space to be allocated, but the contents of the space is undefined. Use RecCnt to initialize the space. InitFileSiz can be used to reduce the fragmentation of a file if you know the approximate size it will grow to. Reducing fragmentation can improve product performance. A remote target system might ignore this information.

MaxFileSiz

Specifies the maximum number (LONG) of records that can be allocated to the file. A value of -1 indicates that the file size is unlimited.

DupFilOpt

The value (CODEPOINT) indicating the action to take if a file with the same name already exists. The valid values are:

- DUPFILDO** Return Duplicate File Name (X'1459').
The function is rejected with DUPFILRM, and the option returns an error condition (SVRCOD=X'0008').
- EXSCNDDO** Return Existing Condition (X'145A').
The function is rejected with EXSCNDRM, and the option returns a warning condition (SVRCOD=X'0004').

DDMCreateRecFile

DftRecOp

The value (CODEPOINT) indicating the action a create file function should take to initialize the data contents of the file.

If the value is specified as NIL, the file is not initialized with default records.

If a value other than NIL is specified, the file is initialized with at least the number of records that are specified. The number of records that are specified is through the RecCnt variable that is related to this parameter. This function ignores the value of the DDM_INIEX parameter flag.

The valid values are:

DFTINAIN Default inactive record initialization (X'1460'). Specifies that the file is to be initialized with inactive records.

If the file is created with initially-varying-length records or with variable-length records, the initialized records have a length equal to RecLen.

If the file is not delete-capable and is not a direct file, DFTRECRM is returned.

DFTTRGIN Default target initialization (X'145F').

Specifies that the file is to be initialized with active records whose contents are determined by the target server. All records have the same initial contents that is defined by the target server. The local VSAM file system initializes records with the '!' character.

If the file is created with initially-varying-length records or with variable-length records, the initialized records have a length equal to RecLen.

DFTSRCIN Default source initialization (X'1449').

Specifies that the file is to be initialized with active records whose contents are defined by the DftRecBuf parameter. The contents of DftRecBuf are replicated or truncated to match the record length of the file. This means that DftRecBuf(X'00') causes the file to be initialized with records that consist of all zeros. A DftRecBuf('ABC') would initialize a file with 10-byte records with 'ABCABCABCA' as the initialization record.

If the file is created with initially-varying-length records or with variable-length records, the initialized records have a length equal to RecLen.

NIL Do not initialize the data content of the file (X'002A').

RecCnt

Specifies the number (ULONG) of records to initialize. This parameter works in conjunction with the DftRecOp parameter. If the DftRecOp parameter is specified as NIL, RecCnt is ignored. Records are initialized in the space that is allocated by InitFileSiz with additional space that is allocated as needed.

DDMCreateRecFile

EABuf

The pointer (PEAOP2) to the file's EA data to be set by DDMCreateRecFile, or NULL if no additional DDM file attributes are to be set at create time. Refer to “Extended Attributes” on page 5 for more information on the format of this buffer.

The following DDM file attributes can be specified in EABuf:

TITLE
MGMCLSNM
DTACLNSM
STGCLSNM

For the definition of these EAs, see Chapter 4, “VSAM API Common Parameters” on page 361.

FileClass

Indicates the value (CODEPOINT) for the class or type of record file to create. Valid values are:

DIRFIL Direct File (X'140C')
KEYFIL Keyed File (X'141E')
SEQFIL Sequential File (X'143B')

DftRecBuf

The pointer (PDDMDFTREC) to the default record initialization buffer or NULL. When this function is called, the format of the buffer is:

LL	X'142B'	Initialization Record
----	---------	-----------------------

Field	Description
LL	The length (ULONG) of the default initialization record from the beginning of LL to the end of the Initialization Record.
X'142B'	The value (CODEPOINT) indicating that the following content is the default initialization record.
Data	The default initialization record information.

See “DFTREC (Default Record)” on page 369 for more information.

Returns

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Access Method
ADDRRM	X'F212'	Address Error
DFTRECRM	X'1204'	Default Record Error
DRCATHRM	X'1237'	Not Authorized to Directory
DUPFILRM	X'1207'	Duplicate File Name
EXSCNDRM	X'123A'	Existing Condition
FILATHRM	X'123B'	Not Authorized to File
FILUSRMR	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found

DDMCreateRecFile

Message ID	Code Point	Message Title
FILSNARM	X'120F'	File Space Not Available
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRMNSPRM	X'1251'	Parameter Not Supported
RSCLMTRM	X'1233'	Target Resource Limits Reached
SYNTAXRM	X'124C'	Data Stream Syntax Error
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

DDMCreateRecFile does not require the file to have any access capabilities. If however, the base file is created without any access capabilities, DDMSetPathInfo must be used to set the access capabilities that are required for further processing.

Effect on Cursor Position

There is no effect on the cursor position because the file is not open.

Locking (for Local VSAM File System Only)

No locks are obtained and held on the file by this function.

Exceptions

If a file exists on the target system with the same name, the DupFilOpt parameter specifies the action to take for this condition.

This Causes the Function to Be Rejected	With This Reply Message
The new file cannot be entered into the directory because the directory is full.	DRCFULRM
The file is not delete-capable and is not a direct file.	DFTRECRM
The KeyDefBuf parameter specifies a key length of zero or a value greater than the maximum allowed by the target system. The KeyDefBuf parameter defines a key that cannot be mapped to the key-field capabilities of the target server.	KEYDEFRM
The DDM_ALDUPKEY is false and DftRecOp is not NIL.	SYNTAXRM
DftRecOp (with a value other than NIL) is specified and RecCnt exceeds the maximum number of record positions that the target system allocates to the file.	VALNSPRM

DDMDelete

DDMDelete (Delete File)

This function deletes a file from the target system, releases all locks that are held on the file, and releases the space the file occupied.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMDelete (PSZ      FileName,
                  ULONG    Flags
                  );
```

Parameters

FileName

The pointer (PSZ) to the name of the record-oriented file to be deleted.

Flags

The Flags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
-----	---------

1–31	Reserved flags
------	----------------

0	DDM_OVRDTA (Overwrite Data)
---	-----------------------------

Specifies that the data being deleted is to be overwritten with binary zeros. This prevents the data from being read by subsequent users of the allocated file space.

If the file is a keyed file or alternate index file, this flag specifies whether the indexes for the file are also overwritten.

Returns

Message ID	Code Point	Message Title
EXSCNDRM	X'123A'	Existing Condition
ADDRRM	X'F212'	Address Error
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILUSRMR	X'120D'	File In Use
FILNAMRM	X'1212'	Invalid File Name
FILNFRMR	X'120E'	File Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request

DDMDelete

Remarks

For an alternate index file, only the index is deleted. The base file of an alternate index file is not deleted.

The primary index file for the specified keyed file is also deleted.

Any alternate index files, using the specified file as a base file, must be deleted before the specified file can be deleted.

Effect on Cursor Position

There is no effect on the cursor position because the file is not open.

Locking (for Local VSAM File System Only)

DDMDelete does the following:

1. Attempts to obtain a MODNONLK lock on the file.
If the MODNONLK lock is obtained, the function is processed (successfully or unsuccessfully). If the MODNONLK lock is not obtained, the function is rejected with FILIUSRM.
2. Releases the MODNONLK lock it obtained on the file.

If DDMDelete ends with a reply message that has a severity code value of: ERROR or higher, then:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to Terminate Normally	With This Reply Message
The file specified by FileName cannot be found.	FILNFNRM

This Causes the Function to be Rejected	With This Reply Message
The file specified by FileName is the base file for one or more alternate index files.	INVRQSRM
The file specified by FileName is a protected file.	
The requester has the file open.	FILIUSRM

DDMDeleteRec

DDMDeleteRec (Delete Record)

This function deletes a record that has an update intent on it.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMDeleteRec (HDDMFILE   FileHandle,
                    ULONG       Flags
                    );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

Flags

The Flags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
-----	---------

1-31	Reserved flags
------	----------------

0	DDM_OVRDTA (Overwrite Data)
---	-----------------------------

Specifies whether the record being deleted is to be overwritten with binary zeros to prevent the data from being read by non-DDM applications.

Note: This flag is obsolete, but supported for compatibility with earlier releases. The deleted record space is always overwritten with binary zeros.

Returns

Message ID	Code Point	Message Title
EXSCNDRM	X'123A'	Existing Condition
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
RECIUSRM	X'124A'	Record in Use
RECDMGRM	X'1249'	Record Damaged
UPDINTRM	X'124E'	No Update Intent on Record

Remarks

DDMDeleteRec has the following effects:

- The data content of the record is no longer available.
- The record position becomes inactive and its length is preserved if it is initially variable or fixed.

DDMDeleteRec

- If the file contains variable-length records, the length of the record position goes to the maximum record length for the file. See “RECINA (Inactive Record)” on page 389 for a detailed description.
- If the file is a keyed file or an alternate index file, the associated indexes are updated to show that the record has been deleted.
- If the record's position is overwritten, it is overwritten with binary zeros.
- Update intent is removed.

Before this function can be used, an update intent must be placed on a record in the file. A DDMSetxxx or DDMGetRec function can be used to place an update intent on a record.

If the record that is deleted was the last active record in a direct file, EOF is backed up to the previous active record.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message determines the state of the cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

DDMDeleteRec does the following:

If the file was opened for multiple updates, the access method attempts to acquire an EXCRECLK lock on the record that has an update intent placed on it. If the EXCRECLK lock cannot be obtained because of a lock conflict, the DDMDeleteRec is rejected with the RECIUSRM reply message.

If the EXCRECLK lock is obtained:

1. DDMDeleteRec is processed.
2. Because all record modifications are committed at the time of modification, the EXCRECLK lock is released from the record.
3. Even if DDMDeleteRec is rejected with an error reply, the obtained EXCRECLK lock is released from the record.

If DDMDeleteRec ends with a reply message that has a severity code value of ERROR or higher, then:

DDMDeleteRec

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record locks.

Exceptions

This Causes the Function to Terminate Normally	With This Reply Message
The file handle is not valid.	HDLNFNRM
Any reserved bits in Flags are active.	INVFLGRM
The DELAI access intent was not specified when the file was opened. The file is not delete-capable. The file is direct and the file is empty.	INVRQSRM
A damaged record (not an active or inactive record) is encountered.	RECDMGRM
There is not a record with update intent placed on it in the file.	UPDINTRM

This Causes the Function to be Rejected	With This Reply Message
The record is already inactive.	EXSCNDRM
An EXCRECLK lock cannot be obtained.	RECIUSRM

Example

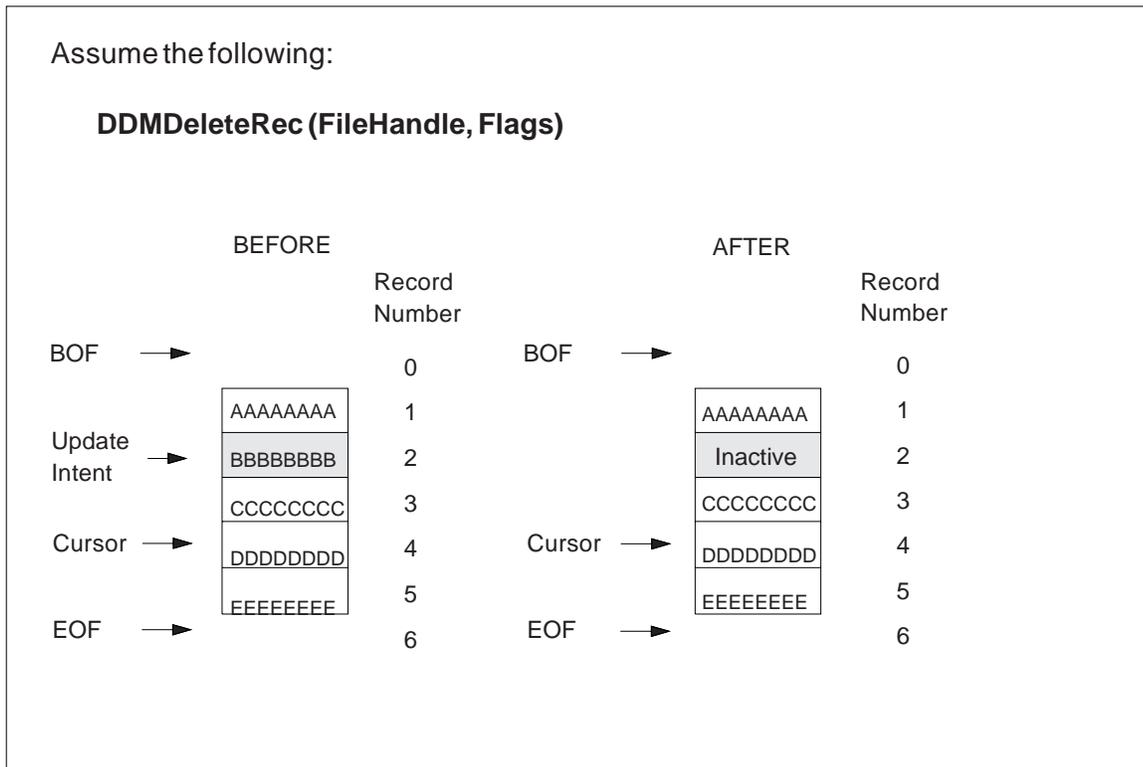


Figure 10. DDMDeleteRec Function

DDMForceBuffer

DDMForceBuffer (Commit a File's Cached Information)

This function commits a file's cached information to non-volatile storage. The file's directory entry and EAs are updated (as if the file had been closed with a DDMClose), but the file remains in the open state.

Syntax

```
#include dub.h /* Required for all platforms */  
  
APIRET DDMForceBuffer (HDDMFILE      FileHandle  
                      );
```

Parameters

FileHandle

The handle (HDDMFILE) of the open file whose cached information is to be committed to non-volatile storage. A value of X'FFFFFFFF' for this parameter causes all open files for this process to have their caches written to non-volatile storage.

Returns

Message ID	Code Point	Message Title
HDLNFNRM	X'1257'	File Handle Not Found

Remarks

This function is analogous to the DosResetBuffer command.

To obtain the current EA values while a file is being used, you must issue a DDMForceBuffer and request the EA values to be returned. The EA values are returned through DDMQueryFileInfo or DDMOpen.

When a file is being used in the local VSAM file system, the file system maintains certain extended attributes in memory for each I/O operation that affects the file. When an alternate index file is being used, the local VSAM file system maintains only the base file EAs in memory. These changed EAs are permanently updated for the file as well as for all associated index files when a DDMForceBuffer or DDMClose function is issued.

Effect on Cursor Position

If the file is opened without GETAI access intent, there is no effect on the cursor position.

If the file was opened with GETAI access intent, the cursor is positioned to EOF.

DDMForceBuffer

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

The locks on the requester's files are the same before and after the DDMForceBuffer function. All record locks that are held by the requester are released.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is not valid.	HDLNFNRM

DDMGetRec

DDMGetRec (Get Record)

This function gets and returns the record that is indicated by the current cursor position. This function also optionally returns the record number and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMGetRec (HDDMFILE      FileHandle,
                  ULONG          AccessFlags,
                  PDDMRECORD    RecordBuf,
                  ULONG          RecordBufLen
                  );
```

Parameters

FileHandle

The file handle (HDDMFILE) that is obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
4–31	Reserved flags
3	DDM_RTININA (Return Inactive Records)
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

Note: DDM_KEYVALFB is ignored for nonkeyed files.

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer to the record buffer (PDDMRECORD) for the returned record. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples of RecordBuf Data Formats” on page 75.

RecordBufLen

The length (ULONG) of the Record Buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
CSRNSARM	X'1205'	Cursor Not Selecting a Record Position
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYMODRM	X'1260'	Key value was modified since cursor was last set
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use

Remarks

As an option, DDMGetRec can:

- Specify whether inactive records should be returned (DDM_RTNINA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message determines the cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file is opened for multiple updates, the access method acquires an implicit SHRRECLK on the record. This occurs if the requester does not lock the record with a SHRRECLK. If a different record is already locked, the lock on that record is released before the SHRRECLK on the current record is obtained.

The SHRRECLK is released when one of the following occurs:

- The record is updated (DDMModifyRec or DDMDelRec).
- The cursor is moved to a different record.
- The DDMUnLockRec function is issued.
- Any function is issued that references a record other than the one currently pointed to by the cursor. Examples of these functions are DDMInsertRecEOF,

DDMGetRec

DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum.

- The file is closed.

If none of these conditions are met, the record remains locked.

If the record lock is not obtained, the function is rejected with RECIUSRM.

If DDM_UPDINT(TRUE) is specified and the file is *not* opened for multiple updates, an update intent is placed on the record. However, the access method does *not* acquire any record locks.

If the function ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message. determines the state of the record locks.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
DDM_RECNRFB or DDM_KEYVALFB is set or DDM_NODATA is not set and RecordBuf does not contain an address.	ADDRRM
The cursor is positioned to outside the bounds of the file. The cursor position is unknown.	CSRNSARM
The file handle is not valid.	HDLNFNRM
Any of the reserved bits in AccessFlags are set.	INVFLGRM
The file was opened without GETAI specified. DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The file is a keyed or alternate index file, the cursor was last positioned by key value, and the key value has changed or the record has become inactive since the cursor was positioned to its current location.	KEYMODRM
The RecordBuf is not large enough to hold the returned record.	LENGTHRM
The record returned is damaged (not an active or inactive record).	RECDMGRM
The DDM_RTNINA parameter specifies that inactive records are <i>not</i> to be returned and the current record is inactive.	RECINARM
A record lock cannot be obtained.	RECIUSRM

Examples of RecordBuf Data Formats

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_RTNINA(FALSE)

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_RTNINA(TRUE)

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the buffer from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is an ULONG length of an inactive record.
Data	Either the record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_RTNINA(FALSE)

RecordBuf

DATA FORMAT

DDMGetRec

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_RTININA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of CP.

DDMGetRec

CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is an ULONG length of an inactive record.
Data	Either the record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_RTNINA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_RTNINA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

DDMGetRec

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) of the field from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is an ULONG length of an inactive record.
Data	Either the record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_RTNINA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) of the field from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).

DDMGetRec

RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) of the field from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) of the field from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_RTNINA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) of the field from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number.
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) of the field from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMGetRec

L3	The length (ULONG) of the field from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is an ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

DDMGetReplyMessage

DDMGetReplyMessage (Get Reply Message)

This function gets and returns a reply message that is issued from the previously requested function in the current thread of execution.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMGetReplyMessage (PBYTE          RpyMsgBuf,
                          ULONG          RpyMsgBufLen,
                          ULONG          RpyMsgFlags
                          );
```

Parameters

RpyMsgBuf

The pointer (PBYTE) to the reply message buffer for the returned reply message. For information on how to interpret the reply message data, see Chapter 6, “VSAM API Reply Messages” on page 411.

RpyMsgBufLen

The length (ULONG) of the reply message buffer. The length of the reply message buffer should be the same as the largest reply message plus four bytes for the length and four bytes for the code point fields.

RpyMsgFlags

The RpyMsgFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
-----	---------

1–31	Reserved flags
------	----------------

0	Full Error Reply Message
---	--------------------------

If this flag is set, at least one full error reply message is returned. If this flag is not set, only the USHORT code point that identifies the error reply message is returned. A subsequent repeat invocation of DDMGetReplyMessage with this flag that is not set causes the code point of the next reply message to be returned. The previous error reply message is lost.

DDMGetReplyMessage

Returns

On return, APIRET contains one of the SVRCOD error codes. For a detailed description of the severity code values, see "SVRCOD (Severity Code)" on page 393.

APIRET	Description
X'00000000'	All reply messages for last requested function have been received.
X'00000004'	There are more reply messages to be received. Call the DDMGetReplyMessage function again to get the next message. The reply messages are put in a process thread-based FIFO (first-in first-out) queue. Each call of DDMGetReplyMessage gets the next reply message from the queue. If the currently executing thread issues a function other than DDMGetReplyMessage, before all of the reply messages have been received, the remaining reply messages are discarded. The process thread-based queue is filled with the reply messages from the requested function.
X'00000008'	Reply buffer is too small. The reply message buffer is not large enough to hold the reply message. If the buffer length is at least 1 ULONG, the first ULONG of the reply message buffer contains the length of the reply message.
X'00000010'	Warning error. A reply message was requested but there are no reply messages available.
X'00000020'	Error. An invalid reply message buffer address was specified.
X'00000040'	Severe error. An un-architected reply message object was encountered. One or more additional reply messages are available. The cause may be a target problem.
X'00000080'	Severe error. A reserved bit was set on in the RpyMsgFlags parameter.

DDMInsertRecEOF

DDMInsertRecEOF (Insert Records at EOF)

This function inserts records at the end of the file and optionally returns the record number and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMInsertRecEOF (HDDMFILE      FileHandle,
                       ULONG          AccessFlags,
                       PDDMRECORD    RecordBuf,
                       ULONG          RecCount,
                       PDDMOBJECT    FdbkBuf,
                       ULONG          FdbkBufLen
                       );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
12–31	Reserved flags
11	DDM_HLDUPD (Hold Update Intent)
10	DDM_UPDCSR (Update Cursor)
3–9	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	Reserved flag

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the records to be inserted.

When DDMInsertRecEOF is called, the format of RecordBuf is:

LL	CP	Data	...
----	----	------	-----

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following data is either Record Data or an Inactive Record Length.

DDMInsertRecEOF

X'144A' Indicates that the following data is Record Data.

X'142D' Indicates that the following data is an ULONG Inactive Record Length. The number of record descriptions (Record Data or Inactive Record Lengths) should be the same as the number indicated in RecCount.

Data The record data.

Examples of the DDMInsertRecEOF function are shown in “Examples” on page 90.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

FdbkBuf

The pointer (PDDMOBJECT) to the Feedback Buffer for the requested returned feedback data, or NULL if no information has been requested. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned feedback data formats are shown in “Remarks.”

FdbkBufLen

The length (ULONG) of the feedback buffer or 0.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DTARECRM	X'1206'	Invalid Data Record
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
FILFULRM	X'120C'	File is Full
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYVALRM	X'1240'	Invalid Key Value
LENGTHRM	X'F211'	Field Length Error
OBJNSPRM	X'1253'	Object Not Supported
RECIUSRM	X'124A'	Record in Use
RECLENRM	X'1215'	Record Length Mismatch
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

For files with variable-length records, new record positions that have the same length as the inserted records are created. Records to be inserted are contained in the record buffer.

After successful completion of this command, EOF points to the record position after the last inserted record.

DDMInsertRecEOF

This function processes the records in a Record Buffer as a group. This function treats inactive records in the group as place holders between the active records, as the group is inserted into the file. How the EOF is updated depends on the type of the file.

- If the file is a direct file, the EOF is only updated when an active record in the group is inserted. Therefore, inactive records that follow the last active record in a group are located at or beyond the EOF and are subject to overlay by other functions. The method of inserting records into a direct file can affect the file contents; for example:
 - When multiple records are inserted at a time, both active and inactive records can occur before the EOF (see Figure 11).
 - When individual records are inserted one at a time, only the active records will occur before the EOF (see Figure 12 on page 86).
- If the file is not a direct file, the EOF is updated as each record in the group is inserted. If all the records in the group are inserted successfully, the EOF is positioned after the last record in the group. This is true whether the record is an active or inactive record. The method of inserting records into these files does not affect the file contents.

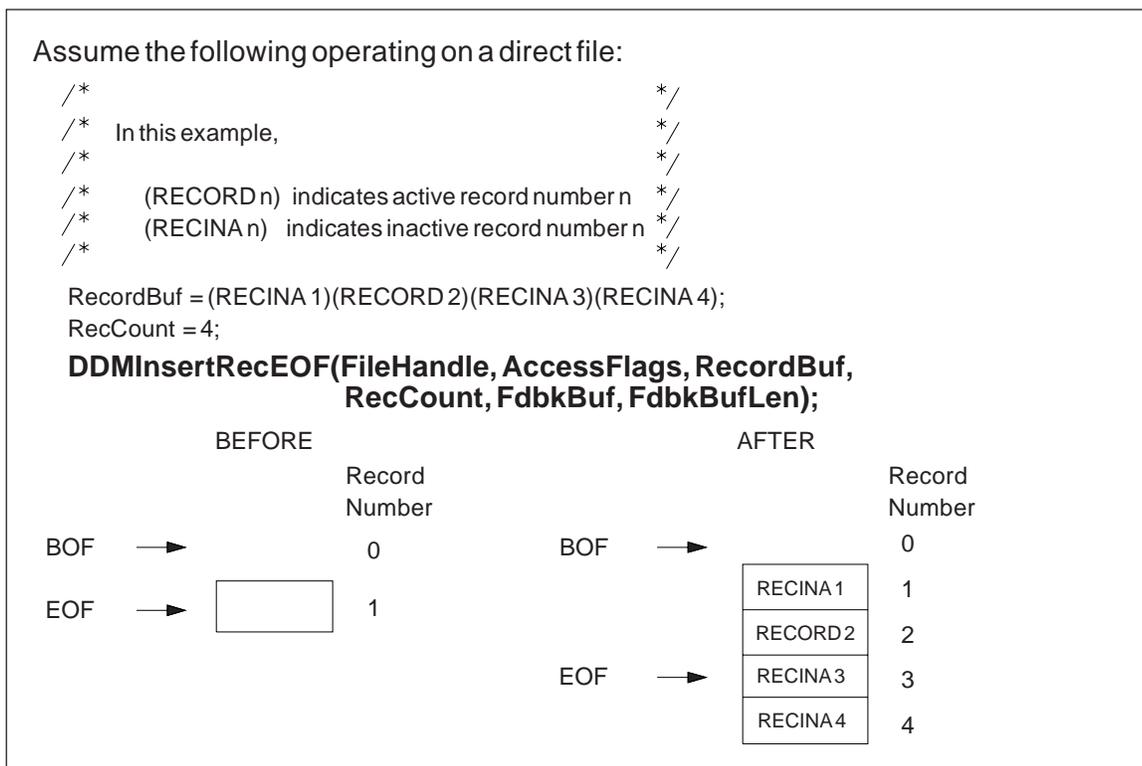


Figure 11. DDMInsertRecEOF. Insert Multiple Records at the Same Time into a Direct File

DDMInsertRecEOF

Assume the following operating on a direct file:

```

/*                                     */
/* In this example,                   */
/*                                     */
/* (RECORD n) indicates active record n */
/* (RECINA n) indicates inactive record n */
/*                                     */

```

```

RecordBuf = (RECINA 1);
RecCount = 1;

```

```

DDMInsertRecEOF(FileHandle, AccessFlags, RecordBuf,
RecCount, FdbkBuf, FdbkBufLen);

```

```

RecordBuf = (RECORD 2);
RecCount = 1;

```

```

DDMInsertRecEOF(FileHandle, AccessFlags, RecordBuf,
RecCount, FdbkBuf, FdbkBufLen);

```

```

RecordBuf = (RECINA 3);
RecCount = 1;

```

```

DDMInsertRecEOF(FileHandle, AccessFlags, RecordBuf,
RecCount, FdbkBuf, FdbkBufLen);

```

```

RecordBuf = (RECINA 4);
RecCount = 1;

```

```

DDMInsertRecEOF(FileHandle, AccessFlags, RecordBuf,
RecCount, FdbkBuf, FdbkBufLen);

```



Figure 12. DDMInsertRecEOF. Insert One Record into a Direct File

RecCount specifies the number of records to be inserted at EOF. An instance of a record or an inactive record length must be set for each record to be inserted.

Depending on the value of the DDM_UPDCSR flag, this function sets the cursor to the inserted record or keeps its current setting. If RecCount specifies a value greater than 1 and the DDM_UPDCSR flag is set, the cursor is updated after each record is successfully inserted at the end of file.

DDMInsertRecEOF

If the DDM_RECNRFB flag is set, the record number of the last inserted record is returned in FdbkBuf.

If the DDM_KEYVALFB flag is set, the key value of the last inserted record is returned in FdbkBuf.

If the DDM_HLDUPD flag is not set, the update intent on any record in the file is released. If the DDM_HLDUPD flag is set, the update intent on any record in the file remains in place.

When inserting records into a keyed or alternate index file, this function updates the file index and all associated indexes.

Inactive records can only be inserted if the file is delete-capable.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

This function positions the cursor that is based on the DDM_UPDCSR flag. If DDM_UPDCSR is set, this function moves the cursor to the inserted record. If DDM_UPDCSR is not set, the cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was called. If the RecCount is greater than 1, the cursor position is the same as it was before the last iteration of the function.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message indicates the cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

DDMInsertRecEOF does the following:

1. If DDM_HLDUPD(FALSE) is specified and the requester has a SHRRECLK lock on a record in the file, the SHRRECLK lock is released.
2. If the file is opened for multiple updates, DDM_HLDUPD(TRUE) is specified, and the requester has a SHRRECLK lock on a record in the file. The SHRRECLK lock is *not* released.
3. In all cases, the access method attempts to acquire an EXCRECLK lock on the record. If the EXCRECLK lock cannot be obtained due to a lock conflict, the function is rejected with RECIUSRM.

If the EXCRECLK lock is obtained:

- a. The record insert function is performed.

DDMInsertRecEOF

- b. The EXCRECLK lock is released from the record, because all record modifications are committed at the time of modification.
 - c. The obtained EXCRECLK lock is released from the record, even if the function is rejected with an error reply.
4. If the function ends with a reply message that has a severity code of ERROR or higher, then:
- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued. If RecCount is greater than 1, the record locks are the same as before the last iteration of the function.
 - For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the record locks.

Exceptions

This Causes the Function to be Terminated	With This Reply Message
The data in the RecordBuf is not a valid record type.	OBJNSPRM

This Causes the Function to be Rejected	With This Reply Message
The DDM_KEYVALFB or DDM_RECNRFB access flags are set and a pointer is not supplied to the FdbkBuf.	ADDRRM
The file is not delete-capable and the record to be inserted is RECINA.	DTARECRM
One of the following sets of conditions exists: <ul style="list-style-type: none"> • The file is the base file for an alternate index file. • The alternate index file does not allow duplicate keys. • The inserted record would result in a duplicate key value in the alternate index. or <ul style="list-style-type: none"> • The file is an alternate index file. • The file's base file does not allow duplicate keys, or another alternate index file with the same base file does not allow duplicate keys. • The inserted record would result in a duplicate key value in the file's base file or in another alternate index file with the same base file. 	DUPKDIRM
The following conditions exist: <ul style="list-style-type: none"> • The file is a keyed file or alternate index file. • The file does not allow duplicate keys. • The inserted record would result in a duplicate key value in the file index. 	DUPKSIRM
Inserting the record would cause the file to become full.	FILFULRM
The file handle is invalid.	HDLNFNRM
Any of the reserved bits are set in the access flags.	INVFLGRM

DDMInsertRecEOF

This Causes the Function to be Rejected	With This Reply Message
The file was opened without INSAI (Insert Record) access intent.	INVRQSRM
The file supports variable-length records, the file is a keyed file or an alternate index file, and the record to be inserted does not contain all of the fields for the specified file key.	KEYVALRM
The FdbkBuf is not large enough to hold the returned information.	LENGTHRM
An EXCRECLK lock cannot be obtained on the file.	RECIUSRM
<p>If the following are <i>not</i> true:</p> <ol style="list-style-type: none"> 1. If the record class is fixed and the record to be inserted is an active record, the length of the record must be equal to the length of the header plus the record length. (See "RECORD (Record)" on page 391 for more information.) 2. If the record to be inserted is an inactive record, the record length represented by the inactive record must be the same as the length defined for a record in the file. (See "RECINA (Inactive Record)" on page 389 for more information.) 	RECLENRM
RecCount is not greater than zero.	VALNSPRM

DDMInsertRecEOF

Examples

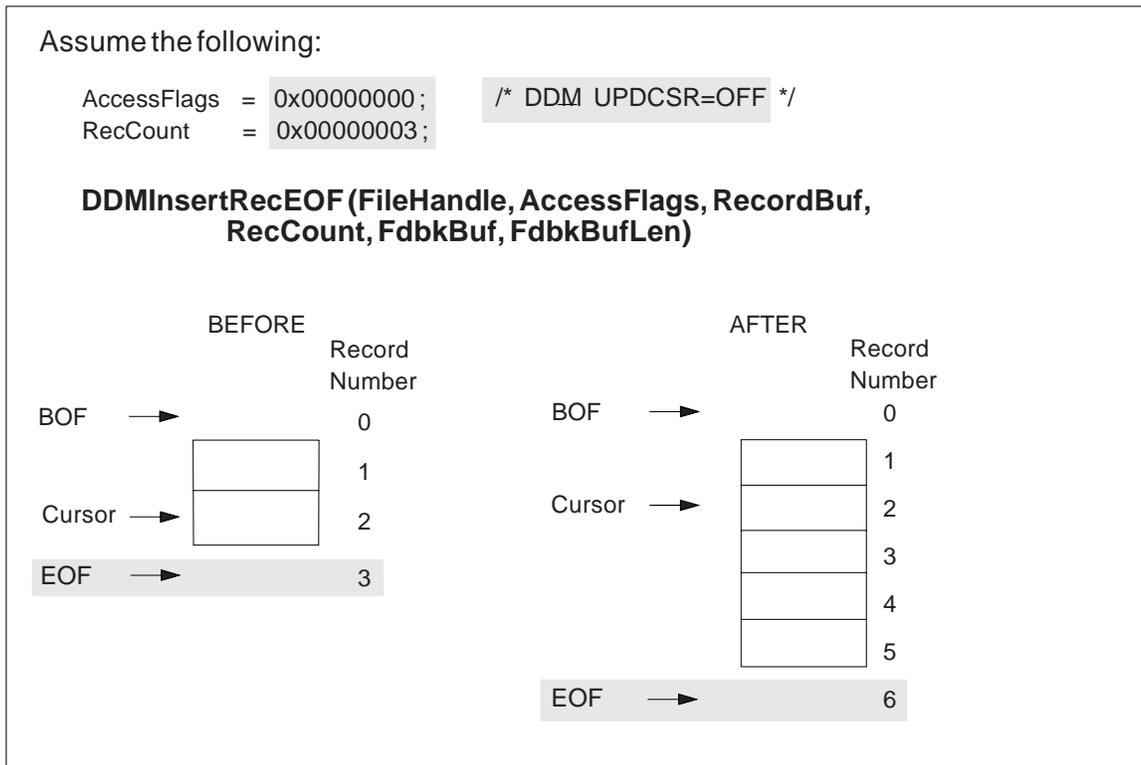


Figure 13. `DDMInsertRecEOF` Function

DDMInsertRecEOF

Assume the following:

```
AccessFlags = 0x00000400 ; /* _DDM UPDCSR=ON */
RecCount   = 0x00000001
```

**DDMInsertRecEOF (FileHandle, AccessFlags, RecordBuf,
RecCount, FdbkBuf, FdbkBufLen)**

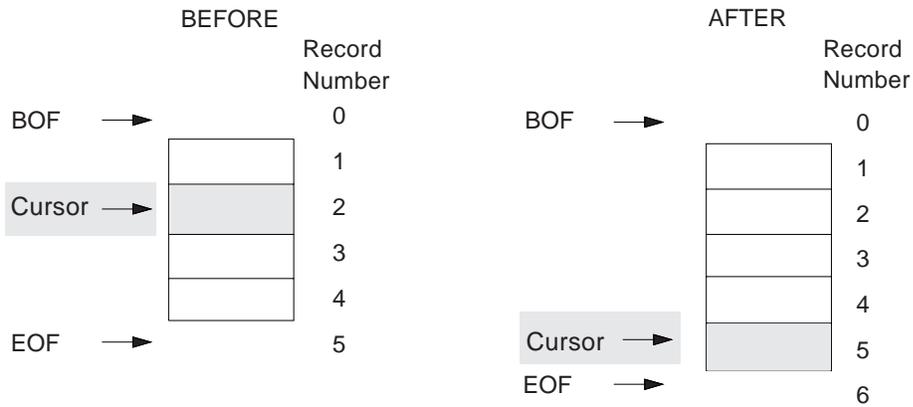


Figure 14. DDMInsertRecEOF Function with DDM_UPDCSR

Examples of FdbkBuf returned data formats are:

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE)

FdbkBuf

This parameter returns nothing.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE)

FdbkBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field Description

LL The length (ULONG) of the buffer from the beginning of LL to the end of RN.

DDMInsertRecEOF

X'111D' The value (CODEPOINT) indicating that the following data is a record number (RECNBR).

RN The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE)

FdbkBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) of the field from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE)

FdbkBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMInsertRecKey

DDMInsertRecKey (Insert Records by Key Value)

This function inserts records according to their key values and optionally returns the record number of the last record that is inserted.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMInsertRecKey (HDDMFILE      FileHandle,
                       ULONG          AccessFlags,
                       PDDMRECORD    RecordBuf,
                       PRECNUM        RecordNumber,
                       ULONG          RecCount
                       );
```

Parameters

FileHandle

The file handle (HDDMFILE) that is obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
12–31	Reserved flags
11	DDM_HLDUPD (Hold Update Intent)
10	DDM_UPDCSR (Update Cursor)
2–9	Reserved flags
1	DDM_RECNRFB (Record Number Feedback)
0	Reserved flag

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the record descriptions and the records to be inserted by key. The format of the record buffer on when DDMInsertRecKey is called:

LL	X'144A'	Data	...
----	---------	------	-----

Field	Description
-------	-------------

LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
----	---

X'144A'	The value (CODEPOINT) indicating that the following data is Record Data. The number of record descriptions (record data's) should be the same as the number indicated in RecCount.
---------	--

DDMInsertRecKey

Data Record data.

RecordNumber

The pointer (PRECNUM) to an output variable of type RECNUM for the Record Number Feedback from the last record inserted. If the Record Number Feedback flag of AccessFlags has not been set, this parameter is ignored.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DTARECRM	X'1206'	Invalid Data Record
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
FILFULRM	X'120C'	File is Full
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYVALRM	X'1240'	Invalid Key Value
RECLENRM	X'1215'	Record Length Mismatch
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

For files with a RECLENCL (record length class) of variable-length record, either:

- A new record position having the same length as the record to be inserted is created, or
- An existing record position containing an inactive record the same length as the record to be inserted is used.

The record structure must be consistent with the key definition on DDMCreateRecFile and DDMCreateAltIndex.

If the file supports variable-length records whose lengths are changeable, the length of the record position is changed to equal the length of the inserted record.

If RecCount specifies a value greater than 1, multiple records are inserted into the file. RecCount specifies the number of times the DDMInsertRecKey function will be performed. If the DDM_UPDCSR flag is set, the cursor position is updated after each iteration of the DDMInsertRecKey.

Depending on the setting of the DDM_UPDCSR flag, the cursor can be set to the inserted record or can retain its current setting.

If the DDM_RECNRFB flag specifies that the record number of the inserted record is to be returned, RecordNumber is returned with the record number of the last record inserted.

DDMInsertRecKey

At the completion of the function, any existing update intent is released unless the DDM_HLDUPD flag is set. In this case, the existing update intent remains in effect.

The file index and all other indexes that are associated with the file are updated to show the inserted records.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is based on the DDM_UPDCSR flag. If DDM_UPDCSR is set, the cursor is moved to the last inserted record. If DDM_UPDCSR is not set, the cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same. If RecCount is greater than 1, the cursor position is the same as before the last iteration of the function.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message indicates the cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

DDMInsertRecKey does the following:

1. If the file was opened for multiple updaters for each record to be inserted:
 - a. If DDM_HLDUPD(FALSE) was specified and the requester currently has a SHRRECLK on a record in the file, the SHRRECLK is released.
 - b. If DDM_HLDUPD(TRUE) was specified and the requester currently has a SHRRECLK on a record in the file, the SHRRECLK is not released.

In all cases, the local VSAM file system attempts to acquire an EXCRECLK. If the EXCRECLK cannot be obtained due to a lock conflict, the function is rejected with RECIUSRM. If the EXCRECLK is obtained, the record insert function is performed. Since all record modifications are committed at the time of modification, the EXCRECLK is released from the record. Even if the function is rejected with an error reply, the obtained EXCRECLK is released from the record.

2. If the function ends with a reply message that has a severity code of ERROR or higher, then:
 - For error termination (SVRCOD of 8): The record locks are the same as before the function was issued. If RECCNT is greater than 1, the record locks are the same as before the last iteration of the function.
 - For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record locks.

DDMInsertRecKey

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file is not delete-capable and the record to be inserted is RECINA.	DTARECRM
<p>One of the following sets of conditions exists:</p> <ul style="list-style-type: none"> The file is the base file for an alternate index file. The alternate index file does not allow duplicate keys. The inserted record would result in a duplicate key value in the alternate index. <p>or</p> <ul style="list-style-type: none"> The file is an alternate index file. The file's base file does not allow duplicate keys, or another alternate index file with the same base file does not allow duplicate keys. The inserted record would result in a duplicate key value in the file's base file or in another alternate index file with the same base file. 	DUPKDIRM
<p>The following conditions exist:</p> <ul style="list-style-type: none"> The file is a keyed file or alternate index file. The file does not allow duplicate keys. The inserted record would result in a duplicate key value in the file index. 	DUPKSIRM
The file handle is invalid.	HDLNFNRM
<p>The following conditions exist:</p> <ul style="list-style-type: none"> The file supports variable-length records. The file is a keyed file or an alternate index file. The record to be inserted does not contain all of the fields for the specified file key. 	KEYVALRM
An EXCRECLK record lock cannot be obtained.	RECIUSRM
<p>The RELENCL (Record Length Class) is fixed, and the length of the record to be inserted (LL) is not equal to the record length (RECLen) of the file <i>plus</i> the length of the record header (see "RECORD (Record)" on page 391 for more information).</p> <p>The record length of the record to be inserted exceeds the maximum record length of the file.</p>	RECLenRM

DDMInsertRecKey

Example

Assume the following:

```
AccessFlags = 0x00000400 ; /* DDM UPDCSR=ON */  
RecCount   = 0x00000001 ;
```

**DDMInsertRecKey (FileHandle, AccessFlags, RecordBuf,
RecordNumber, RecCount)**

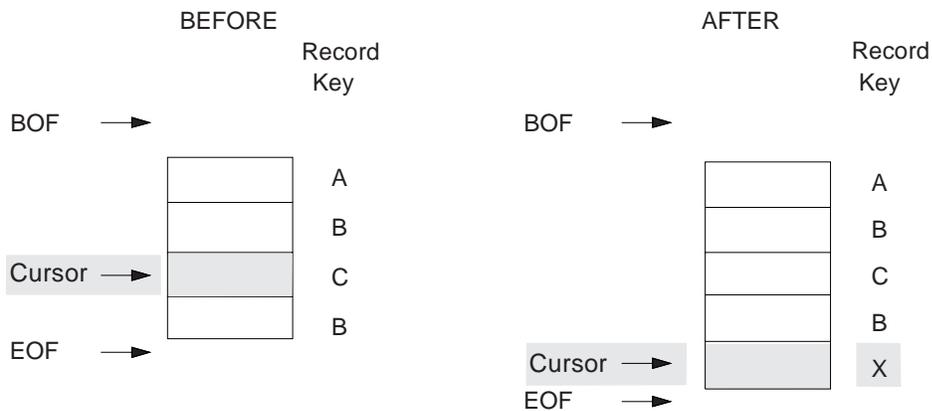


Figure 15. `DDMInsertRecKey` Function with `DDM_UPDCSR`

DDMInsertRecNum

DDMInsertRecNum (Insert by Record Number)

This function inserts records at the position that is specified by the RecordNumber parameter and optionally returns the record key.

Syntax

```
APIRET DDMInsertRecNum (HDDMFILE      FileHandle,  
                        ULONG          AccessFlags,  
                        PDDMRECORD     RecordBuf,  
                        PDDMOBJECT     KeyFdbk,  
                        ULONG          KeyFdbkLen,  
                        RECNUM         RecordNumber,  
                        ULONG          RecCount  
                        );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMSOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
11–31	Reserved flags
10	DDM_UPDCSR (Update Cursor)
3–9	Reserved flags
2	DDM_KEYVALFB (Key Value Feedback)
0–1	Reserved flags

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the records to be inserted at the specified record number. When this function is called, the format of the record buffer is:

LL	CP	Data	...
----	----	------	-----

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following data is either record data or an inactive record length. X'144A' Indicates that the following data is record data.

DDMInsertRecNum

X'142D' Indicates that the following data is an ULONG inactive record length. The number of record descriptions (record Data or inactive record lengths) should be the same as the number indicated in the RecCount.

Data The data associated with this code point.

KeyFdbk

The pointer (PDDMOBJECT) to the key value feedback buffer of the last record inserted. If the DDM_KEYVALFB flag of AccessFlags has not been set, this parameter is ignored. The format of the key value feedback buffer on return from the function is:

LL	X'1115'	Key Value
----	---------	-----------

Field	Description
LL	The length (ULONG) of the response from the beginning of LL to the end of the Key Value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value.
Key Value	The key value.

KeyFdbkLen

The length (ULONG) of the key value feedback buffer. The key value feedback buffer should be the same length as a key value, with an additional six bytes for the length and code point fields. If the DDM_KEYVALFB flag of AccessFlags has not been set, this parameter is ignored.

RecordNumber

The length (RECNUM) of the record number for the first record to be inserted. All other records are placed in consecutive record positions.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DTARECRM	X'1206'	Invalid Data Record
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
DUPRNBRM	X'120A'	Duplicate Record Number
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYVALRM	X'1240'	Invalid Key Value
OBJNSPRM	X'1253'	Object Not Supported
RECDMGRM	X'1249'	Record Damaged

DDMInsertRecNum

Message ID	Code Point	Message Title
RECIUSRM	X'124A'	Record in Use
RECLNRM	X'1215'	Record Length Mismatch
RECNBRRM	X'1224'	Record Number Out of Bounds
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

Records can only be inserted within the bounds of the file and only in inactive record positions:

- For sequential and keyed files, the bounds are record number 1 (inclusive) and the current EOF (exclusive).
- For direct files, the bounds are record number 1 For direct files, the bounds are:
 - Record number 1 (inclusive), and
 - The physical boundaries of the file (inclusive)as defined by the application when the file was created (this can be beyond the EOF position of the file).
- An alternate index file has the same bounds as its base file.

Depending on the value of the DDM_UPDCSR flag, the cursor can be set to the inserted record position or can retain its current setting.

The records in a Record Buffer are processed as a group. Inactive records in the group are treated as place holders between the active records as the group is inserted into the file. How the EOF is updated depends on the type of file. For example, if the file is a direct file and records are added at or beyond the current EOF, the EOF is only updated when an active record is inserted. Inactive records that follow the last active record will be located in the file at or beyond the EOF and are subject to overlay by other functions. See "DDMInsertRecEOF (Insert Records at EOF)" on page 83 for additional information and examples.

The RecCount parameter specifies the number of records to be inserted. The insertion of records begins at RecordNumber.

If RecCount specifies a value other than 1, the record number is increased after each record is inserted. The new record number must meet the same validity criteria as the original (previous) record number before the next record can be inserted. The validity criteria for record number refers to the file boundary rules for record insertion.

If RecCount specifies a value greater than 1 and the DDM_UPDCSR flag is set, the cursor is updated after each record is successfully inserted.

If Key Value Feedback is requested (DDM_KEYVALFB), the key value of the last record inserted is returned.

If the DDM_KEYVALFB flag is set and the file is not keyed, the flag is ignored.

DDMInsertRecNum

The file index is updated when inserting records into a keyed or alternate index file or into the base file of an alternate index file.

If the file supports variable-length records whose lengths are changeable, the length of the record position is changed to match the length of the inserted record.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is based on the DDM_UPDCSR flag. If the DDM_UPDCSR flag is set, the cursor is moved to the inserted record. If the DDM_UPDCSR flag is not set, the cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was called. If RecCount is greater than 1, the cursor position is the same as before the last iteration of the function.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message determines the cursor position.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters, then:

1. If the requester currently has a SHRRECLK on a record in the file, the SHRRECLK is released.
2. The access method attempts to acquire an EXCRECLK on the record.

If the EXCRECLK cannot be obtained because of a lock conflict, the function is rejected with RECIUSRM. If the EXCRECLK is obtained, then:

- a. The record insert function is performed, and because all record modifications are committed at the time of modification, the EXCRECLK is released from the record.
- b. The obtained EXCRECLK is released from the record, even if the function is rejected with an error reply.

If the function ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued. If RecCount is greater than 1, the record locks are the same as before the last iteration of the function.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record locks.

DDMInsertRecNum

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The record buffer address is not greater than zero. DDM_KEYVALFB access flag is specified and KeyFdbk is not specified.	ADDRRM
The file is not delete-capable and the record to be inserted is RECINA.	DTARECRM
One of the following sets of conditions exists: <ul style="list-style-type: none"> The file is the base file for an alternate index file. The alternate index file does not allow duplicate keys. The inserted record would result in a duplicate key value in the alternate index. or <ul style="list-style-type: none"> The file is an alternate index file. The file's base file does not allow duplicate keys, or another alternate index file with the same base file does not allow duplicate keys. The inserted record would result in a duplicate key value in the file's base file or in another alternate index file with the same base file. 	DUPKDIRM
The following are true: <ul style="list-style-type: none"> The file is a keyed file or an alternate index file. The file does not allow duplicate keys. The inserted record would result in a duplicate key value in the file index. 	DUPKSIRM
The RecordNumber parameter specifies a record position that contains an active record.	DUPRNBRM
The file handle is invalid.	HDLNFNRM
Any of the reserved bits are set in the access flags.	INVFLGRM
The file was opened without INSAI (Insert Record) access intent.	INVRQSRM
The following are true: <ul style="list-style-type: none"> The file supports variable-length records. The file is a keyed file or an alternate index file. The record to be inserted does not contain all of the fields for the specified file key. 	KEYVALRM
The Keyfdbk is not large enough to hold the returned key.	LENGTHRM
The data in the record is not a valid record.	OBJNSPRM
The record is to be inserted at a position that does not contain an active or inactive record.	RECDMGRM
An EXCRECLK lock cannot be obtained on the file.	RECIUSRM

DDMInsertRecNum

This Causes the Function to be Rejected	With This Reply Message
<p>The combination of the following two are true:</p> <ul style="list-style-type: none"> The file supports variable-length records whose lengths are not changeable (RECIVL). The record length of the record to be inserted is not equal to the record position length. <p>The record length of the record to be inserted exceeds the maximum record length of the file or is less than the minimum record length.</p> <p>The following conditions are <i>not</i> true:</p> <ul style="list-style-type: none"> If the record length class (RECLNCL) is fixed and the record to be inserted is an active record, the length of the record to be inserted (LL) must be equal to the record length (RECLN) plus the length of the record header. (See "RECORD (Record)" on page 391 for more information.) If the record to be inserted is an inactive record, the record length specified in the inactive record (Data) must be equal to the record length (RECLN) for the file. (See "RECINA (Inactive Record)" on page 389 for more information.) 	<p>RECLNRM</p>
<p>The RecordNumber parameter specifies a value that is outside the bounds of the file, for example:</p> <ul style="list-style-type: none"> The record is outside the bounds for a direct file. The record would be inserted past the EOF for nondirect files. RecordNumber is not greater than zero. 	<p>RECNBRRM</p>
<p>RecCount is not greater than zero.</p>	<p>VALNSPRM</p>

DDMInsertRecNum

Examples

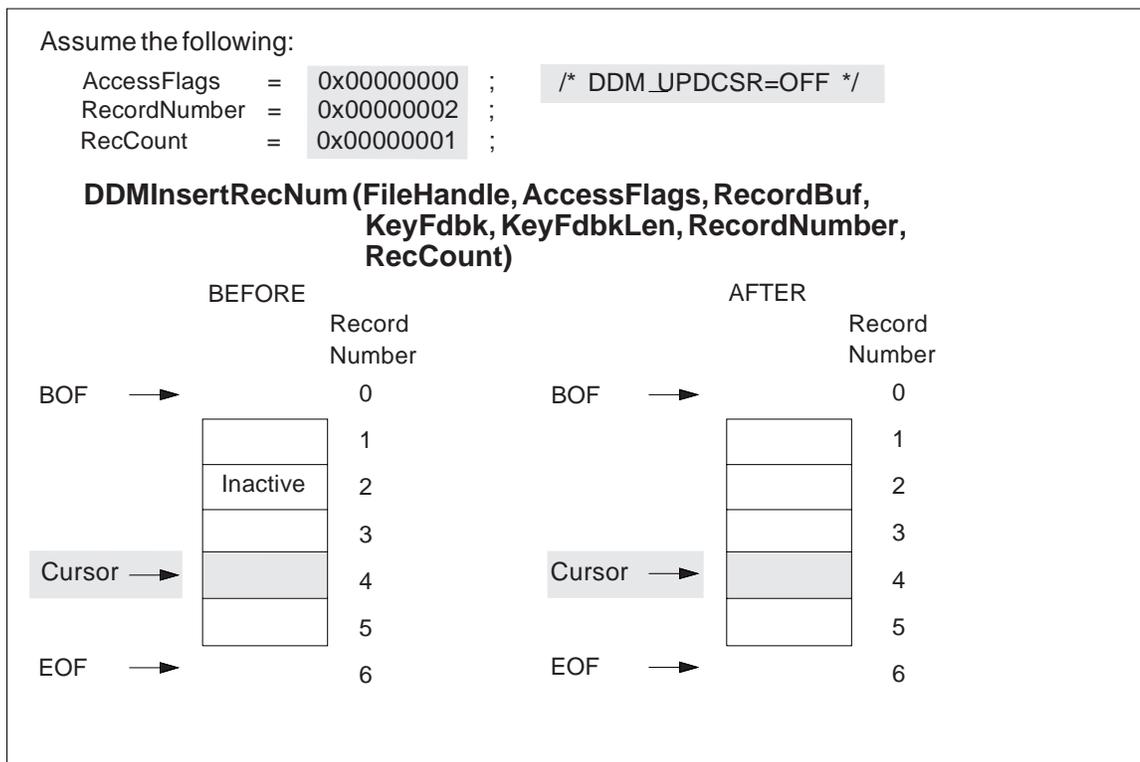


Figure 16. DDMInsertRecNum Function

DDMInsertRecNum

Assume the following:

```
AccessFlags = 0x00000400 ; /* DDM_UPDCSR=ON */  
RecordNumber = 0x00000002  
RecCount = 0x00000002
```

**DDMInsertRecNum (FileHandle, AccessFlags, RecordBuf,
KeyFdbk, KeyFdbkLen, RecordNumber,
RecCount)**

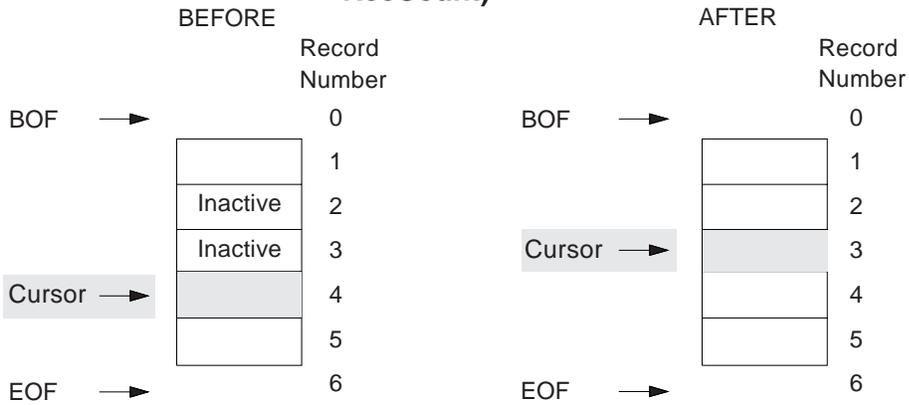


Figure 17. `DDMInsertRecNum` Function with Multiple Records

DDMLoadFileFirst

DDMLoadFileFirst (Load Records into File)

This function loads a file with one or more records that are contained in the record buffer.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMLoadFileFirst (PSZ          FileName,
                        PHDDMLOAD     LoadHandle,
                        ULONG          Flags,
                        PDDMRECORD     RecordBuf,
                        ULONG          RecCount,
                        );
```

Parameters

FileName

The pointer (PSZ) to the name of the record-oriented file to be loaded.

LoadHandle

The pointer (PHDDMLOAD) to the location where the system returns a handle value that is to be used with a subsequent corresponding DDMLoadFileNext function.

Flags

The Flags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
1–31	Reserved flags
0	DDM_CHAIN

This bit notifies the system to keep system resources allocated on behalf of this LoadFile. When the chaining bit is on, any unwritten chained (related) buffers are to be written out or sent to the target system. This occurs on the completion of a DDMLoadFileNext function that has the DDM_CLOSE flag bit set to a value of 1.

When the chaining bit is off:

- The DDM server is allowed to deallocate resources on completion of the DDMLoadFileFirst function.
- A NULL value is returned for LoadHandle.

RecordBuf

The pointer (PDDMRECORD) to the record buffer. The record buffer can contain the following objects:

```
RECORD
RECINA
```

DDMLoadFileFirst

RECAL

These objects can be in mixed order, and they can be repeated.

The format of the record buffer when calling DDMLoadFileFirst is:

LL	CP	Data	...
----	----	------	-----

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data, an inactive record length, or a record attribute list containing a record number and record data.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is an inactive record (RECINA).
X'1430'	Indicates that the following data is a Record Attribute List (RECAL) and can contain RECCNT, RECNR, or both: If CP is a record attribute list, the format of DATA is:

L2	X'111A'	RC	L3	X'111D'	RN
----	---------	----	----	---------	----

L4	CP	Data
----	----	------

Field	Description
L2	The length (ULONG) from the beginning of L2 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT (Record Count) parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where $N \geq 0$, without replicating the contents of the record.
RC	The number (ULONG) of duplicate records in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).

DDMLoadFileFirst

RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record. Each subsequent record has a record number one greater than the previous record.
L4	The length (ULONG) of the record description from beginning of L4 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is a length (ULONG) of an inactive record.
Data	The record data or the length (ULONG) of an inactive record.

If CP is a record or inactive record description, the format of Data is the record data or the length (ULONG) of an inactive record.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

The number of record descriptions (record data and inactive record lengths) should be the same number as indicated in the record count. When a RECAL (Record Attribute List) is specified in RecordBuf and RECCNT of N is specified within the RECAL, the RecCount parameter reflects the N duplicate records. Therefore if RecordBuf contained 10 data records and a RECAL, with RECCNT having a value of 100, the value of RecCount would be 110.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DTARECRM	X'1206'	Invalid Data Record
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
DUPRNBRM	X'120A'	Duplicate Record Number
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILFULRM	X'120C'	File is Full
FILIU SRM	X'120D'	File In Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYVALRM	X'1240'	Invalid Key Value
LENGTHRM	X'F211'	Field Length Error
OBJNSPRM	X'1253'	Object Not Supported
RECDMGRM	X'1249'	Record Damaged
RECLNRM	X'1215'	Record Length Mismatch
RECNBRM	X'1224'	Record Number Out of Bounds
VALNSPRM	X'1252'	Parameter Value Not Supported

DDMLoadFileFirst

Remarks

A set of records can be transferred to a target server and:

- Placed in an empty, existing file.
- Appended to the records in an existing file.
- Distributed into record positions of an existing file.

The record buffer can contain any of the following items and any combination of these items:

- One or more inactive records.
- One or more records.
- One or more RECAL (Record Attribute List) parameters that contains a record and record number. If the record attribute list contains a key value attribute, the key value attribute is ignored.

The DDMLoadFileFirst function begins to load records into a file that is based on the following:

- If the first object is a record or an inactive record, the records are loaded at the EOF position for the file. In this case, the operation of DDMLoadFileFirst is similar to the DDMLoadFileFirst function.
- If the first object is a record attribute list, the records are loaded at the record position that is specified by the record number attribute. In this case, the operation of DDMLoadFileFirst is similar to the DDMLoadFileFirst function.

Subsequent records are loaded in the next higher record position until a RECAL (Record Attribute List) is found or until the entire RecordBuf has been processed. If a RECAL parameter is found, the records that follow are loaded starting with the record position that is specified by the record number value (RN). This allows nonsequential loading of the file.

The records in RecordBuf are processed as a group. Inactive records in the group are treated as place holders between the active records as the group is inserted into the file. How the EOF is updated depends on the type of file. For example, if the file is a direct file and records are added at or beyond the current EOF. The EOF is only updated when an active record is inserted. Inactive records that follow the last active record will be located in the file at or beyond the EOF and are subject to overlay by other functions. See “DDMLoadFileFirst (Insert Records at EOF)” on page 83 for additional information and examples.

If the target file is a keyed file or the base file for an alternate index file, the appropriate indexes are updated as the records are loaded.

An inactive record can be loaded to an inactive record position of a delete-capable file that causes the record position to remain inactive.

If an error condition is encountered, do not use the file handle in a DDMLoadFileNext.

DDMLoadFileFirst

Effect on Cursor Position

There is no effect on the cursor position because the file is not open.

Locking (for Local VSAM File System Only)

DDMLoadFileFirst does the following:

1. Attempts to obtain a MODNONLK on the file.
If the MODNONLK is obtained, the function is processed (successfully or unsuccessfully). If the MODNONLK is not obtained, the function is rejected with FILIUSRM.
2. Releases the MODNONLK it obtained on the file if the DDM_CHAIN bit is not active. If the DDM_CHAIN bit is active, the lock is released by DDMLoadFileNext with the DDM_CLOSE bit active.

If the function ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to be Terminated	With This Reply Message
The file gets full when loading.	FILFULRM
The MODNONLK cannot be obtained on the file.	FILIUSRM
The function tried to load the records outside the bounds of the file. Note: This can occur if the RecCount parameter did not include the actual number of records that was specified in the RECAL descriptor.	RECNBRRM

This Causes the Function to be Rejected	With This Reply Message
The record buffer address is not greater than zero.	ADDRRM
The file is not delete-capable and the record to be inserted is RECINA.	DTARECRM
The following are true: <ul style="list-style-type: none">• The file is a keyed file.• An associated alternate index file does not allow duplicate keys.• The loading of records would result in a duplicate key value.	DUPKDIRM

DDMLoadFileFirst

This Causes the Function to be Rejected	With This Reply Message
<p>The following are true:</p> <ul style="list-style-type: none"> • The file is a keyed file. • The file does not allow duplicate keys. • The loaded record would result in a duplicate key value. 	DUPKSIRM
An attempt is made to load a record at an active record position.	DUPRNBRM
The file that the records are loaded into is a non-DDM file.	FILATHRM
The file has already been opened by DDMOpen, DDMLoadFileFirst (DDM_CHAIN flag on), or DDMUnloadFileFirst (More Data flag on).	FILIUSRM
Any of the reserved bits are set in the access flags.	INVFLGRM
DDM_CHAIN is specified and LoadHandle is not specified. The file does not have insert or modify capability.	INVRQSRM
<p>The following are true:</p> <ul style="list-style-type: none"> • The file supports variable-length records. • The file is a keyed file or the base file of an alternate index file. • The record to be loaded does not contain all of the fields for the specified file key. 	KEYVALRM
The records to be loaded are not active or inactive.	OBJNSPRM
The active or inactive records to be loaded are too long or too short for the record positions in the file.	RECLENRM
A RECAL specifies a RECNR that is outside the boundaries of the file (see "DDMInsertRecNum (Insert by Record Number)" on page 98 for definition of file boundaries).	RECNRMRM
RecCount is not greater than zero.	VALNSPRM

This Causes a Reply Message to be Generated with SRVCOD = X'04' for each out-of-sync file in the file object. The Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>DDMLoadFileFirst or DDMLoadFileNext re-synchronizes the file-change date and time during close processing unless a higher severity condition prevents it from doing so.</p>	FILDMGRM

DDMLoadFileFirst

Examples

Assume the following:

```
RecordBuf = { 0x0000000A, 0x144A, 'XXXX',  
              0x0000000A, 0x144A, 'YYYY' } ;  
Flags     = 0x00000000 ;  
RecCount  = 0x00000002 ;
```

**DDMLoadFileFirst (FileName, LoadHandle, Flags,
RecordBuf, RecCount)**

Has the following effect:

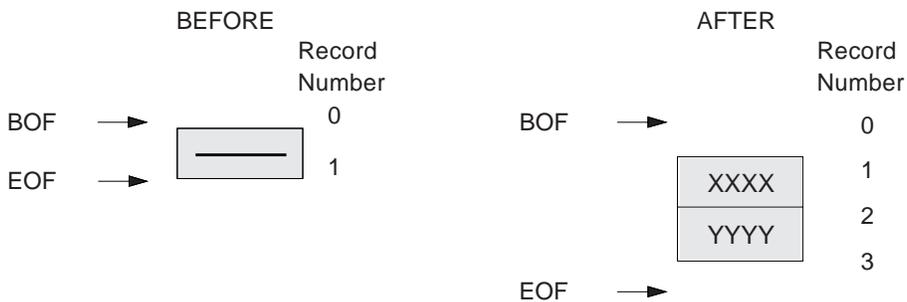


Figure 18. DDMLoadFileFirst Function to a New File

DDMLoadFileFirst

Assume the following:

```
RecordBuf = { 0x0000000A, 0x144A, 'XXXX',
              0x0000000A, 0x144A, 'YYYY' } ;
Flags      = 0x00000000 ;
RecCount   = 0x00000002 ;
```

DDMLoadFileFirst (FileName, LoadHandle, Flags, RecordBuf, RecCount)

Has the following effect:

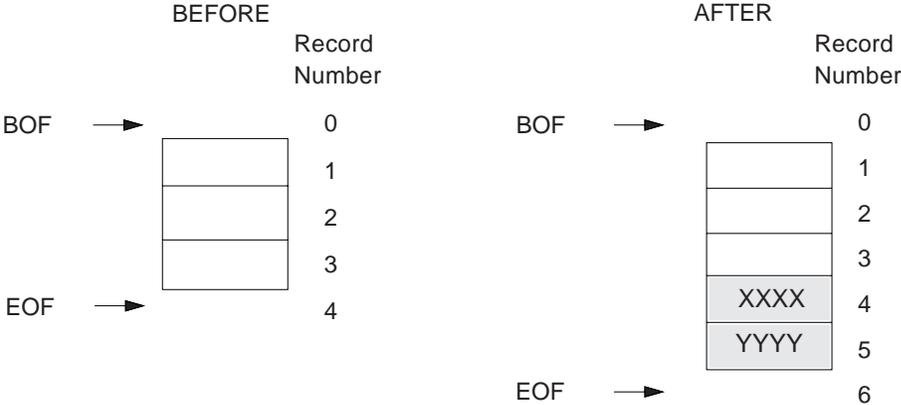


Figure 19. DDMLoadFileFirst Function to Append to a File

DDMLoadFileFirst

Assume the following:

```
RecordBuf = {{0x0000001A, 0x1430, 0x000A, 0x111D,
              0x00000002, 0x0000000A, 0x144A, 'XXXX' },
             {0x0000000A, 0x144A, 'YYYY' },
             {0x0000001A, 0x1430, 0x000A, 0x111D,
              0x00000001, 0x0000000A, 0x144A, 'ZZZ' }} ;
Flags      = 0x00000000 ;
RecCount   = 0x00000003 ;
```

**DDMLoadFileFirst (FileName, LoadHandle, Flags,
RecordBuf, RecCount)**

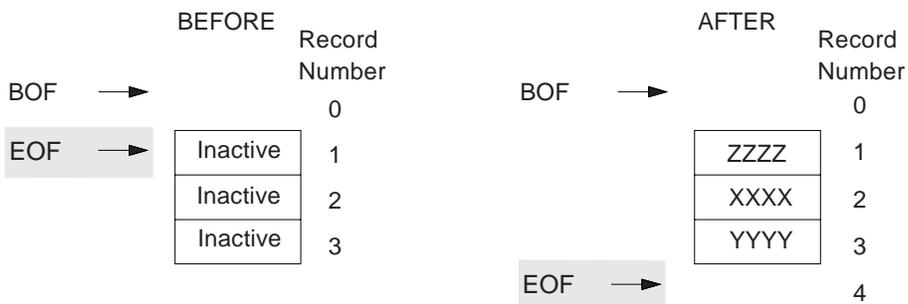


Figure 20. DDMLoadFileFirst Function to Random Load a Direct File

DDMLoadFileNext

DDMLoadFileNext (Load Records into File)

This function continues the load of a file with the records that are contained in the record buffer.

Note: This function should be called after the DDMLoadFileFirst function.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMLoadFileNext (HDDMLOAD      LoadHandle,
                       ULONG          Flags,
                       PDDMRECORD     RecordBuf,
                       ULONG          RecCount
                       );
```

Parameters

LoadHandle

The handle value (HDDMLOAD) previously returned to the caller with DDMLoadFileFirst.

Flags

The Flags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
1–31	Reserved flags
0	DDM_CLOSE (Close LoadFile Requests). A value of 1 for this bit flag notifies the system to end LoadHandle-based chaining and to deallocate LoadHandle-based system resources for this function. Any unwritten chained (related) buffers are written out or sent to the target system on the completion of the DDMLoadFileNext function.

RecordBuf

The pointer (PDDMRECORD) to the record buffer. The record buffer can contain the following objects:

RECORD
RECINA
RECAL

These objects can be in mixed order, and they can be repeated. It is not an error for the record buffer to be null when the DDM_CLOSE flag is set to 1. The format of the record buffer when calling DDMLoadFileNext is:

LL	CP	Data	...
----	----	------	-----

DDMLoadFileNext

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data, an inactive record length, or a record attribute list containing a record number and record data.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is an ULONG length of an inactive record (RECINA).
X'1430'	Indicates that the following data is a RECAL (Record Attribute List), and can contain RECCNT, RECNR, or both.

If CP is a record attribute list, the format of the DATA is:

L2	X'111A'	RC	L3	X'111D'	RN	L4	CP	Data
----	---------	----	----	---------	----	----	----	------

Field	Description
L2	The length (ULONG) from the beginning of L2 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a RECCNT (Record Count). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where $N \geq 0$, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record. Each subsequent record has a record number one greater than the previous record.
L4	The length (ULONG) of the record description from beginning of L4 to the end of Data.

DDMLoadFileNext

CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is an ULONG length of an inactive record.
Data	The record data or the length (ULONG) of an inactive record.

If CP is a record or inactive record description, the format of DATA is the record data or the length (ULONG) of an inactive record.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

The number of record descriptions (record data and inactive record lengths) should be the same number as indicated in the record count. When a RECAL (Record Attribute List) is specified in RecordBuf and RECCNT of N is specified within the RECAL, the RecCount parameter reflects the N duplicate records. Therefore if RecordBuf contained 10 data records and a RECAL, with RECCNT having a value of 100, the value of RecCount would be 110.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DTARECRM	X'1206'	Invalid Data Record
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
DUPRNBRM	X'120A'	Duplicate Record Number
FILFULRM	X'120C'	File is Full
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
KEYVALRM	X'1240'	Invalid Key Value
LENGTHRM	X'F211'	Field Length Error
OBJNSPRM	X'1253'	Object Not Supported
RECDMGRM	X'1249'	Record Damaged
RECLNRM	X'1215'	Record Length Mismatch
RECNBRM	X'1224'	Record Number Out of Bounds
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

A set of records can be transferred to a target server and either:

- Appended to the records in an existing file, or
- Distributed into record positions of an existing file.

The record buffer can contain any of the following items and any combination of these items:

DDMLoadFileNext

- One or more inactive records.
- One or more records.
- One or more RECAL (Record Attribute List) parameters that contains a record and record number. If the record attribute list contains a key value attribute, the key value attribute is ignored.

DDMLoadFileNext begins to load records into a file that is based on the following:

- If the first object is a record or an inactive record, the records are loaded at the EOF position for the file. In this case, the operation of DDMLoadFileNext is similar to the DDMLoadFileNext function.
- If the first object is a record attribute list, the records are loaded at the record position that is specified by the record number attribute. In this case, the operation of DDMLoadFileNext is similar to the DDMLoadFileNext function.

Subsequent records are loaded in the next higher record position until a RECAL (Record Attribute List) is found or until the entire RecordBuf has been processed. If a record attribute list is found, the records that follow are loaded starting with the record position that is specified by the record number value (RN). This allows nonsequential loading of the file.

The records in a Record Buffer are processed as a group. Inactive records in the group are treated as place holders between the active records as the group is inserted into the file. How the EOF is updated depends on the type of file. For example, if the file is a direct file and records are added at or beyond the current EOF, the EOF is only updated when an active record is inserted. Inactive records that follow the last active record will be located in the file at or beyond the EOF and are subject to overlay by other functions. See "DDMLoadFileNext (Insert Records at EOF)" on page 83 for additional information and examples.

If the target file is a keyed file or the base file for an alternate index file, the appropriate indexes are updated as the records are loaded.

An inactive record can be loaded to an inactive record position of a delete-capable file causing the record position to remain inactive.

If an error condition is encountered, do not use the file handle in a DDMLoadFileNext.

Effect on Cursor Position

There is no effect on the cursor position because the file is not open.

Locking (for Local VSAM File System Only)

DDMLoadFileNext releases the MODNONLK that was obtained by DDMLoadFileFirst on the file, provided the DDM_CLOSE bit is active.

If this function ends with a reply message that has a severity code of ERROR or higher, then:

DDMLoadFileNext

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The record buffer address is not greater than zero.	ADDRRM
The file is not delete-capable and the record to be inserted is RECINA. Note: An inactive record can be loaded to an inactive record position of a delete-capable file causing the record position to remain inactive. DTARECRM is not returned in this case.	DTARECRM
The following are true: <ul style="list-style-type: none"> • The file is the base file for an alternate index file. • The alternate index file does not allow duplicate keys. • The inserted record would result in a duplicate key value. 	DUPKDIRM
The following are true: <ul style="list-style-type: none"> • The file is a keyed file. • The file does not allow duplicate keys. • The loaded record would result in a duplicate key value. 	DUPKSIRM
An attempt is made to load an active or inactive record at an active record position.	DUPRNBRM
The handle from DDMLoadFileFirst is not used as the handle for a DDMLoadFileNext.	HDLNFNRM
The file gets full when loading.	FILFULRM
Any of the reserved bits are set in Flags.	INVFLGRM
The following are true: <ul style="list-style-type: none"> • The file supports variable-length records. • The file is a keyed file or the base file of an alternate index file. • The record to be loaded does not contain all of the fields for the specified file key. 	KEYVALRM
The records to be loaded were not valid records.	OBJNSPRM
The active or inactive records to be loaded are too long or too short for the record positions in the file.	RECLENRM

DDMLoadFileNext

This Causes the Function to be Rejected	With This Reply Message
<p>A RECAL specifies a RECNR that is outside the boundaries of the file (see "DDMInsertRecNum (Insert by Record Number)" on page 98 for definitions of file boundaries).</p> <p>The function tried to load records outside the bounds of the file.</p> <p>Note: This can occur if the record count parameter did not include the actual number of records that was specified in the RECAL descriptor.</p>	RECNBRRM
RecCount is not greater than zero.	VALNSPRM

DDMLoadFileNext

Examples

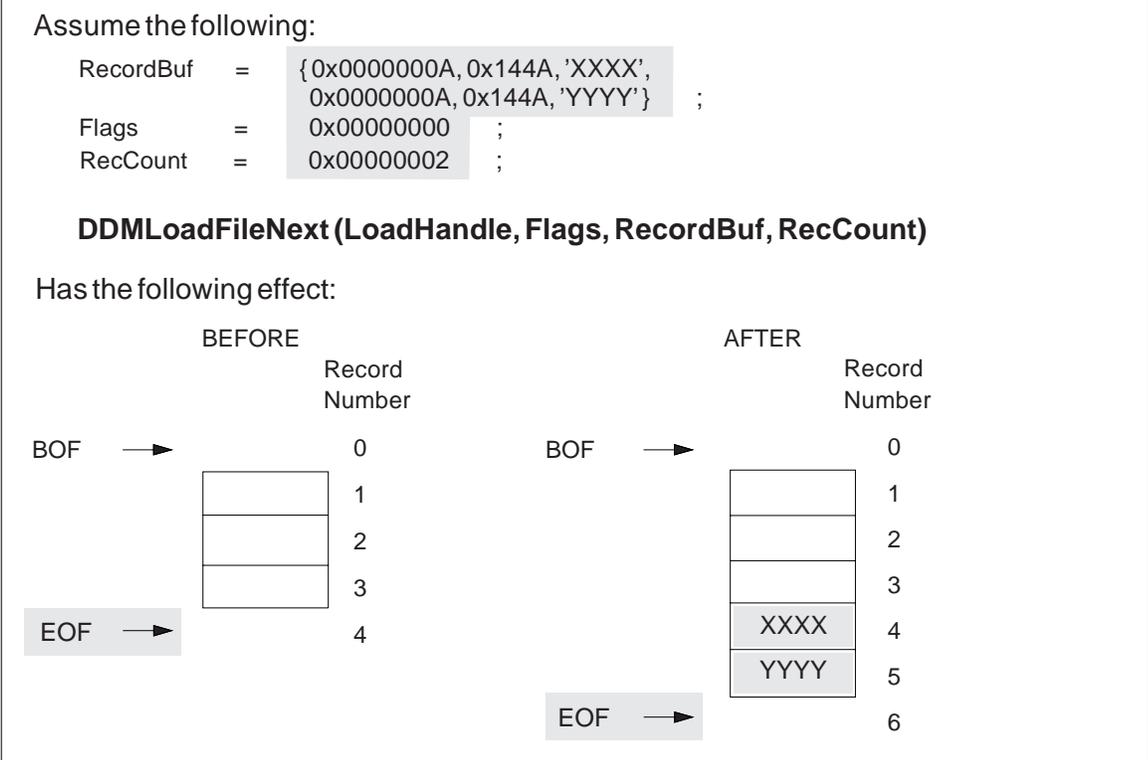


Figure 21. DDMLoadFileNext Function to Append to a File

DDMModifyRec

DDMModifyRec (Modify Record)

This function modifies a record that has an update intent placed on it.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMModifyRec (HDDMFILE      FileHandle,
                    ULONG          AccessFlags,
                    PDDMRECORD     RecordBuf
                    );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
10–31	Reserved flags
9	DDM_INHMODKY (Inhibit Modified Keys)
0–8	Reserved flag

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the record. The format of the record buffer when calling the function is:

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of record data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
HDLNFNRM	X'1257'	File Handle Not Found

DDMModifyRec

Message ID	Code Point	Message Title
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYUDIRM	X'1201'	Key Update Not Allowed by Different Index
KEYUSIRM	X'123F'	Key Update Not Allowed by Same Index
KEYVALRM	X'1240'	Invalid Key Value
OBJNSPRM	X'1253'	Object Not Supported
RECLENRM	X'1215'	Record Length Mismatch
UPDINTRM	X'124E'	No Update Intent on Record

Remarks

DDMModifyRec has the following effects:

- For a sequential or direct file, the contents of the record with the update intent are replaced with the supplied record.
- If the modification affects the key field (or fields) and DDM_INHMODKY is set, the function fails with a Key Update Not Allowed (KEYUDIRM or KEYUSIRM) message. Otherwise, the contents of the record with the update intent are replaced with the replacement record.
- For keyed and alternate index files, the associated indexes are updated.
- The record position becomes active if it was not active before.
- The cursor position does not change; it points to the same record position at the completion of the function.
- If the file supports variable-length records whose length is changeable, the length of the record position is changed to match the length of the modified record.
- Update intent is removed.

Before DDMModifyRec can be used, an update intent must be placed on a record in the file. A DDMSetxxxx or DDMGetRec function can be used to place an update intent on a record.

For direct files, EOF may change if the modified record was an inactive record that was past the current EOF.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message determines the cursor position.

DDMModifyRec

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters:

1. The access method attempts to acquire an EXCRECLK lock on the record that has an update intent placed on it. If the EXCRECLK lock cannot be obtained because of a lock conflict, the function is rejected with the RECIUSRM reply message.
2. If the EXCRECLK lock is obtained, the DDMModifyRec function is performed. Because all record modifications are committed at the time of modification, the EXCRECLK lock is released from the record. Even if the function is rejected with an error reply, the obtained EXCRECLK lock is released from the record.

If DDMModifyRec ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record lock.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
RecordBuf address is not supplied.	ADDRRM
Modification would result in duplicate keys in associated index file.	DUPKDIRM
Modification would result in duplicate keys in current index file.	DUPKSIRM
The file handle is invalid.	HDLNFNRM
Any reserved bits are set in the access flags.	INVFLGRM
The MODAI access intent was not specified when the file was opened.	INVRQSRM
KEYUDIRM	Modification would cause key in associated index file to be modified.
Modification would cause key in current index file to be modified.	KEYUSIRM
The file supports variable-length records; the file is a keyed file or an alternate index file; and the modified record does not contain all of the fields for the specified file key.	KEYVALRM
A record other than an active record is sent as the modified record.	OBJNSPRM

DDMModifyRec

This Causes the Function to be Rejected	With This Reply Message
The following are true: <ul style="list-style-type: none">• The file supports variable-length records whose length is not changeable (initially variable).• The record length of the modified record is not equal to the record position length. The record in the RecordBuf is not the correct length.	RECLENRM
The EXCRECLK lock cannot be obtained on the file.	RECIUSRM
No record in the file has an update intent placed on it.	UPDINTRM

DDMModifyRec

Example

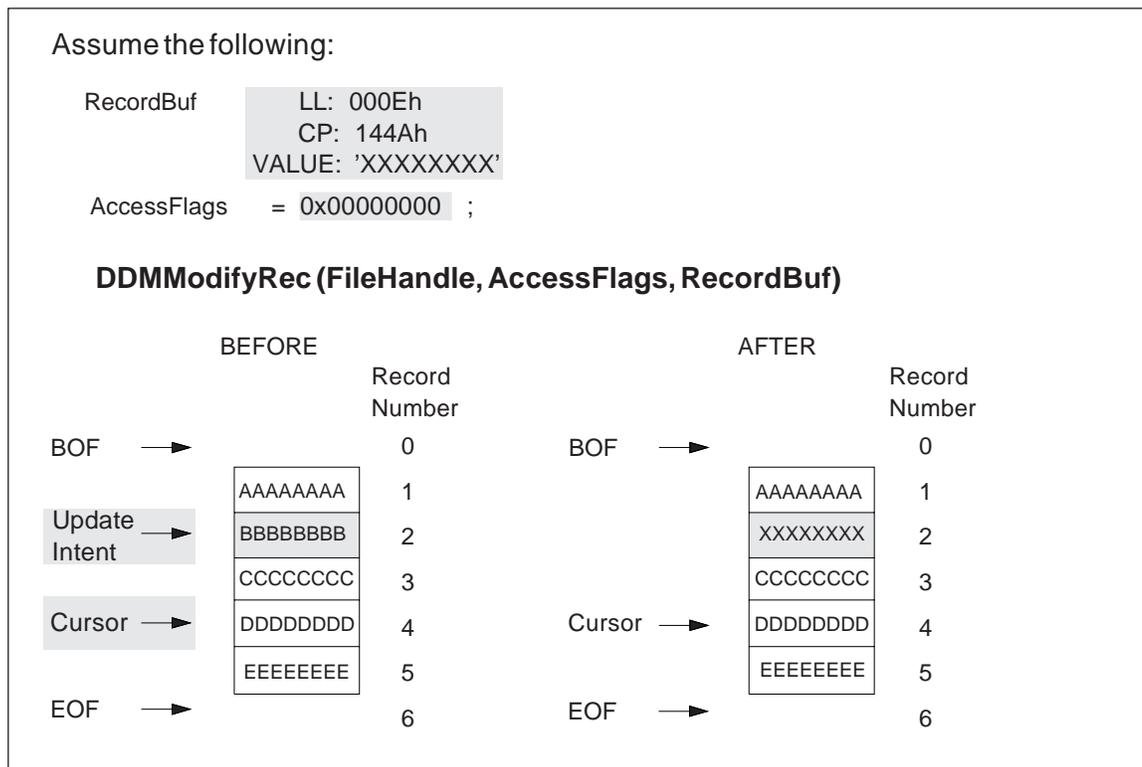


Figure 22. DDMModifyRec Function. The BEFORE state illustrates a case where the cursor and the update intent are on different records. This occurs when a function like DDMSetUpdateNum or DDMSetRecNum is issued using the DDM_HLDCSR and DDM_UPDINT flags.

DDMOpen (Open File)

This function establishes a logical connection between the using program on the source system and the accessed file on the target system.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMOpen (PSZ          FileName,
                PHDDMFILE    FileHandle,
                CODEPOINT    AccessMethod,
                ULONG         AccIntList,
                USHORT        FileShare,
                PBYTE         EABuf,
                PBYTE         (reserved)
                );
```

Parameters

FileName

The pointer (PSZ) to the name of the record-oriented file to be opened.

FileHandle

The pointer (PHDDMFILE) to the file handle returned for use on all subsequent file access and close requests for the file that is being opened.

AccessMethod

The value (CODEPOINT) indicating the requested access method for the file. Specifying the appropriate value identifies the requested access method. Valid values are:

Value	Description
X'1433'	RELNRBAM (Relative by Record Number)
X'1435'	RNDRNBAM (Random by Record Number)
X'1407'	CMBNRBAM (Combined Record Number)
X'1432'	RELKEYAM (Relative by Key)
X'1434'	RNDKEYAM (Random by Key)
X'1406'	CMBKEYAM (Combined Keyed)
X'1405'	CMBACCAM (Combined Access)

The choice of access method can affect read performance. For more information about access methods, see "Access Methods" on page 18.

AccIntList

The value (ULONG) that specifies the access functions that will be used based on whether the bit flag is set. The bit flags are:

Bit	Meaning
7-31	Reserved flags

DDMOpen

- 6** DDM_FAILONERROR (Fail-Errors)
Specifies the handling of media I/O errors.
This bit is the same as DosOpen with OpenMode bit FAIL_ON_ERROR.
- 5** Reserved For Future Use
- 4** DDM_WRITETHRU (File Write-Through)
The file is opened as follows:
- 0 — any data that is written to the file may be cached in memory and written to the media at a later time.
 - 1 — any data that is written to the file may be cached in memory. However, the data is immediately written to the media synchronously with the request.
- This bit is the same as DosOpen with OpenMode bit OPEN_FLAGS_WRITE_THROUGH.
- 3** DDM_DELAI (Delete Record)
Specifies that the requester intends to delete records from the file. If DDM_DELAI is not specified, the DDDeleteRec function is rejected with the INVRQSRM reply message.
This bit is the same as DosOpen with OpenMode bit OPEN_ACCESS_READWRITE.
- 2** DDM_MODAL (Modify Record)
Specifies that the requester intends to modify existing records in the file. If the DDM_MODAL intent is not specified, the following functions are rejected with the INVRQSRM reply message.
- DDMTruncFile
 - DDMModifyRec
- This bit is the same as DosOpen with OpenMode bit OPEN_ACCESS_READWRITE.
- 1** DDM_INSAI (Insert Record)
Specifies that the requester intends to insert records into the file. If the DDM_INSAI intent is not specified, the following functions are rejected with the INVRQSRM reply message.
- DDMInsertRecNum
 - DDMInsertRecEOF
 - DDMInsertRecKey
- This bit is the same as DosOpen with OpenMode bit OPEN_ACCESS_READWRITE.

DDMOpen

0 DDM_GETAI (Get Record)

Specifies that the requester intends to retrieve records from the file. If DDM_GETAI is not specified, the DDMGetRec function is rejected with the INVRQSRM reply message.

If DDM_GETAI is not specified and DDM_NODATA is not set, the following functions are rejected with the INVRQSRM reply message.

- DDMGetRec
- DDMSetFirst
- DDMSetKey
- DDMSetKeyFirst
- DDMSetKeyLast
- DDMSetKeyNext
- DDMSetKeyPrevious
- DDMSetLast
- DDMSetMinus
- DDMSetRecNum
- DDMSetNextRec
- DDMSetNextKeyEqual
- DDMSetPlus
- DDMSetPrevious
- DDMSetUpdateKey
- DDMSetUpdateNum.

This bit is the same as DosOpen with OpenMode bit OPEN_ACCESS_READONLY (if no other access intent is specified along with GETAI).

FileShare

Specifies the value (USHORT) for the concurrent users with which the requester is willing to share the file. The valid values are:

- X'0001'** DDM_NOSHARE (None). This value allows no concurrent users.
This bit is the same as DosOpen with OpenMode bit OPEN_SHARE_DENYREADWRITE.
- X'0002'** DDM_READERS (Readers). This value allows sharing with concurrent users who only intend to read records from the file.
This bit is the same as DosOpen with OpenMode bit OPEN_SHARE_DENYWRITE.
- X'0003'** DDM_UPDATERS (Updaters). This value allows sharing with concurrent users who intend to update records in the file.
This bit is the same as DosOpen with OpenMode bit OPEN_SHARE_DENYNONE.

Note: The combination of the AcclntList and the FileShare value that are specified determines what implicit lock is obtained on the file.

DDMOpen

EABuf

The pointer (PBYTE) to the file's EA data to be returned by DDMOpen or NULL. See "Extended Attributes" on page 5 for more information on the format of this buffer.

(reserved)

This pointer (PBYTE) is reserved for future use and must be specified as NULL.

Returns

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Access Method
ACCINTRM	X'1266'	Access Intent List Error
ACCMTHRM	X'1231'	Invalid Access Method
ADDRRM	X'F212'	Address Error
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILUSRMR	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
FILSNARM	X'120F'	File Space Not Available
HDLNFNRM	X'1257'	File Handle Not Found
INTATHRM	X'125C'	Not Authorized to Open Intent for Named File
INVFLGRM	X'F205'	Invalid Flags
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRMNSPRM	X'1251'	Parameter Not Supported
RSCLMTRM	X'1233'	Target Resource Limits Reached

Remarks

Once the connection is established, access method commands can flow between the source and target systems.

The target server uses both AccessMethod and AcclntList to determine the access method that is required by the user. If the required support is not available in the target server, the function is rejected with the ACCMTHRM reply message.

The DDM architecture permits the DDM server to promote a user-specified lower-level access method class to a file to a higher-level access method class. All subsequent access to this file are processed as though the promoted access method class has been specified by the user. The promotion values for record-oriented access methods are described in "Access Methods" on page 18.

The AcclntList is used to limit the use of valid functions in an access method.

The FileShare value indicates the type of concurrent users with which the requester is willing to share the file while processing the file. This permits the requester to ensure that concurrency problems do not occur.

In the local VSAM file system, to process a keyed file via an associated alternate index file, it is only necessary for the user to issue a DDMOpen for the alternate index file. Issuing a subsequent DDMOpen for another alternate index (of the same keyed file) or

DDMOpen

for the keyed file itself, is considered *concurrent use* by the local VSAM file system. Concurrent use requires that the `AccIntList` and `FileShare` parameters of the `DDMOpen` functions be compatible. For example, if an alternate index file is opened with `AccIntList=MODAI` and `FileShare=Readers`, any subsequent `DDMOpen` function issued for another alternate index of the same keyed file requires `AccIntList=GETAI` and `FileShare=Updaters`. Otherwise, the subsequent `DDMOpen` will fail. (See “Locking (for Local VSAM File System Only)” for more information.)

When the file is opened, the cursor is set to the BOF position.

An example of requesting Extended Attributes (EAs) is provided on page 5.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is created and moved to the beginning of the file.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The `CSRPOSST` (Cursor Position Status) parameter on the reply message determines the cursor position.

Locking (for Local VSAM File System Only)

`DDMOpen` does the following:

1. Acquires a file lock on the file. For keyed and alternate index files, an equivalent file lock is placed on the keyed file and each of its associated index files. This occurs when the command is issued for the keyed file or for any of its associated alternate index files. The type of lock that is acquired is dependent on the values of the `AccIntList` and `FileShare` parameters. Table 18 on page 132 specifies the type of file lock the `DDMOpen` function acquires.

For keyed and alternate index files, an equivalent file lock is placed on the keyed file and each of its associated index files. This occurs when the function is issued for the keyed file or any of its associated alternate index files.

2. Acquires only one file lock on the file. This file lock is not released until the file is closed.

For keyed and alternate index files, only one lock per file is acquired. The file locks are not released until the file is closed.

3. If the function ends with a reply message that has a severity code of `ERROR` or higher, then:
 - For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
 - Severe Termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

DDMOpen

Table 18. File Locks Obtained by DDMOpen for Record Files

File Sharing (FileShare)	File Access Intents (AcclntList)	
	GETAI Only	MODAI, DELAI, INSAI
None	GETNONLK	MODNONLK
Reader	GETGETLK	MODGETLK
Updater	GETMODLK	(See Note)

Note: In this case, the file is being opened so that both the requester and concurrent users can update the file. (This is referred to as “opened for multiple updaters” elsewhere in this document.) For the files where the local VSAM file system supports implicit record locks, a MODMODLK lock is acquired. Otherwise, a MODGETLK file lock is acquired.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
<p>If the user attempts to open a file without setting at least one of the following bits in AcclntList:</p> <ul style="list-style-type: none"> • GETAI • INSAI • MODAI • DELAI 	ACCINTRM
The target server does not support the access method specified by AccessMethod and AcclntList.	ACCMTHRM
DDMOpen is issued against a keyed file or any of its associated indexes and the associated indexes have recorded, in DDM_BASCHGDT, the last-change date/time for the base file that is different from the current system last-change date/time (System Object Attribute).	FILDMGRM
The file lock cannot be acquired because of a lock conflict.	FILIUSRM
The user attempts to open a file with an access intent (specified in AcclntList) for which the file is not allowed.	INTATHRM
The file lock cannot be acquired because of insufficient lock manager resources or because of an implementation file lock maximum.	RSCLMTRM

This Causes a Reply Message to be Generated with SRVCOD = X'04' for each out-of-sync file in the file object and the Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>If the file was opened for write access, DDMClose will re-synchronize the file-change date and time unless a higher severity condition prevents it from doing so.</p>	FILDMGRM

DDMQueryFileInfo (Get a File's Information)

This function returns information for a specific file.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMQueryFileInfo (HDDMFILE      FileHandle,
                        ULONG          FileInfoLevel,
                        PBYTE         FileInfoBuf,
                        ULONG          FileInfoBufSize
                        );
```

Parameters

FileHandle

The handle (HDDMFILE) of the open file.

FileInfoLevel

The level (ULONG) of file information that is required.

Level 0x00000001 is the only defined level. This is the same as DosQueryFileInfo, ulFileInfoLevel bit (FILE_STANDARD).

Level 0x00000001 returns a subset of the EA information for the file. On input, FileInfoBuf maps to an EAOP2 structure. fpGEA2List points to a GEA2 list defining the attribute names whose values are returned. fpFEA2List points to a data area where the relevant FEA2 list is returned. The length field of this FEA2 list is valid, giving the size of the FEA2 list buffer. oError is ignored.

On output, FileInfoBuf is unchanged because the buffer pointed to by fpFEA2List is the one that is filled in with the returned information.

FileInfoBuf

The pointer (PBYTE) to the storage area where the system returns the requested level of file information. Refer to "Extended Attributes" on page 5 for more information on the format of this buffer.

FileInfoBufSize

The length (ULONG) of the FileInfoBuf.

DDMQueryFileInfo

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
CMDCHKRM	X'1254'	Command Check
FILIUSRM	X'120D'	File in Use
HDLNFNRM	X'1257'	File Handle Not Found
LENGTHRM	X'F211'	Field Length Error
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

This function is similar to the OS/2 DosQueryFileInfo command.

An example of requesting Extended Attributes (EAs) is provided on page 5.

When requesting information on the variable-length EAs (ALTINDLS and KEYDEF), it is possible for the user to provide inadequate buffer space in the FileInfoBuf parameter. If this is the case, the function is rejected with the LENGTHRM reply message and a server diagnostic code of 0004 (Extended Attribute reply buffer too small). If the buffer that is provided was at least 4 bytes long, it contains the required buffer length. This buffer length should be used to create a FileInfoBuf of FileInfoBufSize that is large enough to contain the requested list of EAs.

File information, where applicable, is at least as accurate as the most recent DDMClose, DDMForceBuffer, or DDMSetFileInfo.

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

For the local VSAM file system on AIX, the file needs to be opened for DDMQueryFileInfo. The level of locking in effect is the same as what was specified in the DDMSOpen call for the file.

For the local VSAM file system on OS/2, the locking behaviour is the same as that for DOSQueryFileInfo. See *OS/2 WARP Control Program Programming Reference*.

Record File Attributes by File Class

Refer to Table 13 on page 37.

DDMQueryPathInfo (Get File or Subdirectory Information)

This function returns information for a specific file or subdirectory.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMQueryPathInfo (PSZ          PathName,
                        ULONG         PathInfoLevel,
                        PBYTE         PathInfoBuf,
                        ULONG         PathInfoBufSize
                        );
```

Parameters

PathName

The pointer (PSZ) to the full path name of the file or subdirectory.

PathInfoLevel

The level (ULONG) of path information that is required.

Level 0x00000001 is the only defined level. This is the same as DosQueryPathInfo, ulFileInfoLevel bit (FILE_STANDARD).

Level 0x00000001 returns a subset of the EA information for the file. On input, PathInfoBuf maps to an EAOP2 structure. fpGEA2List points to a GEA2 list defining the attribute names whose values are returned. fpFEA2List points to a data area where the relevant FEA2 list is returned. The length field of this FEA2 list is valid, giving the size of the FEA2 list buffer. oError is ignored.

On output, PathInfoBuf is unchanged since the buffer pointed to by fpFEA2List is the one that is filled in with the returned information.

PathInfoBuf

The pointer (PBYTE) to the storage area where the system returns the requested level of path information. Refer to “Extended Attributes” on page 5 for more information on the format of this buffer.

PathInfoBufSize

The length (ULONG) of PathInfoBuf.

DDMQueryPathInfo

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
CMDCHKRM	X'1254'	Command Check
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
LENGTHRM	X'F211'	Field Length Error
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

This function is similar to the OS/2 DosQueryPathInfo command.

An example of requesting Extended Attributes (EAs) is provided on page 5.

When requesting information on the variable-length EAs (.DDM_ALTINDLS and .DDM_KEYDEF), it is possible for the user to provide inadequate buffer space in the PathInfoBuf parameter. If this is the case, the function is rejected with the LENGTHRM reply message and a server diagnostic code of 0004 (Extended Attribute reply buffer too small). If the buffer that is provided was at least 4 bytes long, it contains the required buffer length. This buffer length should be used to create a PathInfoBuf of PathInfoBufSize that is large enough to contain the requested list of EAs.

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

For the OS/2 local VSAM file system, the file locking rules are the same as for DOSFindFirst. These rules do not permit access to the file attributes if the file is already opened by another process. See *OS/2 WARP Control Program Programming Reference*.

For the AIX local VSAM file system, two processes can call this API concurrently.

Exceptions

This Causes a Reply Message to be Generated with SRVCOD = X'04'. The Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>DDMQueryPathInfo re-synchronizes the file-change date and time if the file is not open to another process unless a higher severity condition prevents it from doing so.</p>	FILDMGRM

DDMQueryPathInfo

Record File Attributes by File Class

Refer to Table 13 on page 37.

DDMRename

DDMRename (Rename File)

This function changes the name of an existing file.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMRename (PSZ      FileName,
                  PSZ      NewFileName
                  );
```

Parameters

FileName

The pointer (PSZ) to the name of the record-oriented file to be renamed.

NewFileName

The pointer (PSZ) to the new file name.

Returns

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Access Method
DRCATHRM	X'1237'	Not Authorized to Directory
DRCFULRM	X'1258'	Directory Full
EXSCNDRM	X'123A'	Existing Condition
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILIUSRM	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
FILSNARM	X'120F'	File Space Not Available
HDLNFNRM	X'1257'	File Handle Not Found
INVRQSRM	X'123C'	Invalid Request
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRMNSPRM	X'1251'	Parameter Not Supported
RSCLMTRM	X'1233'	Target Resource Limits Reached

DDMRename

Remarks

Naming that directory as part of the new file name (NewFileName) can move a file to a different directory.

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

The DDMRename function:

1. Attempts to obtain a MODNONLK lock on the file.
If the MODNONLK lock is obtained, the function is processed (successfully or unsuccessfully). If the MODNONLK lock is not obtained, the function is rejected with the FILIUSRM reply message.
2. Releases the MODNONLK lock it obtained on the file.

If the function ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes	This Reply Message to be Returned
The new name for the file is the same as the existing name for the file.	EXSCNDRM

This Causes the Function to be Rejected	With This Reply Message
The new file cannot be entered into the directory because the directory is full.	DRCFULRM
The requester has the named file open.	FILIUSRM

DDMRename

This Causes a Reply Message to be Generated with SRVCOD = X'04' for each out-of-sync file in the file object. The Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>DDMRename re-synchronizes the file-change date and time unless a higher severity condition prevents it from doing so.</p>	FILDMGRM

DDMSetBOF (Set Cursor to Beginning of File)

This function sets the cursor to the beginning-of-file (BOF) position of the file.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetBOF (HDDMFILE      FileHandle
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

Returns

Message ID	Code Point	Message Title
HDLNFNRM	X'1257'	File Handle Not Found

Remarks

DDMSetBOF sets the cursor to the BOF position in the file to allow relative accesses (for example, DDMSetNextRec, DDMSetPlus, and DDMSetKeyNext) to be performed. Any attempt to retrieve, insert, or modify a record at this file position is rejected.

If the hold cursor indicator of the cursor is on, it is set off by this function.

Resets any key limits that were set on a keyed file.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the BOF position of the file.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The CSRPOSST (Cursor Position Status) parameter on the reply message determines the cursor position.

DDMSetBOF

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters and the requester currently has a SHRRECLK lock on a record in the file, the SHRRECLK lock is released.

If the function ends with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The DTALCKST (Data Lock Status) parameter on the reply message determines the state of the record locks.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is invalid.	HDLNFNRM

Example

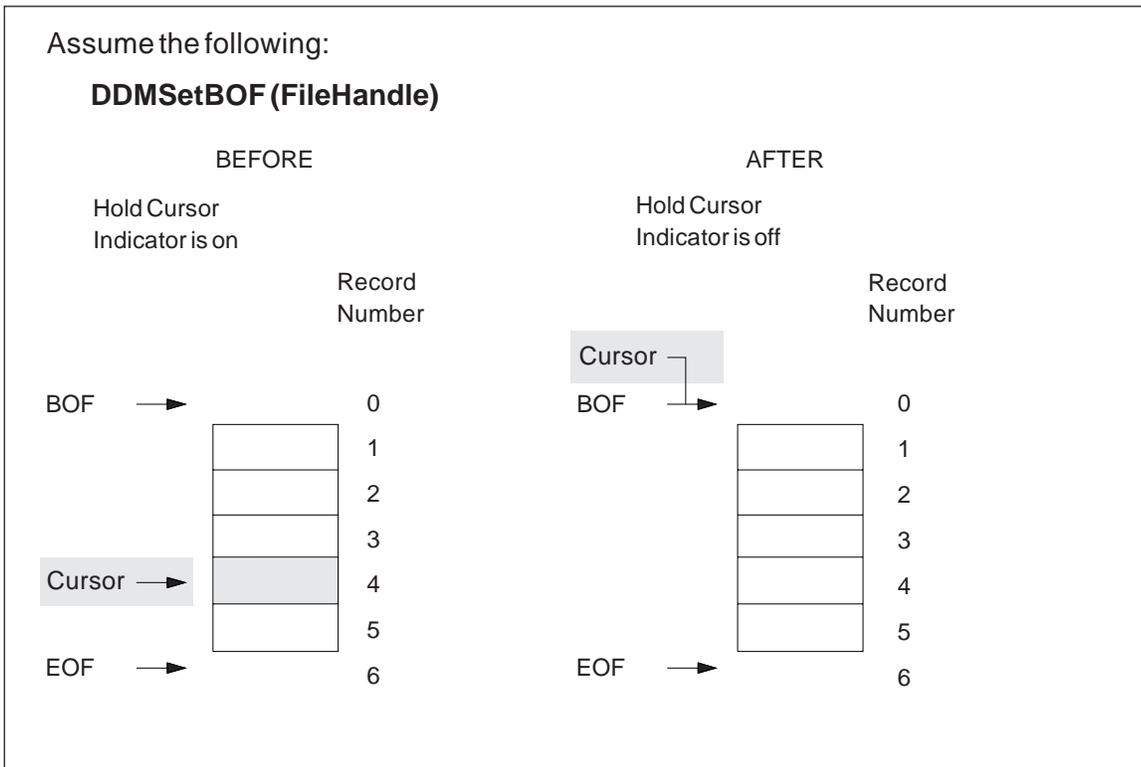


Figure 23. DDMSetBOF Function

DDMSetEOF

DDMSetEOF (Set Cursor to End of File)

This function sets the cursor to the end-of-file (EOF) position of the file.

Syntax

```
#include dub.h /* Required for all platforms */  
  
APIRET DDMSetEOF (HDDMFILE      FileHandle
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

Returns

Message ID	Code Point	Message Title
HDLNFRM	X'1257'	File Handle Not Found

Remarks

The cursor position is defined by each file class.

The cursor is placed at the EOF position to allow relative accesses (for example DDMSetPrevious, DDMSetMinus, and DDMSetKeyPrevious) to be performed.

If the hold cursor indicator of the cursor is turned on, it is set off by this function.

Resets any key limits that were set on a keyed file.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the EOF position of the file.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters and the requester currently has a SHRRECLK lock on a record in the file, the SHRRECLK lock is released.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

DDMSetEOF

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is not invalid.	HDLNFNRM

Example

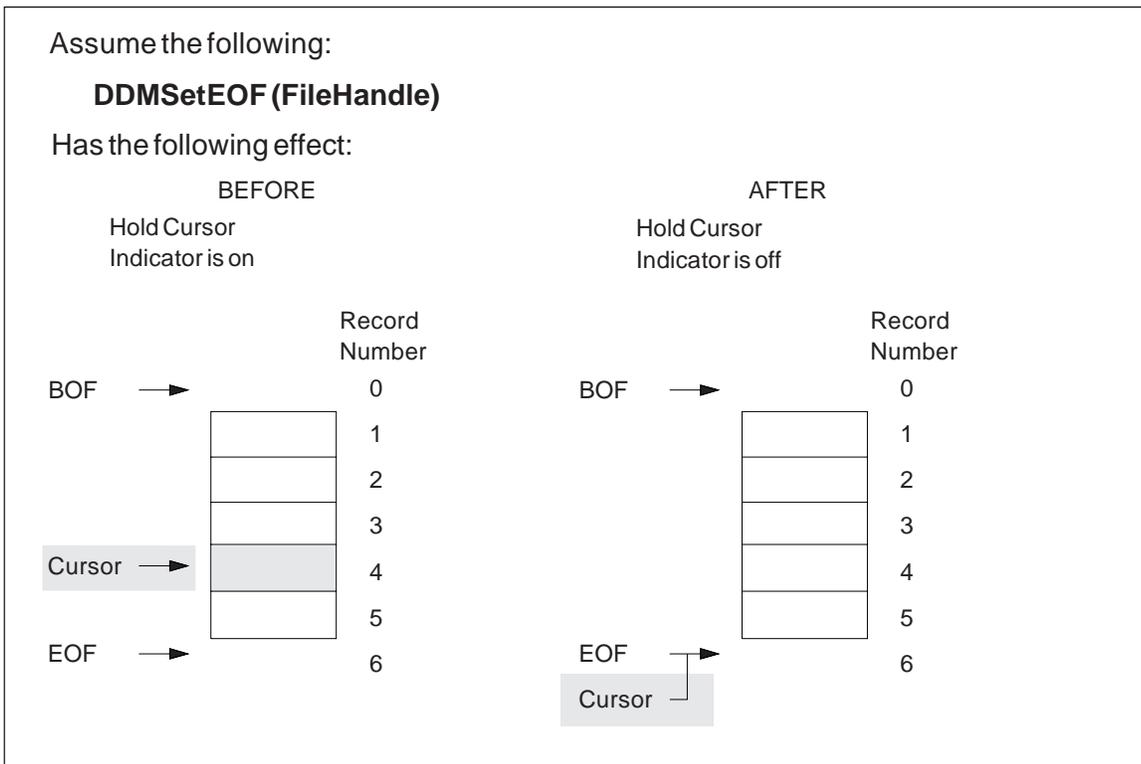


Figure 24. DDMSetEOF Function

DDMSetFileInfo

DDMSetFileInfo (Set File Information)

This function specifies information for a file or a directory. File information support is specific to the DDM server implementation and is dependent on the operating system.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetFileInfo (HDDMFILE      FileHandle,
                      ULONG          FileInfoLevel,
                      PBYTE         FileInfoBuf,
                      ULONG          FileInfoBufSize
                      );
```

Parameters

FileHandle

The handle (HDDMFILE) of the open file.

FileInfoLevel

The level (ULONG) of file/directory information being defined.

Level 0x00000001 information is the only defined level. This is the same as DosSetFileInfo, ulFileInfoLevel bit (FILE_STANDARD).

Level 0x00000001 file information sets a series of EA name/value pairs. On input, FileInfoBuf maps to an EAOP2 structure. fpGEA2List is ignored. fpFEA2List points to a data area where the relevant FEA2 list is to be found. oError is ignored.

On output, fpGEA2List is unchanged. fpFEA2List is unchanged as is the area pointed to by fpFEA2List. If an error occurred during the set, oError is the offset of the FEA2 where the error occurred. The return code is the error code corresponding to the condition generating the error. If no error occurred, oError is undefined.

FileInfoBuf

The pointer (PBYTE) to the storage area where the system gets the file information. Refer to "Extended Attributes" on page 5 for more information on the format of this buffer.

FileInfoBufSize

The length (ULONG) of FileInfoBuf.

DDMSetFileInfo

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
CMDCHKRM	X'1254'	Command Check
HDLNFNRM	X'1257'	File Handle Not Found
LENGTHRM	X'F211'	Field Length Error
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

This function is similar to the DosSetFileInfo command.

An example of requesting Extended Attributes (EAs) is provided on page 5.

Effect on Cursor Position

There is no effect on the cursor position.

Locking (for Local VSAM File System Only)

For the OS/2 local VSAM file system, the locking behaviour is the same as for DOSSetFileInfo. See *OS/2 WARP Control Program Programming Reference*.

For the AIX local VSAM file system, an exclusive lock is requested for the file.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is not invalid.	HDLNFNRM

Record File Attributes by File Class

These are modifiable record file attributes.

Refer to Table 14 on page 38.

When the FILINISZ EA is changed, it has no effect on the current space already allocated to the file.

When the DELCP EA of an alternate index file is changed, the DELCP of the base file and all other indexes is also changed.

When the GETCP EA of an alternate index file is changed, the GETCP of the base file and all other indexes are also changed.

When the INSCP EA of an alternate index file is changed, the INSCP of the base file and all other indexes are also changed.

When the MODCP EA of an alternate index file is changed, the MODCP of the base file and all other indexes are also changed.

DDMSetFirst

DDMSetFirst (Set Cursor to First Record)

This function sets the cursor to the first record of the file and optionally returns the record, record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetFirst (HDDMFILE      FileHandle,
                   ULONG          AccessFlags,
                   PDDMRECORD     RecordBuf,
                   ULONG          RecordBufLen
                   );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	DDM_ALLREC (All Records, Active and Inactive)
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats are found in “Examples” on page 153.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetFirst

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record In Use
RECNFNRM	X'1225'	Record Not Found

Remarks

The DDM_ALLREC bit flag is used to determine the first record of a file. If DDM_ALLREC is not set, the cursor is set to the first active record in the file. Otherwise the cursor is set to record 1 in the file. For direct files, DDM_ALLREC must be set off.

As an option, DDMSetFirst can:

- Set the hold cursor indicator (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Key limits are reset after completion of function.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetFirst

<i>Table 19. DDMSetFirst (DDM_NODATA or DDM_ALLREC) Decision Table</i>					
If the DDMSetFirst function is issued:					
When initial system states are:					
Record State	I	I	I	A	A
DDM_ALLREC	F	T	T	*	*
DDM_NODATA	*	F	T	F	T
The final system states are:	↓	↓	↓	↓	↓
RECINARM (returned)	F	F	T4	F	F
RECINA (returned)	F	T	F	F	F
RECORD (returned)	F	F	F	T	F
CURSOR (returned)	F	T	T	T	T
Repeat table after bypassing record	T	F	F	F	F
Legend					
A	Active				
I	Inactive				
T	TRUE (On)				
F	FALSE (Off)				
T4	TRUE with SVRCOD (Warning)				
*	Either TRUE or FALSE				

Effect on Cursor Position

Normal completion (SVRCOD of 0 or 4)

The cursor is moved to record number 1 if DDM_ALLREC is set. The cursor is moved to the first active record in the file if DDM_ALLREC is not set.

Error termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

DDMSetFirst

- The record is updated (DDMModifyRec, DDMDelRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnlockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, or DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with the RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes a Reply Message to be Generated and the Function Continues	With This Reply Message
DDM_ALLREC and DDM_NODATA are active and an inactive record is read.	RECINARM

This Causes the Function to be Terminated	With This Reply Message
Access flag DDM_NODATA is not set and the file was opened without GETAI.	INVRQSRM
The RecordBuf is not large enough to hold the returned record.	LENGTHRM

This Causes the Function to be Rejected	With This Reply Message
DDM_RECNRFB or DDM_KEYVALFB is set or DDM_NODATA is not set and RecordBuf doesn't contain an address.	ADDRRM
The file handle is not valid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM

DDMSetFirst

This Causes the Function to be Rejected	With This Reply Message
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified. DDM_ALLREC is set and the file is a direct file.	INVRQSRM
The record is damaged (not an active or inactive record).	RECDMGRM
A record lock cannot be obtained.	RECIUSRM
Bypassing inactive records is requested (DDM_ALLREC is off) and the file only contains inactive records. The file does not contain any records. Note: The cursor position is not changed.	RECNFNRM

DDMSetFirst

Examples

Assume the following:

```
AccessFlags = 0x00000010 ; /* DDM_ALLREC=ON */
```

DDMSetFirst (FileHandle, AccessFlags, RecordBuf, RecordBufLen)

Has the following effect:

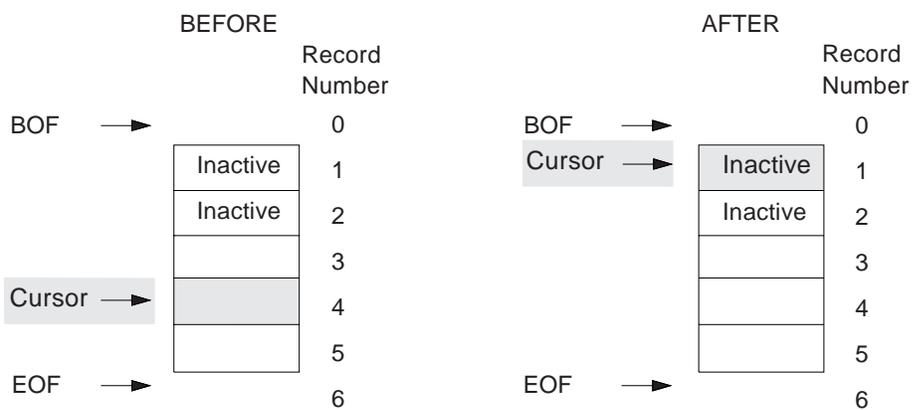


Figure 25. `DDMSetFirst` Function with `DDM_ALLREC` Set

DDMSetFirst

Assume the following:

```
AccessFlags = 0x00000000 ; /* DDM__ ALLREC=OFF */
```

DDMSetFirst (FileHandle, AccessFlags, RecordBuf, RecordBufLen)

Has the following effect:

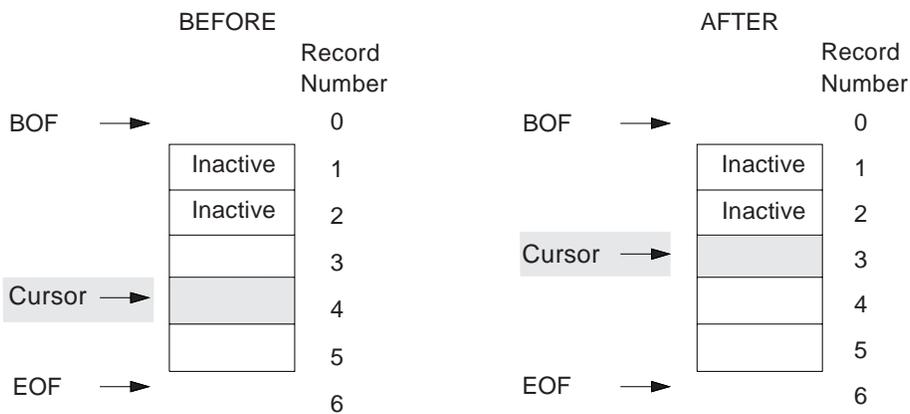


Figure 26. DDMSetFirst Function with DDM_ALLREC Not Set

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field

Description

LL

The length (ULONG) of the record buffer from the beginning of LL to the end of Data.

CP

The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.

X'144A'

Indicates that the following data is record data (RECORD).

DDMSetFirst

X'142D' Indicates that the following data is a ULONG length of an inactive record (RECINA).

Data Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
	X'144A' Indicates that the following data is record data (RECORD).
	X'142D' Indicates that the following data is a ULONG length of an inactive record (RECINA).
Data	Either record data or the length (ULONG) of the inactive record.

DDMSetFirst

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is a ULONG length of an inactive record (RECINA).

DDMSetFirst

Data Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.

DDMSetFirst

X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetKey (Set Cursor by Key)

This function positions the cursor based on the key value and relational operator specified, and optionally returns the record, record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKey (HDDMFILE      FileHandle,
                  ULONG         AccessFlags,
                  PDDMOBJECT    KeyValBuf,
                  CODEPOINT     RelOpr,
                  PDDMRECORD    RecordBuf,
                  ULONG         RecordBufLen,
                  );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

Specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on access flags, see Chapter 5, “VSAM API Flags” on page 399.

KeyValBuf

Pointer to the buffer which contains the key to which the cursor should be moved. The format of the key value buffer upon invocation of the function is:

LL	X'1115'	Key Value
----	---------	-----------

Field	Description
LL	The length (ULONG) of the key value description (from the beginning of LL to the end of Key Value).

DDMSetKey

X'1115' The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).

RelOpr

Specifies the relational test that should be used to test the specified key value against the file index key values. Valid values are:

- X'1445'** KEYAE (Key After or Equal)
Specifies that the relational test between the specified key value and the index key values is *after or equal to*. *After* is towards the end of file in the key sequence.
- X'1446'** KEYAF (Key After)
Specifies that the relational test between the specified key value and the index key values is *after*. *After* is towards the end of file in the key sequence.
- X'1447'** KEYEQ (Key Equal)
Specifies that the relational test between the specified key value and the index key values is *equal to*.
- X'144B'** KEYBE (Key Before or Equal)
Specifies that the relational test between the specified key value and the index key values is *before or equal to*. *Before* is towards the beginning of file in the key sequence.
- X'144C'** KEYBF (Key Before)
Specifies that the relational test between the specified key value and the index key values is *before*. *Before* is towards the beginning of file in the key sequence.

These values are described in detail on page 161.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in "Examples" on page 165.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetKey

Returns

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Access Method
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
DRCATHRM	X'1237'	Not Authorized to Directory
FILATHRM	X'123B'	Not Authorized to File
FILIUSRM	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
FILSNARM	X'120F'	File Space Not Available
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYLENRM	X'122D'	Invalid Key Length
LENGTHRM	X'F211'	Field Length Error
OBJNSPRM	X'1253'	Object Not Supported
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRMNSPRM	X'1251'	Parameter Not Supported
RECDMGRM	X'1249'	Record Damaged
RECNFNRM	X'1225'	Record Not Found
RSCLMTRM	X'1233'	Target Resource Limits Reached

Remarks

The cursor can be moved to the key value that is equal to, after, after or equal to, before, or before or equal to the specified key value. This function is only valid for keyed and alternate index files. The following list describes how this function sets the cursor for specific values for the RelOpr parameter.

Value	The Cursor Is Set by Key Sequence to:
KEYEQ	The first record in the file that has a key equal to the key specified in the key value buffer.
KEYAE	The first record in the file that has a key after or the last record in the file that has a key equal to the key specified in the key value buffer. If there is more than one record that has a key equal to the specified key, the cursor is set to the last record with an equal key. If there is no record with an equal key and there are multiple records that have a key equal to the next key in sequence, the cursor is set to the first of these records.
KEYAF	The first record in the file that has a key after the key specified in key value buffer.
KEYBE	The first record of the file that has a key equal to the key specified in key value buffer. If no equal key is found, the cursor, by key sequence, is set to the last record of the file with a key before the key specified in key value buffer.
KEYBF	The last record in the file with a key before the key specified in key value buffer.

DDMSetKey

If the key value specified in key value buffer is shorter than the file record keys, a generic search is performed. Only the first record of all records satisfying the generic search can be accessed with this function. DDMSetKeyNext can be used to access additional records that satisfied the generic search.

If the key value specified in key value buffer has duplicate entries in the file (duplicate keys), only the first or last record, depending upon the value of RelOpr, of all records having the duplicate key value can be accessed with this function. See “DDMSetKeyNext (Set Cursor to Next Record in Key Sequence)” on page 202 or “DDMSetKeyPrevious (Set Cursor to Previous Record in Key Sequence)” on page 220 for accessing additional records with the same key value.

As an option, DDMSetKey can:

- Set the hold cursor indicator (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Effect on Cursor Position

Normal completion (SVRCOD of 0 or 4)

The cursor is moved to the record that satisfies the relational operator specification.

Error termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.

DDMSetKey

- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, or DDMSetUpdateNum functions).

If the record lock is not obtained, the DDMSetKey function is rejected with the RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetKey

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is not invalid.	HDLNFNRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The key length specified for KEYVAL is larger than the key length used to build the index. Note: The cursor position is not changed.	KEYLENRM
The file does not contain any records or a record does not exist that satisfies RelOpr. Note: The cursor position is not changed.	RECNFNRM

Examples

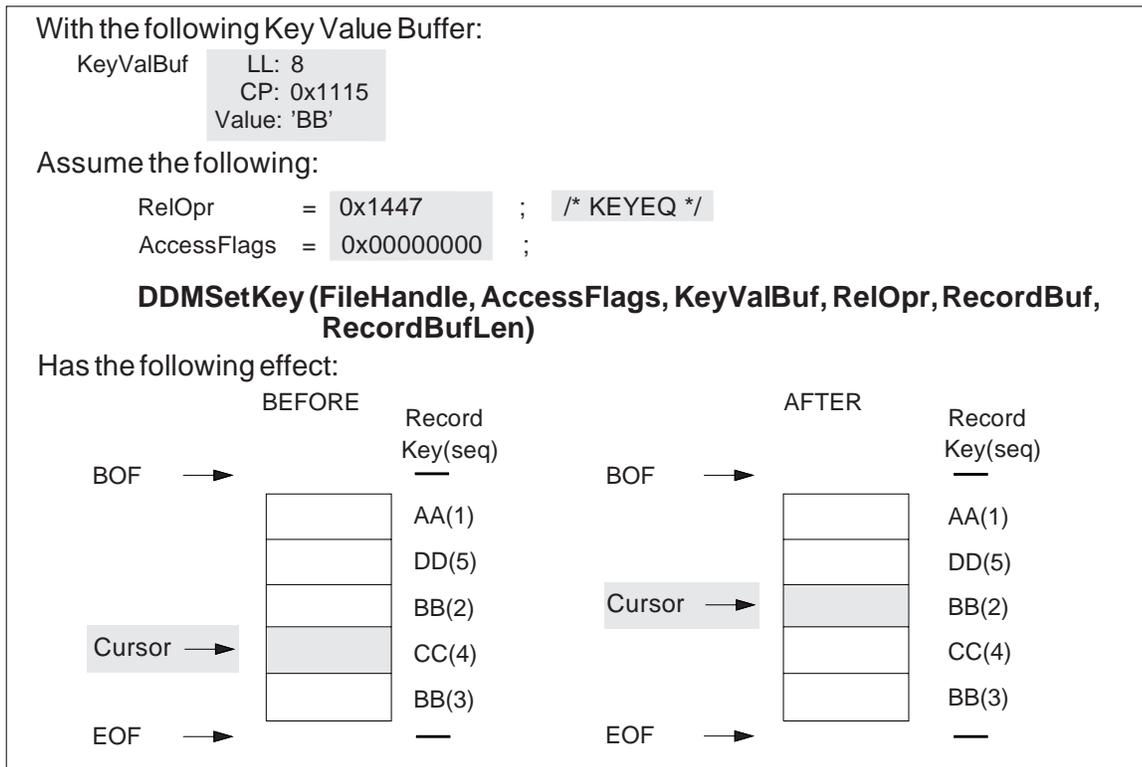


Figure 27. DDMSetKey Function with RelOpr Set to KEYEQ

DDMSetKey

With the following Key Value Buffer:

```
KeyValBuf  LL: 8
           CP: 0x1115
           Value: 'AD'
```

Assume the following:

```
RelOpr     = 0x1445 ; /* KEYAE */
AccessFlags = 0x00000000 ;
```

DDMSetKey (FileHandle, AccessFlags, KeyValBuf, RelOpr, RecordBuf, RecordBufLen)

Has the following effect:

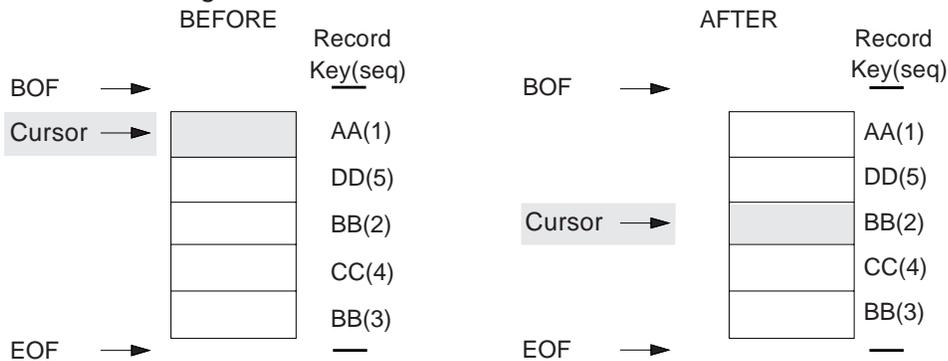


Figure 28. `DDMSetKey` Function with `RelOpr` Set to `KEYAE`

DDMSetKey

With the following Key Value Buffer:

```
KeyValBuf  LL: 8  
           CP: 0x1115  
           Value: 'CA'
```

Assume the following:

```
RelOpr     = 0x1446 ; /* KEYAF */  
AccessFlags = 0x00000000 ;
```

DDMSetKey (FileHandle, AccessFlags, KeyValBuf, RelOpr, RecordBuf, RecordBufLen)

Has the following effect:

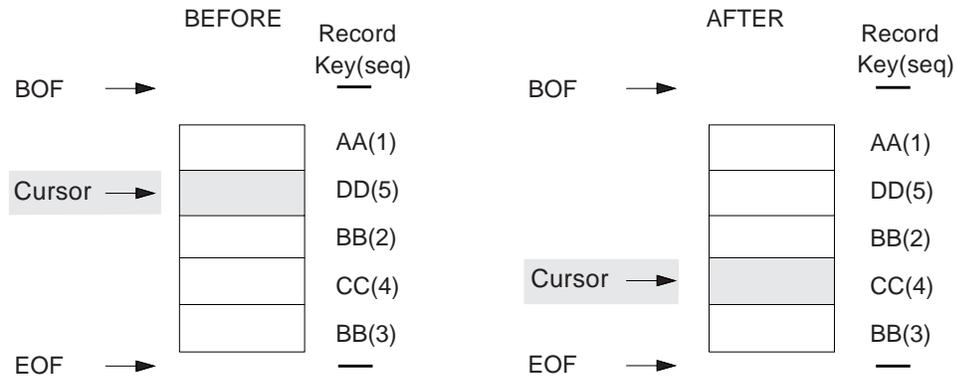


Figure 29. DDMSetKey Function with RelOpr Set to KEYAF

DDMSetKey

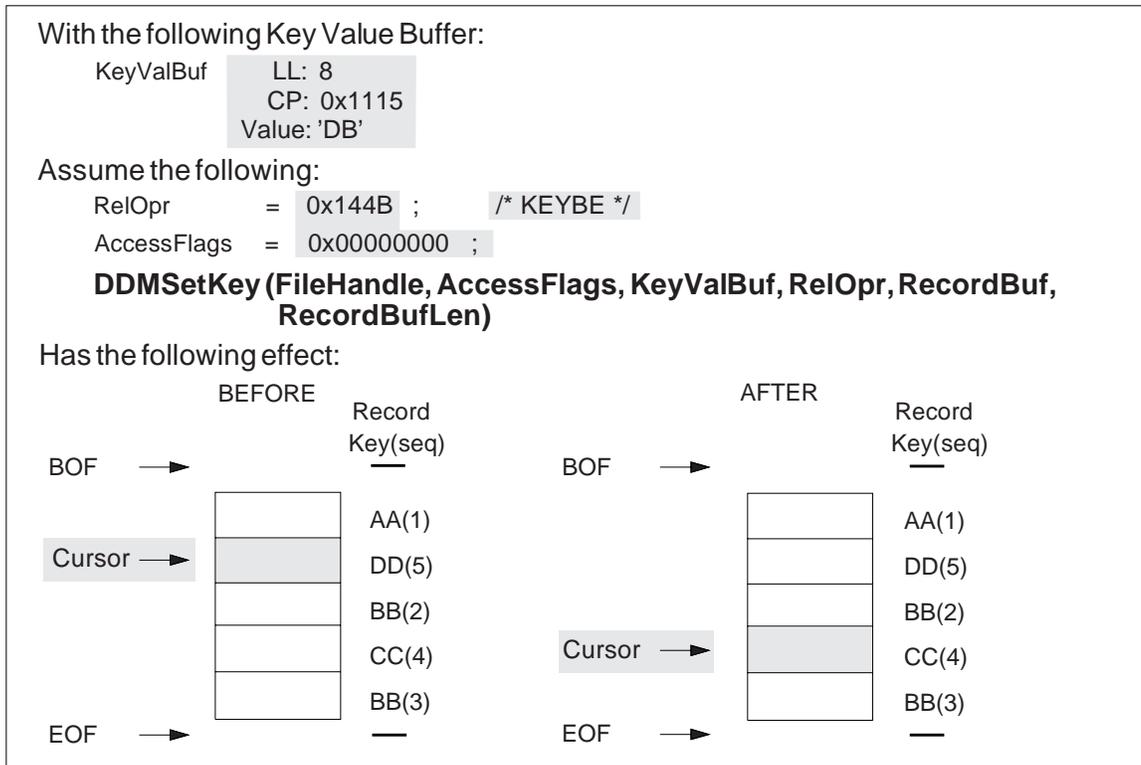


Figure 30. DDMSetKey Function with RelOpr Set to KEYBE

DDMSetKey

With the following Key Value Buffer:

```
KeyValBuf  LL: 8
           CP: 0x1115
           Value: 'BB'
```

Assume the following:

```
RelOpr     = 0x1446 ; /* KEYAE */
AccessFlags = 0x00000000 ;
```

DDMSetKey (FileHandle, AccessFlags, KeyValBuf, RelOpr, RecordBuf, RecordBufLen)

Has the following effect:

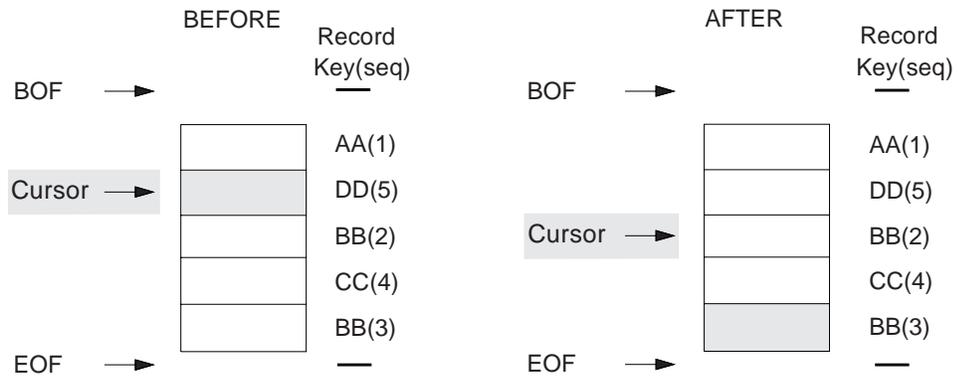


Figure 31. DDMSetKey Function with RelOpr Set to KEYAE

DDMSetKey

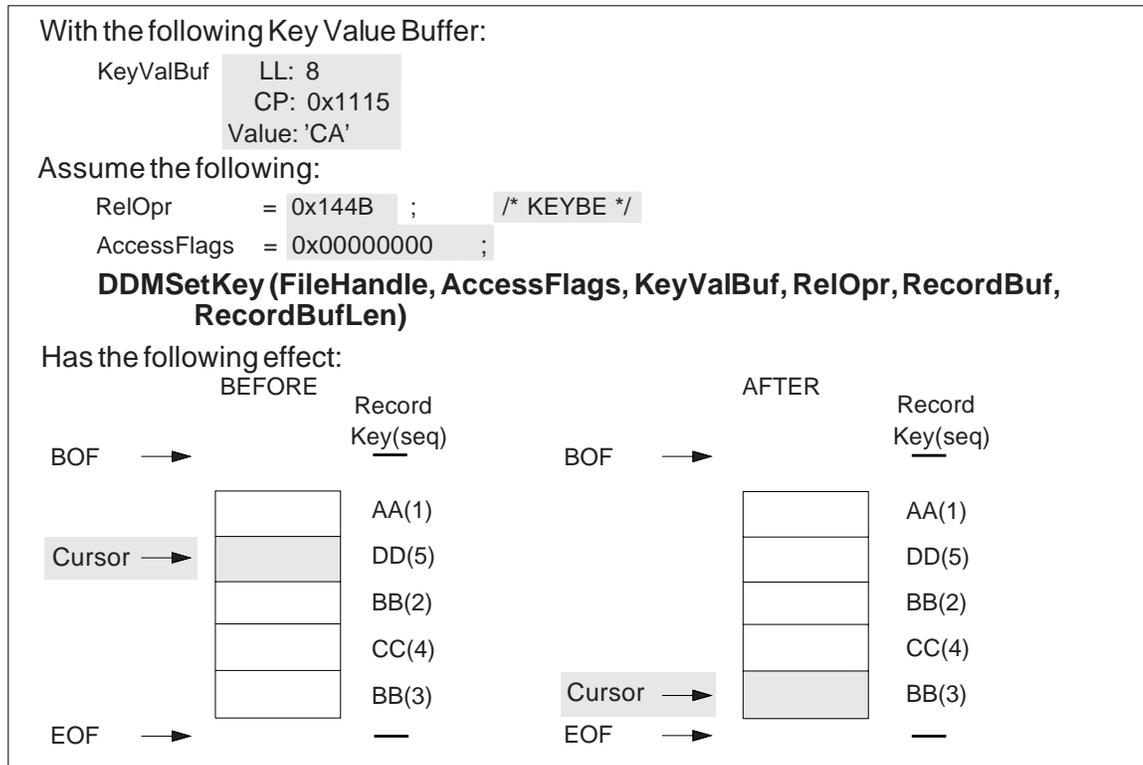


Figure 32. DDMSetKey Function with RelOpr Set to KEYBE

DDMSetKey

With the following Key Value Buffer:

```
KeyValBuf  LL: 8  
           CP: 0x1115  
           Value: 'BB'
```

Assume the following:

```
RelOpr     = 0x144B ; /* KEYBE */  
AccessFlags = 0x00000000 ;
```

DDMSetKey (FileHandle, AccessFlags, KeyValBuf, RelOpr, RecordBuf, RecordBufLen)

Has the following effect:

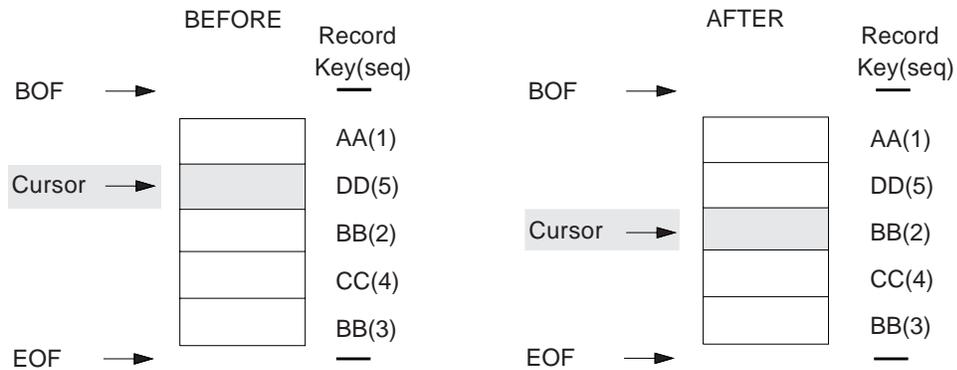


Figure 33. DDMSetKey Function with RelOpr Set to KEYBE

DDMSetKey

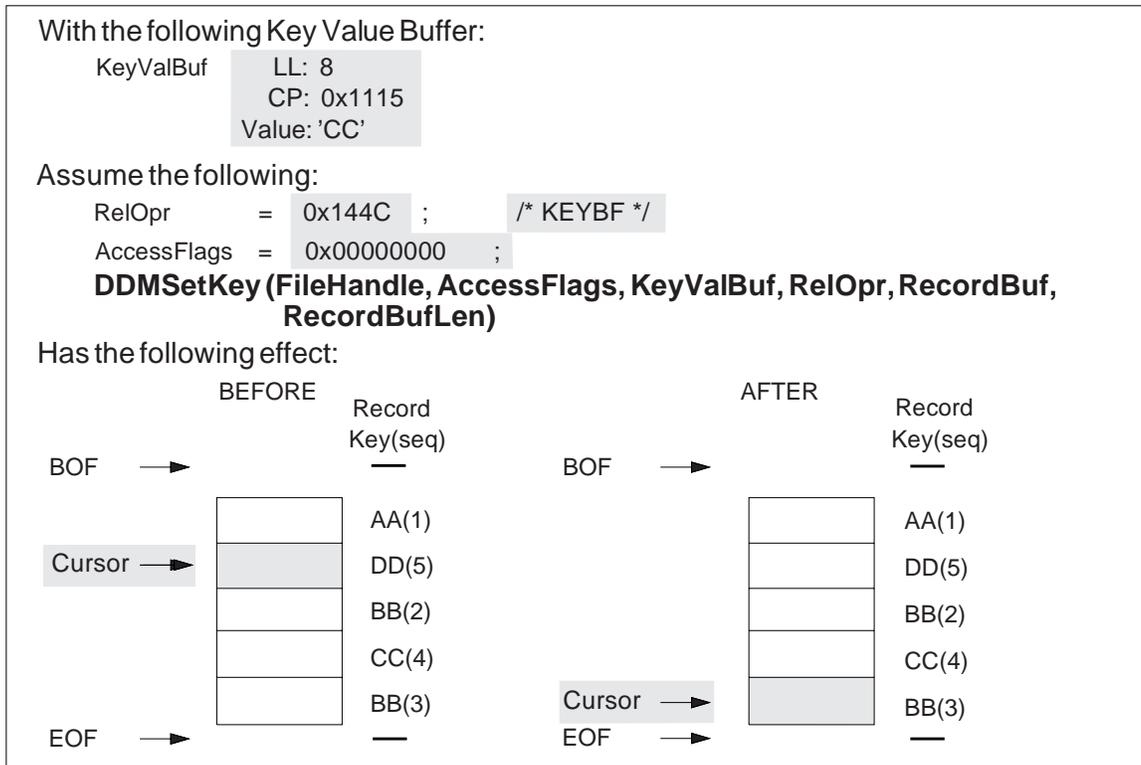


Figure 34. DDMSetKey Function with RelOpr Set to KEYBF

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKey

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

DDMSetKey

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

DDMSetKey

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKey

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetKeyFirst

DDMSetKeyFirst (Set Cursor to First Record in Key Sequence)

This function sets the cursor to the first record in key sequence and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKeyFirst (HDDMFILE      FileHandle,
                      ULONG          AccessFlags,
                      PDDMRECORD     RecordBuf,
                      ULONG          RecordBufLen
                      );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
6–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 179.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetKeyFirst

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
FILATHRM	X'123B'	Not Authorized to File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
KEYLENRM	X'122D'	Invalid Key Length
LENGTHRM	X'F211'	Field Length Error
OBJNSPRM	X'1253'	Object Not Supported
RECDMGRM	X'1249'	Record Damaged
RECIUSRM	X'124A'	Record in Use
RECNFNRM	X'1225'	Record Not Found

Remarks

As an option, DDMSetKeyFirst can:

- Set the hold cursor indicator (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECnbrFB).
- Place an update intent on the record (DDM_UPDINT).

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the first record according to the index key sequence.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

DDMSetKeyFirst does the following:

- If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:
 - The record is updated (DDMModifyRec or DDMDeleteRec).
 - The cursor is moved to a different record.

DDMSetKeyFirst

- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function is issued that references a record other than the one currently pointed to by the cursor (for example, the DDMLInsertRecEOF, DDMLInsertRecKey, DDMLInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).
- If the record lock is not obtained, the function is rejected with the RECIUSRM reply message.
- If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.
- If the function terminates with a reply message that has a severity code of ERROR or higher, then:
 - For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
 - For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is not invalid.	HDLNFNRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The record lock cannot be obtained.	RECIUSRM
The file does not contain any records. Note: The cursor position is not changed.	REC�FNRM

Examples

DDMSetKeyFirst

Assume the following:

```
AccessFlags = 0x00000000 ;
```

DDMSetKeyFirst (FileHandle, AccessFlags, RecordBuf, RecordBufLen)

Has the following effect:

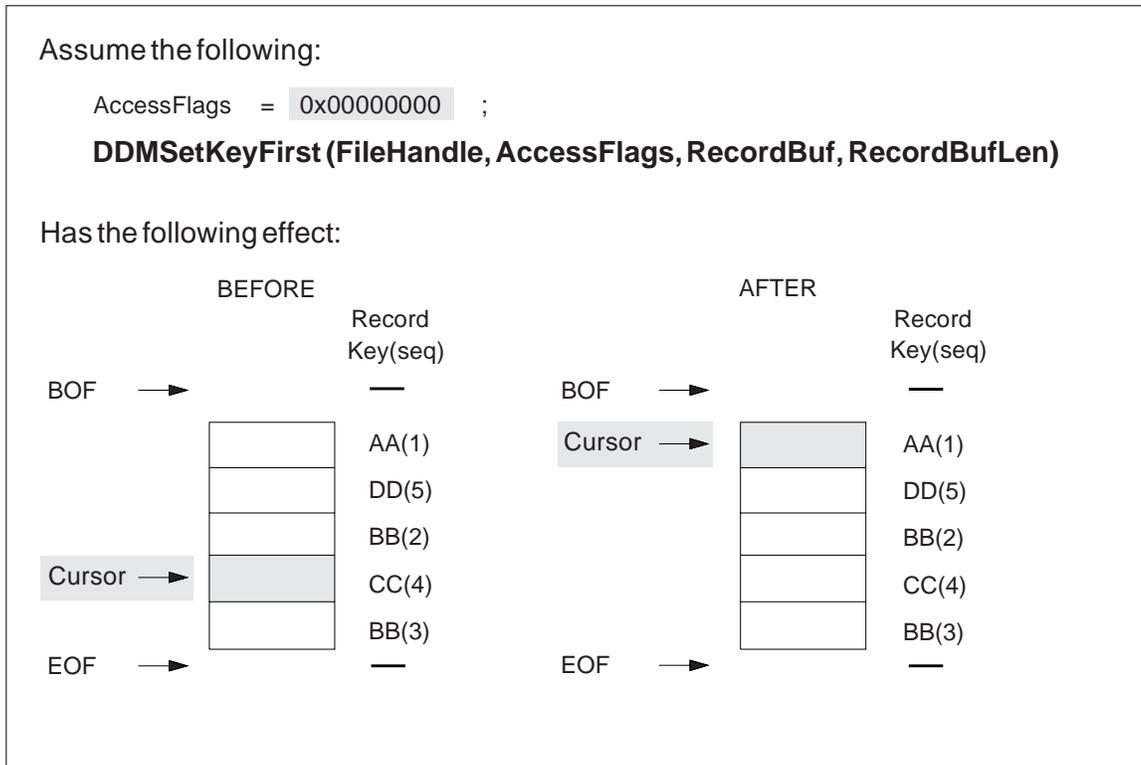


Figure 35. `DDMSetKeyFirst` Function for Ascending Sequence

DDMSetKeyFirst

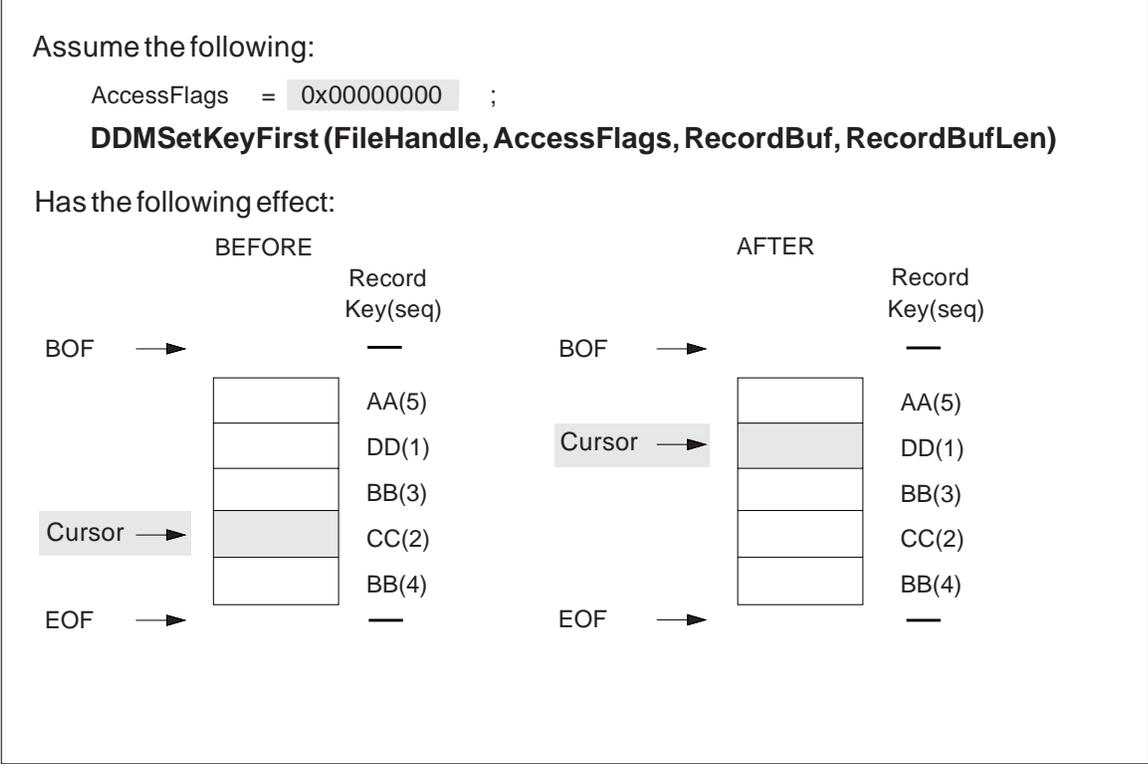


Figure 36. DDMSetKeyFirst Function for Descending Sequence

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKeyFirst

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

DDMSetKeyFirst

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

DDMSetKeyFirst

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list (from the beginning of LL to the end of Data).
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKeyFirst

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list (from the beginning of LL to the end of KEY).
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetKeyLast

DDMSetKeyLast (Set Cursor to Last Record in Key Sequence)

This function sets the cursor to the last record of the file in key sequence order and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKeyLast (HDDMFILE      FileHandle,
                     ULONG          AccessFlags,
                     PDDMRECORD    RecordBuf,
                     ULONG          RecordBufLen
                     );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
3–4	Reserved flags
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 189.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetKeyLast

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	address error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
ENDFILRM	X'120B'	End of File
FILATHRM	X'123B'	Not Authorized to File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECIUSRM	X'124A'	Record in Use
RECNFNRM	X'1225'	Record Not Found

Remarks

If the file permits duplicate keys and the last record in the file has a duplicate key, the cursor is set to the last record of the duplicates in key sequence.

As an option, DDMSetKeyLast can:

- Set the hold cursor indicator on (DDM_HLDCSR).
- Return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the last record in the index key sequence.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record, if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (for example, DDMMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnlockRec function is issued.

DDMSetKeyLast

- Any function is issued that references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with the RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is invalid.	HDLNFNRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The record lock cannot be obtained.	RECIUSRM
The file does not contain any records. Note: The cursor position is not changed.	RECNFNRM

DDMSetKeyLast

Examples

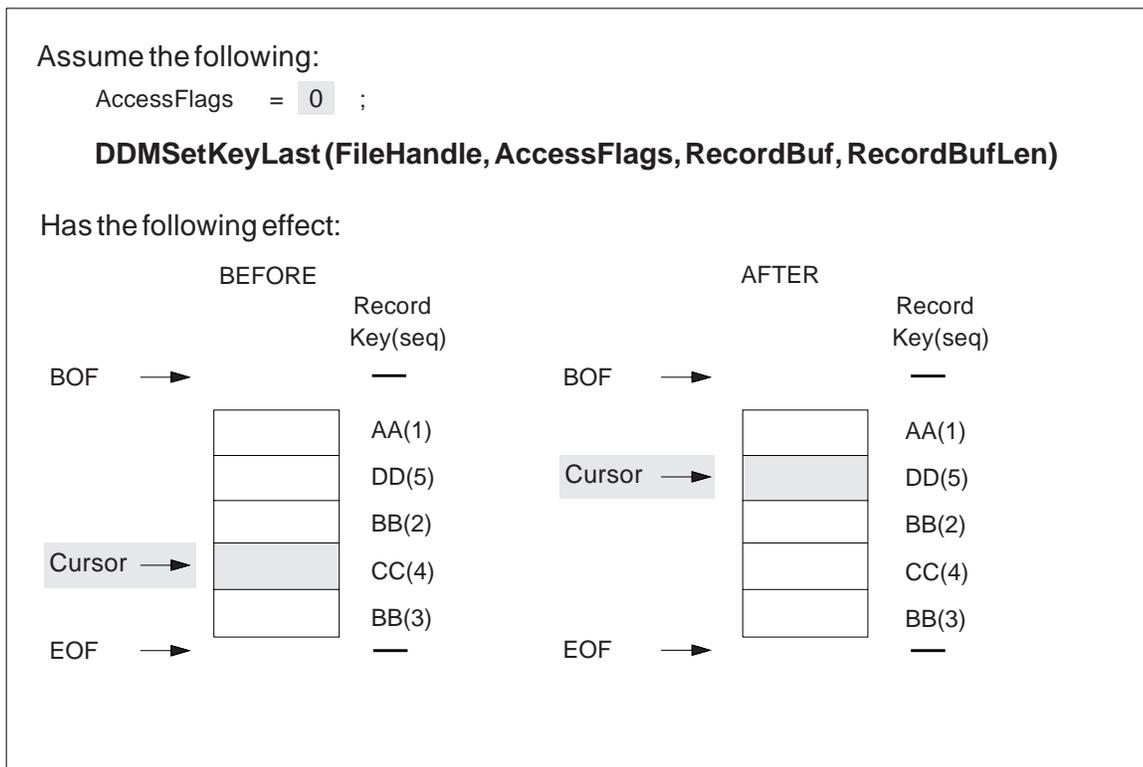


Figure 37. DDMSetKeyLast Function for Ascending Sequence

DDMSetKeyLast

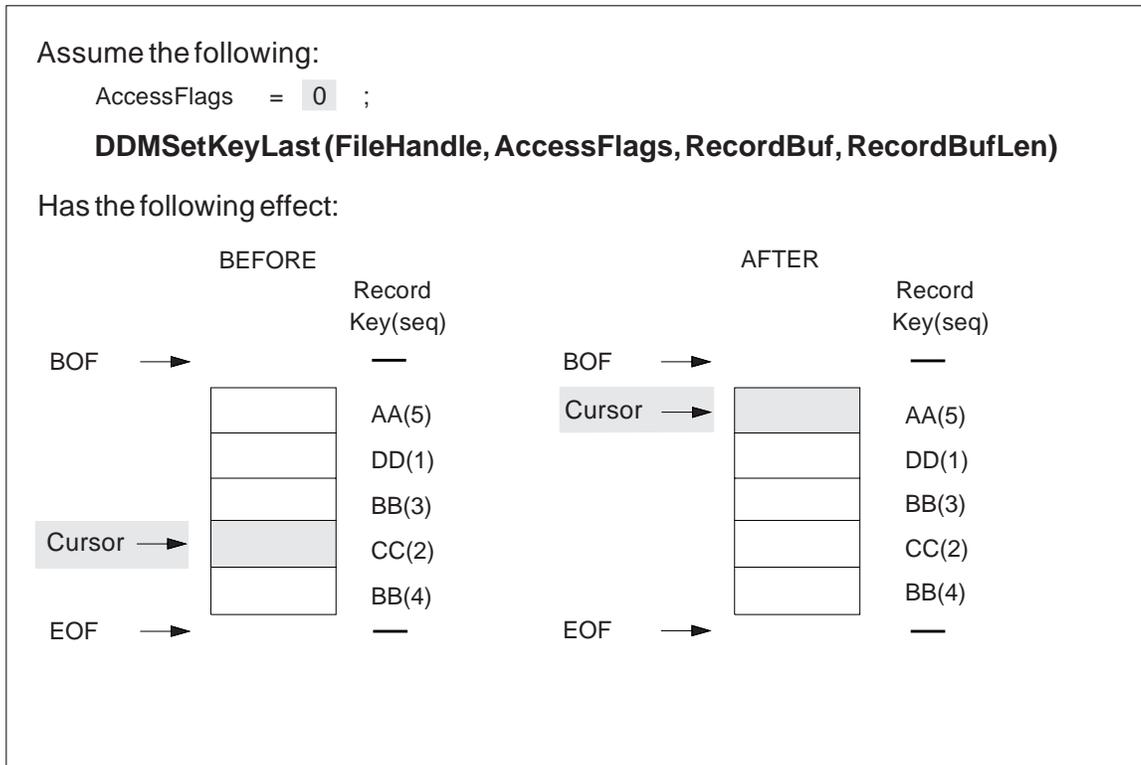


Figure 38. DDMSetKeyLast Function for Descending Sequence

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field Description

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKeyLast

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

DDMSetKeyLast

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

DDMSetKeyLast

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetKeyLast

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetKeyLimits

DDMSetKeyLimits (Set Key Limits)

This function sets the limits of the key values for subsequent DDMSetKeyNext and DDMSetNextKeyEqual functions.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKeyLimits (HDDMFILE      FileHandle,
                       PDDMOBJECT    LowKeyLim,
                       PDDMOBJECT    HiKeyLim
                       );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

LowKeyLim

The pointer (PDDMOBJECT) to the key buffer for the lower key value limit. The format of the low key limit buffer upon invocation of the function is:

LL	X'1130'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the key value description from the beginning of LL to the end of the key value. This field may be set to 6 and no key value need be provided. This has the special meaning of first key value of the file.
X'1130'	The value (CODEPOINT) indicating that the following data is a key value, representing a low key limit.
Data	The key value (BYTE) for a record. The key value can be a maximum of 255 bytes.

HiKeyLim

The pointer (PDDMOBJECT) to the key buffer for the higher key value limit. The format of the high key limit buffer upon invocation of the function is:

LL	X'112F'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the key value description from the beginning of LL to the end of the key value. This field may be set to 6 and no key value need be provided. This has the special meaning of last key value of the file.

DDMSetKeyLimits

X'112F'	The value (CODEPOINT) indicating that the following data is a key value, representing a high key limit.
Data	The key value (BYTE) for a record. The key value can be a maximum of 255 bytes.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVRQSRM	X'123C'	Invalid Request
KEYLENRM	X'122D'	Invalid Key Length
OBJNSPRM	X'1253'	Object Not Supported

Remarks

The DDMSetKeyLimits function is only valid for files with ascending keys.

The DDMSetKeyLimits function:

- Establishes the key limits and associates them with the active cursor.
- Sets the cursor to the record position of the lower limit or the first key after the low key limit if it is not in the file.
- Sets the hold cursor indicator to the on value so the first DDMSetKeyNext or DDMSetNextKeyEqual function remains at the first record within the limits.

When key limits have been established, the DDMSetKeyNext or DDMSetNextKeyEqual function only operates within the defined limits.

DDMSetKeyNext or DDMSetNextKeyEqual sets the cursor, in key sequence, to the next record that is within the bounds of the key limits. If the cursor is already positioned at the highest key limit, the function is terminated with the ENDFILRM reply message, the key limits are reset, and the cursor is set to the EOF position.

The key limits remain in effect until one of the following occurs:

- The file is closed by a DDMClose function or termination of communications.
- A cursor positioning function other than DDMSetKeyNext or DDMSetNextKeyEqual is performed. This includes the following functions:

- DDMSetBOF
- DDMSetEOF
- DDMSetFirst
- DDMSetKey
- DDMSetKeyFirst
- DDMSetKeyLast
- DDMSetKeyPrevious
- DDMSetLast
- DDMSetMinus

DDMSetKeyLimits

DDMSetRecNum
DDMSetNextRec
DDMSetPrevious

- A DDMInsertRecxxx function with the DDM_UPDCSR bit in the AccessFlags set on is performed.
- An ENDFILRM reply message is returned from a DDMSetKeyNext or DDMSetNextKeyEqual function.
- A DDMSetKeyLimits function specifies new limits.

When the key limits are reset, they are logically reset with a low key limit value of beginning of file and high key limit value of end of file. The cursor is not directly affected by resetting the key limits, but its position may be changed by the function that resets the key limits.

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the first record in the file with a key value equal to or greater than the low key limit (LowKeyLim) in the index key sequence. If an ENDFILRM reply message results, the cursor is set to the end of file.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters and the requester currently has a SHRRECLK lock on a record in the file, the SHRRECLK lock is released.

If the DDMSetKeyLimits function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetKeyLimits

Exceptions

This Causes	This Reply Message to be Returned
The LowKeyLim specified is after the last key. The HiKeyLim specified is before the first key.	ENDFILRM

This Causes the Function to be Rejected	With This Reply Message
The file handle is not invalid.	HDLNFNRM
The HiKeyLim specifies a key value that is before the LowKeyLim. The file was created with a key (or composite key) whose parts are not all ascending.	INVRQSRM
Either the HiKeyLim or LowKeyLim parameter specifies a partial key.	KEYLENRM

DDMSetKeyLimits

Examples

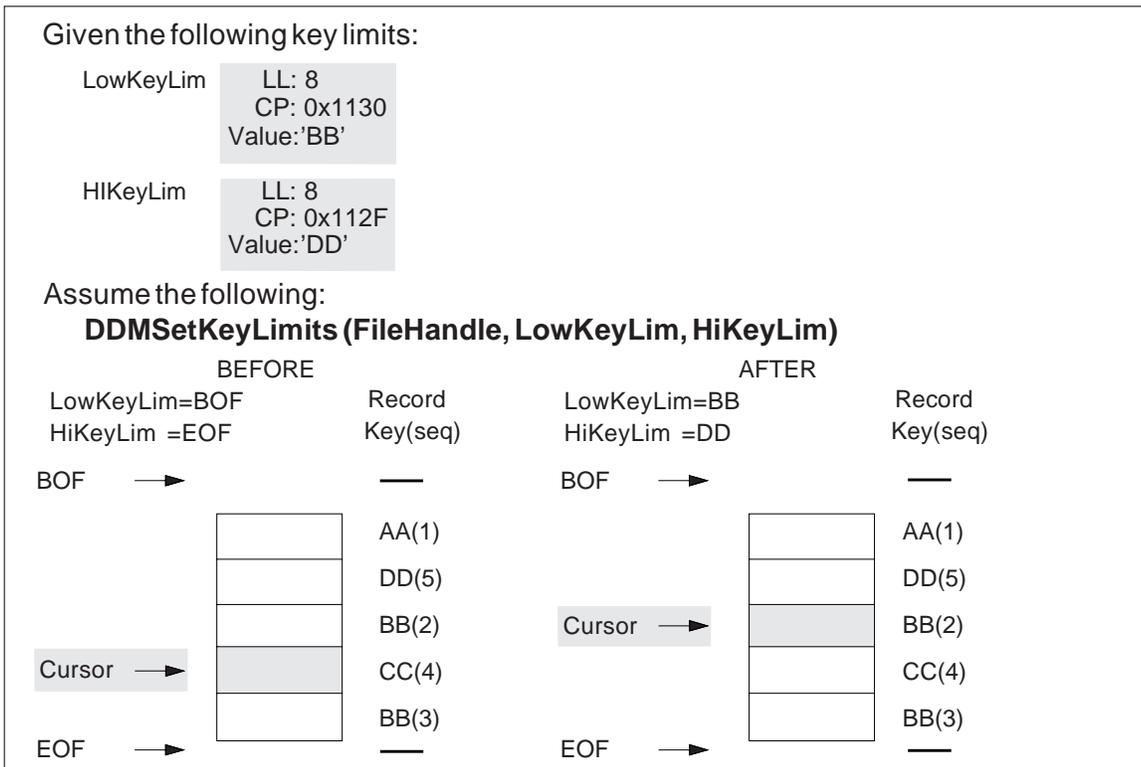


Figure 39. DDMSetKeyLimits Function

DDMSetKeyLimits

Assume the following:

```
AccessFlags = 0 ;
RecCount    = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

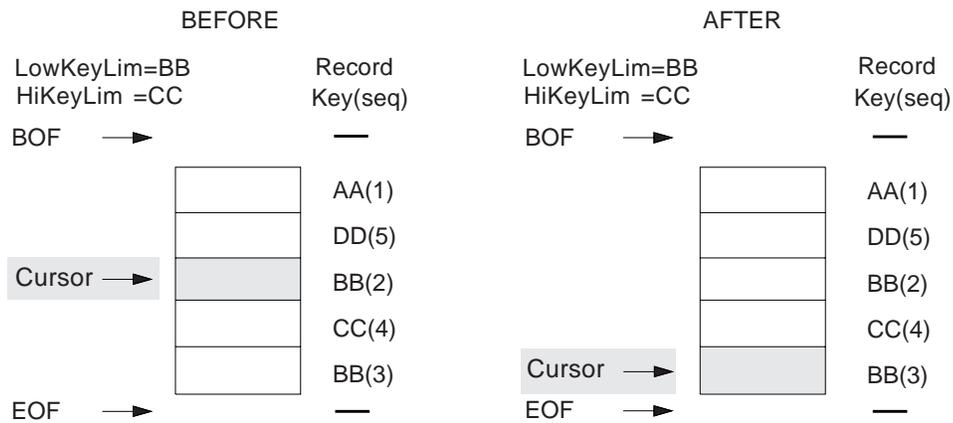


Figure 40. DDMSetKeyNext Function with Key Limits Set

DDMSetKeyLimits

Given the following key value buffer:

```
KeyValBuf    LL: 8
              CP: 0x1115
              Value: 'AA'
```

Assume the following:

```
DDMSetKey (FileHandle, AccessFlags, KeyValBuf, RelOpr, RecordBuf,
           RecordBufLen)
```

```
RelOpr      = 0x1447 ; /* KEYEQ */
```

```
AccessFlags = 0 ;
```

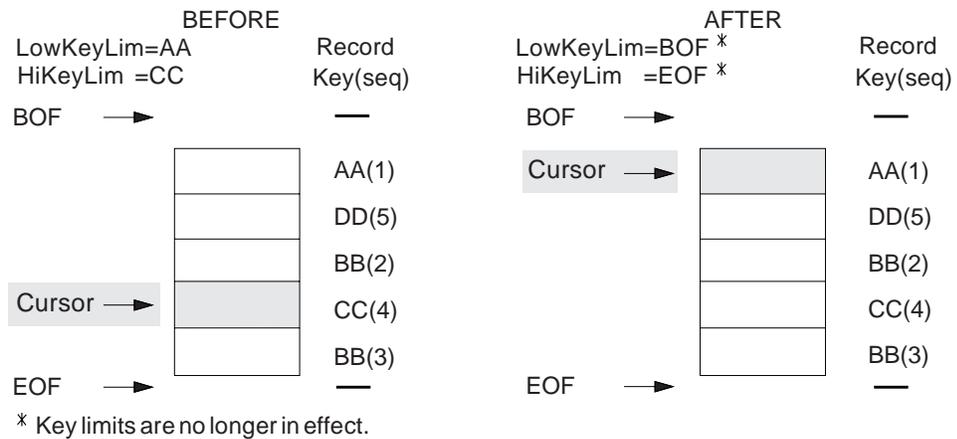


Figure 41. Resetting Limits with DDMSetKey Function

DDMSetKeyNext

DDMSetKeyNext (Set Cursor to Next Record in Key Sequence)

This function moves the cursor to the next record of the file in key sequence order and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKeyNext (HDDMFILE      FileHandle,
                     ULONG          AccessFlags,
                     PDDMRECORD    RecordBuf,
                     ULONG          RecordBufLen,
                     ULONG          RecCount,
                     PULONG         RecRtnCnt
                     );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	DDM_BYPDMG (Bypass Damaged Record)
5	DDM_NODATA (No Record Data Returned)
3–4	Reserved flags
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 207.

RecordBufLen

The length (ULONG) of the record buffer.

RecCount

Specifies the number (ULONG) of records requested.

DDMSetKeyNext

RecRtnCnt

The pointer (PULONG) to the count of the records actually returned. When RECAL (Record Attribute List) parameters are specified in RecordBuf and RECCNT is specified within the RECAL, the RecRtnCnt parameter (ULONG) reflects the RECCNT number of duplicate records. Therefore, if RecordBuf contained 25 data records, one of which included a RECAL with RECCNT having a value of 150, the value of RecRtnCnt would be 175.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

The cursor is set to the next record in key sequence even if that record has a key equal to the key of the current record.

If key limits have been established (see “DDMSetKeyLimits (Set Key Limits)” on page 195), DDMSetKeyNext sets the cursor to the next record in key sequence, as long as that record has a key value which is before or equal to the value specified by the high key limit parameter on the DDMSetKeyLimits function. If the cursor is currently at the high key limit, the function is terminated with the ENDFILRM reply message, the cursor is set to EOF, and the key limits are reset (unspecified value).

As an option, DDMSetKeyNext can:

- Specify whether more than one record is being requested (RecCount).
- Set the hold cursor indicator to on (DDM_HLDCSR).
- Specify whether damaged records should be bypassed (DDM_BYPDGM).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

If the hold cursor indicator in the cursor is set to on, the DDM_HLDCSR bit in AccessFlags is FALSE, and the record is active, the cursor remains at its current position. For all other conditions, the cursor is updated.

If RecCount specifies a value greater than 1, multiple records are sent to the source agent. RecCount specifies the number of times the DDMSetKeyNext function is to be performed. This moves the cursor to the last record processed by the DDMSetKeyNext

DDMSetKeyNext

function. If RecCount specifies a number and DDM_NODATA is set, the cursor is still updated but no records are sent; this is not an error.

If RecCount specifies a number greater than the remaining records in the file, the remaining records are sent to the source agent, the cursor position is changed to EOF, and an ENDFILRM reply message is sent.

If the DDM_BYPDGM bit of AccessFlags is set, any damaged record encountered by the DDMSetKeyNext function sends a RECDMGRM reply message, updates the cursor, and decreases RecCount by one. This allows the maximum number of undamaged records to be sent to the source system.

Effect on Cursor Position

Normal completion (SVRCOD of 0 or 4)

The cursor is moved to the next record in the index key sequence or remains in the same position in the current record based on:

- The hold cursor indicator in the cursor
- The DDM_HLDCSR flag
- Whether the record is active.

If the ENDFILRM reply message results, the cursor is set to the end of file.

Error termination (SVRCOD of 8)

The cursor position is the same as before the function was issued. If RecCount is greater than 1, the cursor position is the same as before the last iteration of the function.

Severe termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, then the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (for example, DDMMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnlockRec function is issued.

DDMSetKeyNext

- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued, or if RecCount is greater than 1, the record locks are the same as before the last iteration of the function.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetKeyNext

Exceptions

This Causes the Function to be Rejected	With this Reply Message
<p>The file does not contain any records initially after a DDMCreateRecFile.</p> <p>The file does not contain any records beyond the current cursor position.</p> <p>The cursor had previously been set to an inactive record.</p> <p>The file does not contain any records beyond the current cursor position, within the limits set by the DDMSetKeyLimits function.</p> <p>RecCount specifies a number greater than the number of records remaining in the file.</p>	ENDFILRM
<p>The file handle is invalid.</p>	HDLNFNRM
<p>DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.</p>	INVRQSRM
<p>The record lock cannot be obtained.</p>	RECIUSRM

DDMSetKeyNext

Examples

Assume the following:

```

AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
RecCount = 1 ;

```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

BEFORE		AFTER	
Hold Cursor indicator is off	Record Key(seq)	Hold Cursor indicator is off	Record Key(seq)
BOF →	—	BOF →	—
	AA(1)		AA(1)
	DD(5)		DD(5)
Cursor →	BB(2)		BB(2)
	CC(4)		CC(4)
	BB(3)	Cursor →	BB(3)
EOF →	—	EOF →	—

Figure 42. DDMSetKeyNext Function with Duplicate Key Values

DDMSetKeyNext

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
RecCount   = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

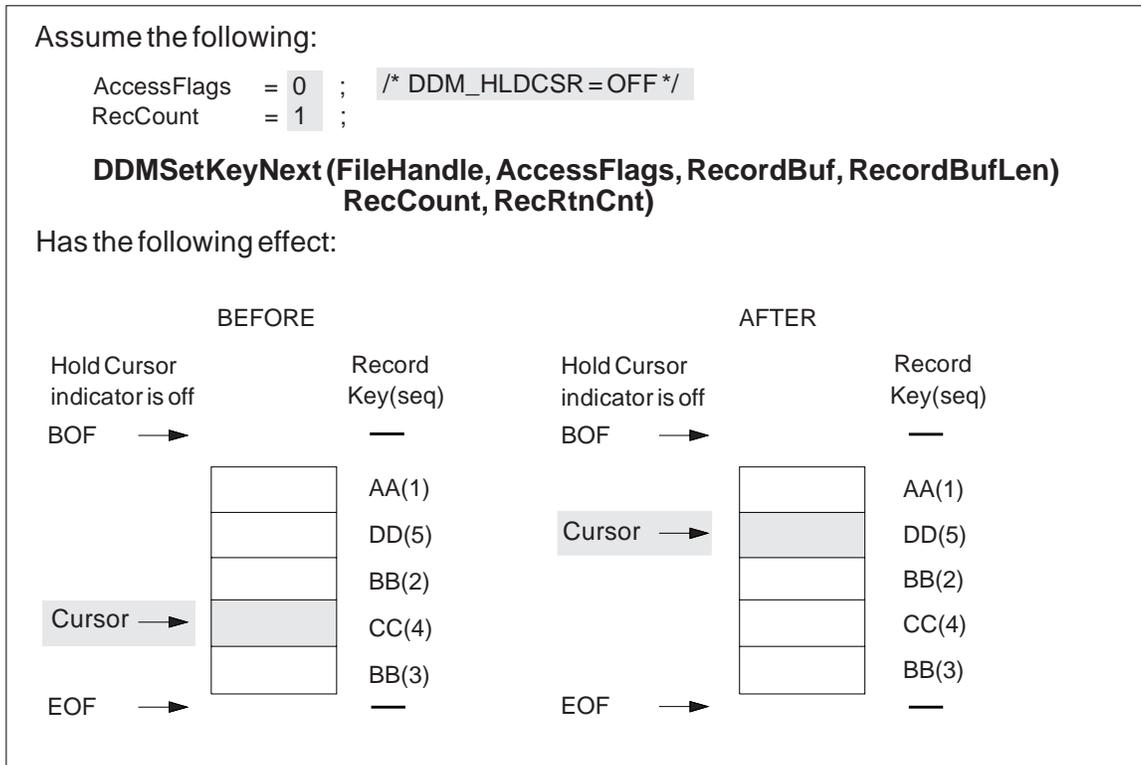


Figure 43. DDMSetKeyNext Function for Ascending Sequence

DDMSetKeyNext

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
RecCount   = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

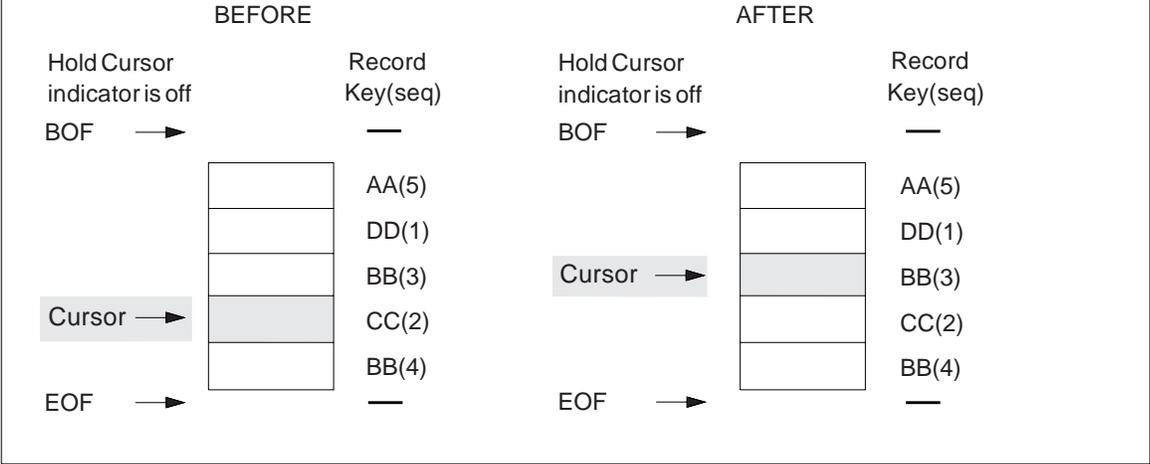


Figure 44. DDMSetKeyNext Function for Descending Sequence

DDMSetKeyNext

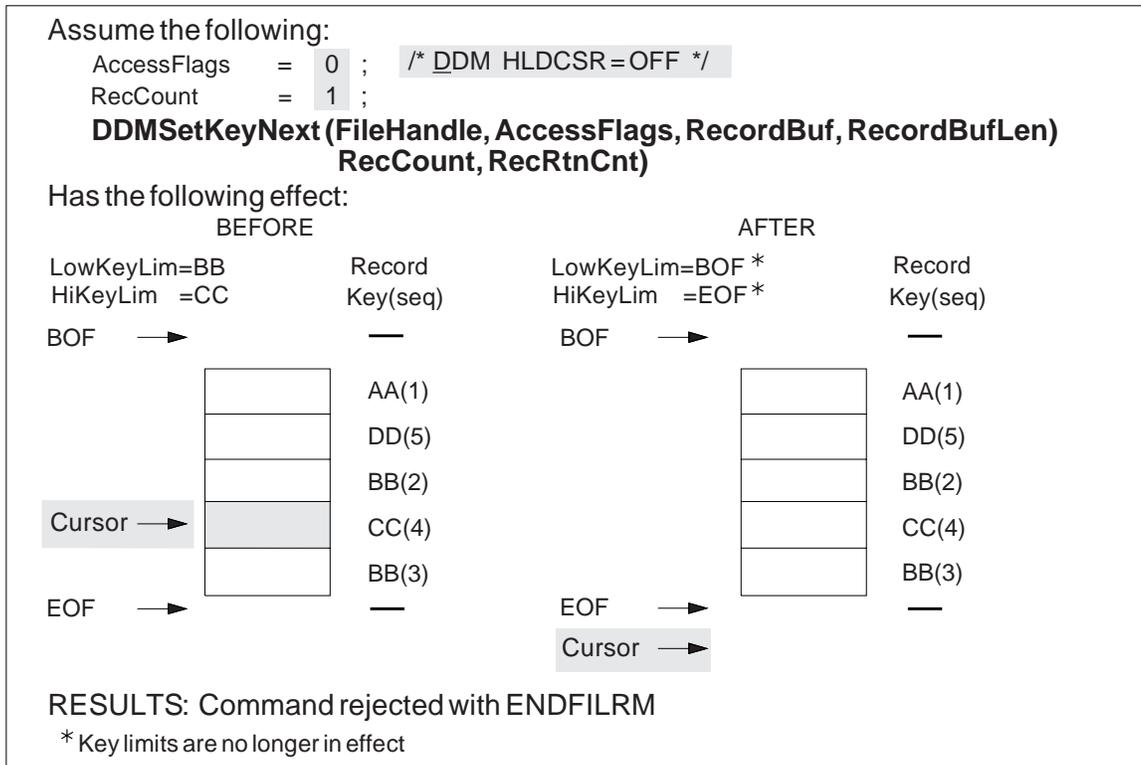


Figure 45. DDMSetKeyNext Function with Key Limits Set

DDMSetKeyNext

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
RecCount = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

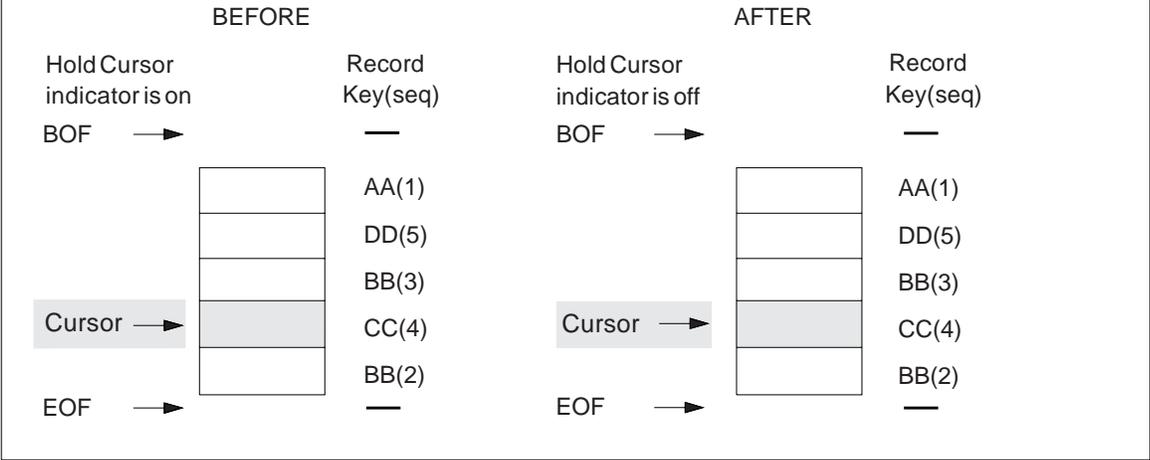


Figure 46. DDMSetKeyNext Function with Hold Cursor Initially On

DDMSetKeyNext

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR = ON */
RecCount   = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

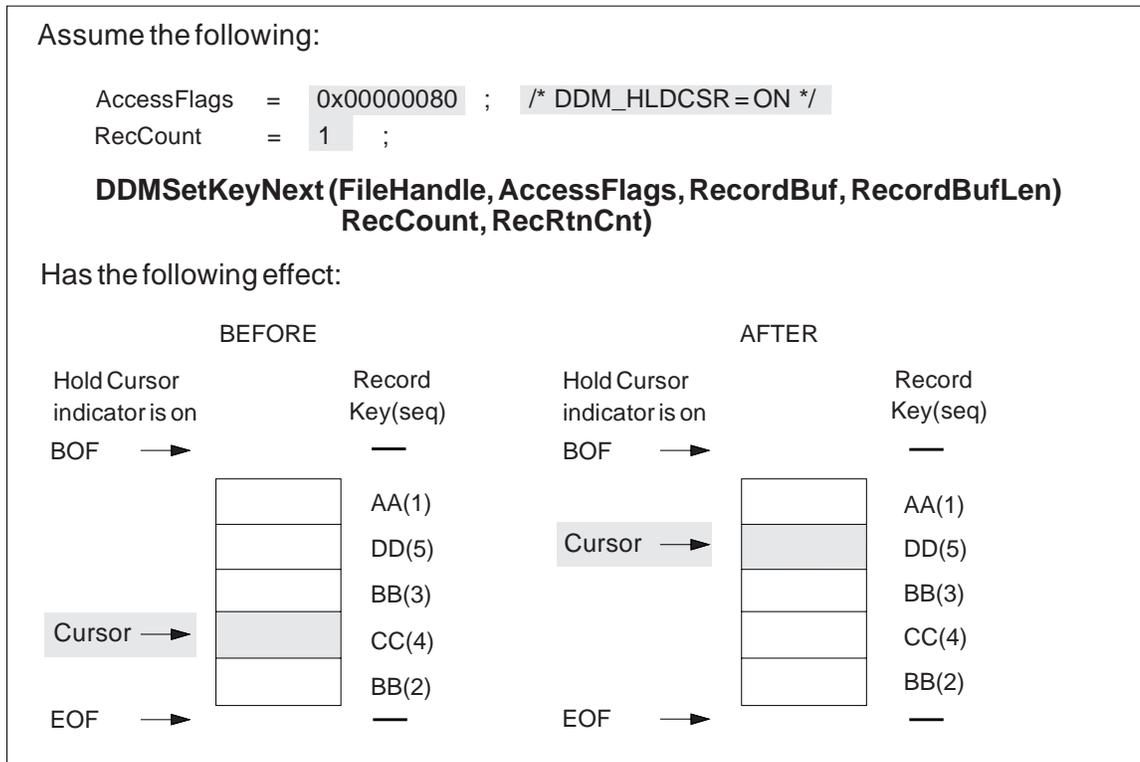


Figure 47. DDMSetKeyNext Function with Hold Cursor Initially On

DDMSetKeyNext

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR=ON */
RecCount   = 1 ;
```

**DDMSetKeyNext (FileHandle, AccessFlags, RecordBuf, RecordBufLen)
RecCount, RecRtnCnt)**

Has the following effect:

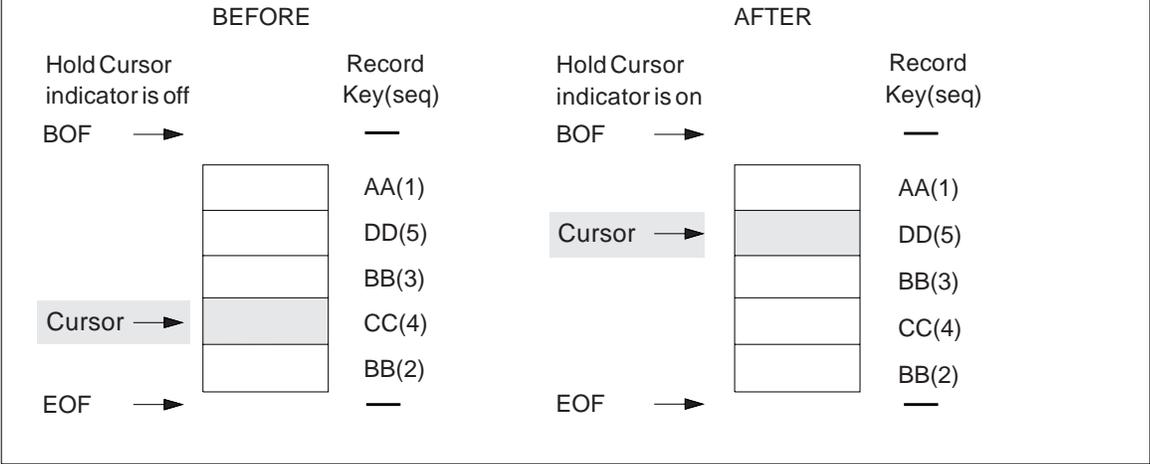


Figure 48. DDMSetKeyNext Function with Hold Cursor Initially Off

DDMSetKeyNext

These are examples of RecordBuf data formats:

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count. The RC parameter is used to indicate the number of duplicate records. It provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list. Note: RC is not included unless identical, consecutive records are being returned.
L2	The length (ULONG) from the beginning of L2 to the end of data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

DDMSetKeyNext

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents. Note: RECCNT is not included unless identical, consecutive records are being returned.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record and each subsequent record has a record number one greater than the previous record. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L3	The length (ULONG) from the beginning of L3 to the end of Data.

DDMSetKeyNext

X'144A' The value (CODEPOINT) indicating that the following data is record data.

Data The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.

DDMSetKeyNext

Note: RECCNT is not included unless identical, consecutive records are being returned.

RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record key value.
X'1430'	The value (CODEPOINT) indicating that the following key is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following key is a key count. The RC parameter is used to indicate the number of duplicate keys. It provides a shorthand way of specifying N keys, where N>1, without replicating the key's contents. Note: RC is not included unless identical, consecutive keys are being returned.
RC	The number (ULONG) of duplicate keys in the record attribute list.

DDMSetKeyNext

L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'1115'	KEY	L4	X'144A'	Data
----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents. Note: RECCNT is not included unless identical, consecutive records are being returned.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of the key value.

DDMSetKeyNext

X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetKeyPrevious

DDMSetKeyPrevious (Set Cursor to Previous Record in Key Sequence)

This function moves the cursor to the previous record of the file in key sequence and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetKeyPrevious (HDDMFILE      FileHandle,
                          ULONG          AccessFlags,
                          PDDMRECORD    RecordBuf,
                          ULONG          RecordBufLen,
                          ULONG          RecCount,
                          PULONG        RecRtnCnt
                          );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
3–4	Reserved flags
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 224.

RecordBufLen

The length (ULONG) of the record buffer.

RecCount

Specifies the number (ULONG) of records requested.

DDMSetKeyPrevious

RecRtnCnt

The pointer (PULONG) to the count of the records actually returned. When RECAL (Record Attribute List) parameters are specified in RecordBuf and RECCNT is specified within the RECAL, the RecRtnCnt parameter (ULONG) reflects the RECCNT number of duplicate records. Therefore, if RecordBuf contained 25 data records, one of which included a RECAL with RECCNT having a value of 150, the value of RecRtnCnt would be 175.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
ENDFILRM	X'120B'	End of File
FILATHRM	X'123B'	Not Authorized to File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECIUSRM	X'124A'	Record in Use
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

If the file contains records with duplicate keys, the cursor is set to the previous record with the same or next key in the key sequence.

As an option, DDMSetKeyPrevious can:

- Set the hold cursor indicator to on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

If RecCount gives a value greater than 1, multiple records are sent to the source agent. RecCount requests that the DDMSetKeyPrevious function be performed the number of times specified by RecCount. This moves the cursor to the last record processed by the DDMSetKeyPrevious function.

If RecCount gives a number greater than the remaining records in the file, the remaining records are sent to the source agent, the cursor position is changed to BOF, and an ENDFILRM reply message is sent.

DDMSetKeyPrevious

Effect on Cursor Position

Normal completion (SVRCOD of 0 or 4)

The cursor is moved to the previous record in the index key sequence. If an ENDFILRM reply message results, the cursor is moved to the beginning of file.

Error termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (for example, DDMMModifyRec or DDMDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function issued references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetKeyPrevious

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file handle is invalid.	HDLNFNRM
The file does not contain any records initially after a DDMCreateRecFile. Note: The cursor position is set to BOF. The file does not contain any records before the current cursor position. Note: The cursor position is set to BOF. RecCount specifies a number greater than the number of records remaining in the file.	ENDFILRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The record lock cannot be obtained.	RECIUSRM

DDMSetKeyPrevious

Examples

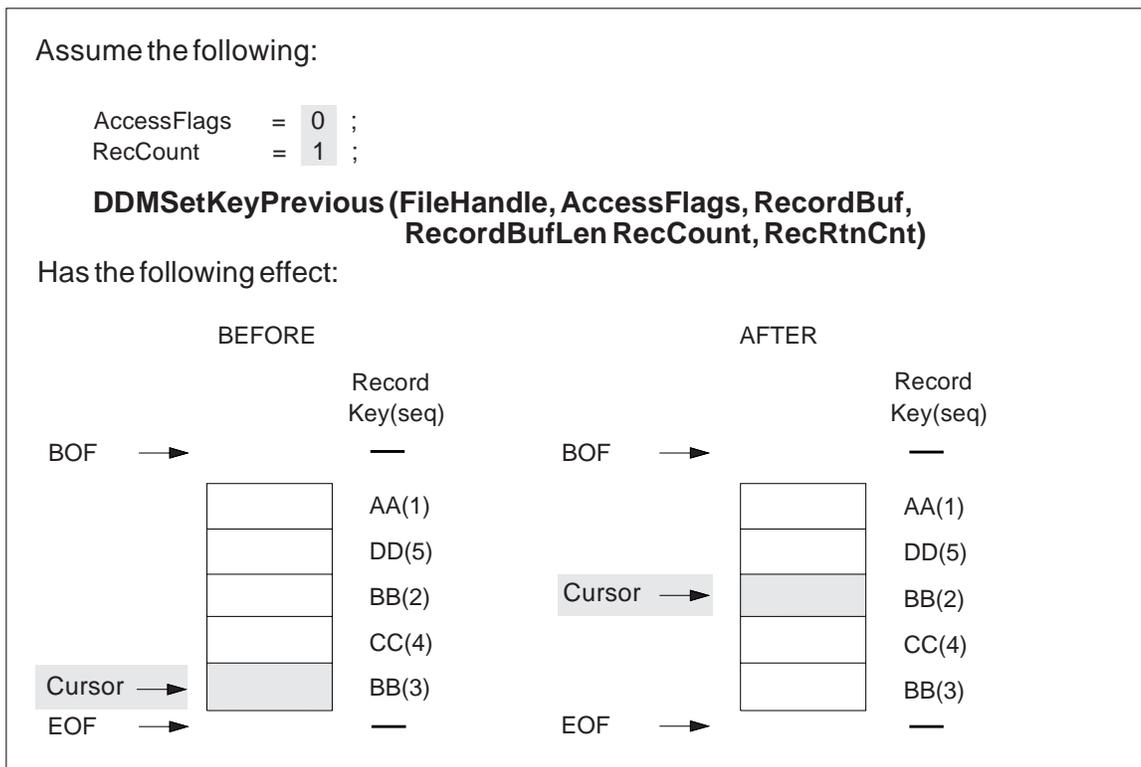


Figure 49. DDMSetKeyPrevious Function with Duplicate Key Values

DDMSetKeyPrevious

Assume the following:

```
AccessFlags = 0 ;  
RecCount   = 1 ;
```

**DDMSetKeyPrevious (FileHandle, AccessFlags, RecordBuf,
RecordBufLen RecCount, RecRtnCnt)**

Has the following effect:

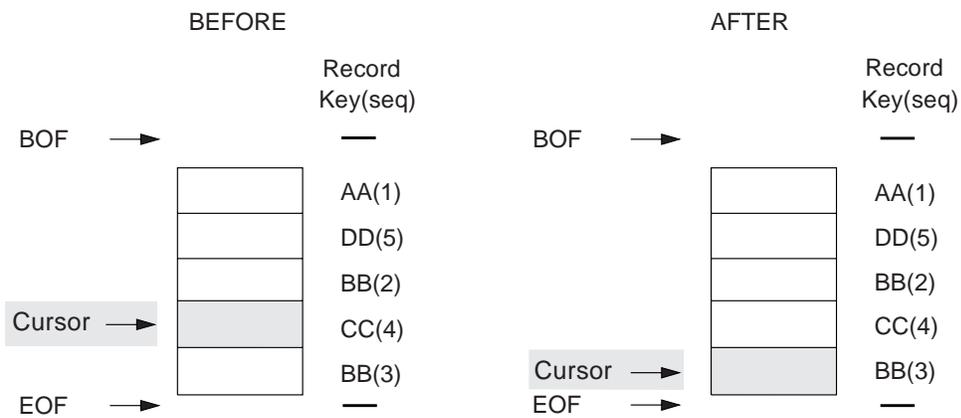


Figure 50. `DDMSetKeyPrevious` Function for Ascending Sequence

DDMSetKeyPrevious

Assume the following:

```
AccessFlags = 0 ;
RecCount    = 1 ;
```

**DDMSetKeyPrevious (FileHandle, AccessFlags, RecordBuf,
RecordBufLen RecCount, RecRtnCnt)**

Has the following effect:

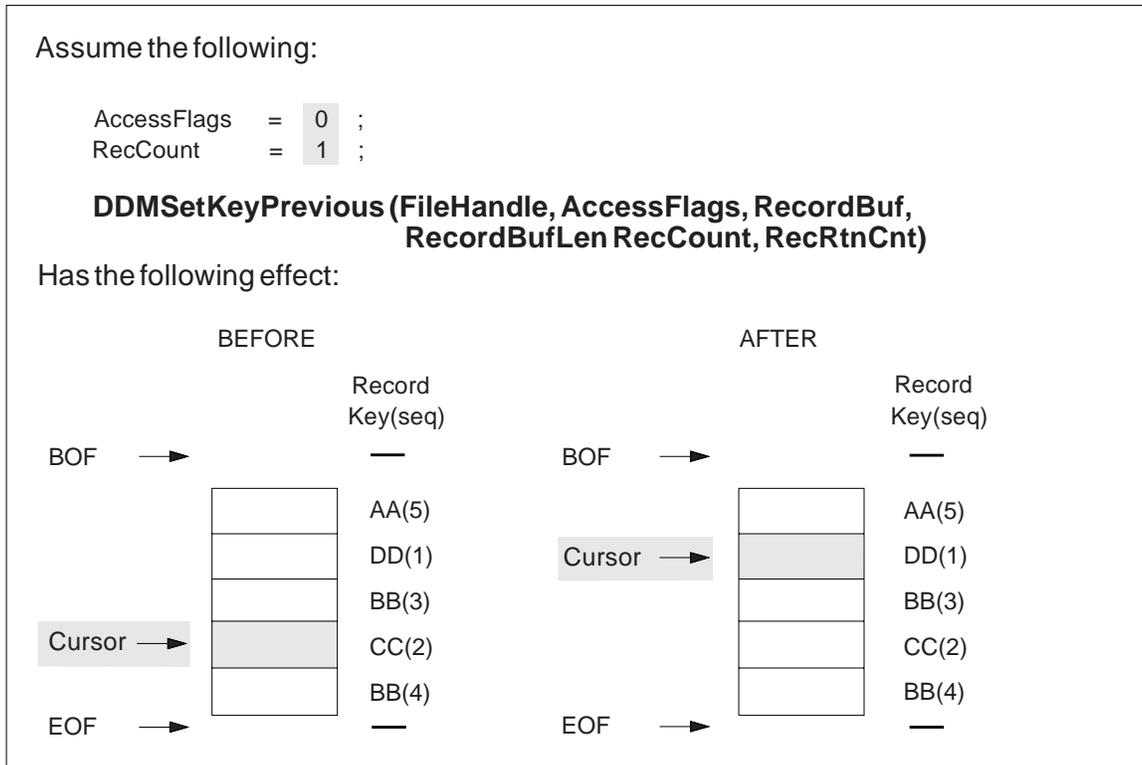


Figure 51. DDMSetKeyPrevious Function for Descending Sequence

DDMSetKeyPrevious

These are examples of RecordBuf data formats:

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count. The RC parameter is used to indicate the number of duplicate records. It provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list. Note: RC is not included unless identical, consecutive records are being returned.
L2	The length (ULONG) from the beginning of L2 to the end of data.
X'144A'	The value (CODEPOINT) indicating that the following data is a record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

DDMSetKeyPrevious

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents. Note: RECCNT is not included unless identical, consecutive records are being returned.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record and each subsequent record has a record number one greater than the previous record. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L3	The length (ULONG) from the beginning of L3 to the end of Data.

DDMSetKeyPrevious

X'144A' The value (CODEPOINT) indicating that the following data is record data.

Data The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.

DDMSetKeyPrevious

	Note: RECCNT is not included unless identical, consecutive records are being returned.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record key value.
X'1430'	The value (CODEPOINT) indicating that the following key is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following key is a key count. The RC parameter is used to indicate the number of duplicate keys. The RC parameter provides a shorthand way of specifying N keys, where N>1, without replicating the key's contents.
	Note: RC is not included unless identical, consecutive keys are being returned.

DDMSetKeyPrevious

RC	The number (ULONG) of duplicate keys in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'1115'	KEY	L4	X'144A'	Data
----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents. Note: RECCNT is not included unless identical, consecutive records are being returned.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.

DDMSetKeyPrevious

L3	The length (ULONG) from the beginning of L3 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetLast (Set Cursor to Last Record)

This function sets the cursor to the last record of the file and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetLast (HDDMFILE      FileHandle,
                  ULONG          AccessFlags,
                  PDDMRECORD    RecordBuf,
                  ULONG          RecordBufLen
                  );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	DDM_ALLREC (All Records, Active or Inactive)
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 237.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetLast

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
RECNFNRM	X'1225'	Record Not Found

Remarks

The DDM_ALLREC bit flag is used to determine the last record of the file. If DDM_ALLREC is not set, the cursor is set to the last active record in the file. Otherwise, the cursor is set to the last record in the file (the record preceding EOF). For direct files, DDM_ALLREC must be set off.

As an option, DDMSetLast can:

- Set the hold cursor indicator to on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Any key limits set are reset when the function completes.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetLast

Table 20. DDMSetLast (DDM_NODATA or DDM_ALLREC) Decision Table					
If the DDMSetLast function is issued:					
When initial system states are:					
Record State	I	I	I	A	A
DDM_ALLREC	F	T	T	*	*
DDM_NODATA	*	F	T	F	T
The final system states are:	↓	↓	↓	↓	↓
RECINARM (returned)	F	F	T4	F	F
RECINA (returned)	F	T	F	F	F
RECORD (returned)	F	F	F	T	F
CURSOR (returned)	F	T	T	T	T
Repeat table after bypassing record	T	F	F	F	F
Legend					
A	Active				
I	Inactive				
T	TRUE (On)				
F	FALSE (Off)				
T4	TRUE with SVRCOD (Warning)				
*	Either TRUE or FALSE				

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the last record position in the file if DDM_ALLREC is set on. The cursor is moved to the last active record in the file if DDM_ALLREC is set to off.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

DDMSetLast

- The record is updated (for example, DDMMModifyRec or DDMDeleteRec.)
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMSInsertRecEOF, DDMSInsertRecKey, DDMSInsertRecNum, DDMSSetUpdateKey, and DDMSSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes a Reply Message to be Generated and the Function Continues	With This Reply Message
DDM_ALLREC and DDM_NODATA are active and an inactive record is read.	RECINARM

This Causes the Function to be Terminated	With This Reply Message
Accessflag DDM_NODATA is not set and the file was opened without GETAL.	INVRQSRM
The RecordBuf is not large enough to hold the returned record.	LENGTHRM

This Causes the Function to be Rejected	With This Reply Message
DDM_RECNRFB or DDM_KEYVALFB is set or DDM_NODATA is not set and RecordBuf doesn't contain an address.	ADDRRM
The file handle is not valid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM

DDMSetLast

This Causes the Function to be Rejected	With This Reply Message
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified. DDM_ALLREC is set and the file is a direct file.	INVRQSRM
The record is damaged (not an active or inactive record).	RECDMGRM
A record lock cannot be obtained.	RECIUSRM
Bypassing inactive records is requested (DDM_ALLREC is off) and the file only contains inactive records. The file does not contain any records initially after a DDMCreateRecFile. Note: The cursor position is not changed.	REC�FNRM

Examples

Assume the following:

```
AccessFlags = 0x00000000 ; /* DDM_ALLREC=OFF */
```

DDMSetLast (FileHandle, AccessFlags, RecordBuf, RecordBufLen)

Has the following effect:

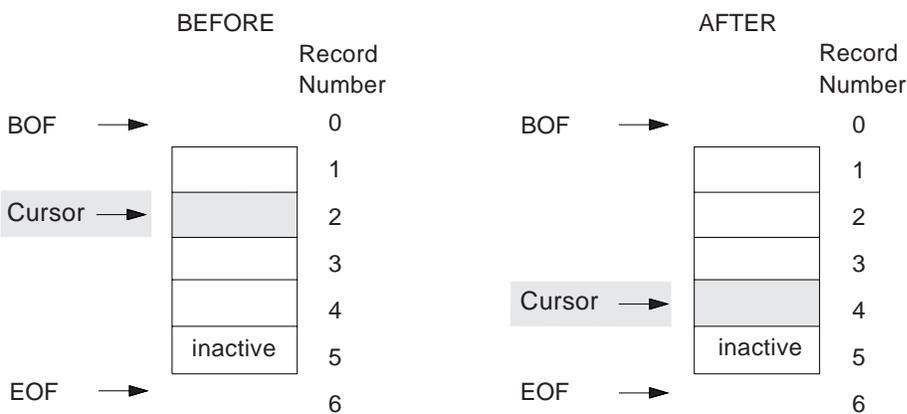


Figure 52. DDMSetLast DDM_ALLREC Set Off for Sequential File

DDMSetLast

Assume the following:

```
AccessFlags = 0x00000010 ; /* DDM_ALLREC=ON */
```

DDMSetLast (FileHandle, AccessFlags, RecordBuf, RecordBufLen)

Has the following effect:

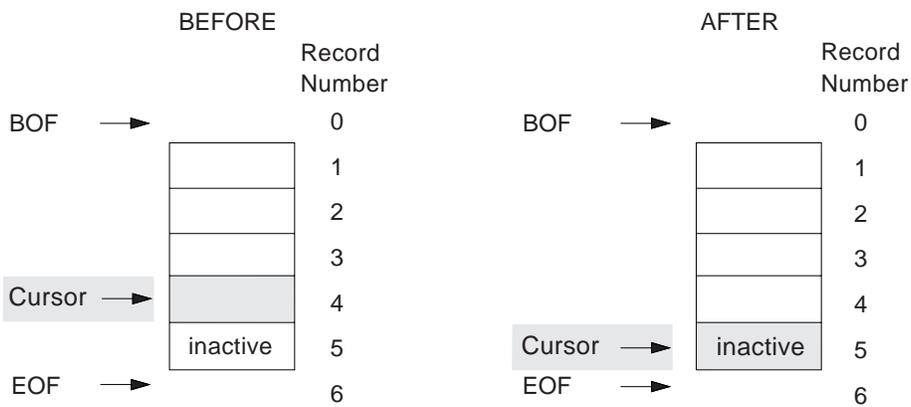


Figure 53. `DDMSetLast` `DDM_ALLREC` Set On for Sequential File

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field

Description

LL

The length (ULONG) of the record buffer from the beginning of LL to the end of Data.

CP

The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.

X'144A'

Indicates that the following data is record data (RECORD).

DDMSetLast

X'142D' Indicates that the following data is a ULONG length of an inactive record (RECINA).

Data Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
	X'144A' Indicates that the following data is record data (RECORD).
	X'142D' Indicates that the following data is a ULONG length of an inactive record (RECINA).
Data	Either record data or the length (ULONG) of the inactive record.

DDMSetLast

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
Data	Either record data or the length (ULONG) of the inactive record.

DDMSetLast

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY	L3	CP	Data
----	---------	----	---------	----	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetLast

L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is record data or a ULONG length inactive record length.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is a ULONG length of an inactive record (RECINA).
Data	Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetMinus (Set Cursor Minus)

This function sets the cursor to the record number of the file indicated by the cursor, minus the number of record positions specified by the CsrDisp (Cursor Displacement) parameter. This function can also return the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetMinus (HDDMFILE      FileHandle,
                   ULONG          AccessFlags,
                   ULONG          CsrDisp,
                   PDDMRECORD     RecordBuf,
                   ULONG          RecordBufLen
                   );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
9–31	Reserved flags
8	DDM_ALWINA (Allow Cursor on Inactive Record)
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	DDM_RTININA (Return Inactive Record)
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

CsrDisp

Specifies the cursor displacement (ULONG) in the negative direction.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Example” on page 247.

DDMSetMinus

RecordBufLen

The length (ULONG) of the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
RECNBRRM	X'1224'	Record Number Out of Bounds

Remarks

The type of the records in the file (active or inactive) bypassed by DDMSetMinus has no effect on the cursor positioning.

As an option, DDMSetMinus can:

- Specify whether the cursor can be set to an inactive record position (DDM_ALWINA).
- Set the hold cursor indicator on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether inactive records should be returned (DDM_RTININA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Any key limits set are reset when the function completes.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetMinus

Table 21. DDMSetMinus (DDM_ALWINA, DDM_RTNINA, or DDM_NODATA) Decision Table							
If the DDMSetMinus function is issued:							
When initial system states are:							
Record State	I	I	I	I	I	A	A
DDM_ALWINA	T	T	T	F	F	*	*
DDM_RTNINA	T	*	F	*	*	*	*
DDM_NODATA	F	T	F	F	T	F	T
The final system states are:	↓	↓	↓	↓	↓	↓	↓
RECINARM (returned)	F	T4	T4	T8	T8	F	F
RECINA (returned)	T	F	F	F	F	F	F
RECORD (returned)	F	F	F	F	F	T	F
CURSOR (changed)	T	T	T	F	F	T	T
Legend							
A	Active						
I	Inactive						
T	TRUE (On)						
F	FALSE (Off)						
T4	TRUE with SVRCOD (Warning)						
T8	TRUE with SVRCOD (Error)						
*	Either TRUE or FALSE						

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is positioned to the record position CsrDisp records prior to where the cursor was positioned before the DDMSetMinus function was issued.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

DDMSetMinus

- The record is updated (DDMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMinertRecEOF, DDMinertRecKey, DDMinertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to Continue and Return	This Reply Message
An inactive record is read and DDM_ALWINA is active, and DDM_RTNINA is not set or DDM_NODATA is set.	RECINARM

This Causes the Function to be Rejected	With This Reply Message
DDM_RECNRFB or DDM_KEYVALFB is set, or DDM_NODATA is not set, and RecordBuf doesn't contain an address.	ADDRRM
The file handle is invalid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified. Access flag DDM_NODATA is not set and the file was opened without GETAI.	INVRQSRM
RecordBuf is not large enough to hold the returned record.	LENGTHRM
The record is damaged (not an active or inactive record).	RECDMGRM

DDMSetMinus

This Causes the Function to be Rejected	With This Reply Message
<p>The record is inactive and the cursor is not allowed to be set to an inactive record position (DDM_ALWINA is not set).</p> <p>Note: The cursor is not changed.</p>	RECINARM
<p>The record lock cannot be obtained.</p>	RECIUSRM
<p>The CsrDisp value places the cursor prior to the first record in the file.</p> <p>Note: The cursor position does not change.</p> <p>The file contains no records after a DDMCreateRecFile.</p> <p>Note: The cursor position does not change.</p> <p>The cursor is placed outside the bounds of the file; before BOF in a sequential file, and past the physical boundary in a direct file.</p>	RECNBRRM

Example

Assume the following:

```
AccessFlags = 0 ;
CsrDisp     = 2 ;
```

DDMSetMinus (FileHandle, AccessFlags, CsrDisp, RecordBuf, RecordBufLen)

Has the following effect:

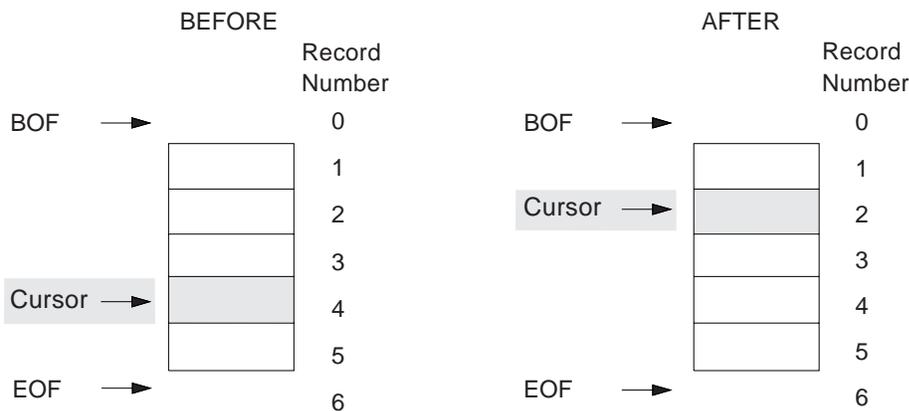


Figure 54. DDMSetMinus Function

DDMSetMinus

These are examples of RecordBuf data formats:

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).

DDMSetMinus

L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

DDMSetMinus

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

DDMSetMinus

RecordBuf DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY	L3	CP	Data
----	---------	----	---------	----	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	Either record data or the length (ULONG) of the inactive record.

DDMSetMinus

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetNextKeyEqual

DDMSetNextKeyEqual (Set Cursor to Next Record with Equal Key)

The DDMSetNextKeyEqual function moves the cursor to the next record in the key sequence. This happens only if the key field of that record has a value that equals the value specified in KeyValBuf (Key Value Buffer) parameter. This function can also return the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetNextKeyEqual (HDDMFILE      FileHandle,
                           ULONG         AccessFlags,
                           PDDMOBJECT    KeyValBuf,
                           PDDMRECORD    RecordBuf,
                           ULONG         RecordBufLen
                           );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
3–4	Reserved flags
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

KeyValBuf

Pointer to the buffer which contains the key to which the cursor should be moved. The format of the key value buffer upon invocation of the function is:

LL	X'1115'	Key Value
----	---------	-----------

Field	Description
LL	The length (ULONG) of the key value description from the beginning of LL to the end of Key Value.

DDMSetNextKeyEqual

X'1115' The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 257.

RecordBufLen

The length (ULONG) of the record buffer. The record buffer length should be the same size as the largest possible record plus the number of bytes required for the RECAL (Record Attribute List).

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
ENDFILRM	X'120B'	End of File
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECNFNRM	X'1225'	Record Not Found
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

Generic keys can be specified in KeyValBuf.

If the key value of the next key in the key sequence is not equal to the value specified by KeyValBuf, an ENDFILRM reply message is returned and the cursor is moved to EOF. The requester must reposition the cursor before another DDMSetNextKeyEqual function can be requested.

The cursor remains at its current position if the key value of the current record in key sequence is equal to the value specified by the key value buffer and:

- The hold cursor indicator in the cursor is set to on.
- The DDM_HLDCSR bit in AccessFlags is FALSE.
- The record is active.

For all other conditions, the cursor is updated.

As an option, DDMSetNextKeyEqual can:

- Set the hold cursor indicator to on (DDM_HLDCSR).
- Return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

DDMSetNextKeyEqual

If key limits have been established DDMSetNextKeyEqual sets the cursor to the next record if it equals the specified key value, and the key value of the record is before or equal to the value specified by high key limit on DDMSetKeyLimits. If the next record is after the high key limit, the function is rejected with an ENDFILRM reply message and the cursor is set to the EOF position of file. See “DDMSetKeyLimits (Set Key Limits)” on page 195.

If the hold cursor indicator in the cursor is set to on, the DDM_HLDCSR bit in AccessFlags is FALSE, and the record is active, the cursor remains at its current position. For all other conditions, the cursor is updated.

Effect on Cursor Position

Normal completion (SVRCOD of 0 or 4)

The cursor is moved to the selected record, or remains in the current record based on the hold cursor indicator in the cursor, DDM_HLDCSR bit in AccessFlags, and whether the record is active. If an ENDFILRM reply message results, the cursor is moved EOF.

Error termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDelRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnlockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

DDMSetNextKeyEqual

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes	This Reply Message to be Returned
The key field of the next record in key sequence is not equal to the key value specified by the KeyValBuf parameter.	ENDFILRM

This Causes the Function to be Rejected	With This Reply Message
The file does not contain any records beyond the current cursor position, within the limits set by the DDMSetKeyLimits function. The cursor had previously been set to an inactive record. Note: The cursor is positioned to EOF.	ENDFILRM
The file handle is invalid.	HDLNFNRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents.	INVRQSRM
The record lock cannot be obtained.	RECIUSRM
The file does not contain any records initially after a DDMCreateRecFile. Note: The cursor position is not changed.	RECNFNRM

DDMSetNextKeyEqual

Examples

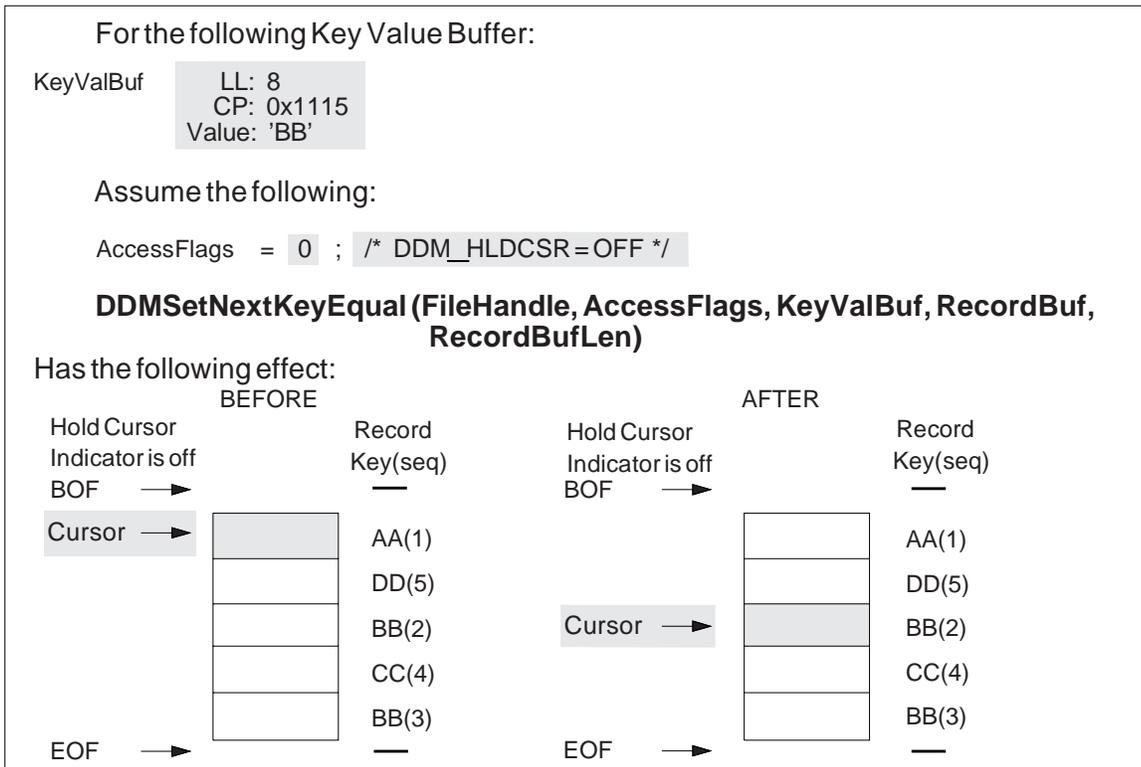


Figure 55. DDMSetNextKeyEqual to Access First Duplicate Key. From the current cursor position, the next record in the key sequence is examined for a key value of BB. The cursor is moved to that record and the record is returned.

DDMSetNextKeyEqual

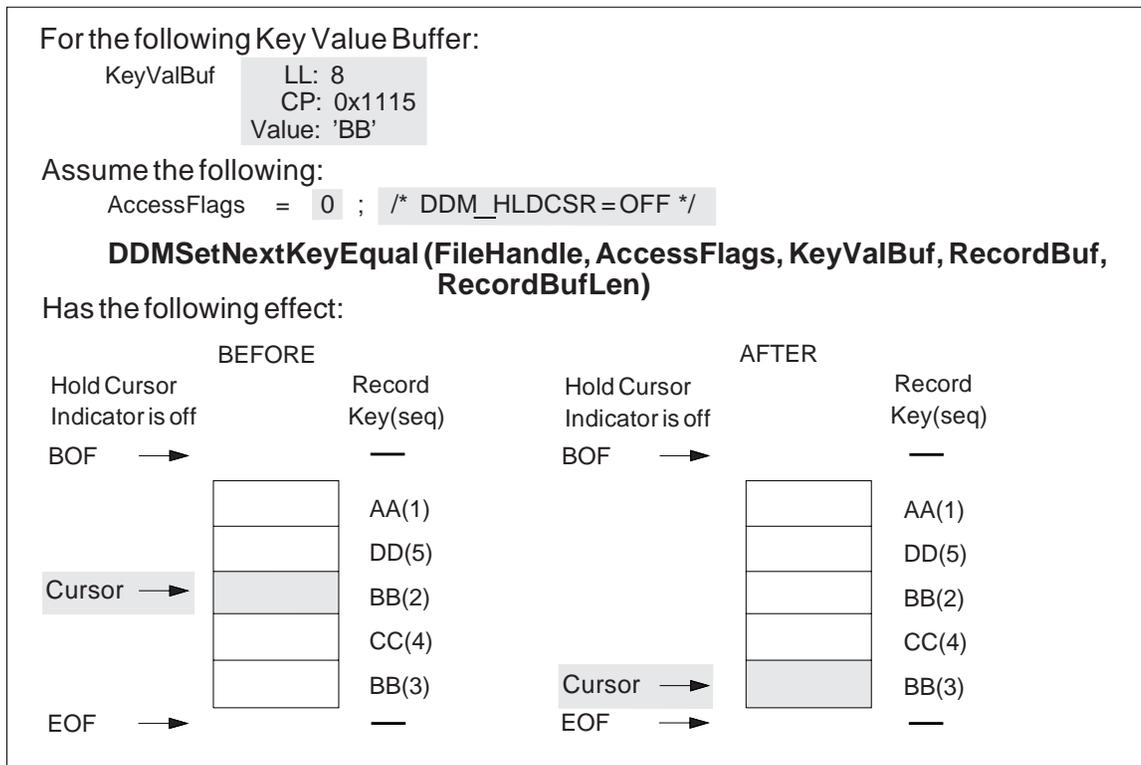


Figure 56. *DDMSetNextKeyEqual* to Access the Next Duplicate Key. From the current cursor position, within a set of records with duplicate keys, the next record in key sequence is examined for a key value of BB. The cursor is positioned at that record and the record is returned.

DDMSetNextKeyEqual

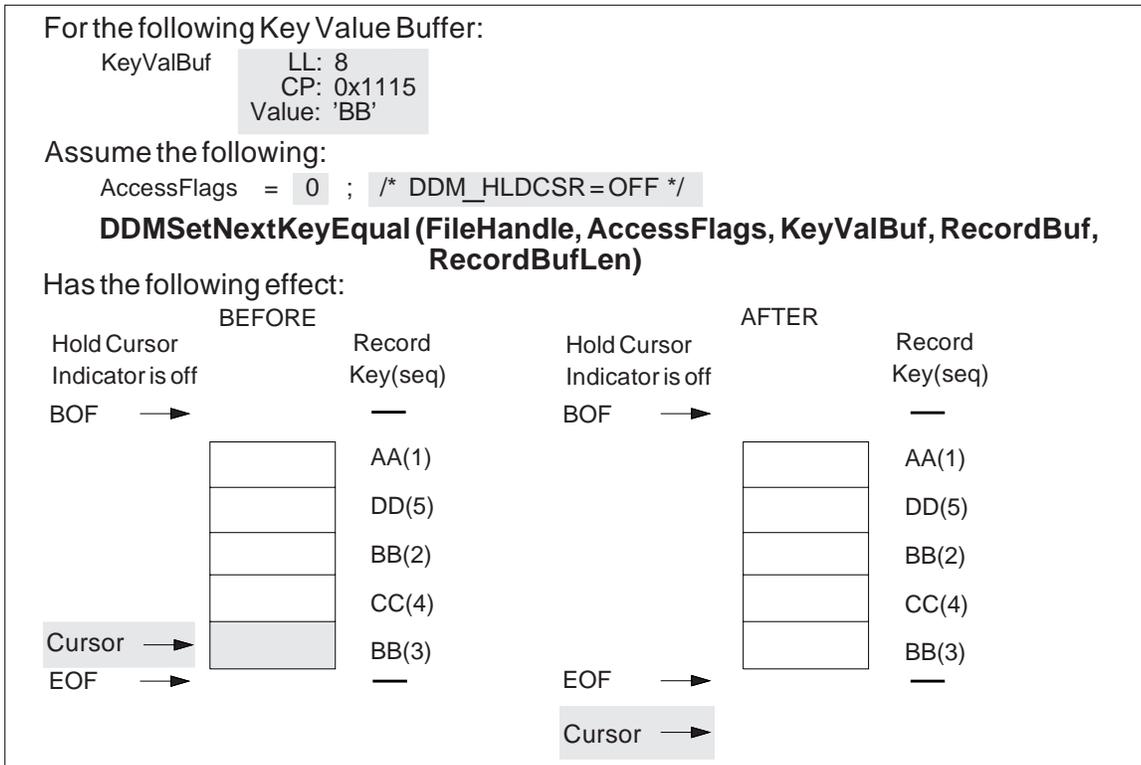


Figure 57. DDMSetNextKeyEqual to Access Past the Last Duplicate Key. From the current cursor position, at the last record in a set of records with duplicate keys, the next record in the key sequence is examined for a key value of BB. This record does not contain a key field of BB. The cursor is set to EOF, and ENDFILRM is returned.

DDMSetNextKeyEqual

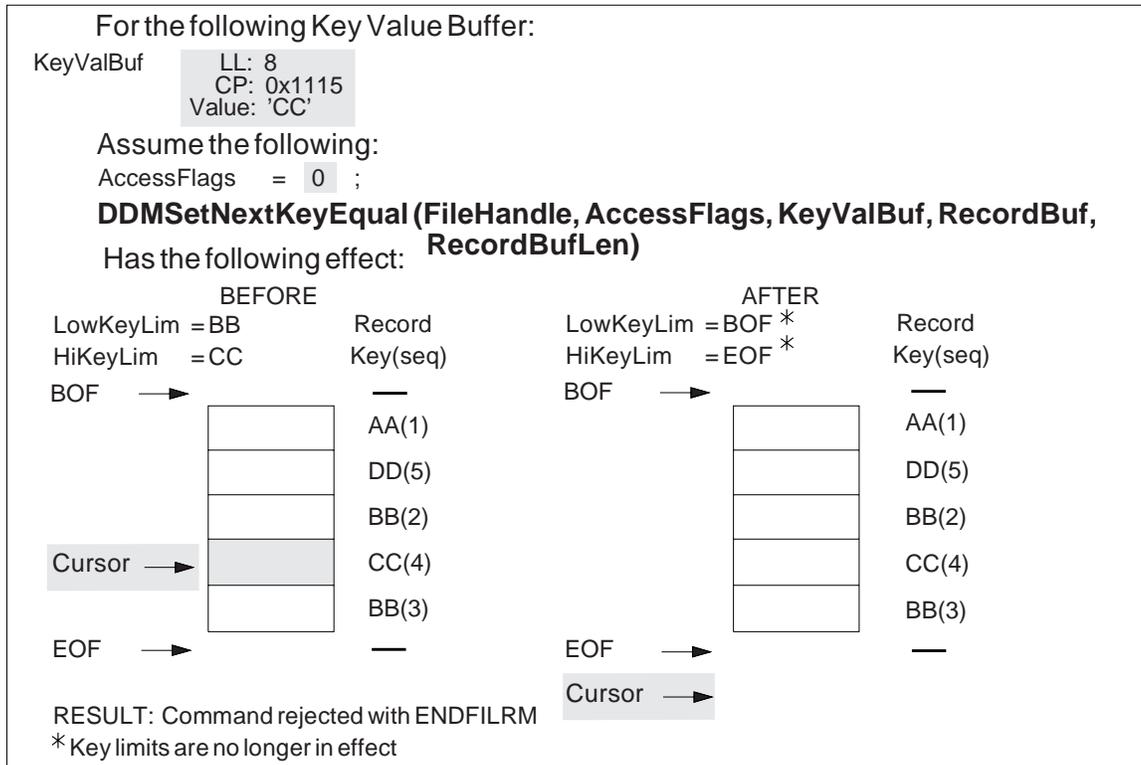


Figure 58. DDMSetNextKeyEqual Function with Key Limits Set. If key limits have been established (see DDMSetKeyLimits), the DDMSetNextKeyEqual command sets the cursor to the next record if it equals the specified key value, and the key value of the record is before or equal to the value specified by High Key Limit on DDMSetKeyLimits. If the next record is after the High Key Limit limit, the command is rejected with ENDFILRM and the cursor is set to the end of file.

DDMSetNextKeyEqual

For the following Key Value Buffer:

```
KeyValBuf  LL: 8
           CP: 0x1115
           Value: 'CC'
```

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR=OFF */
```

DDMSetNextKeyEqual (FileHandle, AccessFlags, KeyValBuf, RecordBuf, RecordBufLen)

Has the following effect:

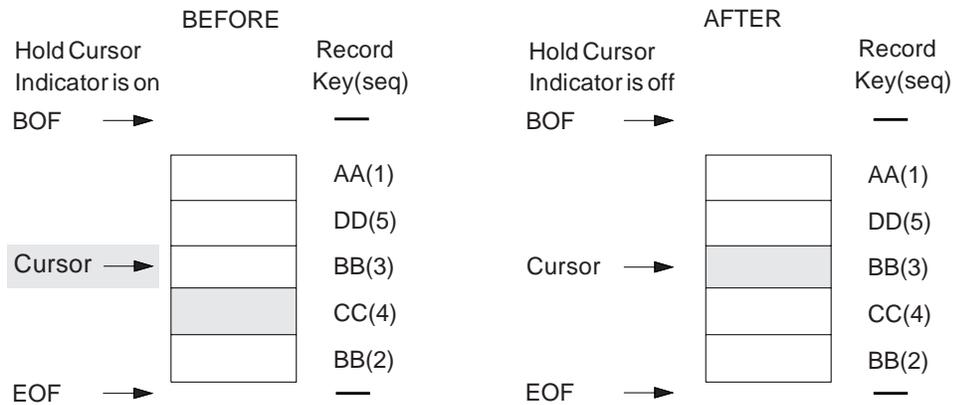


Figure 59. DDMSetNextKeyEqual Function with Hold Cursor Initially On. If the hold cursor indicator in the cursor is set to on, the HLDCSR bit in the Access Flags is FALSE, and the record is active, the cursor remains at its current position. For all other conditions, the cursor is updated.

DDMSetNextKeyEqual

For the following Key Value Buffer:

```
KeyValBuf  LL: 8
           CP: 0x1115
           Value: 'DD'
```

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR=ON */
```

DDMSetNextKeyEqual (FileHandle, AccessFlags, KeyValBuf, RecordBuf, RecordBufLen)

Has the following effect:

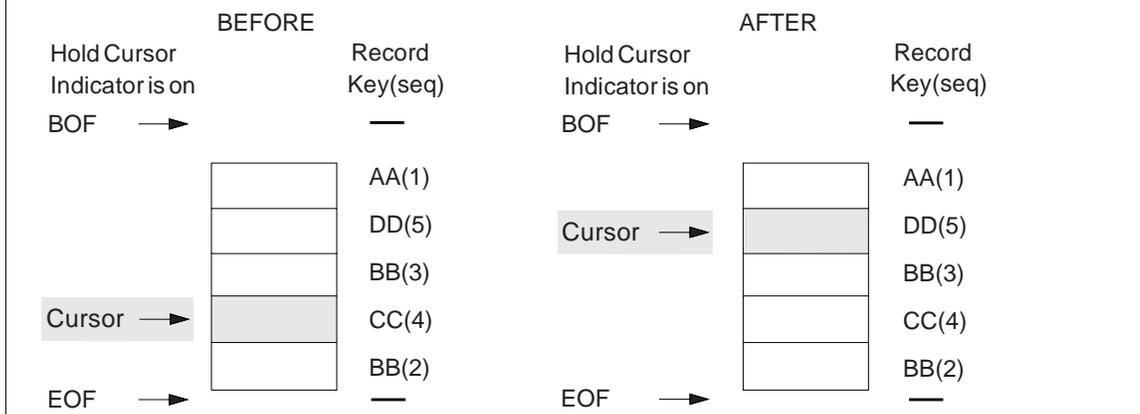


Figure 60. DDMSetNextKeyEqual function with Hold Cursor Initially On

DDMSetNextKeyEqual

For the following Key Value Buffer:

```
KeyValBuf  LL: 8
           CP: 0x1115
           Value: 'DD'
```

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR=ON */
```

DDMSetNextKeyEqual (FileHandle, AccessFlags, KeyValBuf, RecordBuf, RecordBufLen)

Has the following effect:

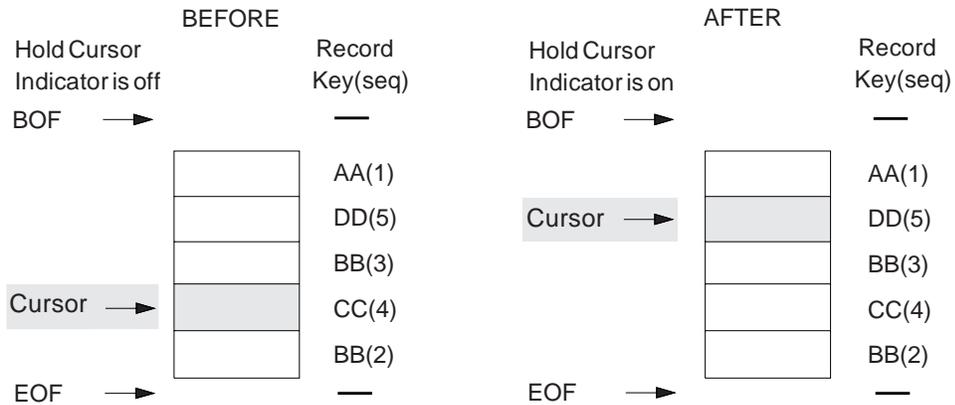


Figure 61. DDMSetNextKeyEqual function with Hold Cursor Initially Off

DDMSetNextKeyEqual

These are examples of RecordBuf data formats:

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).

DDMSetNextKeyEqual

RN	The record number (ULONG) of the record in the record attribute list. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.

DDMSetNextKeyEqual

X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).

DDMSetNextKeyEqual

L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetNextKeyEqual

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetNextRec (Set Cursor to Next Record)

This function sets the cursor to the record that has a record number one greater than the current cursor position and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetNextRec (HDDMFILE      FileHandle,
                     ULONG          AccessFlags,
                     PDDMRECORD     RecordBuf,
                     ULONG          RecordBufLen,
                     ULONG          RecCount,
                     PULONG         RecRtnCnt
                     );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	DDM_BYPDMG (Bypass Damaged Records)
5	DDM_NODATA (No Record Data Returned)
4	DDM_ALLREC (All Records, Active and Inactive)
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Examples” on page 276.

RecordBufLen

The length (ULONG) of the record buffer.

DDMSetNextRec

RecCount

Specifies the number (ULONG) of records requested.

RecRtnCnt

The pointer (PULONG) to the count of the records actually returned. When RECAL (Record Attribute List) parameters are specified in RecordBuf and RECCNT is specified within the RECAL, the RecRtnCnt parameter (ULONG) reflects the RECCNT number of duplicate records. Therefore, if RecordBuf contained 25 data records, one of which included a RECAL with RECCNT having a value of 150, the value of RecRtnCnt would be 175.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

If inactive records are to be bypassed (DDM_ALLREC not set), the cursor is set to the next active record that has a record number greater than the current cursor position. For direct files, the only valid specification for DDM_ALLREC is DDM_ALLREC not set.

As an option, DDMSetNextRec can:

- Specify whether more than one record should be returned (RecCount).
- Set the hold cursor indicator on (DDM_HLDCSR).
- Specify whether damaged records should be bypassed (DDM_BYPDMG).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

If DDM_HLDCSR in AccessFlags is FALSE, the cursor remains at its current position when the hold cursor indicator in the cursor was previously set and either the record is active, or the record is inactive and DDM_ALLREC in AccessFlags is TRUE. Under all other conditions, the cursor is updated. This decision process is illustrated in Table 22 on page 272.

If RecCount specifies a value greater than 1, multiple records are sent to the requestor. RecCount specifies the number of times that the DDMSetNextRec function be performed, with the following exceptions:

DDMSetNextRec

- For all iterations of the function except the last iteration, the RECINARM is not sent. All other reply messages resulting from the iteration of the function are sent.
- For the last iteration of the function, any reply message resulting from the last iteration of the function, including RECINARM, is sent.

This moves the cursor to the last record processed by the DDMSetNextRec function. Bypassed records (as a result of DDM_ALLREC not being set) are *not* counted to satisfy RecCount. If RecCount specifies a number and DDM_NODATA is set, no records are sent.

If the RecCount specifies a number greater than the remaining records in the file:

- The remaining records are sent to the source agent.
- The cursor position is changed.
- A ENDFILRM reply message is sent.

If DDM_BYPDGM is set, any damaged record encountered by the DDMSetNextRec function:

- Sends a RECDMGRM reply message.
- Updates the cursor.
- Is counted to satisfy RecCount.

This allows the maximum number of undamaged records to be sent to the source system.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetNextRec

<i>Table 22. DDMSetNextRec (DDM_ALLREC or DDM_NODATA) Decision Table (Part 1 of 2)</i>						
If the DDMSetNextRec function is issued, two decision tables are processed sequentially starting with Decision Table 1:						
Decision Table 1: DDM_HLDCSR / DDM_ALLREC						
When initial system states are:						
hldcsr indicator in cursor	T	T	T	T	F	F
DDM_HLDCSR	T	F	F	F	T	F
Record State	*	A	I	I	*	*
DDM_ALLREC	*	*	F	T	*	*
The next system states are:	↓	↓	↓	↓	↓	↓
hldcsr indicator in cursor set	T	F	F	F	T	F
move cursor to next record	Y	N	Y	N	Y	Y
go to Table 23 on page 273	Y	Y	Y	Y	Y	Y
Legend						
A	Active					
I	Inactive					
T	TRUE (On)					
F	FALSE (Off)					
T4	TRUE with SVRCOD (Warning)					
*	Either TRUE or FALSE					
Y	YES					
N	NO					

DDMSetNextRec

Table 23. DDMSetNextRec (DDM_ALLREC or DDM_NODATA) Decision Table (Part 2 of 2)					
Decision Table 2: DDM_ALLREC / DDM_NODATA					
When the system states are:					
Record State	I	I	I	A	A
DDM_ALLREC	F	T	T	*	*
DDM_NODATA	*	F	T	F	T
The next system states are:	↓	↓	↓	↓	↓
RECINARM (returned)	F	F	T4	F	F
RECINA (returned)	F	T	F	F	F
RECORD (returned)	F	F	F	T	F
cursor position saved	F	T	T	T	T
move cursor to next record and repeat Table 2	T	F	F	F	F
DDMSetNextRec complete	N	Y	Y	Y	Y
Legend					
A	Active				
I	Inactive				
T	TRUE (On)				
F	FALSE (Off)				
T4	TRUE with SVRCOD (Warning)				
*	Either TRUE or FALSE				
Y	YES				
N	NO				

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is determined in two steps: step 1 determines the first record to be considered, step 2 determines if the contents of the record are acceptable to the user.

Step 1. The cursor remains at the current record if:

- The hold indicator in the cursor is on, DDM_HLDCSR is off, and the record is active.
- The hold indicator in the cursor is on, DDM_HLDCSR is off, the record is inactive, and DDM_ALLREC is on.

Otherwise, the cursor is advanced to the next record.

The cursor may be advanced more than one record.

See Table 22 on page 272 for an illustration of this step.

Step 2. If DDM_ALLREC is off, and the record is inactive, the cursor is advanced until it points to an active record.

DDMSetNextRec

Otherwise, the cursor is pointing to the correct record.

Step 3. If an ENDFILRM results from the advancing of the cursor, the cursor is moved to EOF.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued. If RecCount is greater than 1, the cursor position is the same as before the last iteration of the function.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, then the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (for example, DDMMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, or DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued. If RecCount is greater than 1, the record locks are the same as before the last iteration of the function.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetNextRec

Exceptions

This Causes the Function to Continue and Returns	This Reply Message
DDM_ALLREC is not set and the cursor is at the last active record.	ENDFILRM
The record is damaged and DDM_BYPDMG flag is set.	RECDMGRM

This Causes the Function to be Rejected	With This Reply Message
Any data is to be returned and RecRtnCnt has not been specified. DDM_RECNRFB or DDM_KEYVALFB is set, or DDM_NODATA is not set, and RecordBuf doesn't contain an address.	ADDRRM
The cursor is already positioned at EOF. If one of the following conditions is true about the file: <ul style="list-style-type: none"> It does not contain any records initially after a DDMCreateRecFile. It does not contain any records beyond the current cursor position. It does not contain any active records beyond the current cursor position when DDM_ALLREC is not set. Note: The cursor position is changed to EOF.	ENDFILRM
The file handle is invalid.	HDLNFNRM
Any reserved bits in AccessFlags are set.	INVFLGRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified as one of the access intents. DDM_ALLREC(TRUE) is specified for a direct file. The DDM_NODATA flag is not set and the file was opened without GETAI.	INVRQSRM
The RecordBuf is not large enough to hold the returned record.	LENGTHRM
The record is damaged (record not active or inactive).	RECDMGRM
The record is inactive and DDM_NODATA is set.	RECINARM
The record lock cannot be obtained.	RECIUSRM
The RecCount is not greater than 0.	VALNSPRM

DDMSetNextRec

Examples

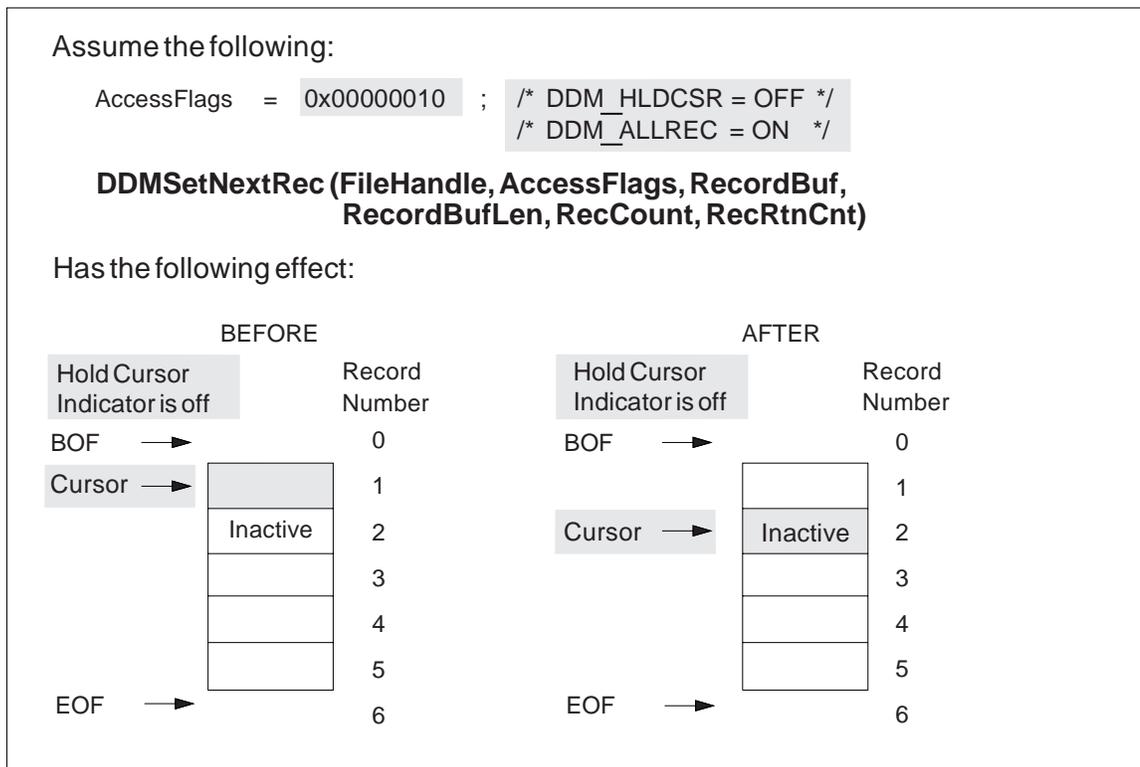


Figure 62. DDMSetNextRec Function with DDM_ALLREC Set

DDMSetNextRec

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
                /* DDM_ALLREC = OFF */
```

DDMSetNextRec (FileHandle, AccessFlags, RecordBuf, RecordBufLen, RecCount, RecRtnCnt)

Has the following effect:

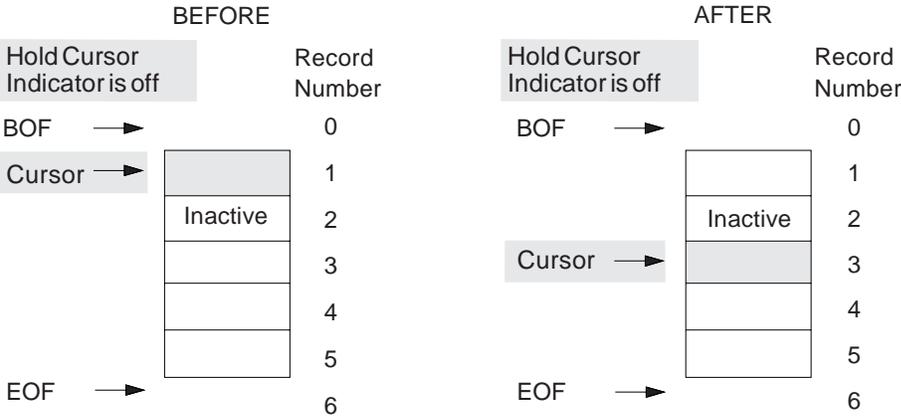


Figure 63. DDMSetNextRec Function with DDM_ALLREC Not Set

DDMSetNextRec

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */  
              /* DDM_ALLREC = OFF */
```

**DDMSetNextRec (FileHandle, AccessFlags, RecordBuf,
RecordBufLen, RecCount, RecRtnCnt)**

Has the following effect:

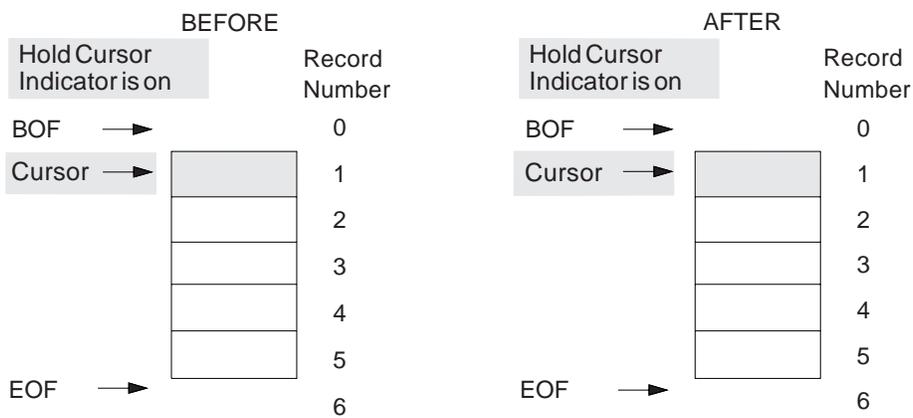


Figure 64. `DDMSetNextRec` Function with `Hold Cursor` Initially On

DDMSetNextRec

Assume the following:

```
AccessFlags = 0 ; /* DDM_HLDCSR = OFF */
                /* DDM_ALLREC = OFF */
```

DDMSetNextRec (FileHandle, AccessFlags, RecordBuf, RecordBufLen, RecCount, RecRtnCnt)

Has the following effect:

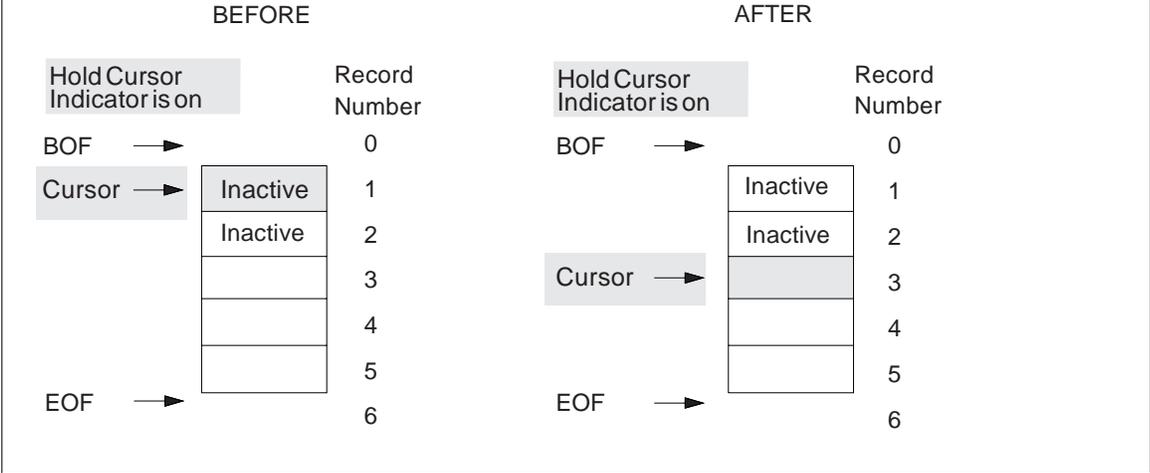


Figure 65. DDMSetNextRec Function with Hold Cursor Initially On

DDMSetNextRec

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR = ON */  
                /* DDM_ALLREC = OFF */
```

**DDMSetNextRec (FileHandle, AccessFlags, RecordBuf,
RecordBufLen, RecCount, RecRtnCnt)**

Has the following effect:

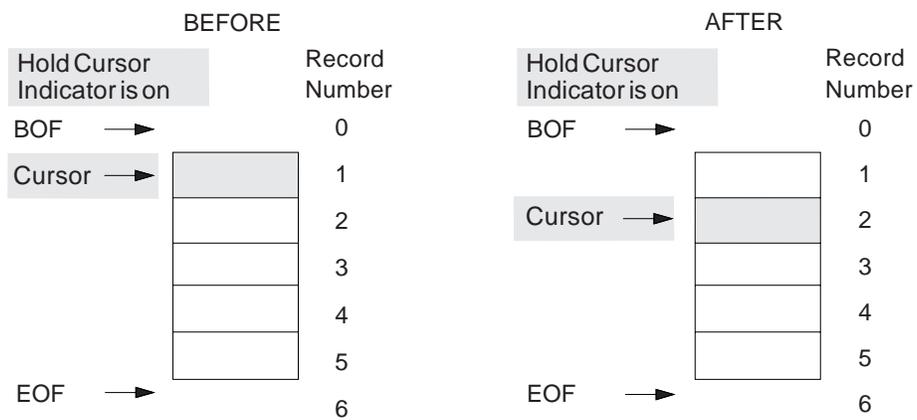


Figure 66. `DDMSetNextRec` Function with Hold Cursor Initially On

DDMSetNextRec

Assume the following:

```
AccessFlags = 0x00000080 ; /* DDM_HLDCSR = ON */
                /* DDM_ALLREC = OFF */
```

**DDMSetNextRec (FileHandle, AccessFlags, RecordBuf,
RecordBufLen, RecCount, RecRtnCnt)**

Has the following effect:

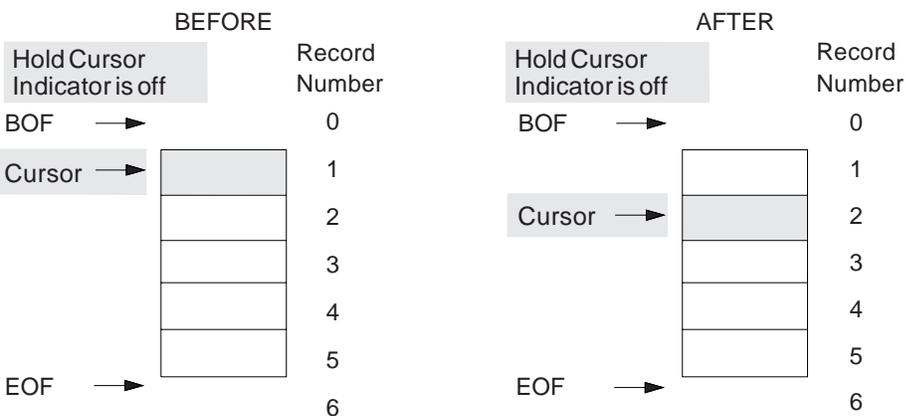


Figure 67. DDMSetNextRec Function with Hold Cursor Initially Off

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field

Description

LL

The length (ULONG) of the record attribute list from the beginning of LL to the end of the record data.

X'1430'

The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).

DDMSetNextRec

L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count. The RC parameter is used to indicate the number of duplicate records. It provides a shorthand way of specifying N record where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list. Note: RC is not included unless identical, consecutive records are being returned.
L2	The length (ULONG) from the beginning of L2 to the end of data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN	L3	CP	Data
----	---------	----	---------	----	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).

DDMSetNextRec

L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record and each subsequent record has a record number one greater than the previous record.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).

DDMSetNextRec

RN The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY	L3	CP	Data
----	---------	----	---------	----	----	---------	-----	----	----	------

Field	Description
-------	-------------

LL	The length (ULONG) of the record attribute list (from the beginning of LL to the end of Data).
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

DDMSetNextRec

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record key value.
X'1430'	The value (CODEPOINT) indicating that the following key is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following key is a key count. The RC parameter is used to indicate the number of duplicate keys. It provides a shorthand way of specifying N keys, where N>1, without replicating the key's contents. Note: RC is not included unless identical, consecutive keys are being returned.
RC	The number (ULONG) of duplicate keys in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'1115'	KEY	L4	CP	Data
----	---------	-----	----	----	------

DDMSetNextRec

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records (where N>1) without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

DDMSetNextRec

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetPathInfo

DDMSetPathInfo (Set File or Directory Information)

This function specifies information for a file or a directory.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetPathInfo (PSZ          PathName,
                      ULONG         PathInfoLevel,
                      PBYTE         PathInfoBuf,
                      ULONG         PathInfoBufSize
                      );
```

Parameters

PathName

The pointer (PSZ) to the full path name of the file or subdirectory.

PathInfoLevel

The level (ULONG) of the file or directory information being defined.

Level 0x00000001 information is the only defined level. This is the same as DosSetPathInfo, ulFileInfoLevel bit (FILE_STANDARD).

Level 0x00000001 file information sets a series of EA name/value pairs. On input, PathInfoBuf maps to an EAOP2 structure. fpGEA2List is ignored. fpFEA2List points to a data area where the relevant FEA2 list is to be found. oError is ignored.

On output, fpGEA2List is unchanged. fpFEA2List is unchanged as is the area pointed to by fpFEA2List. If an error occurred during the set, oError is the offset of the FEA2 where the error occurred. The API return code is the error code corresponding to the condition generating the error. If no error occurred, oError is undefined.

PathInfoBuf

The pointer (PBYTE) to the storage area where the system gets the file information. Refer to "Extended Attributes" on page 5 for more information on the format of this buffer.

PathInfoBufSize

The length (ULONG) of PathInfoBuf.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
CMDCHKRM	X'1254'	Command Check
FILUSRMR	X'120D'	File In Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
LENGTHRM	X'F211'	Field Length Error

DDMSetPathInfo

Message ID	Code Point	Message Title
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

This function is similar to the DosSetPathInfo command.

An example of requesting Extended Attributes (EAs) is provided on page 5.

Effect on Cursor Position

There is no effect on the cursor position, because the file is not open.

Locking (for Local VSAM File System Only)

For the OS/2 local VSAM file system, the locking behaviour is the same as that of DOSSetPathInfo. See *OS/2 WARP Control Program Programming Reference*.

For the AIX local VSAM file system, an exclusive lock is requested for the file.

Exceptions

This Causes a Reply Message to be Generated with SRVCOB = X'04' for each out-of-sync file in the file object and the Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>DDMSetPathInfo re-synchronizes the file-change date and time if the file is not open to another process unless a higher severity condition prevents it from doing so.</p>	FILDMGRM

Record File Attributes by File Class

These are modifiable record file attributes.

Refer to Table 14 on page 38.

When the FILINISZ EA is changed, it has no effect on the current space already allocated to the file.

When the DELCP EA of an alternate index file is changed, the DELCP of the base file and all other indexes is also changed.

When the GETCP EA of an alternate index file is changed, the GETCP of the base file and all other indexes is also changed.

When the INSCP EA of an alternate index file is changed, the INSCP of the base file and all other indexes is also changed.

DDMSetPathInfo

When the MODCP EA of an alternate index file is changed, the MODCP of the base file and all other indexes is also changed.

DDMSetPlus (Set Cursor Plus)

This function sets the cursor to the record number of the file indicated by the cursor, plus the number of record positions specified by the CsrDisp (Cursor Displacement) parameter. This function can also return the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetPlus (HDDMFILE      FileHandle,
                  ULONG          AccessFlags,
                  ULONG          CsrDisp,
                  PDDMRECORD     RecordBuf,
                  ULONG          RecordBufLen
                  );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
9–31	Reserved flags
8	DDM_ALWINA (Allow Cursor on Inactive Record)
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	DDM_RTININA (Return Inactive Record)
2	DDM_KEYVALFB (Key Value Feedback)
1	!DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

CsrDisp

Specifies the cursor displacement (ULONG) in the positive direction.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Example” on page 295.

DDMSetPlus

RecordBufLen

The length (ULONG) of the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
RECNBRRM	X'1224'	Record Number Out of Bounds

Remarks

The type of the records in the file (active or inactive) bypassed by DDMSetPlus has no effect on the cursor positioning.

As an option, DDMSetPlus can:

- Specify whether the cursor can be set to an inactive record position (DDM_ALWINA).
- Set the hold cursor indicator on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether inactive records should be returned (DDM_RTININA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

Any key limits set are reset when function completes.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetPlus

Table 24. DDMSetPlus (DDM_ALWINA, DDM_RTNINA, or DDM_NODATA) Decision Table							
If the DDMSetPlus function is issued:							
When initial system states are:							
Record State	I	I	I	I	I	A	A
DDM_ALWINA	T	T	T	F	F	*	*
DDM_RTNINA	T	*	F	*	*	*	*
DDM_NODATA	F	T	F	F	T	F	T
The final system states are:	↓	↓	↓	↓	↓	↓	↓
RECINARM (returned)	F	T4	T4	T8	T8	F	F
RECINA (returned)	T	F	F	F	F	F	F
RECORD (returned)	F	F	F	F	F	T	F
CURSOR (changed)	T	T	T	F	F	T	T
Legend							
A	Active						
I	Inactive						
T	TRUE (On)						
F	FALSE (Off)						
T4	TRUE with SVRCOD (Warning)						
T8	TRUE with SVRCOD (Error)						
*	Either TRUE or FALSE						

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is positioned to the record position that is beyond its original position by the number of records specified by CsrDisp.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

DDMSetPlus

- The record is updated (DDMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMSInsertRecEOF, DDMSInsertRecKey, DDMSInsertRecNum, DDMSSetUpdateKey, and DDMSSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to Return and Continue	With This Reply Message
An inactive record is read and DDM_ALWINA is active, when DDM_RTNINA is not set or DDM_NODATA is set.	RECINARM

This Causes the Function to be Rejected	With This Reply Message
DDM_RECNRFB or DDM_KEYVALFB is set, or DDM_NODATA is not set and RecordBuf doesn't contain an address.	ADDRRM
The file handle is invalid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified. Access flag DDM_NODATA is not set and the file was opened without GETAI.	INVRQSRM
RecordBuf is not large enough to hold the returned record.	LENGTHRM
The record is damaged (not an active or inactive record).	RECDMGRM

DDMSetPlus

This Causes the Function to be Rejected	With This Reply Message
The record is inactive and the cursor is not allowed to be set to an inactive record position (DDM_ALWINA is not set). Note: The cursor is not changed.	RECINARM
The record lock cannot be obtained.	RECIUSRM
The CsrDisp would cause the cursor to be placed outside the bounds of the file. Note: The cursor position is not changed. The file does not contain any records initially after a DDMSaveRecFile. Note: The cursor position is not changed.	RECNBRRM

Example

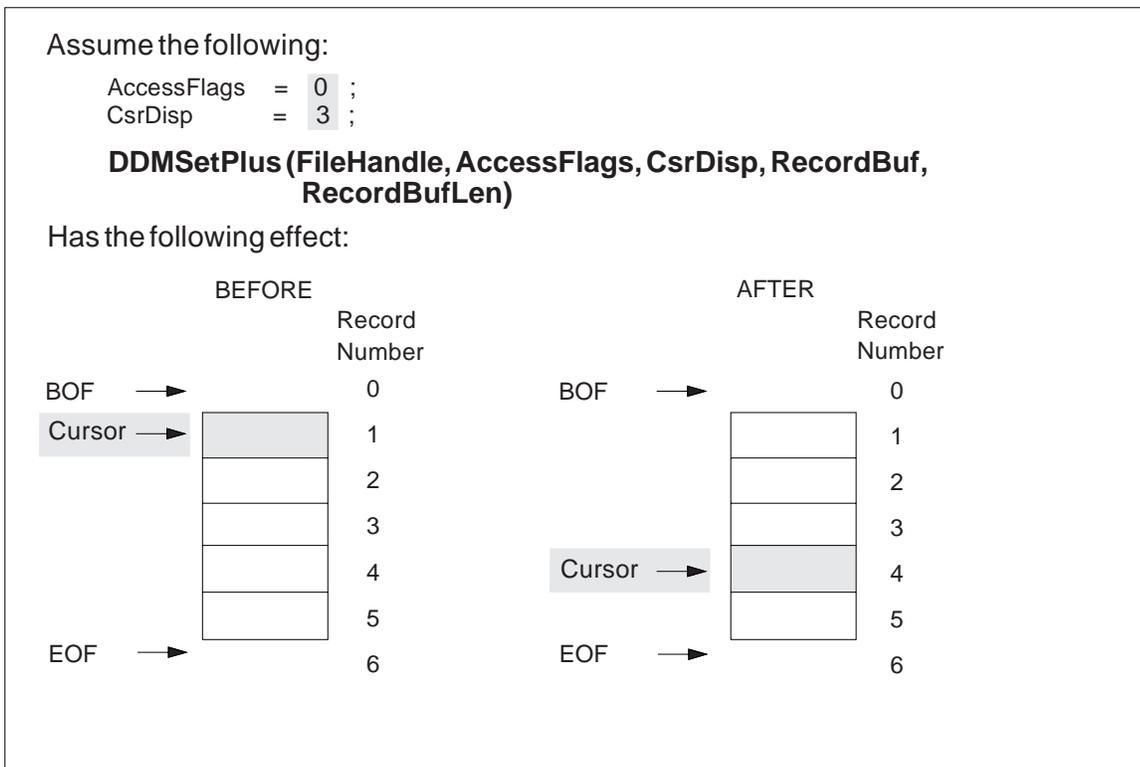


Figure 68. DDMSetPlus Function

These are examples of RecordBuf data formats:

DDMSetPlus

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.

DDMSetPlus

X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.

DDMSetPlus

X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

DDMSetPlus

L3	CP	Data
----	----	------

Field	Description				
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.				
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).				
L1	The length (ULONG) from the beginning of L1 to the end of RN.				
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).				
RN	The record number (ULONG) of the record in the record attribute list.				
L2	The length (ULONG) from the beginning of L2 to the end of the key value.				
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).				
KEY	The record key value.				
L3	The length (ULONG) from the beginning of L3 to the end of Data.				
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.				
	<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%;">X'144A'</td> <td>Indicates that the following data is record data.</td> </tr> <tr> <td>X'142D'</td> <td>Indicates that the following data is a ULONG length of an inactive record.</td> </tr> </table>	X'144A'	Indicates that the following data is record data.	X'142D'	Indicates that the following data is a ULONG length of an inactive record.
X'144A'	Indicates that the following data is record data.				
X'142D'	Indicates that the following data is a ULONG length of an inactive record.				
Data	The record data or the length (ULONG) of the inactive record.				

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.

DDMSetPlus

X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetPrevious (Set Cursor to Previous Record)

This function sets the cursor to the record that has a record number 1 less than the current cursor position and optionally returns the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetPrevious (HDDMFILE      FileHandle,
                      ULONG          AccessFlags,
                      PDDMRECORD    RecordBuf,
                      ULONG          RecordBufLen,
                      ULONG          RecCount,
                      PULONG        RecRtnCnt
                      );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
8–31	Reserved flags
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	DDM_ALLREC (All Records, Active and Inactive)
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. See “Examples” on page 306.

RecordBufLen

The length (ULONG) of the record buffer.

RecCount

Specifies the number (ULONG) of records requested.

DDMSetPrevious

RecRtnCnt

The pointer (PULONG) to the count of the records actually returned. When Record Attribute List (RECAL) parameters are specified in RecordBuf and RECCNT is specified within the RECAL, the RecRtnCnt parameter (ULONG) reflects the RECCNT number of duplicate records. Therefore, if RecordBuf contained 25 data records, one of which included a RECAL with RECCNT having a value of 150, the value of RecRtnCnt would be 175.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
ENDFILRM	X'120B'	End of File
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

If inactive records are bypassed (DDM_ALLREC not set), the cursor is set to the next active record whose record number is less than the current cursor position. For direct files, DDM_ALLREC must be false or a INVRQSRM reply will be sent.

As an option, DDMSetPrevious can:

- Set the hold cursor indicator to on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_RECNRFB).
- Place an update intent on the record (DDM_UPDINT).

If RecCount specifies a value greater than 1, multiple records are sent to the requester.

RecCount

specifies the number of times DDMSetPrevious is to be performed, with the following exceptions:

- For all iterations of the function except the last iteration, the RECINARM is not sent. All other reply messages resulting from the iteration of the function are sent.
- For the last iteration of the function, any reply message resulting from the last iteration of the function, including RECINARM, is sent.

This moves the cursor to the last record processed by DDMSetPrevious. Bypassed records (as a result of the DDM_ALLREC bit not being set) are *not* counted in RecCount. If RecCount specifies a number and DDM_NODATA is set, no records are returned.

If RecCount specifies a number greater than the remaining records in the file:

DDMSetPrevious

- The remaining records are sent to the requestor.
- The cursor position is changed.
- A ENDFILRM reply message is sent.

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

<i>Table 25. DDMSetPrevious (DDM_ALLREC or DDM_NODATA) Decision Table</i>					
If the DDMSetPrevious function is issued:					
When initial system states are:					
Record State	I	I	I	A	A
DDM_ALLREC	F	T	T	*	*
DDM_NODATA	*	F	T	F	T
The final system states are:					
	↓	↓	↓	↓	↓
RECINARM (returned)	F	F	T4	F	F
RECINA (returned)	F	T	F	F	F
RECORD (returned)	F	F	F	T	F
CURSOR (changed)	F	T	T	T	T
Repeat table after bypassing record	T	F	F	F	F
Legend					
A	Active				
I	Inactive				
T	TRUE (On)				
F	FALSE (Off)				
T4	TRUE with SVRCOD (Warning)				
*	Either TRUE or FALSE				

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

One of the following occurs:

- If DDM_ALLREC is set, the cursor is moved to the previous record in the file.
- If DDM_ALLREC is not set, the cursor is moved to the previous active record in the file
- If an ENDFILRM reply message results, the cursor is moved to BOF.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

DDMSetPrevious

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDDeleteRec).
- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions)

If the record lock is not obtained, the function is rejected with RECIUSRM.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetPrevious

Exceptions

This Causes the Function to Return and Continue	With This Reply Message
<p>The following are true:</p> <ul style="list-style-type: none"> The record is inactive. The DDM_NODATA flag is set. The DDM_ALLREC flag is set. <p>Note: If DDM_ALLREC is not set, this record is bypassed and the cursor is set to the previous record, as shown in Figure 70 on page 307.</p>	RECINARM

This Causes the Function to be Rejected	With This Reply Message
<p>DDM_RECNRFB or DDM_KEYVALFB is set, or DDM_NODATA is not set, and RecordBuf doesn't contain an address.</p> <p>Any data is to be returned and RecRtnCnt does not contain an address.</p>	ADDRRM
<p>The cursor is set to BOF in the following cases:</p> <ul style="list-style-type: none"> The file does not contain any records initially after a DDMCreateRecFile. The file does not contain any records before the current cursor position (or any inactive records when DDM_ALLREC is set). <p>Note: Any time the ENDFILRM is returned, the hold cursor indicator is set to OFF in the cursor regardless of the value specified for the DDM_HLDCSR flag.</p>	ENDFILRM
The file handle is invalid.	HDLNFNRM
Any reserved bits in AccessFlags are set.	INVFLGRM
<p>DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI specified.</p> <p>DDM_ALLREC(TRUE) is specified for a direct file.</p> <p>DDM_NODATA is not set and the file was opened without GETAI.</p>	INVRQSRM
The RecordBufLen value is not large enough to contain the number of records that are returned.	LENGTHRM
The record is damaged (record not active or inactive).	RECDMGRM
The record lock cannot be obtained.	RECIUSRM
The RecCount is not greater than 0.	VALNSPRM

DDMSetPrevious

Examples

Assume the following:

```
AccessFlags = 0x00000010 ; /* DDM_ALLREC = ON */  
RecCount = 1 ;
```

**DDMSetPrevious (FileHandle, AccessFlags, RecordBuf, RecordBufLen,
RecCount, RecRtnCnt)**

Has the following effect:

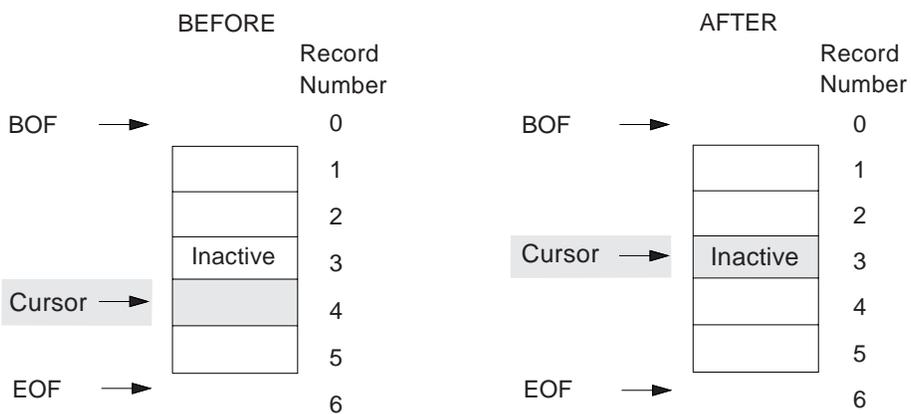


Figure 69. `DDMSetPrevious` Function with `DDM_ALLREC` Set to On

DDMSetPrevious

Assume the following:

```
AccessFlags = 0 ; /* DDM_ALLREC = OFF */
RecCount   = 1 ;
```

DDMSetPrevious (FileHandle, AccessFlags, RecordBuf, RecordBufLen, RecCount, RecRtnCnt)

Has the following effect:

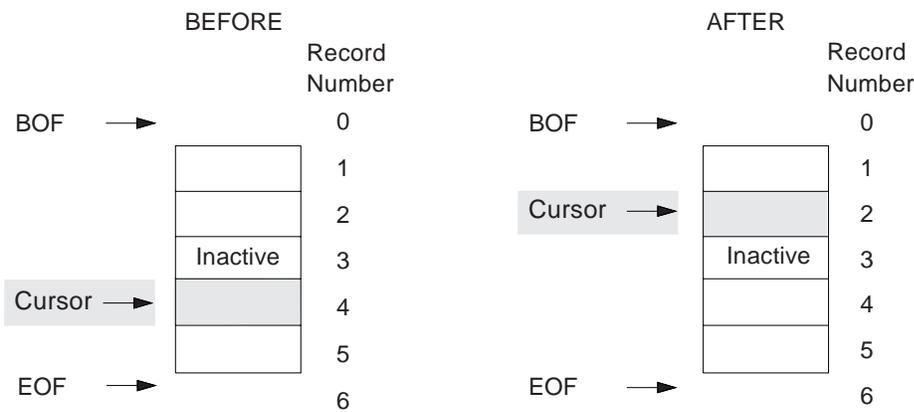


Figure 70. DDMSetPrevious Function with DDM_ALLREC Not Set

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	CP	Data
----	---------	----	---------	----	----	----	------

Field

Description

LL

The length (ULONG) of the record attribute list from the beginning of LL to the end of the record data.

X'1430'

The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).

DDMSetPrevious

L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count. The RC parameter is used to indicate the number of duplicate records. It provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list. Note: RC is not included unless identical, consecutive records are being returned.
L2	The length (ULONG) from the beginning of L2 to the end of data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.

DDMSetPrevious

X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).				
L1	The length (ULONG) from the beginning of L1 to the end of RC.				
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.				
RC	The number (ULONG) of duplicate records in the record attribute list.				
L2	The length (ULONG) from the beginning of L2 to the end of RN.				
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).				
RN	The record number (ULONG) of the record in the record attribute list. When RC and RN are both specified, the record number specified by RN applies to the first occurrence of the record and each subsequent record has a record number one greater than the previous record.				
L3	The length (ULONG) from the beginning of L3 to the end of Data.				
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.				
	<table border="0" style="margin-left: 2em;"> <tr> <td style="vertical-align: top;">X'144A'</td> <td>Indicates that the following data is record data.</td> </tr> <tr> <td style="vertical-align: top;">X'142D'</td> <td>Indicates that the following data is a ULONG length of an inactive record.</td> </tr> </table>	X'144A'	Indicates that the following data is record data.	X'142D'	Indicates that the following data is a ULONG length of an inactive record.
X'144A'	Indicates that the following data is record data.				
X'142D'	Indicates that the following data is a ULONG length of an inactive record.				
Data	The record data or the length (ULONG) of the inactive record.				

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.

DDMSetPrevious

X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG).

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.

DDMSetPrevious

Data The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

If RecCount is greater than one, the RecordBufLen must be provided in the record attribute list (RECAL).

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of the record key value.
X'1430'	The value (CODEPOINT) indicating that the following key is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following key is a key count. The RC parameter is used to indicate the number of duplicate keys. It provides a shorthand way of specifying N keys, where N>1, without replicating the key's contents.
RC	The number (ULONG) of duplicate keys in the record attribute list. Note: RC is not included unless identical, consecutive keys are being returned.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

DDMSetPrevious

L3	X'1115'	KEY	L4	CP	Data
----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, without replicating the record's contents.
RC	The number (ULONG) of duplicate records in the record attribute list.
L2	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

DDMSetPrevious

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetRecNum

DDMSetRecNum (Set Cursor to Record Number)

This function sets the cursor to the record of the file specified by RecordNumber and optionally returns the record and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetRecNum (HDDMFILE      FileHandle,
                    ULONG          AccessFlags,
                    RECNUM         RecordNumber,
                    PDDMRECORD     RecordBuf,
                    ULONG          RecordBufLen
                    );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
9–31	Reserved flags
8	DDM_ALWINA (Allow Cursor on Inactive Record)
7	DDM_HLDCSR (Hold Cursor Position)
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	DDM_RTININA (Return Inactive Record)
2	DDM_KEYVALFB (Key Value Feedback)
1	Reserved flag
0	DDM_UPDINT (Update Intent)

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

RecordNumber

Specifies the record number (ULONG) of the record to which the cursor should be moved.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Example” on page 318.

DDMSetRecNum

RecordBufLen

The length (ULONG) of the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
RECNBRRM	X'1224'	Record Number Out of Bounds
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

As an option, DDMSetRecNum can:

- Specify whether the cursor can be set to an inactive record position (DDM_ALWINA).
- Set the hold cursor indicator on (DDM_HLDCSR).
- Not return the requested record (DDM_NODATA).
- Specify whether inactive records should be returned (DDM_RTNINA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Place an update intent on the record (DDM_UPDINT).

If DDM_KEYVALFB flag is set and the file type is not keyed, the flag is ignored.

DDMSetRecNum

<i>Table 26. DDMSetRecNum (DDM_ALWINA, DDM_RTININA, or DDM_NODATA) Decision Table</i>							
If the DDMSetRecNum function is issued:							
When initial system states are:							
Record State	I	I	I	I	I	A	A
DDM_ALWINA	T	T	T	F	F	*	*
DDM_RTININA	T	*	F	*	*	*	*
DDM_NODATA	F	T	F	F	T	F	T
The final system states are:	↓	↓	↓	↓	↓	↓	↓
RECINARM (returned)	F	T4	T4	T8	T8	F	F
RECINA (returned)	T	F	F	F	F	F	F
RECORD (returned)	F	F	F	F	F	T	F
CURSOR (changed)	T	T	T	F	F	T	T
Legend							
A	Active						
I	Inactive						
T	TRUE (On)						
F	FALSE (Off)						
T4	TRUE with SVRCOD (Warning)						
T8	TRUE with SVRCOD (Error)						
*	Either TRUE or FALSE						

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor is moved to the record specified by RecordNumber.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If DDM_UPDINT(TRUE) is specified and the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if the record is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDDeleteRec).

DDMSetRecNum

- The cursor is moved to a different record.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnlockRec function is issued.
- Any function references a record other than the one currently pointed to by the cursor (for example, the DDMSetRecEOF, DDMSetRecKey, DDMSetRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with the RECIUSRM reply message.

If DDM_UPDINT(TRUE) is specified and the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to Return and Continue	With This Reply Message
The record is inactive and the DDM_ALWINA flag is set on, and either DDM_RTININA is set off or DDM_NODATA is set on.	RECINARM

This Causes the Function to be Rejected	With This Reply Message
If DDM_KEYVALFB is set or DDM_NODATA not set, RecordBuf does not contain a valid address.	ADDRRM
The file handle is invalid.	HDLNFNRM
Any reserved bits in AccessFlags are set.	INVFLGRM
DDM_UPDINT(TRUE) is specified and the file was opened without DELAI or MODAI. DDM_NODATA is not set and the file was opened without GETAI.	INVRQSRM
The record is damaged (not an active or inactive record).	RECDMGRM
The record is inactive and the DDM_ALWINA flag is set off. Note: The cursor position is not changed.	RECINARM
The record lock cannot be obtained.	RECIUSRM

DDMSetRecNum

This Causes the Function to be Rejected	With This Reply Message
<p>The specified record number (RecordNumber) is outside the bounds of the file.</p> <p>Note: File boundaries are discussed in DDMInsertRecNum on page 100.</p> <p>Note: The cursor position is not changed.</p>	RECNBRRM

Example

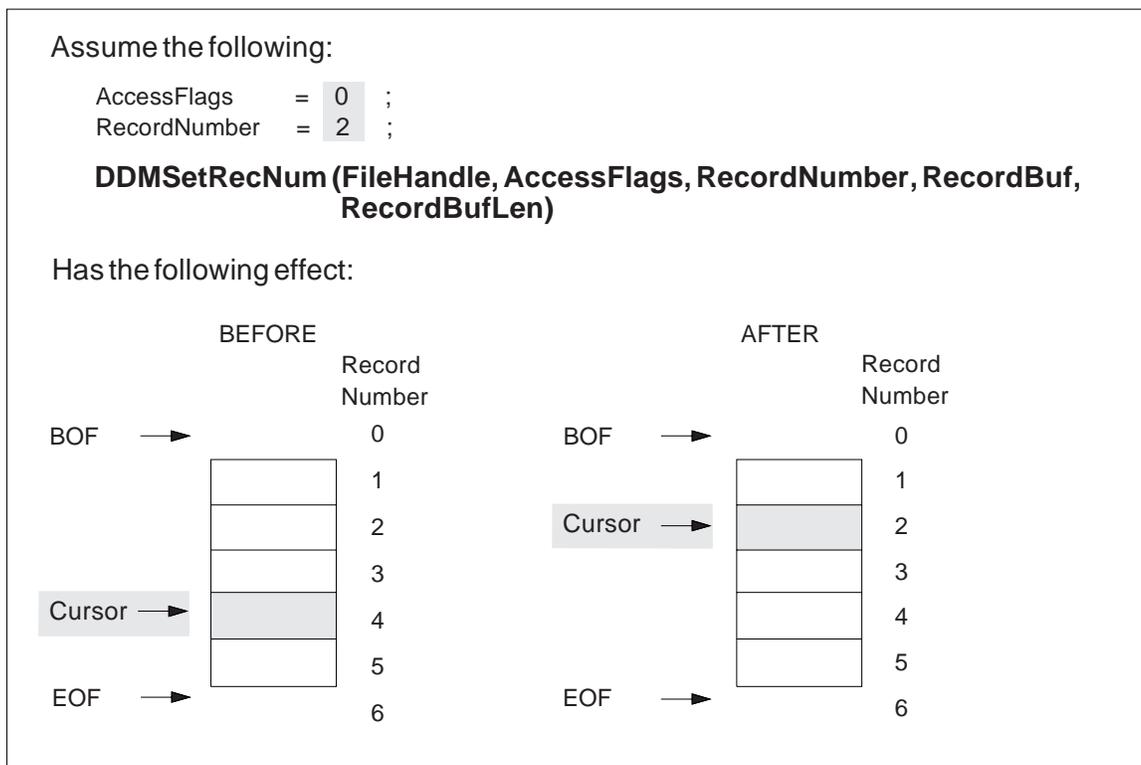


Figure 71. DDMSetRecNum Function

These are examples of RecordBuf data formats:

AccessFlags

DDM_KEYVALFB(FALSE) and & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

DDMSetRecNum

Field	Description
LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(FALSE) and DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(TRUE) and DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) of the field from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) of the field from the beginning of L2 to the end of the key value.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.

DDMSetRecNum

	X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data		The record data or the length (ULONG) of the inactive record.

AccessFlags

DDM_KEYVALFB(TRUE) and DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetUpdateKey

DDMSetUpdateKey (Set Update Intent by Key Value)

This function places an update intent on the record having a key value equal to the key value specified in KeyValBuf (Key Value Buffer). This function can also return the record, the record number, and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetUpdateKey (HDDMFILE      FileHandle,
                       ULONG          AccessFlags,
                       PDDMOBJECT     KeyValBuf,
                       PDDMRECORD     RecordBuf,
                       ULONG          RecordBufLen
                       );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMSOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
6–31	Reserved flags
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	Reserved flag
2	DDM_KEYVALFB (Key Value Feedback)
1	DDM_RECNRFB (Record Number Feedback)
0	Reserved flag

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

KeyValBuf

The pointer (PDDMOBJECT) to the key value buffer for the key of the record on which update intent is placed. The format of the key value buffer upon invocation of the function is:

LL	X'1115'	Key Value
----	---------	-----------

Field	Description
LL	The length (ULONG) of the key value description from the beginning of LL to the end of Key Value.

DDMSetUpdateKey

X'1115' The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Example” on page 325.

Field	Description
-------	-------------

LL	The length (ULONG) of the response from the beginning of LL to the end of record data.
----	--

X'144A'	The value (CODEPOINT) indicating that the following data is record data.
----------------	--

RecordBufLen

The length (ULONG) of the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
KEYLENRM	X'122D'	Invalid Key Length
KEYVALRM	X'1240'	Invalid Key Value
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECIUSRM	X'124A'	Record in Use
REC�FNRM	X'1225'	Record Not Found
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

Partial key values are valid for the DDMSetUpdateKey function. The first record selected receives the update intent.

If the key value specified in key value buffer has duplicate entries in the file (duplicate keys), the first record, in key sequence, of all records with the duplicate key value will have the update intent placed on it.

As an option, DDMSetUpdateKey can:

- Not return the requested record (DDM_NODATA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).
- Specify whether the record number should be returned (DDM_REC�BRFB).

DDMSetUpdateKey

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is the same as before the function was issued.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDelRec).
- The cursor is moved to a different record.
- A DDMGetRec with DDM_UPDINT(TRUE) is issued.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function issued references a record other than the one currently pointed to by the cursor (for example, the DDMSInsertRecEOF, DDMSInsertRecKey, DDMSInsertRecNum, DDMSSetUpdateKey, and DDMSSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

DDMSetUpdateKey

Exceptions

This Causes the Function to be Rejected	With This Reply Message
If DDM_KEYVALFB or DDM_RECNRFB is set, or DDM_NODATA not set, and RecordBuf does not contain a valid address.	ADDRRM
The file handle is invalid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM
The file was opened without DELAI or MODAI specified. The access method is not valid for this function. DDM_NODATA is not set and the file was opened without GETAI.	INVRQSRM
The key length specified for the key value is larger than the key length used to build the index.	KEYLENRM
The record lock cannot be obtained.	RECIUSRM
The file does not contain any records initially after a DDMCreateRecFile A record does not exist with a key value equal to the value contained in KeyValBuf.	REC�FNRM
RecordNumber is invalid.	VALNSPRM

DDMSetUpdateKey

Example

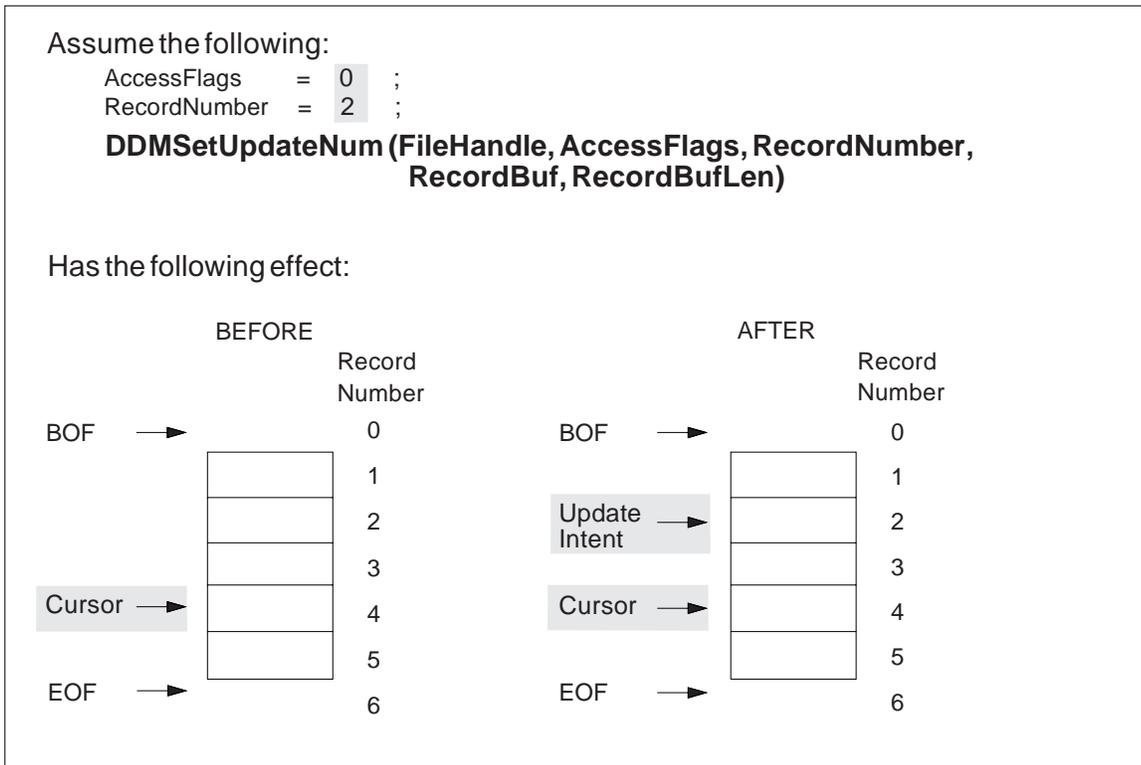


Figure 72. `DDMSetUpdateKey` Function

These are examples of `RecordBuf` data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	X'144A'	Data
----	---------	------

Field Description

LL	The length (ULONG) of the record buffer from the beginning of LL to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetUpdateKey

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(FALSE) &
DDM_NODATA(TRUE)

RecordBuf

Nothing is returned.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'144A'	Data
----	---------	----	---------	----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list. A value of X'FFFFFFFF' for RN indicates that the record number of the first record in the record attribute list is not known.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(FALSE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'111D'	RN
----	---------	----

DDMSetUpdateKey

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG). A value of X'FFFFFFFF' for RN indicates that the record number is not known.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	X'144A'	Data
----	---------	----	---------	-----	----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1115'	KEY
----	---------	-----

DDMSetUpdateKey

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(FALSE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

L3	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L3	The length (ULONG) from the beginning of L3 to the end of Data.
X'144A'	The value (CODEPOINT) indicating that the following data is record data.
Data	The record data.

DDMSetUpdateKey

AccessFlags

DDM_KEYVALFB(TRUE) & DDM_RECNRFB(TRUE) &
DDM_NODATA(TRUE)

RecordBuf

DATA FORMAT

LL	X'1430'	L1	X'111D'	RN	L2	X'1115'	KEY
----	---------	----	---------	----	----	---------	-----

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of KEY.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L2	The length (ULONG) from the beginning of L2 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMSetUpdateNum

DDMSetUpdateNum (Set Update Intent by Record Number)

This function places an update intent on the record of the file that is indicated by the RecordNumber parameter and optionally returns the record and record key.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMSetUpdateNum (HDDMFILE      FileHandle,
                        ULONG          AccessFlags,
                        RECNUM         RecordNumber,
                        PDDMRECORD     RecordBuf,
                        ULONG          RecordBufLen
                        );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
9–31	Reserved flags
8	!DDM_ALWINA (Allow Update Intent on Inactive Record)
7	Reserved flag
6	Reserved flag
5	DDM_NODATA (No Record Data Returned)
4	Reserved flag
3	DDM_RTININA (Return Inactive Record)
2	DDM_KEYVALFB (Key Value Feedback)
0–1	Reserved flags

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

DDMSetUpdateNum

RecordNumber

Specifies the record number (ULONG) of the record on which update intent is placed.

RecordBuf

The pointer (PDDMRECORD) to the record buffer for the returned data. The format of the returned data in the buffer depends on the bit settings in AccessFlags. Examples of the returned data formats can be found in “Example” on page 334.

RecordBufLen

The length (ULONG) of the record buffer.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flag
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record in Use
RECNBRRM	X'1224'	Record Number Out of Bounds

Remarks

As an option, DDMSetUpdateNum can:

- Specify whether an update intent can be placed on an inactive record position (DDM_ALWINA).
- Not return the requested record (DDM_NODATA).
- Specify whether inactive records should be returned (DDM_RTNINA).
- Specify whether the record key value should be returned (DDM_KEYVALFB).

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is the same as before the function was issued.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

DDMSetUpdateNum

Locking (for Local VSAM File System Only)

Record locks apply only to OS/2 local VSAM files on the client OS/2 workstation. For other local VSAM files, locking occurs at the file level.

If the file was opened for multiple updaters, the access method acquires an implicit SHRRECLK on the record if it is not already locked by the requester with a SHRRECLK lock. The SHRRECLK record lock is released when:

- The record is updated (DDMModifyRec or DDMDelRec).
- The cursor is moved to a different record.
- A DDMGetRec with DDM_UPDINT(TRUE) is issued.
- The file is closed.
- The DDMForceBuffer function is issued.
- The DDMUnLockRec function is issued.
- Any function issued references a record other than the one currently pointed to by the cursor (for example, the DDMInsertRecEOF, DDMInsertRecKey, DDMInsertRecNum, DDMSetUpdateKey, and DDMSetUpdateNum functions).

If the record lock is not obtained, the function is rejected with a RECIUSRM reply message.

If the file was *not* opened for multiple updaters, an update intent is placed on the record, but the access method does *not* acquire any record locks.

If the function terminates with a reply message that has a severity code of ERROR or higher, then:

- For error termination (SVRCOD of 8): The record locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the record locks is determined by the DTALCKST (Data Lock Status) parameter on the reply message.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
If DDM_KEYVALFB or DDM_RECNRFB is set, or DDM_NODATA is not set, and RecordBuf does not contain a valid address.	ADDRRM
The file handle is invalid.	HDLNFRM
Any reserved bits in AccessFlags are set.	INVFLGRM
The file was opened without DELAI or MODAI specified. The file was opened with a GETAI access intent and DDM_NODATA(FALSE) was specified.	INVRQSRM

DDMSetUpdateNum

This Causes the Function to be Rejected	With This Reply Message
The RecordBufLen value is not large enough to contain the number of records that are returned.	LENGTHRM
The record position is inactive and an update intent is not allowed to be set to an inactive record position (DDM_ALWINA not set).	RECINARM
The record lock cannot be obtained.	RECIUSRM
The specified record number (RecordNumber) is outside the bounds of the file. Note: The cursor position is not changed.	RECNBRRM

Table 27. DDMSetUpdateNum (DDM_ALWINA, DDM_RTININA, or DDM_NODATA) Decision Table

If the DDMSetUpdateNum function is issued:								
When initial system states are:								
Record State	I	I	I	I	I	A	A	
DDM_ALWINA	T	T	T	F	F	*	*	
DDM_RTININA	T	*	F	*	*	*	*	
DDM_NODATA	F	T	F	F	T	F	T	
The final system states are:	↓	↓	↓	↓	↓	↓	↓	
RECINARM (returned)	F	T4	T4	T8	T8	F	F	
RECINA (returned)	T	F	F	F	F	F	F	
RECORD (returned)	F	F	F	F	F	T	F	
CURSOR (changed, see note)	F	F	F	F	F	F	F	
Legend								
A	Active							
I	Inactive							
T	TRUE (On)							
F	FALSE (Off)							
T4	TRUE with SVRCOD (Warning)							
T8	TRUE with SVRCOD (Error)							
*	Either TRUE or FALSE							
Note: The cursor position does not change.								

DDMSetUpdateNum

Example

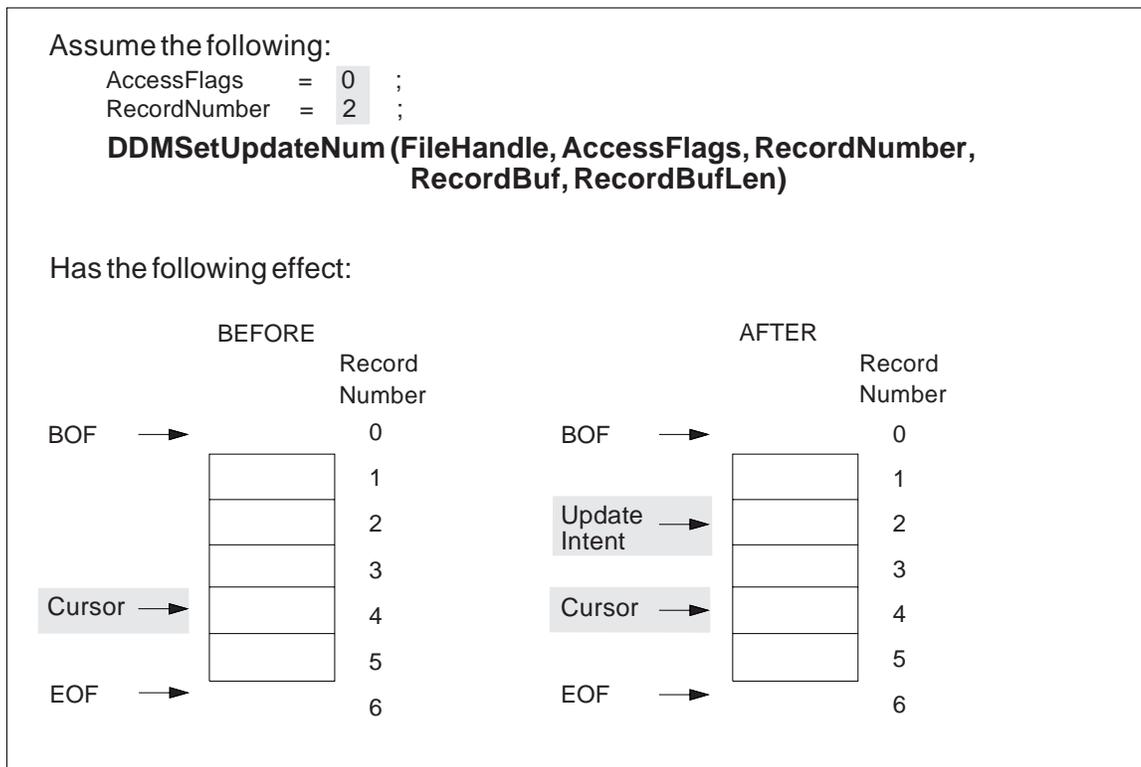


Figure 73. `DDMSetUpdateNum` Function

These are examples of RecordBuf data formats:

AccessFlags

```
DDM_KEYVALFB(FALSE) & DDM_NODATA(FALSE)
```

RecordBuf

DATA FORMAT

LL	CP	Data
----	----	------

Field

Description

LL

The length (ULONG) of the record buffer (from the beginning of LL to the end of Data).

CP

The value (CODEPOINT) indicating that the following is either record data or an inactive record length.

X'144A'

Indicates that the following data is record data.

DDMSetUpdateNum

X'142D' Indicates that the following data is a ULONG length of an inactive record.

Data The record data or the length (ULONG) of the inactive record.

AccessFlags
DDM_KEYVALFB(FALSE) & DDM_NODATA(TRUE)

RecordBuf
Nothing is returned.

AccessFlags
DDM_KEYVALFB(TRUE) & DDM_NODATA(FALSE)

RecordBuf
DATA FORMAT

LL	X'1430'	L1	X'1115'	KEY	L2	CP	Data
----	---------	----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data.
X'1430'	The value (CODEPOINT) indicating that the following data is a record attribute list (RECAL).
L1	The length (ULONG) from the beginning of L1 to the end of the key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.
L2	The length (ULONG) from the beginning of L2 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
	X'144A' Indicates that the following data is record data.
	X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of the inactive record.

AccessFlags
DDM_KEYVALFB(TRUE) & DDM_NODATA(TRUE)

RecordBuf
DATA FORMAT

DDMSetUpdateNum

LL	X'1115'	KEY
----	---------	-----

Field	Description
LL	The length (ULONG) from the beginning of LL to the end of the Key value.
X'1115'	The value (CODEPOINT) indicating that the following data is a key value (KEYVAL).
KEY	The record key value.

DDMTruncFile (Move EOF to Current Cursor Position)

This function moves EOF to the current cursor position.

The records starting at the current cursor position and ending at the old EOF are eliminated.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMTruncFile (HDDMFILE      FileHandle
                    );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMOpen.

Returns

Message ID	Code Point	Message Title
FILIUSRM	X'120D'	File in Use
HDLNFNRM	X'1257'	File Handle Not Found
INVRQSRM	X'123C'	Invalid Request

Remarks

This function is only valid for sequential files.

This function physically shortens the file by the number of records eliminated. Any data in these records is permanently lost.

Effect on Cursor Position

After this function is successfully completed, the cursor is set to the new EOF.

Locking (for Local VSAM File System Only)

The file must be opened with MODNONLK.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file was not opened with MODNONLK.	FILIUSRM
The file handle is invalid.	HDLNFNRM
A direct, keyed, or alternate index file is accessed. MODAI access intent was not specified when the file was opened.	INVRQSRM

DDMTruncFile

Example

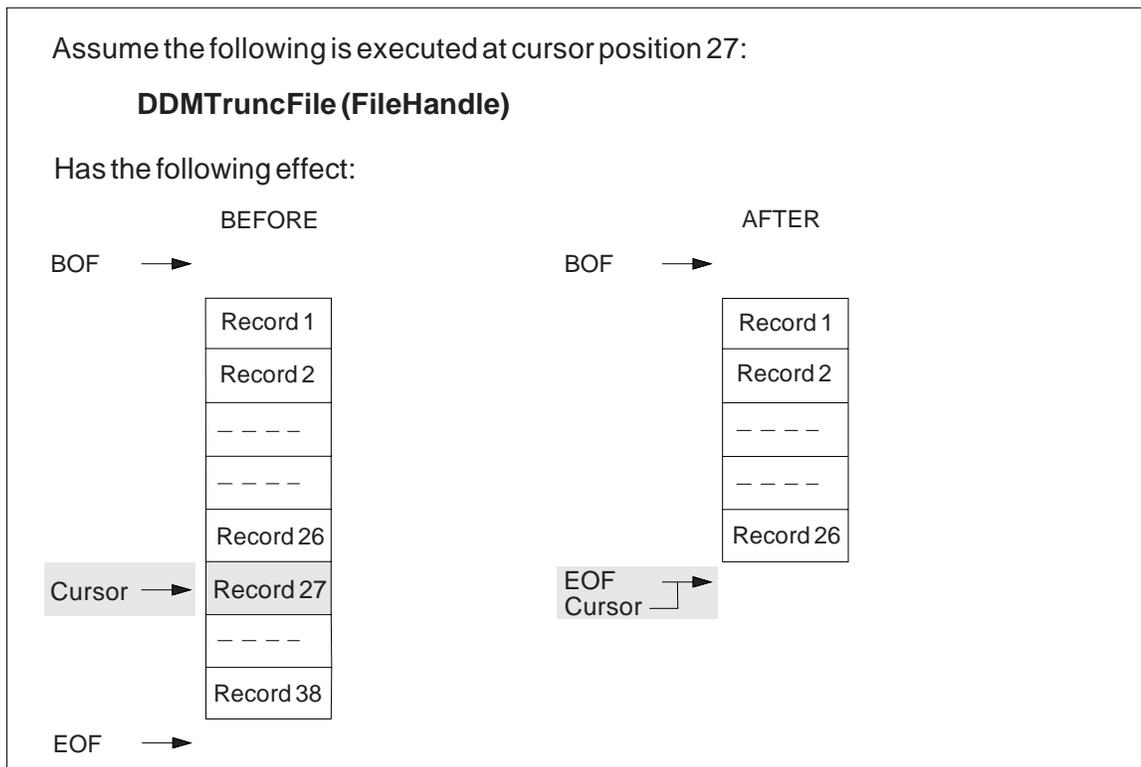


Figure 74. `DDMTruncFile` Function

DDMUnLoadFileFirst

DDMUnLoadFileFirst (Unload Records from File)

This function unloads records from a file and transfers them to the requester's buffers.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMUnLoadFileFirst (PSZ          FileName,
                          PHDDMLOAD    UnLoadHandle,
                          ULONG         AccessFlags,
                          PULONG       Flags,
                          PDDMRECORD   RecordBuf,
                          ULONG         RecordBufLen,
                          CODEPOINT    UnloadOrder,
                          PULONG       RecCount
                          );
```

Parameters

FileName

The pointer (PSZ) to the name of the record-oriented file to be unloaded to the requester's buffers.

UnLoadHandle

The pointer (PHDDMLOAD) to the location where the system returns a handle value that is to be used with a subsequent corresponding DDMUnLoadFileNext function.

AccessFlags

The AccessFlags (ULONG) specify the action to be taken depending on whether the bit flag is set. The bit flags are:

Bit	Meaning
7–31	Reserved flags
6	DDM_BYPDMG (Bypass Damaged Records)
4–5	Reserved flags
3	DDM_RTNINA (Return Inactive Record)
0–2	Reserved flags

For detailed information on the access flags, see Chapter 5, “VSAM API Flags” on page 399.

Flags

The pointer (PULONG) to the bit flags parameter. The bit flags are:

Bit	Meaning
1–31	Reserved flags
0	DDM_MOREDATA flag.

The system sets this bit upon return from DDMUnLoadFileFirst if the record buffer is not large enough to hold all of the target file's

DDMUnLoadFileFirst

existing records. This flag bit notifies the user to issue a subsequent DDMUnLoadFileNext in order to continue the unload function. When the DDM_MOREDATA flag bit is off, the system has completed the unload of the entire file and a NULL value is returned for UnLoadHandle.

RecordBuf

The pointer (PDDMRECORD) to the record buffer. The returned record buffer can contain the following objects:

RECORD
RECAL

These objects can be in mixed order and can be repeated. The format of the record buffer upon return of the function is:

LL	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following data is either record data or a RECAL (Record Attribute List) containing a record number and record data.
X'144A'	Indicates that the following data is record data.
X'1430'	Indicates that the following data is a RECAL (Record Attribute List). A RECAL object is returned when inactive records have been encountered. This is not used when unloading in key order.

If CP is a record attribute list, the format of Data is:

LL	X'111A'	RC	L3	X'111D'	RN	L4	CP	Data
----	---------	----	----	---------	----	----	----	------

Field	Description
L2	The length (ULONG) from the beginning of L2 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate inactive records, where $N \geq 1$.
RC	The number (ULONG) of duplicate records in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of RN.

DDMUnLoadFileFirst

X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNBR).
RN	The record number (ULONG) of the record in the record attribute list.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data.
X'142D'	Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of an inactive record.

If CP is record data, the format is RECORD.

RecordBufLen

The length (ULONG) of the record buffer.

UnloadOrder

(CODEPOINT) Specifies the order in which the function processes the records in the file. The valid values are:

RNBORD	Record Number Order (X'145E')
KEYORD	Key Order Processing (X'145D')

RecCount

The count (ULONG) of the record descriptions in the record buffer.

The number of record descriptions (record data and inactive record lengths) should be the same number as indicated in the record count. When a RECAL (Record Attribute List) is specified in RecordBuf and RECCNT of N is specified within the RECAL, the RecCount parameter reflects the N duplicate records. Therefore, if RecordBuf contained 10 data records and a RECAL, and RECCNT had a value of 100, the value of RecCount would be 110.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
ENDFILRM	X'120B'	End of File
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILIUSRM	X'120D'	File in Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged
VALNSPRM	X'1252'	Parameter Value Not Supported

DDMUnloadFileFirst

Remarks

The DDMUnloadFileFirst function unloads the records of the file in the order specified by UnloadOrder. If the file is not an alternate index or keyed file, then the specified value of UnloadOrder is ignored and the records are unloaded in record number order.

- To unload in record number order:
 1. The DDMUnloadFileFirst function unloads records from a file in a sequence order that begins with the first record and proceeds through the remainder of the file.
 2. If the DDM_RTNINA flag is not set, inactive records are not returned. A Record Attribute List (RECAL) is placed in the RecordBuf that includes the record number (RECNBR) of the next active record in the file. The RECAL also includes the active record.
 3. If inactive records are to be returned, a RECAL object that includes a RECCNT object is placed in the RecordBuf. RECCNT contains the number of duplicate inactive records. RECAL also includes the inactive records.
- To unload in key order:
 1. The DDMUnloadFileFirst function unloads records from a file beginning with the first record in the key sequence and proceeding sequentially through the file in key order.
 2. If the DDM_RTNINA flag is set, then it is ignored for this unload order.

- For all unload orders:

The RecCount is the actual number of records sent on each request; it does not include inactive records that are not returned. The RecCount permits the requester to verify that the total number of records sent is correct. If the total RecCount does not match, the requester can re-issue the DDMUnloadFileFirst function. Before issuing DDMUnloadFileFirst, the requester must first close UnLoadHandle by issuing a DDMUnloadFileNext function with the DDM_CLOSEUNLOAD bit set.

The user can specify that damaged records be bypassed. For each record bypassed, a RECDMGRM reply message with a warning (SVRCOD of 4) is returned. Bypassed damaged records are *not* counted as part of the RecCount.

Multiple DDMUnloadFileFirst functions may be issued on the same file without issuing the corresponding DDMUnloadFileNext close functions. Each DDMUnloadFileFirst function returns a unique unload handle. This allows more than one unload cursor to be active at the same time on the same file.

If an error condition is encountered, do not use the file handle in a DDMUnloadFileNext.

Effect on Cursor Position

There is no effect on the cursor position.

DDMUnloadFileFirst

Locking (for Local VSAM File System Only)

DDMUnloadFileFirst attempts to:

1. Obtain a GETGETLK lock on the file.
If the GETGETLK lock is obtained, the DDMUnloadFileFirst is processed (successfully or unsuccessfully).
If the GETGETLK lock is not obtained, the DDMUnloadFileFirst is rejected with a FILIUSRM reply message.
2. Release the GETGETLK it obtained on the file if the DDM_MOREDATA flag is not active. If the DDM_MOREDATA flag is active, the lock is released by the DDMUnloadFileNext function, provided the Close UnloadFile flag is set.

If the function terminates with a reply message that has a severity code of ERROR or higher then:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The file that the records are loaded into is a non-VSAM file.	ACCATHRM
The RecordBuf address is NULL. The address for the Flags is not valid.	ADDRRM
The file to be unloaded is empty.	ENDFILRM
The file has already been opened by DDMOpen or DDMLoadFileFirst (DDM_CHAIN flag on).	FILIUSRM
Any of the reserved bits are set in AccessFlags.	INVFLGRM
UnLoadHandle is not specified.	INVRQSRM
The RecordBufLen value is not large enough for at least 1 record.	LENGTHRM
UnloadOrder parameter does not contain a correct value.	VALNSPRM

DDMUnloadFileFirst

This Causes a Reply Message to be Generated with SRVCOD = X'04' for each out-of-sync file in the file object. The Function Continues	With This Reply Message
<p>If the file-change date and time recorded by the VSAM API is not the same as that recorded by the file system, either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.</p> <p>DDMUnloadFileFirst and DDMUnloadFileNext will not re-synchronize the file-change date and time during close processing.</p>	FILDMGRM

DDMUnloadFileFirst

Examples

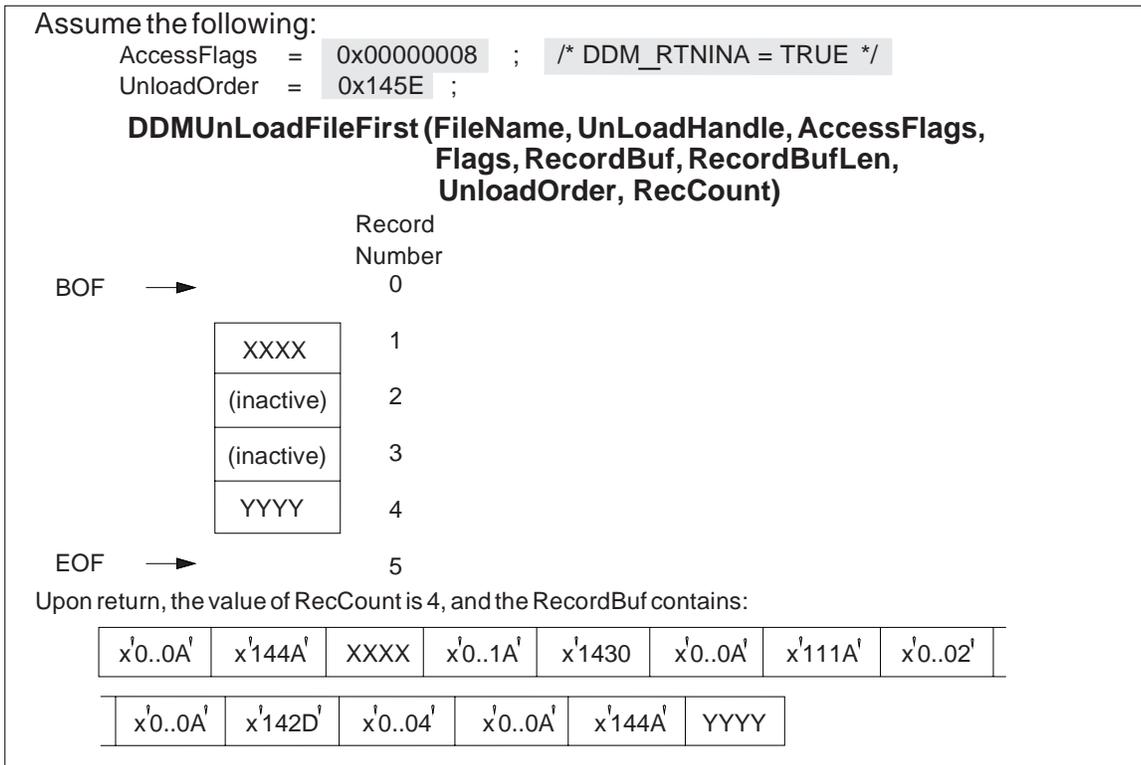


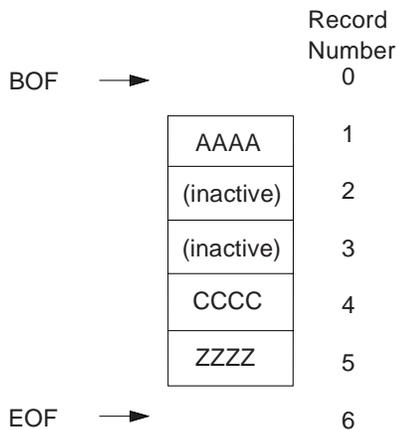
Figure 75. DDMUnloadFileFirst Function When Returning Active or Inactive Records

DDMUnLoadFileFirst

Assume the following:

```
AccessFlags = 0 ; /* DDM_RTNINA = FALSE */
UnloadOrder = 0x145E ;
```

DDMUnLoadFileFirst (FileName, UnLoadHandle, AccessFlags, Flags, RecordBuf, RecordBufLen, UnloadOrder, RecCount)



Upon return, the value of RecCount is 3, and the RecordBuf contains:

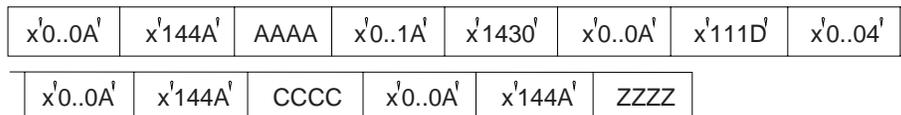


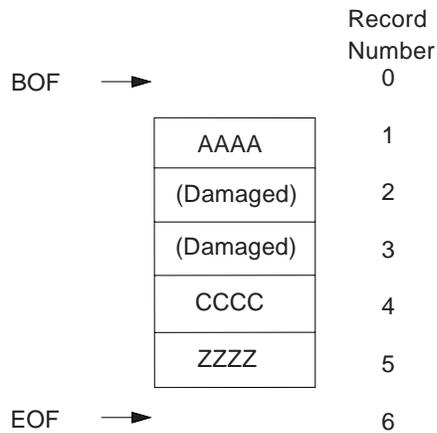
Figure 76. DDMUnLoadFileFirst Function Skipping Inactive Records

DDMUnLoadFileFirst

Assume the following:

```
AccessFlags = 0x00000040 ; /* DDM_BYPDMG = TRUE */
```

DDMUnLoadFileFirst (FileName, UnLoadHandle, AccessFlags, Flags, RecordBuf, RecordBufLen, UnloadOrder, RecCount)



Upon return, the RecCount value is 3, and the RecordBuf contains:

x'0..0A'	x'144A'	AAAA	x'0..0A'	x'144A'	CCCC	x'0..0A'	x'144A'	ZZZZ
----------	---------	------	----------	---------	------	----------	---------	------

In addition, two reply messages are sent, one for each damaged record encountered. The reply message is RECDMGRM. The reply messages contain the record numbers of the damaged records.

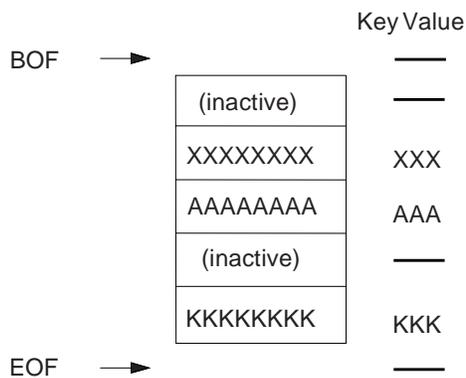
Figure 77. DDMUnLoadFileFirst Function Skipping Damaged Records

DDMUnloadFileFirst

Assume the following:

```
AccessFlags = 0 ; /* DDM_RTININA = False */
UnloadOrder = 0x145D ; /* KeyOrderProcessing */
```

DDMUnloadFileFirst (FileName, UnLoadHandle, AccessFlags, Flags, RecordBuf, RecordBufLen, UnloadOrder, RecCount)



Upon return, the value of RecCount is 3, and the RecordBuf contains:

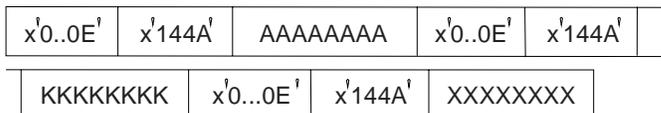


Figure 78. DDMUnloadFileFirst Function Unloading in Key Order

DDMUnLoadFileNext

DDMUnLoadFileNext (Unload Records from File)

This function unloads records from a target server file and transfers them to the requester.

Syntax

```
#include dub.h /* Required for all platforms */

APIRET DDMUnLoadFileNext (HDDMLOAD      UnLoadHandle,
                          ULONG          Flags,
                          PULONG        UnloadFlags,
                          PDDMRECORD    RecordBuf,
                          ULONG          RecordBufLen,
                          PULONG        RecCount
                          );
```

Parameters

UnLoadHandle

The handle value (HDDMLOAD) returned previously to the requester in a corresponding DDMUnLoadFileFirst function.

Flags

The bit flags (ULONG) parameter. The bit flags are:

Bit	Meaning
-----	---------

2–31	Reserved flags
------	----------------

1	DDM_CLOSEUNLOAD flag.
---	-----------------------

The user has the option of setting this flag bit and notifying the system to terminate UnLoadHandle based chaining (for the DDM_MOREDATA flag) and de-allocate UnLoadHandle based system resources for this UnLoadFile function. This flag provides the user with a way of prematurely terminating the unload file operation quickly without having to wait until the entire file has been unloaded.

When this flag is set, no records will be unloaded and the UnloadFlags and RecCount will not be set.

0	Reserved flag.
---	----------------

UnloadFlags

The pointer (PULONG) to the bit unload flags parameter. The bit flags are:

Bit	Meaning
-----	---------

1–31	Reserved flags
------	----------------

DDMUnLoadFileNext

- 0** DDM_MOREDATA flag.
- The system sets this bit upon return from DDMUnLoadFileNext, if the record buffer is not large enough to hold all of the target file's existing records. This flag bit notifies the user to issue a subsequent DDMUnLoadFileNext in order to continue the unload file function. When the DDM_MOREDATA flag bit is off, the system has completed the unload of the entire file and has de-allocated all previously allocated system resources based on UnLoadHandle. No user-initiated action is required to terminate this function.

RecordBuf

The pointer (PDDMRECORD) to the record buffer. The returned record buffer can contain the following objects:

RECORD
RECAL

These objects can be in mixed order and can be repeated. The format of the record buffer upon return of the function is:

LL	CP	Data
----	----	------

Field	Description
LL	The length (ULONG) of the record description from the beginning of LL to the end of Data.
CP	The value (CODEPOINT) indicating that the following data is either record data or a record attribute list containing a record number and record data.
X'144A'	Indicates that the following data is record data.
X'1430'	Indicates that the following data is a record attribute list.

A RECAL object is returned when inactive records have been encountered. This is not used when unloading in key order.

- If CP is a record attribute list, the format of DATA is:

LL	X'111A'	RC	L3	X'111D'	RN	L4	CP	Data
----	---------	----	----	---------	----	----	----	------

Field	Description
L2	The length (ULONG) from the beginning of L2 to the end of RC.
X'111A'	The value (CODEPOINT) indicating that the following data is a record count (RECCNT). The RECCNT parameter is used to indicate the number of duplicate inactive records, where $N \geq 1$.

DDMUnLoadFileNext

RC	The number (ULONG) of duplicate records in the record attribute list.
L3	The length (ULONG) from the beginning of L3 to the end of RN.
X'111D'	The value (CODEPOINT) indicating that the following data is a record number (RECNR).
RN	The record number (ULONG) of the record in the record attribute list.
L4	The length (ULONG) from the beginning of L4 to the end of Data.
CP	The value (CODEPOINT) indicating that the following is either record data or an inactive record length. X'144A' Indicates that the following data is record data. X'142D' Indicates that the following data is a ULONG length of an inactive record.
Data	The record data or the length (ULONG) of an inactive record.

- If CP is record data, the format is record.

The number of record descriptions (record data and inactive record lengths) should be the same as the number indicated in the record count.

RecordBufLen

The length (ULONG) of the record buffer.

RecCount

The count (ULONG) of the record descriptions in the record buffer.

The number of record descriptions (record data and inactive record lengths) should be the same number as indicated in the record count. When a RECAL (Record Attribute List) is specified in RecordBuf and RECCNT of N is specified within the RECAL, the RecCount parameter reflects the N duplicate records. Therefore if RecordBuf contained 10 data records and a RECAL, with RECCNT having a value of 100, the value of RecCount would be 110.

Returns

Message ID	Code Point	Message Title
ADDRRM	X'F212'	Address Error
HDLNFNRM	X'1257'	File Handle Not Found
INVFLGRM	X'F205'	Invalid Flags
INVRQSRM	X'123C'	Invalid Request
LENGTHRM	X'F211'	Field Length Error
RECDMGRM	X'1249'	Record Damaged

DDMUnloadFileNext

Message ID	Code Point	Message Title
VALNSPRM	X'1252'	Parameter Value Not Supported

Remarks

DDMUnloadFileNext starts unloading records for the position in the file that was current from a previous DDMUnloadFileNext or DDMUnloadFileFirst function.

DDMUnloadFileNext unloads the records of the file in the order specified by the UnloadOrder parameter of the originating DDMUnloadFileFirst function. If the file is not an alternate index or keyed file, the specified value of the originating DDMUnloadFileFirst UnloadOrder parameter is ignored and the records are unloaded in record number order. DDM_RTNINA and DDM_BYPDGM flags that were set in DDMUnloadFileFirst are saved and used in the function.

- To load in record number order:
 1. DDMUnloadFileNext unloads records from a file in sequence order specified by the originating DDMUnloadFileFirst.
 2. If the originating access flag DDM_RTNINA was not set, inactive records are not returned. A RECAL (Record Attribute List) is placed in the RecordBuf that includes the record number (RECNBR) of the next active record in the file. The RECAL also includes the active record.
 3. If inactive records are to be returned, a RECAL object that includes a RECCNT object is placed in the RecordBuf. RECCNT contains the number of duplicate inactive records. RECAL also includes the inactive records.
- To unload in key order:
 1. DDMUnloadFileNext unloads records from a file beginning with the first record in the key sequence and proceeding sequentially through the file in key order.
 2. If the originating access flag DDM_RTNINA was set, then it is ignored for this unload order.
- For all unload orders:
 1. RecCount is the actual number of records transferred for each request; it does not include inactive records that are not returned. RecCount lets the requester verify that the total number of records transferred is correct.
 2. If the total record count does not match, the requester can optionally close the UnLoadHandle by issuing a DDMUnloadFileNext function with the DDM_CLOSEUNLOAD flag set and start a new unload file operation by issuing a DDMUnloadFileFirst function.

If an error condition is encountered, do not use the file handle in a DDMUnloadFileNext.

DDMUnloadFileNext

Effect on Cursor Position

There is no effect on the cursor position because the file is not open.

Locking (for Local VSAM File System Only)

DDMUnloadFileNext releases the GETGETLK lock that was obtained by DDMUnloadFileFirst on the file if no more data is to be unloaded or the DDM_CLOSEUNLOAD flag is set.

If DDMUnloadFileNext terminates with a reply message that has a severity code of ERROR or higher then:

- For error termination (SVRCOD of 8): The file locks are the same as before the function was issued.
- For severe termination (SVRCOD of 16 or higher): The state of the file locks may not be the same as before the function was issued.

Exceptions

This Causes the Function to be Rejected	With This Reply Message
The RecordBuf address is NULL. The address for the Flags is not valid.	ADDRRM
The handle from DDMUnloadFileFirst is not used as the handle for a DDMUnloadFileNext.	HDLNFNRM
Any of the reserved bits in AccessFlags are set.	INVFLGRM

This Causes the Function to be Terminated	With This Reply Message
The RecordBufLen value is not large enough for at least 1 record.	LENGTHRM

DDMUnLoadFileNext

Examples

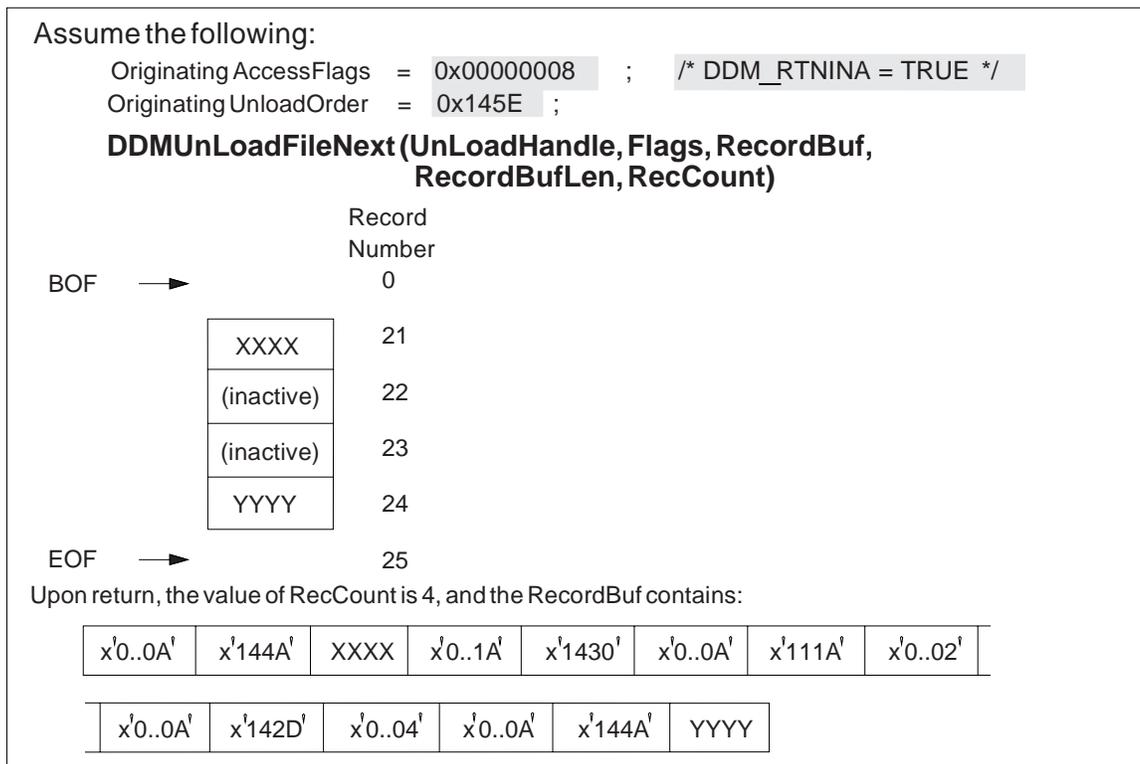


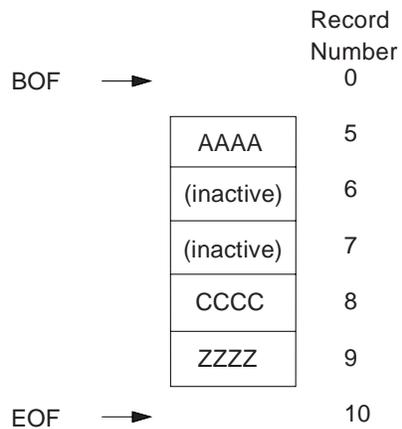
Figure 79. DDMUnLoadFileNext Function

DDMUnLoadFileNext

Assume the following:

```
Originating AccessFlags = 0x00000000 ; /* DDM__RTNINA = FALSE */
Originating UnloadOrder = 0x145E ;
```

**DDMUnLoadFileNext (UnLoadHandle, Flags, RecordBuf,
RecordBufLen, RecCount)**



Upon return, the value of RecCount is 3, and the RecordBuf contains:

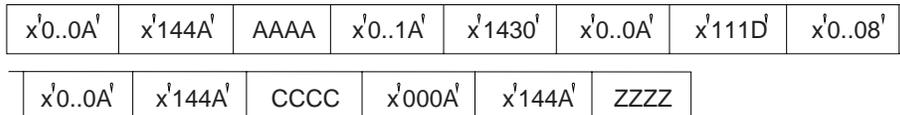


Figure 80. DDMUnLoadFileNext Function Skipping Inactive Records

DDMUnLoadFileNext

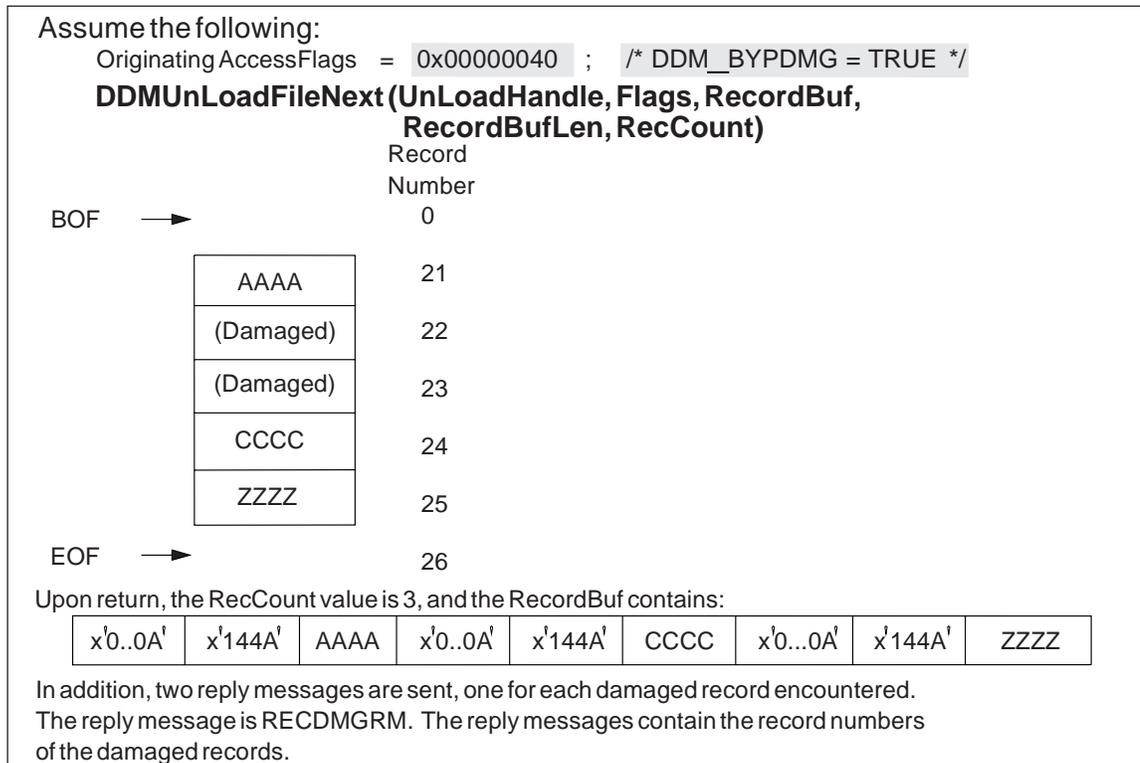


Figure 81. DDMUnLoadFileNext Function Skipping Damaged Records

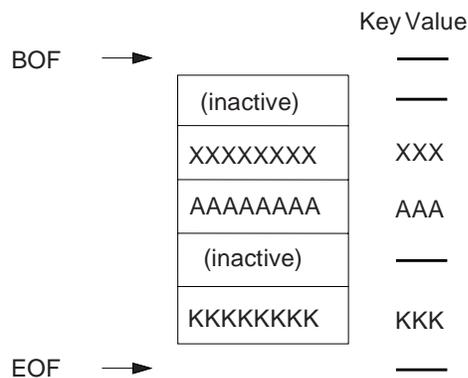
DDMUnLoadFileNext

Assume the following:

```

Originating AccessFlags = 0x00000000 ; /* DDM_RTNINA = False */
Originating Unloader    = 0x145D     ; /* Key Order Processing */
  
```

**DDMUnLoadFileNext (UnLoadHandle, Flags, RecordBuf,
RecordBufLen, RecCount)**



Upon return, the value of RecCount is 3, and the RecordBuf contains:

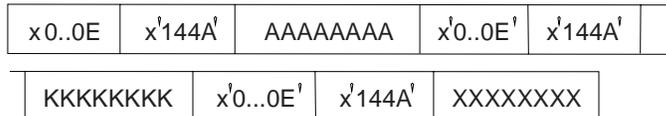


Figure 82. DDMUnLoadFileNext Function Unloading in Key Order

DDMUnlockRec

DDMUnlockRec (Unlock Implicit Record Lock)

This function releases any implicit record lock (that is, an update intent) currently held by the cursor.

Syntax

```
#include dub.h /* Required for all platforms */  
  
APIRET DDMUnlockRec (HDDMFILE      FileHandle  
                    );
```

Parameters

FileHandle

The file handle (HDDMFILE) obtained from DDMSOpen.

Returns

Message ID	Code Point	Message Title
EXSCNDRM	X'123A'	Existing Condition
HDLNFNRM	X'1257'	File Handle Not Found

Remarks

The DDM architecture requires that an "update intent" be set for a record before the record can be deleted or modified.

For the AIX local VSAM file system, DDMUnlockRec releases the update intent (if any) currently in place. However, concurrent data access control is done at the file level (that is, record locking is not supported).

Effect on Cursor Position

Normal Completion (SVRCOD of 0 or 4)

The cursor position is not changed.

Error Termination (SVRCOD of 8)

The cursor position is the same as before the function was issued.

Severe Termination (SVRCOD of 16 or higher)

The cursor position is determined by the CSRPOSST (Cursor Position Status) parameter on the reply message.

Exceptions

This Causes the Function to Terminate Normally	With This Reply Message
The user requests to release the implicit record lock and the cursor does not hold a record lock. Note: If this condition cannot be detected, the function terminates normally.	EXSCNDRM

DDMUnlockRec

This Causes the Function to be Rejected	With This Reply Message
The file handle is invalid.	HDLNFRM

DDMUnlockRec

Chapter 4. VSAM API Common Parameters

This chapter provides detailed information about the common parameters of reply messages, data buffers, and EAs.

The parameters are listed in alphabetical order. Each parameter is described in three parts: purpose, code point, and structure. A code point is a hexadecimal value that uniquely identifies the class of a DDM object. The parameter name is also the name of the pre-defined constant for the code point of the parameter. Each common parameter is a DDM object whose generic structure is defined by the DDMOBJECT type:

LL	CP	Data
----	----	------

LL	Indicates the total length (ULONG) of the data description from the beginning of the length field to the end of Data.
CP	Indicates the code point of the parameter.
Data	Indicates the objects contained in the parameter.

ACCINTLS (Access Intent List)

Purpose Specifies the file access intentions of the requester. One or more file access intents may be returned. The same value should not be returned more than once in the list.

Code Point The code point for this parameter is X'1134'.

Structure

LL	X'1134'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'1134'	The value (code point) indicating that the following data is an access intent list.
Data	A list of access intent values (code point): X'140B' DELAI (Delete Record Access Intent) X'1416' GETAI (Get Record Access Intent) X'1417' INSAI (Insert Record Access Intent)

X'1428' MODAI (Modify Record Access Intent)

ACCMTHCL (Access Method Class)

Purpose Specifies the class of the access method to be opened for file access.

Code Point The code point for this parameter is X'114E'.

Structure

LL	X'114E'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'114E'	The value (code point) indicating that the following data names the class of the access method.
Data	The value (code point) specifying the access method class: dl tsize=15.
X'140B'	DELAI (Delete Record Access Intent)
X'1416'	GETAI (Get Record Access Intent)
X'1417'	INSAI (Insert Record Access Intent)
X'1428'	MODAI (Modify Record Access Intent)

ACCMTHLS (Access Method List)

Purpose Specifies the access methods that can be used to access the records of a file.
When returned by DDMQueryFileInfo or DDMQueryPathInfo, only those access methods supported are listed. If no access method classes are specified, the records of the file cannot be accessed.

Code Point The code point for this parameter is X'1402'.

Structure

LL	X'1402'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of this length field to the end of Data.

X'1402'	The value (code point) indicating that the following data is a list of the access methods.
Data	The values (code point) specifying the access methods:
X'1433'	RELRNBAM (Relative by Record Number Access Method)
X'1435'	RNDRNBAM (Random by Record Number Access Method)
X'1407'	CMBRNBAM (Combined Record Number Access Method)
X'1432'	RELKEYAM (Relative by Key Access Method)
X'1434'	RNDKEYAM (Random by Key Access Method)
X'1406'	CMBKEYAM (Combined Keyed Access Method)
X'1405'	CMBACCAM (Combined Access Access Method)

ALCINISZ (Allocate Initial Extent)—DFM Only

Purpose	Specifies whether storage is to be allocated for the initial extent of a file at the time the file is created. The value can be:
TRUE	Indicates the initial extent should be allocated.
FALSE	Indicates the initial extent should NOT be allocated.
Note:	The value specified in the ALCINISZ parameter is considered a preference. The target system can choose to ignore this parameter.
Code Point	The code point for this parameter is X'1154'.

Structure

X'00000007'	X'1154'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the attribute description (from the beginning of this length field to the end of data).

X'1154'	The value (code point) indicating that the following data is the initial file size.
Status	The 1-byte status of ALCINISZ. The value can be:
X'F1'	Indicates a value of TRUE.
X'F0'	Indicates a value of FALSE.

ALTINDLS (Alternate Index List)

Purpose Specifies a list of alternate index file names associated for a base file. The base file can only be a keyed file.

Code Point The code point for this parameter is X'144E'.

Structure

LL	X'144E'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'144E'	The value (code point) indicating that the following data is a list of alternate index file names.
Data	A list of alternate index file names. The maximum file name length is defined by the underlying file system driver.

LL	X'1103'	Filename	LL	X'1103'	Filename
----	---------	----------	----	---------	----------

...	LL	X'1103'	Filename
-----	----	---------	----------

LL	The length (ULONG) from the beginning of LL to the end of the file name.
X'110E'	The value (code point) indicating that the following data is a file name.
Filename	An ASCII string containing the file name and ending with a null character.

BASFILNM (Base File)

Purpose Specifies the name of the file upon which an alternate index file is based.

The base file cannot be an alternate index file. The base file can *only* be a keyed file.

A DDM file name is an unarchitected string of characters. DDM assumes that a name provided by the user to the source system DDM is in the format required by the target system data manager for locating the file. The named string can contain qualifiers for libraries, catalogs, members, instances, or other levels of identification for the file.

Code Point The code point for this parameter is X'1103'.

Structure

LL	X'1103'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'1103'	The value (code point) indicating that the following data is a base file name.
Data	The base file name.

BASMGMM (Base Management Class Name)

Purpose Specifies the name of the management class for the base file in a reply message.

Code Point The code point for this parameter is X'11D3'.

Structure

LL	X'11D3'	Name
----	---------	------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of Name.
X'11D3'	The value (code point) indicating that the following information is the base management class name.
Name	The character string of up to 16 characters.

BASSTGNM (Base Storage Class Name)

Purpose Specifies the name of the storage class for the base file in a reply message.

Code Point The code point for this parameter is X'11D4'.

Structure

LL	X'11D4'	Name
----	---------	------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of Name.
X'11D4'	The value (code point) indicating that the following information is the base storage class name.
Name	The character string of up to 16 characters.

CODPNT (Code Point Attribute)

Purpose Specifies a value that is a DDM-architected code point.

Code Point The code point for this parameter is X'000C'.

Structure

X'0008'	X'000C'	Code Point
---------	---------	------------

Field	Description
X'0008'	The length (ULONG) of the code point description (from the beginning of this length field to the end of the code point).
X'000C'	The value (code point) indicating that the following information is a code point.
Code Point	The code point.

CSRPOSST (Cursor Position Status)

Purpose Specifies the status of the cursor in a reply message.

If the severity code is at least ERROR:

- The cursor position is the same as before the function that caused the reply message that carried this parameter.
- If the function was DDMInsertRecNum, DDMInsertRecEOF, DDMInsertRecKey, DDMSetNextRec, or DDMSetKeyNext with

RecCount greater than 1, the cursor position is the same as before the function iteration that caused the reply message.

A value of TRUE (X'F1') indicates that the cursor position is the same as before the function was issued or before the function iteration in error. TRUE is the only valid value if the severity code is ERROR.

A value of FALSE (X'F0') indicates that the cursor position may not be the same as before the function was issued, or before the function iteration in error, or that the current cursor position is unknown.

If the severity code is SC_NO_ERROR or SC_WARNING, the value of this parameter is ignored. The cursor status is as specified for the function that returned the reply message with a severity code of SC_NO_ERROR or SC_WARNING.

Code Point The code point for this parameter is X'115B'.

Structure

X'0007'	X'115B'	Status
---------	---------	--------

Field	Description
X'0007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'115B'	The value (code point) for cursor position status.
Status	The 1-byte cursor position status: X'F1' Denotes TRUE. X'F0' Denotes FALSE. Note: This value is always X'F1'.

DATE (Date and Time)

Purpose A date and time can be specified for the required level of resolution. The optional data terms (for example, seconds) can be specified only if the preceding terms (for example, minutes) are also specified.

Dates and times are determined by the calendar and clock of the originator of the date/time stamp. Dates are specified according to the Gregorian calendar. Times are specified according to the military clock.

Code Point The code point of this term is X'000F'.

Structure

LL	X'000F'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'000F'	The value (code point) indicating that the following is date and time data.
Data	The date and time data.
Field	Description
Year	The year: <ul style="list-style-type: none"> • Character digit string • LENGTH 4 • Minimum value is 0000 • Maximum value is 9999.
Month	The month in the year: <ul style="list-style-type: none"> • Character digit string • LENGTH 2 • Enumerated values for this parameter: <ul style="list-style-type: none"> 00 (Month is unknown or special meaning is being conveyed that is server dependent.) 01 (Month of January) 02 (Month of February) 03 (Month of March) 04 (Month of April) 05 (Month of May) 06 (Month of June) 07 (Month of July) 08 (Month of August) 09 (Month of September) 10 (Month of October) 11 (Month of November) 12 (Month of December)
Day	The day of the month: <ul style="list-style-type: none"> • Character digit string • LENGTH 2 • Minimum value is 00 • Maximum value is 31. • A value of 00 means the day is unknown or special meaning is being conveyed that is server dependent.
Hour	The hour of the day: <ul style="list-style-type: none"> • Character digit string • LENGTH 2 • Minimum value is 00

- Maximum value is 23
 - 00 is midnight, 06 is 6 a.m., 12 is noon, and 18 is 6 p.m.
- Minute** The minute of the hour:
- Character digit string
 - LENGTH 2
 - Minimum value is 00
 - Maximum value is 59.
- Second** The second of the minute:
- Character digit string
 - LENGTH 2
 - Minimum value is 00
 - Maximum value is 59.

DELCP (Record Deletion Capability)

- Purpose** Specifies whether records can be deleted from the file.
- Code Point** The code point for this parameter is X'111B'.
- Structure**

X'00000007'	X'111B'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'111B'	The value (code point) indicating that the following data is the record deletion capability.
Status	The 1-byte status of DELCP.
X'F1'	Denotes TRUE.
X'F0'	Denotes FALSE.

DFTREC (Default Record)

- Purpose** The default record is used to initialize a file when it is created. The length of the record data must be at least 1 and not greater than 4096.
- Code Point** The code point for this parameter is X'142B'.
- Structure**

LL	X'142B'	Default Record Data
----	---------	---------------------

Field	Description
LL	The length (ULONG) of the buffer from the beginning of this length field to the end of Default Record Data.
X'142B'	The value (code point) indicating that the following data is the default record data.
Default Record Data	<p>The contents of the default record data are replicated or truncated to match the record length of the file. This means that a value of X'00' causes the file to be initialized with records consisting of all zeroes. A value of 'ABC' would initialize a file with 10-byte records with 'ABCABCABCA' as the initialization record.</p> <p>If the file is created with initially-varying-length records or with variable-length records, the initialized records have a length equal to RecLen.</p> <p>If DFTINAIN record initialization was requested when the file was created, the Default Record Data field will be null, and the LL field will be X'00000006'.</p>

DTACLSNM (Data Class Name)

Purpose Specifies the name of the data class that applies to a file or directory. For the target system, a data class specifies a set of allocation attributes to create a file or directory.

Code Point The code point for this parameter is X'1121'.

Structure

LL	X'1121'	Name
----	---------	------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of Name.
X'1121'	The value (code point) indicating that the following information is the data class name.

Name The character string of up to 16 characters.

DTALCKST (Data Lock Status)

Purpose Specifies the status of the locks held on the records of a file. If the severity code is ERROR or higher, this parameter indicates whether the locks are:

- the same as before the function that caused the reply message which carried this parameter, or
- the same as before the function iteration that caused the reply message, if the function was:

DDMInsertRecNum
DDMInsertRecEOF
DDMInsertRecKey
DDMSetNextRec
DDNSetKeyNext with RecCount greater than 1.

A value of TRUE indicates that the locks are the same as before the function was issued, or before the function iteration in error. TRUE is the only valid value if the severity code is ERROR.

A value of FALSE indicates either:

The record locks are not the same as they were before the function was issued,
The record locks are not the same as they were before function iteration in error, or
The current lock status is unknown.

If the severity code is SC_NO_ERROR or SC_WARNING, the value of this parameter is ignored. The data locks are as specified for the function that returned the reply message with a severity code of SC_NO_ERROR or SC_WARNING.

Code Point The code point for this parameter is X'115C'.

Structure

X'00000007'	X'115C'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'115C'	The value (code point) indicating that the following data is the data lock status.
Status	The 1-byte status of the data locks on the file.

X'F1' Denotes TRUE.
X'F0' Denotes FALSE.
Note: This value is always X'F1'.

EOFNBR (End of File Record Number)

Purpose The record number of the EOF position of the file.
Code Point The code point of this term is X'110B'.

X'0000000A'	X'110B'	EOF Number
-------------	---------	------------

Field	Description
X'0000000A'	The length (ULONG) of the reply message object from the beginning of this length field to the end of EOF Number.
X'110B'	The value (code point) for the EOF record number object.
EOF Number	The ULONG EOF number.

ERRFILNM (Error File Name)

Purpose The error file name is the name of a file, other than the one the function is directly accessing, that caused the error. For example, modification of a record of a file may fail because an alternate index file built over the file does not allow keys to be updated. In this case, the name of the alternate index file would be specified as the error file name.

Code Point The code point for this parameter is X'1126'.

Structure

LL	X'1126'	Error File Name
----	---------	-----------------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of the Error File Name.
X'1126'	The value (code point) indicating the following data is the error file name.
Error File Name	The file name.

FILBYTCN (File Byte Count)

Purpose The file byte count is the total number of bytes currently allocated to a file. The bytes are counted in 1K (1024) byte units. The byte count is rounded to the next higher 1K byte value (for example, 1027 bytes requires a 2K byte value). The minimum value for this parameter is 0.

Code Point The code point for this parameter is X'1139'.

Structure

X'0000000A'	X'1139'	Count
-------------	---------	-------

Field	Description
X'0000000A'	The length (ULONG) of the attribute description (from the beginning of this length field to the end of Count).
X'1139'	The value (code point) indicating that the following data is the file byte count.
Count	The total number of bytes currently allocated to the file. The value of Count is specified in a ULONG.

FILCHGDT (File Change Date)—DFM Only

Purpose The change date of a file is the target system date on which certain operations occurred, such as:

- The file was created
- A record was processed by a DDMMModifyRec, DDMMInsertRec, or DDMDDeleteRec command
- The file was renamed
- The attributes were changed

The file change date can be updated either as each change occurs to the file, or when the file is closed following such a change. This is dependent on the DDM server implementation.

Code Point The code point of this term is X'113A'.

Structure

LL	X'113A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the attribute description from the beginning of LL to the end of Data.
X'113A'	The value (code point) indicating that the following data is the file change date.
Data	Date and time data, rounded down to the even second. See "DATE (Date and Time)" on page 367 for the format of date and time data.

FILCLS (File Class)

Purpose Specifies the class of a file.

Code Point The code point for this parameter is X'1110'.

Structure

X'00000008'	X'1110'	Data
-------------	---------	------

Field	Description
X'00000008'	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'1110'	The value (code point) indicating that the following data is the file class value.
Data	A file class value (USHORT) that can have the following values: <ul style="list-style-type: none"> ALTINDF Alternative Index File (X'1423') DIRFIL Direct File (X'140C') KEYFIL Keyed File (X'141E') SEQFIL Sequential File (X'143B')

FILCRTDT (File Creation Date)

Purpose The creation date of a file is the date on which a DDMCreatexxx function created the file.

Code Point The code point of this term is X'1108'.

Structure

LL	X'1108'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the attribute description from the beginning of LL to the end of Data.
X'1108'	The value (code point) indicating that the following data is the file creation date.
Data	Date and time data, rounded down to the even second. See "DATE (Date and Time)" on page 367 for the format of date and time data.

FILHDD (File Hidden)

Purpose	Specifies whether the file was created with the FILE_HIDDEN attribute.
FALSE	Files or subdirectories with an attribute of FILHDD(TRUE) are not considered a match.
TRUE	Files or subdirectories with an FILHDD attribute value of TRUE or FALSE are considered a match.
Code Point	The code point for this parameter is X'1132'.
Structure	

X'00000007'	X'1132'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'1132'	The value (code point) indicating that the following data is the hidden file attribute.
Status	The 1-byte status of FILHDD. X'F1' Denotes TRUE. X'F0' Denotes FALSE.

FILINISZ (Initial File Size)

Purpose	Specifies the preferred initial file size in records. The maximum initial size is determined by the target system, and the value specified can be rounded up or down to the next unit of allocation. The value is expressed in the number of record positions preferred for the initial file size. The RecLen (Record Length) parameter on the DDCreateRecFile function is used to compute the amount of storage requested.
----------------	--

Note that the value specified in the FILINISZ parameter is considered a preference. The target system can choose to implement another value.

Code Point The code point for this parameter is X'113C'.

Structure

X'0000000A'	X'113C'	Size
-------------	---------	------

Field	Description
X'0000000A'	The length (ULONG) of the attribute description (from the beginning of this length field to the end of Data).
X'113C'	The value (code point) indicating that the following data is the initial file size.
Size	The initial file size in records: <ul style="list-style-type: none"> • The value is specified in a ULONG. • Minimum value is 0, which means that the file exists but has no space allocated to it. • The value of X'FFFFFFFF' means that the file is of unlimited size.

FILNAM (File Name)

Purpose A VSAM API file name is an unarchitected string. A VSAM API assumes that a name provided by the user to the DDM source server is in the format required by the target server for creating or locating the file. The named string can contain qualifiers for directories, libraries, catalogs, members, instances, or other levels of identification of the file.

The target agent validates the file name according to its own rules for naming. This can be done before or after attempting to use the specified file name.

No semantic meaning is assigned to file names.

Code Point The code point for this parameter is X'110E'.

Structure

LL	X'110E'	File Name
----	---------	-----------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of File Name.

X'110E'	The value (code point) indicating that the following information is the file name.
File Name	The file name. The maximum file name length is defined by the underlying file system driver.

FILPRT (File Protected)

Purpose	Specifies whether the file is protected. DDDelete cannot be used on a protected file. Having a protected file attribute does not prevent a file from being opened with access intents of MODAI, DELAI, or INSAI. Nor does this attribute prevent DDModifyRec, DDDeleteRec, or DDInsertRecxxx function from being performed. These functions are controlled by the file capabilities attributes: MODAI, DELAI, or INSAI. The value of TRUE indicates that the file is protected from file management functions that would change the entire contents of the file. The value of FALSE indicates that the file is not protected from file management functions that would change the entire contents of the file.
Code Point	The code point for this parameter is X'112A'.

Structure

X'00000007'	X'112A'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'112A'	The value (code point) indicating that the following data is the file protect attribute.
Status	The 1-byte status of FILPRT. X'F1' Denotes TRUE. X'F0' Denotes FALSE. (This is the default value.)

FILSIZ (File Size)

Purpose	The size of a file is determined by the total number of record positions allocated to the file. This includes all active and inactive records between the BOF and the EOF plus all allocated record positions between the EOF and the end of the last allocated extent. This attribute does not apply to files with variable-length records.
----------------	--

Code Point The code point for this parameter is X'110F'.

Structure

X'0000000A'	X'110F'	Data
-------------	---------	------

Field	Description
X'0000000A'	The length (ULONG) of the attribute description from the beginning of this length field to the end of Data.
X'110F'	The value (code point) for this attribute.
Data	A ULONG binary number: <ul style="list-style-type: none">• The value is specified in a ULONG.• Minimum value is 0.

FILSYS (System File)

Purpose Specifies whether the file was created with the FILE_SYSTEM attribute.

Code Point The code point for this parameter is X'1133'.

Structure

X'00000007'	X'1133'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'1133'	The value (code point) indicating that the following data is the system file attribute.
Status	The 1-byte status of FILSYS. X'F1' Denotes TRUE. X'F0' Denotes FALSE.

GETCP (File Get Capability)

Purpose Specifies whether the contents of a file can be read by DDMGetRec, DDMSETxxx with NODATA(FALSE), or DDMUnLoadFilexxxx. If the file is not get-capable, a DDMGetRec, DDMSETxxx with NODATA(FALSE), or DDMUnLoadFilexxxx is rejected with an INVRQSRM reply message.

Code Point The code point for this parameter is X'1191'.

Structure

X'00000007'	X'1191'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'1191'	The value (code point) indicating that the following data is the file get capability
Status	The 1-byte status of GETCP. X'F1' Denotes TRUE. X'F0' Denotes FALSE.

INSCP (File Insert Capability)

Purpose Specifies whether data records can be inserted into the file by a DDMinertRECxxx or DDMLoadFilexxxx function.
If the file is not insert-capable, an insert function DDMinertRecxxx (an insert function) or DDMLoadFilexxxx is rejected with an INVRQSRM reply message.

Code Point The code point for this parameter is X'1192'.

Structure

X'00000007'	X'1192'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description (from the beginning of this length field to the end of Status).
X'1192'	The value (code point) indicating that the following data is the file insert capability.
Status	The 1-byte status of INSCP. X'F1' Denotes TRUE. X'F0' Denotes FALSE.

KEYDEF (Key Definition)

Purpose The key of a record consists of one or more fields that define an ordering of the records for relative or random access. Key fields are defined in terms of their length and displacement in the record.

Composite keys can be expressed by repeating the KEYFLDDF (Key Field Definition) parameter as many times as necessary. The first KEYFLDDF specifies the most significant part of the key and the last KEYFLDDF specifies the least significant part of the key. The total of all key lengths in a composite key cannot exceed 255 bytes, which is the maximum length key definition.

For a description of the errors that can be detected by target systems in a definition of the keys of a file, see "KEYDEFM (Invalid Key Definition)" on page 446 and "KEYDEFCD (Key Definition Error Code)."

Code Point The code point for this parameter is X'1114'.

Structure

LL	X'1114'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'1114'	The value (code point) indicating that the following data is a key definition.
Data	A list of one or more key field definitions (KEYFLDDF).

KEYDEFCD (Key Definition Error Code)

Purpose Specifies the condition for which the KEYDEFM reply message was returned.

Code Point The code point for this parameter is X'1164'.

Structure

X'0000007'	X'1164'	Data
------------	---------	------

Field	Description		
X'0000007'	The length (ULONG) of the data description from the start of this length field to the end of Data.		
X'1164'	The value (code point) indicating that the following data is a key definition error code.		
Data	A 1-byte value specifying the key definition error code: <table> <tr> <td>X'01'</td> <td>The specified key does not fall within the record boundaries.</td> </tr> </table>	X'01'	The specified key does not fall within the record boundaries.
X'01'	The specified key does not fall within the record boundaries.		

- X'02'** The target system does not support composite keys or the number of composite keys specified.
- X'03'** The total length of the specified key or composite key exceeds the maximum key length supported by the target system. The maximum length key supported in the local VSAM file system is 255 bytes.
- X'04'** The target system does not support overlapping fields. For example, if key field A begins in position 10 for a key length of 10, it is not possible to specify a key field B that overlaps positions 10 through 19.
- X'05'** The target system does not allow a key field to be defined over multiple record fields when the record fields are defined in a target system data dictionary.
- X'06'** The target system does not allow a key field to be defined for a part of a record field.
- X'07'** The target does not allow a key field to be specified for non-character record fields, such as an encoded integer or floating point field.
- X'08'** The target system does not support the specified key sequence for the specified key data class.
- X'09'** The target system does not support the specified key data class.

KEYDUPCP (Duplicate Keys Capability)

Purpose Specifies whether or not duplicate keys are allowed in a file.

Code Point The code point for this parameter is X'113D'.

Structure

X'00000007'	X'113D'	Data
-------------	---------	------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'113D'	The value (code point) that indicates whether duplicate keys are allowed.
Data	A 1-byte value: X'F0' Duplicate keys are not allowed. X'F1' Duplicate keys are allowed.

KEYFLDDF (Key Field Definition)

Purpose The key field defines the location, length, data class, and ordering of a single record key.

Code Point The code point for this parameter is X'140F'.

Structure

X'00000010'	X'140F'	Keyseq	Keycls	Keylen	Keydsp
-------------	---------	--------	--------	--------	--------

KeyLen	KeyDisp
--------	---------

Field	Description
X'00000010'	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'140F'	The value (code point) indicating that the following data is a key field definition.
Keyseq	A value (USHORT) specifying the key sequence: X'1420' Ascending Key Sequence X'1421' Descending Key Sequence
Keycls	A value (USHORT) specifying the key class: X'0044' The key field is a byte string.
Keylen	A value (USHORT) specifying the key length.
Keydsp	The displacement (ULONG) of the start of the key field in the record. If multiple KEYFLDDF (Key Field Definitions) are provided, the fields are concatenated to form a combined key. The maximum length key is 255 bytes.

KEYVAL (Key Value)

Purpose Specifies the value of a record key.

Code Point The code point for this parameter is X'1115'.

Structure

LL	X'1115'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'1115'	The value (code point) that indicates the following is a key value.
Data	The key value (BYTE) for a record. The key value can be up to 255 bytes. If the record is inactive, the key value is set to X'00'.

LSTACCDT (Last Access Date)—DFM Only

Purpose The last access date is the target system date on which the file was last accessed by operations such as a record DDMDelRec, DDMModifyRec, or DDMinRec command.

The last access date can be updated either as these commands are performed or when the file is closed following one of these commands. This is dependent on the DDM server implementation.

Code Point The code point of this term is X'1113'.

Structure

LL	X'1113'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the attribute description from the beginning of LL to the end of Data.
X'1113'	The value (code point) indicating that the following data is the last access date of the file.
Data	Date and time data, rounded down to the even second. See "DATE (Date and Time)" on page 367 for the format of date and time data.

LSTARCDT (Last Archived Date)—DFM Only

Purpose The last archive date is the date on which the file was last archived by the target system.

Code Point The code point of this term is X'118A'.

Structure

LL	X'118A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the attribute description from the beginning of LL to the end of Data.
X'118A'	The value (code point) indicating that the following data is the last archive date of the file.
Data	Date and time data, rounded down to the even second. See "DATE (Date and Time)" on page 367 for the format of date and time data.

MAXARNB (Maximum Active Record Number)

Purpose The maximum active record number is the highest record number at which an active record is stored in a file.

Code Point The code point of this term is X'1159'.

Structure

X'0000000A'	X'1159'	RecordNumber
-------------	---------	--------------

Field	Description
X'0000000A'	The length (ULONG) of the attribute reply data.
X'1159'	The value (code point) indicating that the following data is the maximum active record number.
RecordNumber	The ULONG maximum active record number.

MAXOPN (Maximum Number of Files Opened)

Purpose Specifies the maximum number of times the same file can be opened concurrently by the same agent.

Code Point The code point of this term is X'1157'.

Structure

X'00000008'	X'1157'	MaxNumOpn
-------------	---------	-----------

Field	Description
X'00000008'	The length (ULONG) of the reply message object (for OPNMAXRM).
X'1157'	The value (code point) indicating that the following data is the MAXOPN object.
MaxNumOpn	The maximum number (USHORT) of concurrent opens allowed.

MGMCLSNM (Management Class Name)

Purpose Specifies the name of the management class that applies to a file or directory. The format of a management class is unarchitected. A management class specifies the target system policies related to when and how often the file or directory is to be backed up, saved, or archived.

Code Point The code point for this parameter is X'1140'.

Structure

LL	X'1140'	Name
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of name.
X'1140'	The value (code point) indicating that the following is a management class name.
Name	Character string up to 16 bytes.

MODCP (File Modify Capability)

Purpose Specifies whether the contents of a file can be modified by a DDMMModifyRec or DDMTruncFile function.
If the file is not modify-capable, a DDMMModifyRec or DDMTruncFile function is rejected with an INVRQSRM reply message.

Code Point The code point for this parameter is X'1166'.

Structure

X'00000007'	X'1166'	Status
-------------	---------	--------

Field	Description
X'00000007'	The length (ULONG) of the data description from the beginning of this length field to the end of Status.
X'1166'	The value (code point) indicating that the following data is the file modify capability.
Status	The 1-byte status of MODCP.
X'F1'	Denotes TRUE.
X'F0'	Denotes FALSE.

NEWFILNM (New File Name)

Purpose Specifies the new name to be assigned to a file that was invalid.

The names of files in the VSAM APIs are unarchitected strings of characters with no semantic meaning. A VSAM API assumes that a name provided by the user to the source DDM server is in the format required by the target system data manager for creating or locating the file. The name string can contain qualifiers for libraries, catalogs, members, instances, or other levels of identification for the file.

Code Point The code point of this parameter is X'114F'.

Structure

LL	X'114F'	NewFilNam
----	---------	-----------

Field	Description
LL	The length (ULONG) of the reply message object for NEWNAMRM (Invalid New File Name) reply message from the beginning of this length field to the end of NewFilNam.
X'114F'	The value (code point) indicating that the following data is the new file name object.
NewFilNam	The name of the new file. The maximum file length is determined by the underlying file system.

RECAL (Record Attribute List)

Purpose Specifies a list of attributes of a record as an ordered collection.

A Record Attribute List is used when transmitting more than one attribute of the record (for example, record number or key value and the record itself) as a single unit.

The RECCNT parameter is used to indicate the number of duplicate records.

The elements of a RECAL must be specified in the order in which they are listed in the format of this parameter. If an optional parameter is not included, the order of the remaining variables must be maintained.

If RECNBR and RECCNT are both specified, the record number specified by RECNBR applies to the first occurrence of the record, and each subsequent record has a record number of 1 greater than the previous record.

Note: The returned Record Attribute List structure is contiguous.

**Code Point
Structure**

The code point for this parameter is X'1430'.

LL	X'1430'	L1	X'111A'	RC	L2	X'111D'	RN
----	---------	----	---------	----	----	---------	----

L3	X'1115'	KEY	L4	CP	Data
----	---------	-----	----	----	------

Field	Description
LL	The length (ULONG) of the record attribute list from the beginning of LL to the end of Data. This field is not checked.
X'1430'	The value (code point) indicating that the following data is a RECAL.
L1	The length (ULONG) from the beginning of L1 to the end of RC. This field is not checked.
X'111A'	The value (code point) indicating that the following data is a record count. The RECCNT parameter is used to indicate the number of duplicate records. RECCNT provides a shorthand way of specifying N records, where N>1, and not replicating the record's contents.
RC	The number (ULONG) of duplicate records in the RECAL (RECCNT).
L2	The length (ULONG) from the beginning of L2 to the end of RN. This field is not checked.
X'111D'	The value (code point) indicating that the following data is a record number.
RN	The record number (ULONG) of the record in the RECAL (RECNBR).
L3	The length (ULONG) from the beginning of L3 to the end of the key value.

X'1115'	The value (code point) indicating that the following data is a key value.
KEY	The record key value (KEYVAL).
L4	The length (ULONG) from the beginning of L4 to the end of Data.
CP	The value (code point) indicating that the following is either record data or an inactive record length.
X'144A'	Indicates that the following data is record data (RECORD).
X'142D'	Indicates that the following data is a ULONG of an inactive record (RECINA).
Data	The record data or the length (ULONG) of the inactive record.

RECCNT (Record Count)

Purpose Specifies the number of records:

- Loaded by a DDMLoadFilexxxx function.
- Unloaded by a DDMUnLoadFilexxxx function.
- Initialized when creating a record-oriented file with the DFTRECOF parameter specified on the DDMCreateRecFile function.
- Retrieved by a DDMSETxxx function.

When specified in a reply message, RECCNT specifies the number of records successfully inserted in a file by an DDMLoadFilexxxx function with a Record Count parameter value greater than 1 or by a DDMLoadFilexxxx function.

When used with a RECAL, RECCNT specifies the number of times the contents of the record attribute list is repeated. This provides an efficient way to send multiple copies of the same records.

Code Point The code point for this parameter is X'111A'.

Structure

X'0000000A'	X'111A'	Count
-------------	---------	-------

Field	Description
X'0000000A'	The length (ULONG) of the data description from the beginning of this length field to the end of Count.

X'111A'	The value (code point) indicating the following data is the record count.
Count	The number of records successfully returned or inserted: <ul style="list-style-type: none"> • The value in count is specified in a ULONG. • Minimum value is 0.

RECINA (Inactive Record)

Purpose Represents file record positions at which a record has never been inserted or at which a previously active record has been deleted. The data value of an inactive record is the required length of any record to be inserted at the record position.

Code Point The code point of this term is X'142D'.

Structure

X'0000000A'	X'142D'	Data
-------------	---------	------

Field	Description
X'0000000A'	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'142D'	The value (code point) indicating that the following data is the inactive record length.
Data	The required length of any record to be inserted at the record position. The value of an inactive record is specified in a ULONG. If the special value of -1 is present, this variable-length record has not had a previous value and it can store any length record up to the maximum allowed by the file.

RECLLEN (Record Length)

Purpose The length of the user data in all of the records in files of fixed-length records.

The maximum length of the user data in the records in files of variable-length records or initially-variable-length records.

Code Point The code point of this term is X'111C'.

Structure

X'0000000A'	X'111C'	Data
-------------	---------	------

Field	Description
X'000000A'	The length (ULONG) of the attribute description from the beginning of this length field to the end of Data.
X'111C'	The value (code point) indicating that the following data is the record length.
Data	The record length: <ul style="list-style-type: none"> • The value of Data is specified in a ULONG. • Minimum value is 1. • Maximum value is 64,000.

RECLENCL (Record Length Class)

Purpose Specifies the type of record length that records in a file can have.

If a record length class that is not supported is specified, the record length class can be promoted to a record class that is supported. The record length class cannot be demoted. The promotion scheme for record length classes is:

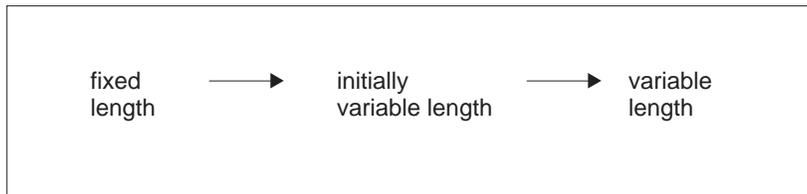


Figure 83. Record Length Class Promotion

Code Point The code point of this term is X'1142'.

Structure

X'00000008'	X'1142'	RLC
-------------	---------	-----

Field	Description
X'0000008'	The length (ULONG) of the attribute description from the beginning of this length field to the end of record length class (RLC).
X'1142'	The value (code point) indicating that the following is a record length class.

RLC A value (code point) specifying the record length class. Valid code points are:

Code Point	Description
X'142E'	Fixed-length record (This is the default value.)
X'142F'	Initially-variable-length record
X'1431'	Variable-length record

RECNBR (Record Number)

Purpose A record number identifies a record at a specific position of the file. Record positions are numbered starting with 1.

Code Point The code point of this term is X'111D'.

Structure

X'0000000A'	X'111D'	RecordNumber
-------------	---------	--------------

Field	Description
X'0000000A'	The length (ULONG) of the data description from the start of length field to the end of RecordNumber.
X'111D'	The value (code point) indicating that the following data is a record number.
RecordNumber	The record number that is being returned: <ul style="list-style-type: none"> • The value is specified in a ULONG. • Minimum value is 1. • The maximum value is target system dependent. • The value of X'FFFFFFFF' means the actual record number is not known.

RECORD (Record)

Purpose Records are the basic unit of data stored in a record-oriented file. They are the basic unit of transfer between requesters and files. A record consists of a record header followed by the record data. This type of record object is also known as an active record.

Code Point The code point of this parameter is X'144A'.

Structure

LL	X'144A'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of Data.
X'144A'	The value (code point) indicating that the following is record data.
Data	Encoded information.

RTNCLS (File Retention Class)

Purpose Specifies the file retention as temporary or permanent.

Code Point The code point for this parameter is X'1148'.

Structure

X'00000008'	X'1148'	Data
-------------	---------	------

Field	Description
X'00000008'	The length (ULONG) of the data description from the beginning of this length field to the end of Data.
X'1148'	The value (code point) indicating that the following data is the file retention class.
Data	A value (code point) specifying the retention class: X'143E' A temporary file. X'142A' A permanent file.

SRVDGN (Server Diagnostic Information)

Purpose Specifies diagnostic information in reply messages that is defined by the responding server. This information can be logged or otherwise used to support problem determination. The contents of this parameter are unarchitected. A maximum of 255 bytes can be sent, but only a server-determined minimum amount of information should be returned.

Code Point The code point for this parameter is X'1153'.

Structure

LL	X'1153'	Data
----	---------	------

Field	Description
LL	The length (ULONG) of this data description from the beginning of LL to the end of Data.
X'1153'	The value (code point) indicating that the following is server diagnostic information.
Data	The diagnostic information. This data is in the format of the server system that generated the reply message.

STGCLSNM (Storage Class Name)

Purpose Specifies the name of a storage class that applies to a file or directory. The format of a storage class is unarchitected. A storage class specifies the target system policies related to the types and speeds of the storage devices that the file or directory can be allocated to.

Code Point The code point for this parameter is X'1141'.

Structure

LL	X'1141'	Name
----	---------	------

Field	Description
LL	The length (ULONG) of the data description from the beginning of LL to the end of name.
X'1141'	The value (code point) indicating that the following is a storage class name.
Name	Character string up to 16 bytes.

SVRCOD (Severity Code)

Purpose Indicates the severity of a condition detected during the execution of a function.

In addition to being in the reply message data, the value is returned in the EAX register after every function.

Code Point The code point of this term is X'1149'.

Structure

X'00000008'	X'1149'	Severity
-------------	---------	----------

Field	Description
X'00000008'	The length (ULONG) of this data description from the beginning of this length field to the end of Severity.
X'1149'	The value (code point) indicating the following is the severity class.
Severity	A ULONG value specifying the severity code.
X'0000'	<p>SC_NO_ERROR (Information Only Severity Code)</p> <p>Specifies that a reply message contains information only and does not describe any problem condition.</p>
X'0004'	<p>SC_WARNING (Warning Severity Code)</p> <p>Specifies that a reply message constitutes the warning of a potential problem in the processing of a request.</p> <p>Further processing of a function depends on the specifications of the specific function, the error condition, and the environment in which it is being executed.</p>
X'0008'	<p>SC_ERROR (Error Severity Code)</p> <p>Specifies that an error condition was detected in the processing of a function. All effects of the condition have been reversed or prevented. For example, any effects on cursor positioning or locks obtained or released have been reversed.</p> <p>Further processing of a function depends on the architected specifications of that function, the error condition, and the environment in which it is being executed. For example, a FILE NOT FOUND error always causes a function to be terminated, but a DUPLICATE FILE error terminates processing of a DDMCreateRecFile function only if specified by the duplicate file option parameter.</p>

X'0010' SC_SEVERE (Severe Error Severity Code)

Specifies that a severe error has occurred during the execution of the function. It was not possible to prevent or reverse all changes to objects affected by the function. For example, record locks or cursor position may have been lost.

It is possible to send further functions to the affected objects.

X'0020' SC_ACCESSDAMAGE (Access Damage Severity Code)

Specifies that damage has occurred to the target agent's ability to access a file as it is currently opened. It is not possible to make further use of that access path, but it may be possible to access other opened files or other objects.

To recover from access damage, it is necessary to close and reopen the file.

X'0040' SC_PERMDAMAGE (Permanent Damage Severity Code)

Specifies that damage has occurred to the state or value of permanent objects of the server. Recovery from permanent damage may require special action that cannot be called through DDM functions; for example, loading a backup file.

Further processing of the function depends on the architected specifications of that request, the permanent damage condition, and the environment in which it is being executed. For example, it may be possible to continue processing with other undamaged resources.

X'0080' SC_SESSIONDAMAGE (Session Damage Severity Code)

Specifies that damage has occurred to the target server's ability to con-

tinue the communication session. It is not possible to make further use of the current session, but it may be possible to use other available communication sessions.

To recover from session damage, it is necessary to terminate the current session and establish a new session.

SYNERRCD (Syntax Error Code)

Purpose	Specifies the condition that caused termination of data stream parsing. The following code points might be returned:
X'01'	Data Stream Structure (DSS) header length less than or equal to 6.
X'02'	DSS header length does not match the number of bytes of data found.
X'03'	DSS header C-byte not X'D0'.
X'04'	DSS header f-bytes not recognized or not supported.
X'05'	DSS continuation specified but not found.
X'06'	DSS chaining specified but no DSS found.
X'07'	Object length less than 4.
X'08'	Object length does not match the number of bytes of data found.
X'09'	Object length greater than maximum allowed.
X'0A'	Object length less than minimum required.
X'0B'	Object length not allowed (for example, if a value must be a multiple of 1 word long but an odd number of bytes is sent).
X'0C'	Incorrect large object length field (see DSS).
X'0D'	Object code point index not supported.
X'0E'	Required object not found.
X'0F'	Too many function data objects sent.
X'10'	Mutually exclusive objects present.
X'11'	Too few function data objects sent.
X'12'	Duplicate object present.
X'13'	Invalid request correlator specified.
X'14'	Required value not found.
X'15'	Reserved value not allowed.
X'16'	DSS continuation less than or equal to 2.
X'17'	Objects not in required order.
X'18'	DSS chaining bit not b'1' but DSSFMT bit3 set to b'1'.
X'19'	Previous DSS indicated current DSS has the same request correlator but the request correlators are not the same.
X'1A'	DSS chaining bit not b'1' but error continuation requested.
X'1B'	Mutually exclusive parameter values specified.
X'1D'	Code point is not a valid function.

Code Point The code point of this term is X'114A'.

Structure

X'0007'	X'114A'	Byte
---------	---------	------

Field	Description
X'0007'	The length (ULONG) of the reply message object (from the start of this length field to the end of Byte).
X'114A'	The value (code point) for the syntax error code object.
Byte	The syntax error code as listed above (1-byte).

TITLE (A Brief Description)

Purpose	A brief description of the file stored in EAs.
Code Point	The code point of this parameter is X'0045'.

Structure

LL	X'0045'	Title
----	---------	-------

Field	Description
LL	The length (ULONG) of this data description from the beginning of this length field to Title.
X'0045'	The value (code point) indicating the following is the title.
Title	Character string up to 255 bytes.

Chapter 5. VSAM API Flags

This chapter provides information about each bit flag in each of the ULONG parameters: AccessFlags and CreateFlags.

Each flag parameter section provides three basic kinds of information about bits:

1. A description of what the flag parameter does
2. An overview of all ULONG bit masks associated with the flag parameter
3. A detailed description of each bit associated with the flag parameter, which includes:
 - A brief explanation of each bit
 - The bit number
 - Bit values.

The bit flags may be set individually or in appropriate combinations by using the bitwise inclusive OR operator (|).

Throughout this publication, these bits are referred to by their bit mask names as defined in DUBCALLS.H.

For more detailed information on each of these bits, see the individual bit names. The individual bit names in the following section are arranged in bit number order.

Bit value:

TRUE, set to 1, ON, or B'1', all have the same meaning.

FALSE, set to 0, OFF, and B'0', all have the same meaning.

AccessFlags (Access Flags)

Purpose Access Flags specify the action to be taken depending on whether the bit flag is set. Not all of the flags are valid on all functions. Flags that are not valid on a particular function are marked as reserved when describing that function. Reserved bits must be set to zero (B'0') or an invalid parameter error occurs.

Bit Mask Names and Descriptions

The total list of bit flags is:

Bit Mask Name	Description
Reserved	(Bits 12–31)
DDM_HLDUPD	Hold Update Intent (Bit 11)
DDM_UPDCSR	Update Cursor (Bit 10)
DDM_INHMODKY	Inhibit Modified Key (Bit 9)
DDM_ALWINA	Allow Cursor on Inactive Record (Bit 8)
DDM_HLDCSR	Hold Cursor Position (Bit 7)
DDM_BYPDMG	Bypass Damaged Record (Bit 6)

DDM_NODATA	No Record Data Returned (Bit 5)
DDM_ALLREC	All Records, Active and Inactive (Bit 4)
DDM_RTNINA	Return Inactive Record (Bit 3)
DDM_KEYVALFB	Key Value Feedback (Bit 2)
DDM_RECNRFB	Record Number Feedback (Bit 1)
DDM_UPDINT	Update Intent (Bit 0)

DDM_HLDUPD (Hold Update Intent)

Purpose	Specifies whether the currently held update intent and record lock, if any, should be released.
Bit number	Bit 11 of the AccessFlags word.
Bit value	A bit value of TRUE indicates that the update intent should not be released. A bit value of FALSE indicates that the update intent is released. In this case, systems that cannot hold locks on two records can reject the function with a VALNSPRM reply message.

DDM_UPDCSR (Update Cursor)

Purpose	Specifies whether the cursor is to be updated to point to the record inserted in the file by the function. When multiple records are being inserted in a file, the cursor points to the last record inserted when DDM_UPDCSR is set.
Bit number	Bit 10 of the AccessFlags word.
Bit value	A bit value of TRUE allows the cursor to be updated. A bit value of FALSE does not allow the cursor to be updated.

DDM_INHMODKY (Inhibit Modified Keys)

Purpose	Specifies whether the key value of an existing record can be modified by the DDMMModifyRec function. The inhibit modified keys bit is only effective when the file is opened with the RELKEYAM, RNDKEYAM, CMBKEYAM, or CMBACCAM access methods. This bit is ignored if the file is opened with any other access method.
Bit number	Bit 9 of the AccessFlags word.
Bit value	A bit value of FALSE permits key fields to be modified if the file permits key fields to be modified. A bit value of TRUE indicates that key fields cannot be modified. An attempt to modify a key field results in the DDMMModifyRec function being rejected with a KEYUSIRM reply message.

DDM_ALWINA (Allow Cursor to Be Set to Inactive Record)

- Purpose** Specifies whether the cursor can be set to point to an inactive record or whether the DDMSetUpdateNum function can set an update intent on an inactive record.
- Bit number** Bit 8 of the AccessFlags word.
- Bit value** A bit value of TRUE allows the cursor to point to an inactive record and specifies that the DDMSetUpdateNum function can set an update intent on an inactive record.
- A bit value of FALSE specifies that the cursor is not allowed to point to an inactive record and that the DDMSetUpdateNum function is *not* allowed to set an update intent on an inactive record.

DDM_HLDCSR (Hold Cursor Position)

- Purpose** Causes the hold cursor indicator to be set ON or OFF for the cursor. The hold cursor indicator is used by the following functions to determine whether the cursor should be moved to the next record or remain at the current record position:

Function	Described on page:
DDMSetKey	159
DDMSetKeyFirst	177
DDMSetKeyLast	186
DDMSetKeyNext	202
DDMSetKeyPrevious	220
DDMSetLast	233
DDMSetMinus	243
DDMSetNextKeyEqual	253
DDMSetNextRec	269
DDMSetPlus	291
DDMSetPrevious	301
DDMSetRecNum	314

- Bit number** Bit 7 of the AccessFlags word.
- Bit value** A bit value of TRUE means that the hold cursor indicator in the cursor is set on. If the hold cursor indicator is already on, it remains on.
- A bit value of FALSE means that the hold cursor indicator in the cursor is set off. If the hold cursor indicator is already off, it remains off.

DDM_BYPDMG (Bypass Damaged Records)

- Purpose** The bypass damaged records bit specifies whether processing is to continue if damaged records are detected for the DDMSetKeyNext, DDMSetNextRec, and DDMUnloadFilexxxx functions.
- Bit number** Bit 6 of the AccessFlags word.

Bit value A bit value of TRUE bypasses damaged records.
A bit value of FALSE does not bypass damaged records.

DDM_NODATA (No Record Data Returned)

Purpose Indicates whether the record, where the cursor is set, is to be returned.

Bit number Bit 5 of the AccessFlags word.

Bit value A bit value of TRUE indicates that the record, where the cursor is set, is not to be returned.
A bit value of FALSE indicates that the record, where the cursor is set, is to be returned. This is the default value.

DDM_ALLREC (All Records, Active and Inactive)

Purpose Specifies whether inactive records are to be bypassed when using one of the DDMSetxxx functions to set the cursor.

Bit number Bit 4 of the AccessFlags word.

Bit value A bit value of TRUE does not bypass inactive records.
A bit value of FALSE bypasses inactive records.

DDM_RTININA (Return Inactive Record)

Purpose Specifies whether an inactive record can be returned if the cursor is set to an inactive record and the record selected by the cursor is to be returned.

Bit number Bit 3 of the AccessFlags word.

Bit value A bit value of TRUE indicates that an inactive record can be returned.
A bit value of FALSE indicates that an inactive record cannot be returned. This is the default value.

DDM_KEYVALFB (Key Value Feedback)

Purpose Specifies whether the key value of the record is to be returned to the requester. If the record is inactive, a null key value (length = 4) is returned.
The local VSAM file system ignores this parameter when the file is opened with the RELRNBAM, RNDRNBAM, or CMBRNBAM access method or the file is not keyed.

Bit number Bit 2 of the AccessFlags word.

Bit value A bit value of TRUE indicates the key value of the record is returned.
A bit value of FALSE indicates the key value of the record is not returned.

DDM_RECNRFB (Record Number Feedback)

Purpose	Specifies whether the record number of the record is to be returned to the requester.
Bit number	Bit 1 of the AccessFlags word.
Bit value	A bit value of TRUE indicates the record number is returned. A bit value of FALSE indicates the record number is not returned.

DDM_UPDINT (Update Intent)

Purpose	Allows a requester to indicate that the user intends to modify the record. This can be specified when the cursor is moved to the record (DDMSetxxx) or when the record at the current cursor position is read (DDMGetRec). An update intent must be placed on a record before a DDMMModifyRec or DDMDDeleteRec function can be performed for the record. An update intent can also be placed on a record by the DDMSetUpdateKey and DDMSetUpdateNum functions.
----------------	--

Update intent is necessary so that a requester can perform operations on a record without interference from concurrent users. For information about the interaction of update intent and sharing and locking files, see "Record Locking (Implementation is Dependent on the Server)" on page 25.

The update intent for the record lasts until one of the following occurs:

- The record is modified (DDMMModifyRec).
- The record is deleted (DDMDDeleteRec).
- The cursor is moved to a different record. All cursor movement DDMSetxxx functions are considered to have moved the cursor even if the result of normal completion of the DDMSetxxx function leaves the cursor position the same as before the DDMSetxxx function was called.
- A DDMInsertRecNum, DDMSetUpdateKey, or DDMSetUpdateNum function for a different record is issued.
- A DDMInsertRecEOF or DDMInsertRecKey function with DDM_HLDUPD (FALSE) specified for a different record is issued.
- A DDMUnLockRec function is issued.
- A DDMGetRec function with update intent is issued.
- The file is closed.

Once the update intent for a record is removed, a new update intent must be placed on the record before a DDMMModifyRec or DDMDDeleteRec function can be performed for the record. Two consecutively issued DDMMModifyRec functions result in the rejection of

the second DDMMModifyRec function with UPDINTRM because the first DDMMModifyRec function removed the update intent for the record.

Bit number Bit 0 of the AccessFlags word.

Bit value A bit value of 1 (TRUE) indicates that the requester intends to modify or delete the record and, therefore, an update intent is to be placed on the record. If the file was opened for multiple modifications, an implicit (exclusive access) lock is placed on the record. Record locking is dependent on the remote server. See the appropriate documentation.

For the local VSAM file system, record locks apply only to OS/2 local VSAM files on the client OS/2 workstation.

A bit value of 0 (FALSE) indicates that the requester does not intend to modify or delete the record.

CopyFlags (Copy Flags)

Purpose Copy Flags specify the action to be taken depending on whether the bit flag is set. Not all of the flags are valid on all functions. Flags that are not valid on a particular function are marked as reserved when describing that function. Reserved bits must be set to zero (B'0') or an invalid parameter error occurs.

Bit Names and Descriptions The total list of bit flags is:

Bit Name	Description
Reserved	(bits 13-31)
DDM_ACCORD	Access Order (Key versus record order processing) (bit 12)
Reserved	(bits 7-11)
DDM_BYPDGM	Bypass Damaged Records (bit 6)
Reserved	(bit 5)
DDM_BYPINA	Bypass Inactive Records (Not applicable to direct files) (bit 4)
Reserved	(bits 0-3)

Throughout this document, these bits are referred to by name.

For more detailed information on each of these bits, see the individual bit names. The individual bit names in the following section are arranged in bit number order.

Bit value An individual bit is referred to as TRUE, set to 1, ON, or B'1', all of which have the same meaning.

An individual bit can also be referred to as FALSE, set to 0, OFF, and B'0', all of which have the same meaning.

DDM_BY PINA (Bypass Inactive Records)

Purpose Specifies whether inactive records are to be bypassed.

Bit number Bit 4 of the CopyFlags word.

Bit value A bit value of TRUE indicates inactive records are to be bypassed.

A bit value of FALSE indicates inactive records are not to be bypassed.

DDM_BY PDMG (Bypass Damaged Records)

Purpose Specifies whether damaged records are to be bypassed.

Bit number Bit 6 of the CopyFlags word.

Bit value A bit value of TRUE indicates damaged records are to be bypassed and that processing continues when the data record is damaged.

A bit value of FALSE indicates damaged records are not to be bypassed and that processing does not continue when a data record is damaged.

DDM_ACCORD (Access Order)

Purpose Specifies the order in which the records of the file are processed.

Bit number Bit 12 of the CopyFlags word.

Bit value A bit value of TRUE specifies key order processing.

A bit value of FALSE specifies record number order processing.

CreateFlags (Create Flags)

Purpose Create Flags specify the action to be taken depending on whether the bit flag is set. Not all of the flags are valid on all functions. Those flags not valid on a particular function are marked as reserved in the section describing that function. Reserved bits must be set to 0 (B'0') or an invalid parameter error occurs.

Bit Mask Names and Descriptions

The total list of bit flags is:

Bit Mask Name	Description
Reserved	(Bits 10–31)
DDM_FILPRT	Specifies Protected File (Bit 9)
DDM_FILSYS	Specifies System File (Bit 8)

DDM_FILHDD	Specifies Hidden File (Bit 7)
DDM_MODCP	Allows Modify Record Capability (Bit 6)
DDM_INSCP	Allows Insert Record Capability (Bit 5)
DDM_GETCP	Allows Get Record Capability (Bit 4)
DDM_INIEX	Inhibit Initial Extent (Bit 3)
DDM_DELCP	Allows Record Deletion (Bit 2)
DDM_TMPFIL	Temporary File (Bit 1)
DDM_ALDUPKEY	Allows Duplicate Keys (Bit 0)

DDM_FILPRT (Protected File)

Purpose	Specifies whether the file is protected. A protected file is protected from the DDMDelete function. If the DDMDelete function is attempted against a protected file, the function is rejected with an INVRQSRM reply message. A protected file does not prevent a file from being opened with access intents of MODAI, DELAI, or INSAI. Nor does a protected file prevent DDMMModifyRec, DDMDeleteRec, or DDMInsertRecxxx functions from being performed. These functions are controlled by the file capabilities attributes: DDM_MODCP, DDM_DELCP, and DDM_INSCP.
Bit number	Bit 9 of the CreateFlags word.
Bit value	A value of TRUE indicates that the file is protected from file management functions that would change the entire contents of the file. A value of FALSE indicates that the file is not protected from file management functions that would change the entire contents of the file. This is the default value.

DDM_FILSYS (System File)

Purpose	DDM_FILSYS(TRUE) indicates that the file was created with the FILE_SYSTEM attribute. A system file is the same as a non-system file in all respects except for the processing done during a directory search or scan in which the FILE_SYSTEM attribute is used to determine whether a file or subdirectory should be considered a match.
Bit number	Bit 8 of the CreateFlags word.
Bit value	A value of TRUE indicates that the file was created with the FILE_SYSTEM attribute. A value of FALSE indicates that the file was not created with the FILE_SYSTEM attribute.

DDM_FILHDD (Hidden File)

Purpose	DDM_FILHDD(TRUE) indicates that the file was created with the FILE_HIDDEN attribute. A hidden file is the same as a non-hidden file in all respects except for the processing done during a directory search or scan in which the FILE_HIDDEN attribute is used to determine whether a file or subdirectory should be considered a match.
Bit number	Bit 7 of the CreateFlags word.
Bit value	A value of TRUE indicates that the file is hidden. A value of FALSE indicates that the file is not hidden.

DDM_MODCP (Allow Modify Record Capability)

Purpose	The allow modify record capability bit specifies whether the data records of a file can be modified by a DDMMModifyRec or DDMTruncFile function. If the file is not modify-capable, a DDMMModifyRec function is rejected with an INVRQSRM reply message.
Bit number	Bit 6 of the CreateFlags word.
Bit value	A value of TRUE indicates that the data records of a file can be modified. A value of FALSE indicates that the data records of a file cannot be modified and that requests to modify the file are rejected.

DDM_INSCP (Allow Insert Record Capability)

Purpose	The allow insert record capability bit specifies whether the data records can be inserted into the file by either: DDMInsertRecxxx, or DDMLoadFilexxx If the file is not insert-capable these functions are rejected with an INVRQSRM reply message.
Bit number	Bit 5 of the CreateFlags word.
Bit value	A value of TRUE indicates that data records can be inserted into the file. A value of FALSE indicates that data records cannot be inserted into the file and that the request is rejected.

DDM_GETCP (Allow Get Record Capability)

Purpose	The get record capability bit specifies whether the contents of a file can be read by either: DDMGetRec, DDMSetxxx with DDM_NODATA(FALSE), or DDMUnloadFilexxx.
----------------	--

	If the file is not get-capable, these functions are rejected with an INVRQSRM reply message.
Bit number	Bit 4 of the CreateFlags word.
Bit value	A value of TRUE indicates that the contents of a file can be read by the requester. A value of FALSE indicates that the contents of a file cannot be read by the requester and the request is rejected.

DDM_INIEX (Inhibit Initial Extent)

Purpose	Specifies whether storage is to be allocated for the initial extent of a file when the file is created.
Bit number	Bit 3 of the CreateFlags word.
Bit value	A bit value of TRUE indicates that storage is not allocated for the initial extent of the file when the file is created. A bit value of FALSE indicates that storage is allocated for the initial extent of the file when the file is created.

DDM_DELCP (Allow Record Deletion)

Purpose	Specifies whether records may be deleted from the file being created.
Bit number	Bit 2 of the CreateFlags word.
Bit value	A bit value of TRUE indicates that records may be deleted from the file. A bit value of FALSE indicates that records may not be deleted from the file.

DDM_TMPFIL (Temporary File)

Purpose	Specifies whether the file being created is a permanent or temporary file.
Bit number	Bit 1 of the CreateFlags word.
Bit value	A bit value of TRUE indicates that the file being created is a <i>temporary</i> file. A temporary file only exists until: <ol style="list-style-type: none"> 1. The file is deleted. 2. Communications with the target are terminated. Temporary files operate exactly like permanent files while they exist. A bit value of FALSE indicates that the file being created is a permanent file. A permanent file exists until it is explicitly deleted. Termination of communications does <i>not</i> affect the existence of a permanent file.

DDM_ALDUPKEY (Allow Duplicate Keys)

Purpose	specifies whether duplicate keys are allowed for a file at the time the file is created.
Bit number	Bit 0 in the Create Flags word.
Bit value	A bit value of TRUE indicates that duplicate keys are allowed for the file being created. A bit value of FALSE indicates that duplicate keys are not allowed.

Chapter 6. VSAM API Reply Messages

This chapter provides detailed information about reply messages. Each reply message is accompanied by a brief explanation of the message, its code point, and its structure, which is defined by parameters.

For information about the parameters returned by the reply messages, see Chapter 4, "VSAM API Common Parameters" on page 361.

Reply Message Interface

A reply message is returned to the sender of a function to provide the sender with information about some condition that occurred during the processing of the function. A single function can generate several reply messages.

When a VSAM API function returns a non-zero return code, a `DDMGetReplyMessage` function should be issued immediately to obtain the reply messages. The Reply Message queue for a thread is cleared every time a new VSAM API function is issued. Therefore, the `DDMGetReplyMessage` must be issued before making any other VSAM API function call under this thread to avoid losing the reply messages corresponding to the function that returned the non-zero code.

All reply messages contain a severity code parameter that characterizes the severity of the condition reported. In addition, each reply message may define specific additional parameters to be returned with the message.

Reply Message Structure

LL	CP	LL	CP	DATA	LL	CP	DATA	LL	CP	DATA
----	----	----	----	------	----	----	------	----	----	------

The first length field (4 bytes) indicates the total length of the reply message, and the first code point (2 bytes) is the code point of the reply message which follows.

Subsequent length fields (4 bytes) are for the objects contained in the reply message. The code point words (2 bytes) indicate what data follows.

All length fields represent the length of the data, the code point, and the length field itself.

For information on how to get access to the reply message, see the "DDMGetReplyMessage (Get Reply Message)" on page 81.

Each reply message has a list of the data that may accompany it. Each data item is tagged with one of two possible return conditions:

- Distributed FileManager returns this information.
- The target server decides whether this information is returned.

Reply Messages

The DDM server is responsible for translating file system exceptions to the DDM-architected reply messages as described in this chapter.

If there is no reply message to which the condition can be translated, the DDM server replies with a CMDCHKRM reply message, which might contain the file system return code.

Mixed-case file names might be converted to upper-case file names. Therefore, any reply messages that contain a filename may not reflect the case that was used as input to the API.

Reply Messages

These VSAM API reply messages are returned by the local VSAM file system. There might be other reply messages returned by other DDM server implementations. See the documentation for your DDM server.

The VSAM reply messages are listed alphabetically in the following table:

Table 28 (Page 1 of 2). VSAM Reply Messages Listed Alphabetically

Message ID	Code Point	Message Title
ACCATHRM	X'1230'	Not Authorized to Use Access Method
ACCINTRM	X'1266'	Access Intent List Error
ACCMTHRM	X'1231'	Invalid Access Method
ADDRRM	X'F212'	Address Error
AGNPRMRM	X'1232'	Permanent Agent Error
BASNAMRM	X'1234'	Invalid Base File Name
CLSDMGRM	X'125E'	File Closed with Damage
CMDCHKRM	X'1254'	Command Check
COMMRM	X'F207'	Communications Error
CSRNSARM	X'1205'	Cursor Not Selecting a Record Position
CVTNFNRM	X'F202'	Conversion Table Not Found
DDFNFNRM	X'F201'	Data Description File Not Found
DFTRECRM	X'1204'	Default Record Error
DRCATHRM	X'1237'	Not Authorized to Directory
DRCFULRM	X'1258'	Directory Full
DTARECRM	X'1206'	Invalid Data Record
DUPFILRM	X'1207'	Duplicate File Name
DUPKDIRM	X'1208'	Duplicate Key Different Index
DUPKSIRM	X'1209'	Duplicate Key Same Index
DUPRNBRM	X'120A'	Duplicate Record Number
ENDFILRM	X'120B'	End of File Condition
EXSCNDRM	X'123A'	Existing Condition
FILATHRM	X'123B'	Not Authorized to File
FILDMGRM	X'125A'	File Damaged
FILERRRM	X'F216'	File Error
FILFULRM	X'120C'	File Is Full
FILIUSRM	X'120D'	File In Use
FILNAMRM	X'1212'	Invalid File Name
FILNFNRM	X'120E'	File Not Found
FILSNARM	X'120F'	File Space Not Available

Reply Messages

Table 28 (Page 2 of 2). VSAM Reply Messages Listed Alphabetically

Message ID	Code Point	Message Title
FILTNARM	X'121E'	File Temporarily Not Available
FUNATHRM	X'121C'	Not Authorized to Function
FUNNSPRM	X'1250'	Function Not Supported
HDLNFNRM	X'1257'	File Handle Not Found
INTATHRM	X'125C'	Not Authorized to Open Intent for Named File
INVFLGRM	X'F205'	Invalid Flag
INVRQSRM	X'123C'	Invalid Request
KEYDEFRM	X'123D'	Invalid Key Definition
KEYLENRM	X'122D'	Invalid Key Length
KEYUDIRM	X'1201'	Key Update Not Allowed by Different Index
KEYUSIRM	X'123F'	Key Update Not Allowed by Same Index
KEYVALRM	X'1240'	Invalid Key Value
LENGTHRM	X'F211'	Field Length Error
NEWNAMRM	X'124F'	Invalid New File Name
OBJNSPRM	X'1253'	Object Not Supported
OPNMAXRM	X'1244'	Concurrent Opens Exceeds Maximum
PRCCNVRM	X'1245'	Conversational Protocol Error
PRMNSPRM	X'1251'	Parameter Not Supported
RECDMGRM	X'1249'	Record Damaged
RECINARM	X'1259'	Record Inactive
RECIUSRM	X'124A'	Record In Use
RECLNRM	X'1215'	Record Length Mismatch
RECNAVRM	X'126F'	Record Not Available
RECNBRM	X'1224'	Record Number Out Of Bounds
RECNFNRM	X'1225'	Record Not Found
RSCLMTRM	X'1233'	Resource Limits Reached on Target System
SRCLMTRM	X'F210'	Resource Limits Reached in Source System
SYNTAXRM	X'124C'	Data Stream Syntax Error
TRGNSPRM	X'125F'	Target Not Supported on Target System
UPDCSRRM	X'124D'	Update Cursor Error
UPDINTRM	X'124E'	No Update Intent on Record
VALNSPRM	X'1252'	Parameter Value Not Supported
XLATERM	X'F203'	Translation Error

The VSAM reply messages are listed in code point order in the following table:

Table 29 (Page 1 of 2). VSAM Reply Messages Listed in Code Point Order

Code Point	Message ID	Message Title
X'1201'	KEYUDIRM	Key Update Not Allowed by Different Index
X'1204'	DFTRECRM	Default Record Error
X'1205'	CSRNSARM	Cursor Not Selecting a Record Position
X'1206'	DTARECRM	Invalid Data Record
X'1207'	DUPFILRM	Duplicate File Name
X'1208'	DUPKDIRM	Duplicate Key Different Index
X'1209'	DUPKSIRM	Duplicate Key Same Index
X'120A'	DUPRNBRM	Duplicate Record Number
X'120B'	ENDFILRM	End of File Condition
X'120C'	FILFULRM	File Is Full
X'120D'	FILIUSRM	File In Use
X'120E'	FILNFNRM	File Not Found

Reply Messages

Table 29 (Page 2 of 2). VSAM Reply Messages Listed in Code Point Order

Code Point	Message ID	Message Title
X'120F'	FILSNARM	File Space Not Available
X'1212'	FILNAMRM	Invalid File Name
X'1215'	RECLNRM	Record Length Mismatch
X'121C'	FUNATHRM	Not Authorized to Function
X'121E'	FILTARM	File Temporarily Not Available
X'1224'	RECNBRRM	Record Number Out Of Bounds
X'1225'	REC�FNRM	Record Not Found
X'122D'	KEYLENRM	Invalid Key Length
X'1230'	ACCATHRM	Not Authorized to Use Access Method
X'1231'	ACCMTHRM	Invalid Access Method
X'1232'	AGNPRMRM	Permanent Agent Error
X'1233'	RSCLMTRM	Resource Limits Reached on Target System
X'1234'	BASNAMRM	Invalid Base File Name
X'1237'	DRCATHRM	Not Authorized to Directory
X'123A'	EXSCNDRM	Existing Condition
X'123B'	FILATHRM	Not Authorized to File
X'123C'	INVRQSRM	Invalid Request
X'123D'	KEYDEFRM	Invalid Key Definition
X'123F'	KEYUSIRM	Key Update Not Allowed by Same Index
X'1240'	KEYVALRM	Invalid Key Value
X'1244'	OPNMAXRM	Concurrent Opens Exceeds Maximum
X'1245'	PRCCNVRM	Conversational Protocol Error
X'1249'	RECDMGRM	Record Damaged
X'124A'	RECIUSRM	Record In Use
X'124C'	SYNTAXRM	Data Stream Syntax Error
X'124D'	UPDCSRRM	Update Cursor Error
X'124E'	UPDINTRM	No Update Intent on Record
X'124F'	NEWNAMRM	Invalid New File Name
X'1250'	FUNNSPRM	Function Not Supported
X'1251'	PRMNSPRM	Parameter Not Supported
X'1252'	VALNSPRM	Parameter Value Not Supported
X'1253'	OBJNSPRM	Object Not Supported
X'1254'	CMDCHKRM	Command Check
X'1257'	HDLNFNRM	File Handle Not Found
X'1258'	DRCFULRM	Directory Full
X'1259'	RECINARM	Record Inactive
X'125A'	FILDMGRM	File Damaged
X'125C'	INTATHRM	Not Authorized to Open Intent for Named File
X'125E'	CLSDMGRM	File Closed with Damage
X'125F'	TRGNsprm	Parameter Not Supported on Target System
X'1266'	ACCINTRM	Access Intent List Error
X'126F'	RECNAVRM	Record Not Available
X'F201'	DDFNFNRM	Data Description File Not Found
X'F202'	CVTNFNRM	Conversion Table Not Found
X'F203'	XLATERM	Translation Error
X'F205'	INVFLGRM	Invalid Flag
X'F207'	COMMRM	Communications Error
X'F210'	SRCLMTRM	Resource Limits Reached in Source System
X'F211'	LENGTHRM	Field Length Error
X'F212'	ADDRRM	Address Error
X'F216'	FILERRRM	File Error

Reply Messages

ACCATHRM (Not Authorized to Use Access Method)

Purpose	The requester is not authorized to use the specified access method.	
Code Point	The code point for this term is X'1230'.	
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.	
	Parameter	Description
	SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
	ACCMTHCL	Access method class <ul style="list-style-type: none">• Code point is X'114E'.• Enumerated values for this parameter are:
X'1433'	RELNRBAM	(Relative by record number access method)
X'1435'	RNDRNBAM	(Random by record number access method)
X'1407'	CMBRNBAM	(Combined record number access method)
X'1432'	RELKEYAM	(Relative by key access method)
X'1434'	RNDKEYAM	(Random by key access method)
X'1406'	CMBKEYAM	(Combined keyed access method)
X'1405'	CMBACCAM	(Combined access access method)
	SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

ACCINTRM (Access Intent List Error)

Purpose	Indicates that the access-intent-list parameter in the DDOpen function is in error for one of the following reasons: <ul style="list-style-type: none">• The file does not support the requested access intent.• The file access capability specified on DDCreateRecFile does not support the requested access intent. For more information, see "DDOpen (Open File)" on page 127.
Code Point	The code point for this term is X'1266'.

Reply Messages

Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

ACCMTHRM (Invalid Access Method)

Purpose	Indicates that the function failed because the specified access method was in error. This can happen because: <ul style="list-style-type: none">• The specified access method class is not supported.• The access method class specified is not a defined access method class.
Code Point	The code point for this term is X'1231'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
ACCMTHCL	Access method class <ul style="list-style-type: none">• Code point is X'114E'.• Enumerated values for this parameter are:

X'1433'	RELNRBAM	(Relative by record number access method)
X'1435'	RNDRNBAM	(Random by record number access method)
X'1407'	CMBRNBAM	(Combined record number access method)
X'1432'	RELKEYAM	(Relative by key access method)
X'1434'	RNDKEYAM	(Random by key access method)
X'1406'	CMBKEYAM	(Combined keyed access method)

Reply Messages

X'1405' CMBACCAM (Combined access access method)

SRVDGN Server diagnostic information

- Code point is X'1153'.
- No information is returned.

ADDRRM (Address Error)

Purpose A buffer address of zero was specified when a non-zero value was expected.

Code Point The code point for this term is X'F212'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter Description

SVRCOD Severity code

- Code point is X'1149'.
- Returned.
- Enumerated value(s) for this parameter:

16 Severe Error Severity Code

SRVDGN Server diagnostic information

- Code point is X'1153'.
- Returned.
- Enumerated value(s) for this parameter:

0001	Record Buffer
0002	Key Buffer
0003	GEA (Get Extended Attribute Buffer)
0004	Record Number Buffer
0005	Get Extended Attribute Reply or Set Extended Attribute Buffer
0006	Record Count Buffer or Returned Record Count Buffer
0007	File Name or Title
0008	File Handle
0009	Flags Buffer
0010	Default Record Buffer
0011	Feedback Buffer

Reply Messages

AGNPRMRM (Permanent Agent Error)

Purpose	The function requested could not be completed because of a permanent error condition detected at the target system.
Code Point	The code point for this term is X'1232'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">16 Severe Error Severity Code32 Access Damage Severity Code64 Permanent Damage Severity Code
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

BASNAMRM (Invalid Base File Name)

Purpose	The base file name is not a valid target system file name.
Code Point	The code point for this term is X'1234'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code
BASFILNM	Base file <ul style="list-style-type: none">• Code point is X'1103'.• VSAM returns this information.

Reply Messages

SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.
---------------	--

CLSDMGRM (File Closed with Damage)

Purpose The file was closed as requested by the DDMClose function, but the file was damaged. That is, the file does not contain all the data of the file in the state required by DDM architecture.

If the target system blocks data for storage, the damage can result from failing to write the last block of data being processed to permanent storage.

Other reasons for this condition may also exist, as defined by the target system.

Code Point The code point for this term is X'125E'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 64 Permanent Damage Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code Point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

CMDCHKRM (Command Check)

Purpose An error occurred in a non-DDM related operating system support function that could not be mapped to an existing DDM error reply message.

Code Point The code point for this term is X'1254'.

Reply Messages

Structure

See “Reply Message Structure” on page 411 for the general structure of reply message data.

Parameter	Description
-----------	-------------

SVRCOD	Severity code
---------------	---------------

- Code point is X'1149'.
- Returned.
- Enumerated value(s) for this parameter:
 - 0** Information Only Severity Code
 - 4** Warning Severity Code
 - 8** Error Severity Code
 - 16** Severe Error Severity Code
 - 32** Access Damage Severity Code
 - 64** Permanent Damage Severity Code
 - 128** Session Damage Severity Code

SVRCOD can also contain an operating system error code. If the error code is from the operating system, SRVDGN is 2.

DTALCKST	Data lock status
-----------------	------------------

- Code point is X'115C'.
- Value is X'F1' (TRUE) if the data locks are the same as before the failure.
- Value is X'F0' (FALSE) if the data locks are not the same as before the failure.

CSRPOSST	Cursor position status
-----------------	------------------------

- Code point is X'115B'.
- Value is X'F1' (TRUE) if the cursor position is the same as before the function iteration that caused the reply message. TRUE is the only valid value if the severity code is **ERROR**.
- Value is X'F0' (FALSE) if the cursor position is not the same as before the function iteration that caused the reply message or is that the current cursor position is unknown.
- The target server determines whether this information is returned.

RECCNT	Record count
---------------	--------------

- Code point is X'111A'.
- Minimum value is 0.
- Information is returned if available.
- Required for requests to insert multiple records in a file.

Reply Messages

SRVDGN	Server diagnostic information
	<ul style="list-style-type: none"> • Code point is X'1153'. • Returned. • The target server determines whether this information is returned. • Enumerated value(s) for this parameter are: <ol style="list-style-type: none"> 1 FileShare parameter on the DDOpen was promoted to NON because the file is remote over the LAN (for local VSAM file system only). 2 An operating system error occurred and cannot be mapped to a reply message. The SVRCOD contains the value for the condition the operating system detected.

COMMRM (Communications Error)

Purpose	A problem was encountered communicating with a target system. The requestor is not authorized to use the specified access method.				
Code Point	The code point for this term is X'F207'.				
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.				
	<table> <thead> <tr> <th>Parameter</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>SVRCOD</td> <td>Severity code <ul style="list-style-type: none"> • Code point is X'1149'. • Returned. • Enumerated value(s) for this parameter: <ol style="list-style-type: none"> 16 Severe Error Severity Code </td> </tr> </tbody> </table>	Parameter	Description	SVRCOD	Severity code <ul style="list-style-type: none"> • Code point is X'1149'. • Returned. • Enumerated value(s) for this parameter: <ol style="list-style-type: none"> 16 Severe Error Severity Code
Parameter	Description				
SVRCOD	Severity code <ul style="list-style-type: none"> • Code point is X'1149'. • Returned. • Enumerated value(s) for this parameter: <ol style="list-style-type: none"> 16 Severe Error Severity Code 				
SRVDGN	Server diagnostic information <ul style="list-style-type: none"> • Code point is X'1153'. • Returned. • SOURCE DDM network enumerated value(s) for this parameter. See Table 30 on page 422. 				

The table below shows the hexadecimal (Hex) and decimal (Dec) values for COMMRM.

Reply Messages

Dec	Hex	Mnemonic	Possible Causes
1	1	APPC_NOT_ACTIVE	DFM cannot access the remote system. The possible causes are: <ul style="list-style-type: none"> • The network services have not been started. • The network link has not been started. • The specified APPC LU is not an LU accessible on the network.
2	2	COMM_ENV_NOT_STARTED	STRTDFMC has not been successfully executed (OS/2 only).
3	3	CONV_UNEXP_ENDED	A conversation with a target system has ended unexpectedly. Possible causes: <ul style="list-style-type: none"> • A problem on the network • A problem in the target system • A problem in the network access software • A problem in the operating system
4	4	INSUFF_LOCAL_RESOURCES	Local resources are not sufficient. Most likely, the stack size of the application is too small.
5	5	INTERNAL_ERROR_IN_DFMCM	An internal error has occurred in the record access communications manager component of DFM. Contact your service representative.
6	6	NO_SESSION_AVAILABLE	DFM tried to allocate a conversation with the remote system, but no session was available. Possible causes: <ul style="list-style-type: none"> • The network access software configuration conflicts with the DFM configuration data. • The network access link is not active. • A cable problem exists. • The target system is not active. • The session limit is exceeded.
7	7	PCS_ROUTER_ERROR	An internal error has occurred in the stream access communication manager component of DFM (OS/2 only). Contact your service representative.
8	8	TDDM_NOT_FOUND	An application requested record access to a file on a target system, but CONFIG.DFM contains no DFM_TARGET entry for that target system.

Reply Messages

Table 30 (Page 2 of 3). SRVDGN Values for COMMRM

Dec	Hex	Mnemonic	Possible Causes
9	9	TDDM_UNEXP_ENDED	The DFM target server has unexpectedly terminated the conversation. The most likely cause is the program that implements the DDM target server contains an error. Contact the supplier of the DDM target server.
10	A	TGT_ISSUED_SEND_ERROR	The DFM target server has issued the SEND_ERROR verb when it was not expected by DFM.
11	B	UNKNOWN_COMM_ERROR	DFM tried to communicate with a target system, but an unknown return code from the network access software occurred. Contact your service representative.
12	C	SRVDGN_DFMINIT_FAILURE	Unable to initialize the DFM control blocks from the binary configuration file <i>dfmcfg.dfm</i> . Ensure that the <i>dfmcfg.dfm</i> file is accessible and valid by issuing dfmcfg -c from the session where the application was started (for Windows only).
13	D	SRVDGN_INVALID_SECURITY_MODE	An unknown security mode was assigned to a server system. The DFM binary configuration file has most likely been corrupted. Recreate the DFM configuration file with the dfmcfg command (for Windows only).
14	E	SRVDGN_INVALID_SECURITY_NONE	A security mode violation was detected for the remote system. The security mode, either specified explicitly in the DFM configuration or by default if not explicitly specified is PROGRAM. The dfmlogon command was not issued for the server system. Remember if you do not explicitly specify the security mode for a server system in the DFM configuration file, the default is PROGRAM. Issue the dfmlogon command to define logon information for the server system (for Windows only).
15	F	SRVDGN_INVALID_SECURITY_PROGRAM	A security mode violation was detected. The dfmlogon command was issued for the server system. However, either the password or the user ID, or both, were not specified (for Windows only).
17	11	SRVDGN_INVALID_UNC_PATHNAME	The specification of a remote file for DFM to access has an not valid UNC format (for Windows only).

Reply Messages

Dec	Hex	Mnemonic	Possible Causes
18	12	BAD_ENV	A problem with the runtime environment has caused a fatal error. One or both of the following files cannot be loaded or is corrupted: <i>dfmmain.dll</i> and <i>dfmext.dll</i> (for Windows only).
19	13	SRVDGN_INVALID_SECURITY_SERVER	The remote server determined that security information is not valid. The possible causes are (for Windows only): <ul style="list-style-type: none"> • The specified type of security access is not acceptable. • The user ID is invalid. • The user ID and password combination is not valid.
20	14	SRVDGN_INVALID_TPN_SERVER	The remote system does not support the SNA registered DDM server transaction program, or the DDM server is not active.
21	15	SRVDGN_INVALID_PARAMETER	The remote system LU name is not valid (cannot be found on the network), or the mode name is not valid for the remote system, or the LU name/mode name combination is not valid. Note, some network access software requires specification of LU name/mode name combinations at configuration time.

CSRNSARM (Cursor Not Selecting a Record Position)

Purpose The function failed because the cursor is not presently selecting a record position. The cursor is either at the BOF or EOF position, or its position is unknown.

Code Point The code point for this term is X'1205'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
-----------	-------------

SVRCOD	Severity code <ul style="list-style-type: none"> • Code point is X'1149'. • Returned. • Enumerated value(s) for this parameter:
---------------	--

8 Error Severity Code

Reply Messages

CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

CVTNFNRM (Conversion Table Not Found)

Purpose	The specified character conversion table was not found. No character translation is performed. This reply message is returned when DFM/2 tries to access a conversion table for a character-to-character field conversion. The conversion table to be loaded depends on the code page IDs related to the from-character field and the to-character field (OS/2 only).
Code Point	The code point for this term is X'F202'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	Conversion Table File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.

DDFNFNRM (Data Description File Not Found)

Purpose	The named Data Description File was not found. No translation is performed during the current function request. This reply message is returned when DFM/2 tries to load the data description information for a remote file and it could not find the related DDF file, as specified in the MAPFMT entry of the DFM/2 configuration file (OS/2 only).
----------------	--

Reply Messages

Code Point	The code point for this term is X'F201'
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data. the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	Data Description File Name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.

DFTRECRM (Default Record Error)

Purpose	The request to initialize a file could not be completed because the default record does not meet the target server's criteria. For example, default inactive record initialization cannot be done on sequential files that do not have delete capability.
Code Point	The code point for this term is X'1204'.
Structure	See the description at the beginning of this section for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code 16 Severe Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

DRCATHRM (Not Authorized to Directory)

Purpose	The user is not authorized to access or update the directory that is specified or implied by a file name.
----------------	---

Reply Messages

Code Point	The code point for this term is X'1237'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

DRCFULRM (Directory Full)

Purpose	The directory specified or implied by a file name is full and does not have space for the file being created or renamed.
Code Point	The code point for this term is X'1258'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

DTARECRM (Invalid Data Record)

Purpose	A record to be inserted in a file cannot contain a data value that specifies an inactive record to the local data management on the target system.
----------------	--

Reply Messages

An inactive record can not be inserted into a non-delete-capable file.

If it is necessary to insert an inactive record into a delete-capable file, send RECINA.

Code Point

The code point for this term is X'1206'.

Structure

See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.• For alternate index files, this is the base file name.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.• Required for requests to insert multiple records in a file.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• Information is returned if available.• This is the record number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

DUPFILRM (Duplicate File Name)

Purpose An attempt to create or rename a file failed because it duplicates an existing file name. The target system does not allow duplicates.

Code Point The code point for this term is X'1207'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code Point is X'1153'.• No information is returned.

DUPKDIRM (Duplicate Key Different Index)

Purpose The function was not completed because the record sent contains a field that duplicates a key in an index different than the one being used to access the file. The other index does not allow duplicate key records.
The target returns the name of the file(s) in which the duplicate key would occur (ERRFILNM).

Code Point The code point for this term is X'1208'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code 16 Severe Error Severity Code

Reply Messages

CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
ERRFILNM	Error file name <ul style="list-style-type: none">• Code point is X'1126'.• Returned.• Only one Error File Name is required. Additional Error File Names may be specified if they are known.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Returned for requests to insert multiple records in a file. In other cases, the DDM server determines whether this information is returned.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• The DDM server determines whether this information is returned.• This is the record number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

DUPKSIRM (Duplicate Key Same Index)

Purpose	The function was not completed because the record duplicates a key in the index being used to access the file. This index does not allow duplicate key records.
Code Point	The code point for this term is X'1209'.

Reply Messages

Structure

See “Reply Message Structure” on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">4 Warning (duplicate record found). Indicates that the API access completed successfully and notifies the caller that the record being returned has a duplicate key. This condition was previously flagged as an error.8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Returned for requests to insert multiple records in a file. In other cases, the DDM server determines whether this information is returned.
RECNR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• This is the record number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

DUPRNBRM (Duplicate Record Number)

Purpose	A record cannot be inserted at a record position that is occupied by an active record.
Code Point	The code point for this term is X'120A'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Returned for requests to insert multiple records in a file. In other cases, the DDM server determines whether this information is returned.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

ENDFILRM (End of File)

Purpose

It is not possible to retrieve a record that is outside the BOF, EOF, or some specified file limit with the following functions:

Function

Limits

DDMSetNextRec

Always the last and first record positions, respectively, in the file.

DDMSetPrevious

Always the last and first record positions, respectively, in the file.

DDMSetKeyPrevious

The first record, in key sequence, of the file.

DDMSetKeyNext

The last record, in key sequence, of the file, or the high key limit established by a DDMSetKeyLimits function.

DDMSetNextKeyEqual

The last record (in key sequence) of the file, the high key limit established by a DDMSetKeyLimits function, or the key value specified by the KEYVAL parameter on the DDMSetNextKeyEqual function.

Code Point

The code point for this term is X'120B'.

Structure

See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter

Description

SVRCOD

Severity code

- Code point is X'1149'.
- Returned.
- Enumerated value(s) for this parameter:

4 Warning Severity Code

FILNAM

File name

- Code point is X'110E'.
- Returned.

SRVDGN

Server diagnostic information

- Code point is X'1153'.
- No information is returned.

Reply Messages

Examples:

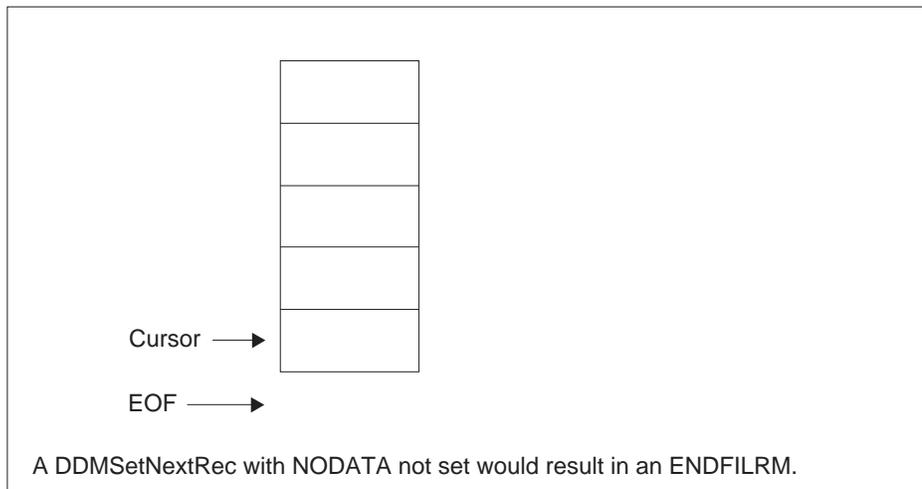


Figure 84. DDMSetNextRec ENDFILRM

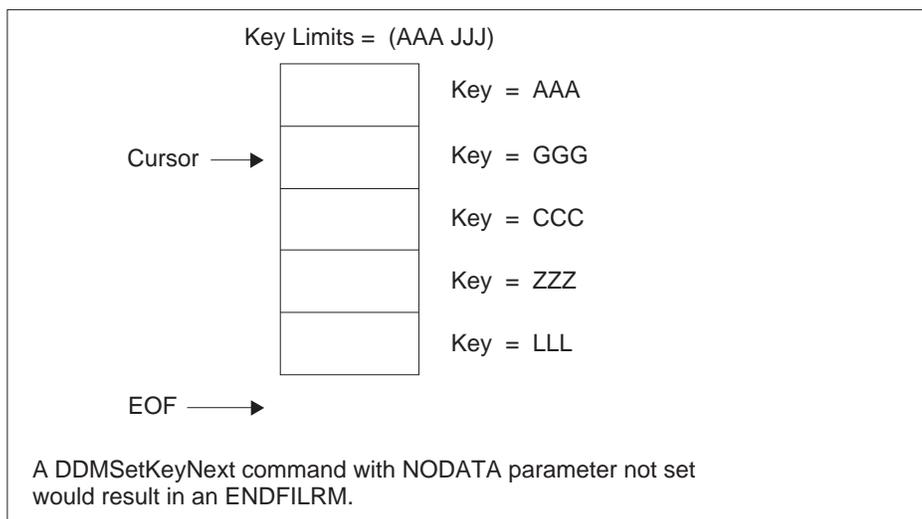


Figure 85. DDMSetKeyNext ENDFILRM

Reply Messages

EXSCNDRM (Existing Condition)

Purpose	A request was made that would have resulted in a condition that already exists. For example: <ul style="list-style-type: none">• A request to create a file when a file by that name already exists.• A request to unlock a record that is not locked.• A request to delete a file that cannot be found.• A request to delete a record that is already deleted.• A request to rename a file to the same name.								
Code Point	The code point for this term is X'123A'.								
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data. <table><thead><tr><th>Parameter</th><th>Description</th></tr></thead><tbody><tr><td>SVRCOD</td><td>Severity code<ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 4 Warning Severity Code</td></tr><tr><td>FILNAM</td><td>File name<ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.</td></tr><tr><td>SRVDGN</td><td>Server diagnostic information<ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.</td></tr></tbody></table>	Parameter	Description	SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 4 Warning Severity Code	FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.	SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.
Parameter	Description								
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 4 Warning Severity Code								
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Information is returned if available.								
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.								

FILATHRM (Not Authorized to File)

Purpose	The user is not authorized to perform the requested function on the file being accessed.
Code Point	The code point for this term is X'123B'.

Reply Messages

Structure	See “Reply Message Structure” on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• Enumerated value(s) for this parameter: 0 The operating system denied access to the file. 1 Access attempt to byte stream file with VSAM API. Byte stream is not a supported record type.

FILDMGRM (File Damaged)

Purpose	<p>The file may be damaged. Some of the indications of a damaged file in the local VSAM file system are:</p> <ul style="list-style-type: none">• The file-change date and time recorded by a VSAM API is not the same as the file-change date and time recorded by the file system. The function continues processing (SVRCOD=4). <p>Either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file. The local VSAM file system resynchronizes the file-change date and time if it can get write access to the file, unless a higher severity condition prevents it from doing so. Re-synchronizing the date and time corrects only this particular file-damaged condition, but the file may still be damaged. To verify that the file is not damaged, use DDMCopyFile or DDMUnloadFileFirst with</p>
----------------	---

Reply Messages

AccessFlags=DDM_BYPDGM|DDM_RTNINA and inspect the result.

- An index file is not consistent with its base file. The function is rejected (SVRCOD=16).

The file-change date and time recorded by the VSAM API for the base file is not the same as the base file's file-change date and time that was recorded as an attribute of the index file. Either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has replaced a base file or an index file without replacing all of the files in the file object. The local VSAM file system does not resynchronize the file-change date and time.

Both of the above conditions can exist at the same time for the same index file, causing two FILDMGRM reply messages to be returned, one for SVRCOD=4 followed by one for SVRCOD=16.

Code Point Structure

The code point for this term is X'125A'.

See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">4 Warning Severity Code8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code64 Permanent Damage Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.

Reply Messages

RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• This is the record number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.• Enumerated value for this parameter:<ul style="list-style-type: none">1 Either an aborted DDM application has left the file in an inconsistent state or a non-DDM application has changed the file.

FILFULRM (File Is Full)

Purpose	A file is full when a record cannot be added to the end of the file because: <ul style="list-style-type: none">• All record positions in the file have been filled and the file is not extendable.• All record positions in the file have been filled and the file has been extended the maximum number of times.• There are not enough bytes available in the file to insert the record and the file is not extendable, or the maximum number of extents have already been made. For example, if there are 45 bytes of space available in the file and an attempt is made to insert a record of 150 bytes, a FILFULRM reply message results.
Code Point	The code point for this term is X'120C'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.

Reply Messages

CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
RECNR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• This is the number of the record being operated on by the function.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

FILIUSRM (File in Use)

Purpose	The named file is locked by another user at a level that prevents the requested function from obtaining the locks it requires.
Code Point	The code point for this term is X'120D'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.

Reply Messages

SRVDGN	Server diagnostic information
	<ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

FILNAMRM (Invalid File Name)

Purpose	The file name specified on the function is not a valid target system file name.
Code Point	The code point for this term is X'1212'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.• This is the file name that is in error.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

FILNFNRM (File Not Found)

Purpose	The named file (specified on the function) cannot be found on the target system.
Code Point	The code point for this term is X'120E'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code

Reply Messages

FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.• This is the file name that is in error.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

FILSNARM (File Space Not Available)

Purpose	The file cannot be created or extended because the operating system does not have sufficient space available.
Code Point	The code point for this term is X'120F'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

FILTNARM (File Temporarily Not Available)

Purpose	The target system has temporarily made the file unavailable to all users. Either the file is damaged and must be repaired before further use, or a target system process, such as disk compression, prevents immediate use.
Code Point	The code point for this term is X'121E'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code64 Permanent Damage Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code Point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

FUNATHRM (Not Authorized to Function)

Purpose	The user is not authorized to perform the requested function.
Code Point	The code point for this term is X'121C'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code

Reply Messages

SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.
---------------	--

FUNNSPRM (Function Not Supported)

Purpose	The function specified is not recognized or not supported for the specified target object.
Code Point	The code point for this term is X'1250'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
CODPNT	Code point attribute <ul style="list-style-type: none">• Code point is X'000C'.• Returned.• Specifies the code point of the function not supported.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

HDLNFNRM (File Handle Not Found)

Purpose	The file handle specified is not known or if the handle from DDMLoadFileFirst or DDMUnloadFileFirst is not used as the handle for a DDMLoadFileNext or DDMUnloadFileNext, this reply message will be returned.
Code Point	The code point for this term is X'1257'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:

Reply Messages

8 Error Severity Code

SRVDGN	Server diagnostic information
	<ul style="list-style-type: none">• Code point is X'1153'.• Handle number is returned.

INTATHRM (Not Authorized to Open Intent for Named File)

Purpose The user is not authorized to open the file with the specified processing intent. This message is returned by servers that validate the user's authorization to access a file when the file is opened. Servers can allow the file to be opened without validation of the requester's specified intents if authorizations are subsequently validated for each function used to access an opened file.

Code Point The code point for this term is X'125C'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
ACCINTLS	Access intent list <ul style="list-style-type: none">• Code point is X'1134'.• Specifies the access intents for which the requester is not authorized.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

INVFLGRM (Invalid Flag)

Purpose One or more reserved bits have been set in a flag word.

Code Point The code point for this term is X'F205'.

Reply Messages

Structure

See “Reply Message Structure” on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'• Returned.• Enumerated value(s) for this parameter: 16 Severe Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'• Returned.• Reflects the reserved bits that had been set on.

INVRQSRM (Invalid Request)

Purpose

A request can be invalid for one of the following reasons:

- There is conflict with a user-specified attribute of the file, such as:
 - The function issues a request to delete a record from a non-delete-capable file.
 - The function violates the access intents specified when the file was opened.
- The requester attempted to delete a file that is the base file for some alternate index files.
- The requested function is supported by the access method but not by the file class to which the access method is opened.
- A DDMSetKeyLimits function was issued for a file that was created with keys such that all parts of the key are not ascending.
- A DDM_ALLREC bit was set on a DDMSetNextRec, DDMSetPrevious, DDMSetFirst, or DDMSetLast function for a direct file.
- An alternate index file was specified as the base file of an alternate index file on the DDMCreateAltIndex function.
- The value of LowKeyLim is after the value of HiKeyLim on a DDMSetKeyLimits function.
- An attempt was made to delete or clear a protected file.
- A DDMTruncFile function:

Reply Messages

- For file opened for read only (GETAI, but not MODAI)
- For a read-only-file (GETCP, but not MODCP).
- The requester attempted to create an alternate index file with a path qualifier that was different than the path qualifier of the base file.

Code Point The code point for this term is X'123C'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• Information is returned if available.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">15 The file is protected.

KEYDEFM (Invalid Key Definition)

Purpose The key definition is invalid for the reason specified by the KEYDEFCD parameter.

Code Point The code point for this term is X'123D'.

Reply Messages

Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
KEYDEFCD	Key definition error code <ul style="list-style-type: none">• Code point is X'1164'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

KEYLENRM (Invalid Key Length)

Purpose	Specifies that the key value provided on a function is not the length required by the requested function. This can be caused by: <ul style="list-style-type: none">• Specifying a partial key on a function that requires full keys.• Specifying a key length greater than the maximum length key supported by the target system.• Specifying a record key value whose length is greater than the defined key length of the file.
Code Point	The code point for this term is X'122D'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code 16 Severe Error Severity Code

Reply Messages

FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

KEYUDIRM (Key Update Not Allowed by Different Index)

Purpose	A different file does not allow its key value (of the record being modified) to be changed.
Code Point	The code point for this term is X'1201'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
ERRFILNM	Error file name <ul style="list-style-type: none">• Code point is X'1126'.• Returned.• Repeatable.• Only 1 error file name is required. Additional error file names may be specified if they are known.
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

KEYUSIRM (Key Update Not Allowed by Same Index)

Purpose	The file index being used to access the file does not allow the key value (of the record being modified) to be changed.
Code Point	The code point for this term is X'123F'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

KEYVALRM (Invalid Key Value)

Purpose	Specifies that the key value provided on a function or a record is not valid. This can be caused by: <ul style="list-style-type: none">• Specifying a variable-length record that does not contain all of the fields for the defined file key.• Specifying a key that is not valid for the target server.
Code Point	The code point for this term is X'1240'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
KEYVAL	Key value in error <ul style="list-style-type: none">• Code point is X'1115'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Returned for requests to insert multiple records in a file.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• This is the number of the record being operated on by the function.

Reply Messages

SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.
---------------	--

LENGTHRM (Field Length Error)

Purpose	A field was found with incorrect length.
Code Point	The code point for this term is X'F211'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">4 Warning Severity Code16 Severe Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">0001 Maximum Record Length Exceeded The maximum record length the local VSAM file system supports is 65,000 bytes. The maximum record length Distributed FileManager/MVS supports is 32000 bytes.0002 Record Buffer Too Small If the buffer is at least 4 bytes long, and no records have been placed in the buffer, the first 4 bytes contain the length of the record that did not fit.0003 Key Definition Buffer Too Small If the buffer is at least 4 bytes long, the first 4 bytes contain the required length of the buffer in order for the key definition information to fit.

Reply Messages

0004	Extended Attribute Reply Buffer Too Small	If the buffer is at least 4 bytes long, the first 4 bytes contain the required length.
0005	Extended Attribute Input Buffer Length Error	
0007	Default Record Buffer Length Error	The default record buffer is outside the allowable limits.

NEWNAMRM (Invalid New File Name)

Purpose	The new file name is not a valid target system file name.
Code Point	The code point for this term is X'124F'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">Code point is X'1149'.Returned.Enumerated value(s) for this parameter: 8 Error Severity Code
NEWFILNM	New file name <ul style="list-style-type: none">Code point is X'114F'.This is the file name that is in error.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">Code point is X'1153'.No information is returned.

OBJNSPRM (Object Not Supported)

Purpose	The object specified as data in a buffer is not recognized or not supported for the function associated with the object. Only active and inactive records are recognized. OBJNSPRM is also returned if an object is found in a valid collection that is part of a buffer (such as the RECAL collection) that is not valid for that collection.
Code Point	The code point for this term is X'1253'.

Reply Messages

Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CODPNT	Code point attribute <ul style="list-style-type: none">• Code point is X'000C'.• Returned.• This is the code point of the object that is not supported.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

OPNMAXRM (Concurrent Opens Exceeds Maximum)

Purpose	The number of concurrent DDOpen functions to the same file exceeds the target server maximum.
Code Point	The code point for this term is X'1244'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.

Reply Messages

MAXOPN	Maximum number of files opened <ul style="list-style-type: none">• Code point is X'1157'.• Specifies the maximum number of opens to the same file.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

PRCCNVRM (Conversational Protocol Error)

Purpose	A conversational protocol error occurred.
Code Point	The code point for this term is X'1245'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code128 Session Damage Severity Code

PRCCNVCD	Conversational protocol error code <ul style="list-style-type: none">• Code point is X'113F'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">0001 RPYDSS received by target communication manager0002 Multiple DSSs sent without chaining or multiple DSS chains sent0003 OBJDSS sent when not allowed0004 The next correlation identifier was not ascending0005 The request correlation identifier of OBJDSS and RPYDSS are not equal0006 EXCSAT was not the first function after the connection was established
-----------------	--

Reply Messages

RECCNT	Recode count <ul style="list-style-type: none">• Code point is X'111A'• Minimum value is 0• Information is returned if available
SVRDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'• No information is returned.

PRMNSPRM (Parameter Not Supported)

Purpose The parameter specified is not recognized or not supported for the associated function.

Code Point The code point for this term is X'1251'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
CODPNT	Code point attribute <ul style="list-style-type: none">• Code point is X'000C'.• Returned.• Specifies the code point of the parameter not supported.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECDMGRM (Record Damaged)

Purpose A record in the file is damaged and cannot be accessed. A damaged record is one in which the Code point is not an active or inactive record.

Damaged records can be bypassed as an option of the following functions:

DDMSetKeyNext
DDMSetNextRec
DDMUnloadFileFirst
DDMUnloadFileNext

Reply Messages

See “DDM_BYPDMG (Bypass Damaged Records)” on page 401.

RECDMGRM is returned with a severity code of WARNING for every damaged record that is bypassed. The record number of the bypassed record is also returned. If damaged records cannot be bypassed, this message is returned with a severity code of ERROR or greater.

Code Point

The code point for this term is X'1249'.

Structure

See “Reply Message Structure” on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">4 Warning Severity Code8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code64 Permanent Damage Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

Reply Messages

RECINARM (Record Inactive)

Purpose	RECINARM is returned with the following severity codes:
	SVRCOD Reason
	X'0004' This is returned when a DDMSxxxx function has moved the cursor to an inactive record.
	X'0008' or higher This is returned when the record is inactive, and the function cannot be executed.
Code Point	The code point for this term is X'1259'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
	Parameter Description
	SVRCOD Severity code
	<ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">4 Warning Severity Code8 Error Severity Code16 Severe Error Severity Code
	FILNAM File name
	<ul style="list-style-type: none">• Code point is X'110E'.• Returned.
	SRVDGN Server diagnostic information
	<ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECIUSRM (Record in Use)

Purpose	The record cannot be locked or accessed. This happens because another user has the record locked at a level that prevents the record from being locked or accessed by other users.
Code Point	The code point for this term is X'124A'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
	Parameter Description
	SVRCOD Severity code
	<ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code

Reply Messages

	16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• OPTIONAL.• Information is returned if available.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• Information is returned if available.• This is the number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECLNRM (Record Length Mismatch)

Purpose	<p>The length of a data record does not match the length of the current record position.</p> <p>If the record class is fixed and the record to be inserted is an active record, the length of the record object must be equal to the length of the record object header (length and code point) plus the length of the record object data. See "RECORD (Record)" on page 391 for more information.</p> <p>If the record to be inserted is an inactive record, the record length represented by the inactive record must be the same as the length defined for a record in the file. (See "RECINA (Inactive Record)" on page 389 for more information.)</p>
Code Point	The code point for this term is X'1215'

Reply Messages

Structure

See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
RECNR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• Information is returned if available.• This is the number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECNAVRM (Record Not Available)

Purpose	The requested record cannot be retrieved because it is not available to the file.
Code Point	The code point for this term is X'126F'.

Reply Messages

Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code 16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECNBRRM (Record Number Out of Bounds)

Purpose	The specified record number is outside the boundaries of the file. For a definition of file boundaries, see "DDMInsertRecNum (Insert by Record Number)" on page 98.
Code Point	The code point for this term is X'1224'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code 16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.

Reply Messages

DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
RECNBR	Record number <ul style="list-style-type: none">• Code point is X'111D'.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RECNFNRM (Record Not Found)

Purpose The cursor cannot be positioned because a record that satisfies the absolute or relative positioning parameters of a function does not exist.

Code Point The code point for this term is X'1225'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.

Reply Messages

FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

RSCLMTRM (Resource Limits Reached on Target System)

Purpose	The requested function could not be completed because of insufficient target server resources. Examples of resource limits are: <ul style="list-style-type: none">• The target agent has insufficient memory to keep track of more open files.• The lock manager cannot obtain another lock.• The communication manager's send or receive buffer overflowed.• The MAX_SEND_LIMIT in a TARGET_SYSTEM statement of the DFM configuration file is set to a low value.
Code Point	The code point for this term is X'1233'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter:<ul style="list-style-type: none">8 Error Severity Code16 Severe Error Severity Code32 Access Damage Severity Code64 Permanent Damage Severity Code128 Session Damage Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• The target server determines whether this information is returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• The target server determines whether this information is returned.

Reply Messages

FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned when the FILNAM parameter is specified for the function. In other cases, the target server determines whether this information is returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

SRCLMTRM (Resource Limit Reached in Source System)

Purpose	Some resource has reached its limit in the source system.
Code Point	The code point for this term is X'F210'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'• Returned.• Enumerated value(s) for this parameter: 16 Severe Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'• No information is returned.

SYNTAXRM (Data Stream Syntax Error)

Purpose	The data sent to the target agent does not conform to the structural requirements of DDM architecture. The target agent terminated parsing of the Data Stream Structure (DSS) when the condition specified by the Syntax Error Code parameter was detected.
Code Point	The code point for this term is X'124C'.

Reply Messages

Structure	See the description at the beginning of this section for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated value(s) for this parameter: 8 Error Severity Code
SYNERRCD	Syntax error code <ul style="list-style-type: none">• Code point is X'114A'.• Returned.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Information is returned if available.
CODPNT	Code point attribute <ul style="list-style-type: none">• Code point is X'000C'.• Returned.• Specifies the code point of the object that caused the syntax error.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

TRGNSPRM (Parameter Not Supported on Target System)

Purpose	The parameter specified cannot be supported on the target system.
Code Point	The code point for this term is X'125F'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated values for this parameter: 8 Error Severity Code
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

UPDCSRRM (Update Cursor Error)

Purpose The cursor cannot be updated to point to the last record inserted in the file.

This error can be sent only if the function set the UPDCSR bit flag for the Access Flags parameter.

Code Point The code point for this term is X'124D'.

Structure See "Reply Message Structure" on page 411 for the general structure of reply message data.

Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none"> Code point is X'1149'. Returned. Enumerated values for this parameter: <ul style="list-style-type: none"> 8 Error Severity Code 16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none"> Code point is X'115B'. Returned.
DTALCKST	Data lock status <ul style="list-style-type: none"> Code point is X'115C'. Returned.
FILNAM	File name <ul style="list-style-type: none"> Code point is X'110E'. Returned.
RECCNT	Record count <ul style="list-style-type: none"> Code point is X'111A'. Minimum value is 0. Returned for requests to insert multiple records in a file.
RECNR	Record number <ul style="list-style-type: none"> Code point is X'111D'. Information is returned if available. This is the number of the record being operated on by the function.
SRVDGN	Server diagnostic information <ul style="list-style-type: none"> Code point is X'1153'. No information is returned.

Reply Messages

UPDINTRM (No Update Intent on Record)

Purpose	The record cannot be updated for one of the following reasons: <ul style="list-style-type: none">• An update intent has <i>not</i> been placed on the record by the requester.• The update intent may have been removed because of a previous function issued by the requester.
Code Point	The code point for this term is X'124E'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated values for this parameter: 8 Error Severity Code 16 Severe Error Severity Code
CSRPOSST	Cursor position status <ul style="list-style-type: none">• Code point is X'115B'.• Returned.
DTALCKST	Data lock status <ul style="list-style-type: none">• Code point is X'115C'.• Returned.
FILNAM	File name <ul style="list-style-type: none">• Code point is X'110E'.• Returned.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

VALNSPRM (Parameter Value Not Supported)

Purpose	The parameter value specified is not recognized or not supported for the named parameter. The function parameter in error is returned as a parameter in this message.
Code Point	The code point for this term is X'1252'.

Reply Messages

Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated values for this parameter: 8 Error Severity Code
CODPNT	Code point attribute <ul style="list-style-type: none">• Code point is X'000C'.• Returned.• Return the code point of the parameter whose value is not supported.
RECCNT	Record count <ul style="list-style-type: none">• Code point is X'111A'.• Minimum value is 0.• Required for requests to insert multiple records in a file.
SRVDGN	Server diagnostic information <ul style="list-style-type: none">• Code point is X'1153'.• No information is returned.

XLATERM (Translation Error)

Purpose	An error occurred during translation of a record or field. The record or field is not translated. This reply message is returned when DFM tries to convert a record from source into target format, or vice versa, by using the data description sequences.
Code Point	The code point for this term is X'F203'.
Structure	See "Reply Message Structure" on page 411 for the general structure of reply message data.
Parameter	Description
SVRCOD	Severity code <ul style="list-style-type: none">• Code point is X'1149'.• Returned.• Enumerated values for this parameter: 4 Warning Severity Code 8 Error Severity Code 16 Severe Error or Severity Code

Reply Messages

SVRDGN

Server diagnostic information

- Code point is X'1153'.
- Returned.
- Enumerated values for this parameter:

0001 Rounding error

0002 Truncation error

0006 Possible causes:

- CDRASRV environment variable not set (Windows or AIX only).
- CDRA conversion table not available.

0101 Range error

0102 Untranslated data

0103 Modification intent, but the view does not cover the entire base record (reduced view)

0104 A partial numeric key field cannot be translated

Other server diagnostic values might be returned. See *SMARTdata UTILITIES Data Description and Conversion*.

Appendix A. Programming Extended Attributes in VSAM APIs

The following example from a C program illustrates how extended attribute information can be prepared for a VSAM API. The particular APIs used are DDMSetPathInfo and DDMQueryPathInfo. It is assumed that a sequential file already exists and the file name coded in the C application has its value in SeqFN.

See "Extended Attributes" on page 5 for an overview of extended attributes used by the VSAM APIs and the relationship of the DOS-based EAOP2, GEA2List, and FEA2List structures.

```
/*-----  
-- SYMBOLIC CONSTANTS  
-----*/  
#define FILCLS_NAME ".DDM_FILCLS"  
#define DELCP_NAME ".DDM_DELCP" /*@WQA*/  
#define TITLE_NAME ".DDM_TITLE" /*@WQC*/  
#define TitleString "Title String" /*@WQC*/  
#define FILCLS_SIZE sizeof(OBJLENGTH) + (2 * sizeof(CODEPOINT)) /*@WQM*/  
#define DELCP_SIZE sizeof(OBJLENGTH) + sizeof(CODEPOINT) + 1 /*@WQM*/  
#define TITLE_SIZE sizeof(OBJLENGTH) + sizeof(CODEPOINT) + strlen(TitleString) /*@WQC*/  
  
.  
.  
.  
  
/* OS/2 extended attribute structures */  
EAOP2 Eaop; /* EA structure for DDMQueryPathInfo @WQC*/  
EAOP2 Eaop2; /* EA structure for DDMSetPathInfo @WQC*/  
PFEA2 pFEA; /* Pointer to FEA2 list entry @WQC*/  
PGEA2 pGEA; /* Pointer to GEA2 list entry @WQC*/  
INT FEASize; /* Tally size of FEA2 list area @WQC*/  
INT GEASize; /* Tally size of GEA2 list area @WQC*/  
INT FEA2Size; /* Tally size of second FEA2 list @WQC*/  
  
ULONG Remainder; /* Holds remainder-byte offset calc @WQA*/  
LONG i; /* Controls FEA2 WHILE loop @WQA*/  
  
.  
.  
.
```

Figure 86 (Part 1 of 11). Example of C Program using Extended Attributes

```

/*****@WOA*/
/* Prepare and execute a DDMQueryPathInfo call to query a @WOA*/
/* file's Extended Attributes. @WOA*/
/*****@WOA*/
/* The DDM call to query a file's extended attributes is @WOA*/
/* based on the OS/2 extended attributes model. As such, the @WOA*/
/* calls to DDMQueryFileInfo and DDMQueryPathInfo must pass a @WOA*/
/* pointer to an EAOP2 structure which, in turn, contains @WOA*/
/* pointers to the GEA2LIST area and the FEA2LIST area. @WOA*/
/* The GEA2LIST area contains a header and variable length @WOA*/
/* GEA2 list entries. Each list entry identifies one EA @WOA*/
/* being queried. The FEA2LIST area is where the returned @WOA*/
/* information will be set. @WOA*/
/* @WOA*/
/* The EAOP2, FEA2LIST, GEA2LIST, FEA2 and GEA2 are defined @WOA*/
/* in DUBDEFS.H which is included by DUB.H. The format of the @WOA*/
/* values which can be returned are documented in the VSAM @WOA*/
/* API Reference manual in the "VSAM API Common Parameters" @WOA*/
/* chapter. @WOA*/

/* Steps: @WOA*/
/* 1. Calculate the sizes of the GEA2LIST and FEA2LIST areas @WOA*/
/* 2. Do the GEA2LIST + FEA2LIST malloc + set EAOP2 pointers @WOA*/
/* 3. Fill in the GEA2LIST area @WOA*/
/* 4. Fill in the FEA2LIST area @WOA*/
/* 5. Issue the DDMQueryPathInfo @WOA*/
/* 6. Extract DDM attribute data from the FEA2LIST area @WOA*/
/* 7. Free the GEA2LIST and FEA2LIST areas @WOA*/

/-----*/
-- Set up for DDMQueryPathInfo: @WOM
--
-- Build an extended attribute GEA2LIST area with two GEA2 @WOC
-- list entries specifying the DELCP and FILCLS EAs. The @WOC
-- attributes queried and structure content match those in @WOC
-- the "Extended Attributes" section of the VSAM API Reference @WOC
-- manual. @WOC
--

/-----*/
/***** STEP 1 *****/@WOA*/
/* 1. Calculate the sizes of the GEA2LIST and FEA2LIST areas @WOA*/
/***** STEP 1 *****/@WOA*/
/* @WOA*/
/* First calculate size of GEA2LIST area to be passed. @WOA*/
/* The GEA2LIST header : ULONG-Length of GEA2 list area @WOA*/
/* (pointed to by fpGEA2LIST in EAOP2 @WOA*/
/* The GEA2 list entry : ULONG-oNextEntryOffset, @WOA*/
/* : UCHAR-cbName (len of name) @WOA*/
/* : CHAR-szName[1] (char .DDM_XXX) @WOA*/
/* Note 1: GEA2 list entries must start on 4 byte boundaries @WOA*/
/* 2: The cbName does not count null string terminator @WOA*/
/* 3: Last entry is identified by oNextEntryOffset=0 @WOA*/

```

Figure 86 (Part 2 of 11). Example of C Program using Extended Attributes

```

/* Calculate the GEA2LIST area size @WOA*/

/* GEA2 list area begins with the GEA2LIST header @WOA*/
GEASize = sizeof(Eaop.fpGEA2List->cbList); /*@WOA*/
/* Each attribute to be queried needs a GEA2 list entry @WOA*/
/* Add on size for first GEA2 list entry - .DDM_DELCP @WOA*/
/* DELCP_NAME is defined as: ".DDM_DELCP" @WOA*/
GEASize = GEASize
+ sizeof(Eaop.fpGEA2List->list[0].oNextEntryOffset)
+ sizeof(Eaop.fpGEA2List->list[0].cbName)
+ strlen(DELCP_NAME)
+ 1; /* + null string terminator @WOA*/
/* GEAOffset entry must be on 4 byte boundary @WOA*/
Remainder = GEASize % 4; /*@WOA*/
if (Remainder != 0) /*@WOA*/
    GEASize=GEASize + (4-Remainder); /*@WOA*/
/* Now add on next GEA2 list entry - .DDM_FILCLS @WOA*/
/* FILCLS_NAME is defined as: ".DDM_FILCLS" @WOA*/
GEASize = GEASize
+ sizeof(Eaop.fpGEA2List->list[0].oNextEntryOffset)
+ sizeof(Eaop.fpGEA2List->list[0].cbName)
+ strlen(FILCLS_NAME)
+ 1; /* + name string terminator @WOA*/
/* This is last GEA2 list entry so the 4 byte boundary rule @WOA*/
/* does not apply. i.e. you don't need to pad this entry. @WOA*/

/* Now calculate size of FEA2LIST area to hold returned info. @WOA*/
/* The FEA2LIST header: ULONG-Length of FEA2 list area @WOA*/
/* (pointed to by fpFEA2LIST in EAOP2 @WOA*/
/* The FEA2 list entry: ULONG-oNextEntryOffset, @WOA*/
/* : UCHAR-fEA (flag @WOA*/
/* : UCHAR-cbName (len of name) @WOA*/
/* : USHORT-cbValue (len of value) @WOA*/
/* : CHAR-szName[1] (char .DDM_xxx) @WOA*/
/* : followed by DDMOBJECT encoded value @WOA*/
/* Note 1: FEA2 list entries start on 4 byte boundaries @WOA*/
/* 2: The cbName does not count null string terminator @WOA*/
/* 3: Last entry is identified by oNextEntryOffset=0 @WOA*/
/* 4: A cbValue of 0 means value field is null @WOA*/

/* Calculate the FEA2LIST area size @WOA*/
/* FEA2LIST area begins with the FEA2LIST header @WOA*/
FEASize = sizeof(Eaop.fpFEA2List->cbList); /*@WOA*/

/* Add on size for returned FEA2 list entry - .DDM_DELCP @WOA*/
/* DELCP_NAME is defined as: ".DDM_DELCP" @WOA*/
/* DELCP_SIZE is defined as: @WOA*/
/* sizeof(OBJLENGTH) + sizeof(CODEPOINT) + 1 @WOA*/
FEASize = FEASize
+ sizeof(Eaop.fpFEA2List->list[0].oNextEntryOffset)
+ sizeof(Eaop.fpFEA2List->list[0].fEA)
+ sizeof(Eaop.fpFEA2List->list[0].cbName)
+ sizeof(Eaop.fpFEA2List->list[0].cbValue)
+ strlen(DELCP_NAME)
+ 1 /* + null string terminator @WOA*/
+ DELCP_SIZE; /*@WOA*/

```

Figure 86 (Part 3 of 11). Example of C Program using Extended Attributes

```

/* FEAOffset entry must be on 4 byte boundary @W0A*/
Remainder = FEASize % 4; /*@W0A*/
if (Remainder != 0) /*@W0A*/
    FEASize=FEASize + (4-Remainder); /*@W0A*/
/* Add on size for returned FEA2 list entry - .DDM_FILCLS @W0A*/
/* FILCLS_NAME is defined as: ".DDM_FILCLS" @W0A*/
/* FILCLS_SIZE is defined as: @W0A*/
/* sizeof(OBJLENGTH) + (2 * sizeof(CODEPOINT)) @W0A*/
FEASize = FEASize
    + sizeof(Eaop.fpFEA2List->list[0].oNextEntryOffset)
    + sizeof(Eaop.fpFEA2List->list[0].fEA)
    + sizeof(Eaop.fpFEA2List->list[0].cbName)
    + sizeof(Eaop.fpFEA2List->list[0].cbValue)
    + strlen(FILCLS_NAME)
    + 1 /* + null string terminator @W0A*/
    + FILCLS_SIZE ;
/* Order of returned attributes is up to server so allow for @W0A*/
/* each returned entry to be on a 4 byte boundary. @W0A*/
Remainder = FEASize % 4; /*@W0A*/
if (Remainder != 0) /*@W0A*/
    FEASize=FEASize + (4-Remainder); /*@W0A*/

/* Note, we have calculated the minimum FEA2LIST size to hold @W0A*/
/* the returned information. We are permitted to pass a much @W0A*/
/* bigger buffer for the FEA2LIST if we wish so we could have @W0A*/
/* skipped doing a precise FEA2LIST size calculation. However, @W0A*/
/* if we pass too small a FEA2LIST area, we will get a @W0A*/
/* LENGTHR error reply message. @W0A*/
/****** STEP 2 *****@W0A*/
/* 2. Do the GEA2LIST + FEA2LIST malloc + set EAOP2 pointers @W0A*/
/****** STEP 2 *****@W0A*/
/* @W0A*/
/* The call to DDMQueryPathInfo will include a pointer to the @W0A*/
/* EAOP2 structure (locally defined as Eaop) and it has @W0A*/
/* pointers to the GEA2LIST and FEA2LIST areas. @W0A*/
/* @W0A*/
/* The EAOP2 struct: PGEA2LIST-fpGEA2List (ptr to GEA2LIST) @W0A*/
/* : PFEA2LIST-fpFEA2List (ptr to FEA2LIST) @W0A*/
/* : ULONG-oError @W0A*/

/* OK, now do the mallocs for GEA2LIST and FEA2LIST areas and @W0A*/
/* put the pointers in the EAOP2 structure. @W0A*/
if ((Eaop.fpFEA2List = (PFEA2LIST)malloc(FEASize)) == NULL)
{
    printf("Out of memory\n");
    Cleanup(SeqFN,DirFN,KeyFN,AltFN,KeyFN2);
    return(1);
}
if ((Eaop.fpGEA2List = (PGEA2LIST)malloc(GEASize)) == NULL)
{
    printf("Out of memory\n");
    Cleanup(SeqFN,DirFN,KeyFN,AltFN,KeyFN2);
    return(1);
}

Eaop.oError = 0L;

```

Figure 86 (Part 4 of 11). Example of C Program using Extended Attributes

```

/***** STEP 3 *****/
/* 3. Fill in the GEA2LIST area @WOA*/
/***** STEP 3 *****/
/* Initialize the GEA2LIST area @WOA*/
memset(&(Eaop.fpGEA2List->cbList), '\0', GEASize); /*@WOA*/

/* OK now start filling in the GEA2LIST area detail @WOA*/
/* Fill in the GEA2 header which has the area length @WOA*/
Eaop.fpGEA2List->cbList = GEASize;
/* The GEA2LIST struct: ULONG-cbList (len of GEA2 area ) @WOA*/
/* GEA2-list[1] (orient first entry) @WOA*/
/* The pGEA pointer will point to the specific GEA2 list @WOA*/
/* entry on which we are working. Use the GEA2LIST structure @WOA*/
/* definition to orient to the first list entry. @WOA*/
pGEA = (PGEA2)(&(Eaop.fpGEA2List->list[0]));

/* Fill out the first GEA2 list entry - DELCP_NAME @WOA*/
pGEA->cbName = (CHAR)(strlen(DELCP_NAME)); /*@WOA*/
strcpy(pGEA->szName, DELCP_NAME); /*@WOA*/

/* Calculate size for first GEA2 list entry - .DDM_DELCP @WOA*/
pGEA->oNextEntryOffset =
    sizeof(pGEA->oNextEntryOffset)
    + sizeof(pGEA->cbName)
    + pGEA->cbName +1; /*@WOA*/

/* The next GEA2 list entry must begin on 4 byte boundary @WOA*/
Remainder = pGEA->oNextEntryOffset % 4; /*@WOA*/
if (Remainder != 0) /*@WOA*/
    pGEA->oNextEntryOffset = pGEA->oNextEntryOffset + (4-Remainder); /*@WOA*/

/* Now move the GEA list entry pointer for the next list entry @WOA*/
pGEA = (PGEA2)((PBYTE)pGEA + pGEA->oNextEntryOffset); /*@WOA*/

/* Set up the next GEA2 list entry - .DDM_FILCLS @WOA*/
pGEA->cbName = (CHAR)(strlen(FILCLS_NAME)); /*@WOA*/
strcpy(pGEA->szName, FILCLS_NAME); /*@WOA*/

/* This GEA2 list entry is the last in this request. So @WOA*/
/* set the NextEntryOffset to 0 to indicate this is last @WOA*/
/* list entry. @WOA*/
pGEA->oNextEntryOffset = 0L;
/***** STEP 4 *****/
/* 4. Fill in the FEA2LIST area @WOA*/
/***** STEP 3 *****/
/* Initialize the FEA2LIST area @WOA*/
memset(&(Eaop.fpFEA2List->cbList), '\0', FEASize); /*@WOA*/

/* Fill in the FEA2LIST header which has the area length @WOA*/
Eaop.fpFEA2List->cbList = FEASize;

/* The remainder of the FEA2LIST area is untouched. It will @WOA*/
/* contain the returned EA FEA2 list entries from the @WOA*/
/* DDMQueryPathInfo call. @WOA*/

```

Figure 86 (Part 5 of 11). Example of C Program using Extended Attributes

```

/***** STEP 5 *****/
/* 5. Issue the DDMQueryPathInfo @WOA*/
/***** STEP 5 *****/
/*-----
-- Query a file to get .DDM_DELCPC and .DDM_FILCLS EA. @WOC
-- Then display the returned EAs. @WOC
-----*/
SevCode = DDMQueryPathInfo
        (SeqFN,          /* PathName */
         IUL,           /* PathInfoLevel */
         (PBYTE)&Eaop, /* PathInfoBuf */
         (ULONG)sizeof(EAOP2) /* PathInfoBufSize */
        );
if (SevCode == SC_NO_ERROR)
{ printf("\n\nSuccessful DDMQueryPathInfo call to file %s\n",SeqFN);
  /***** STEP 6 *****/
  /* 6. Extract DDM attribute data from the FEA2LIST area @WOA*/
  /***** STEP 6 *****/
  /* OK, we got a good return code, so it is time to look at @WOA*/
  /* the FEA2LIST area which now holds the returned info. @WOA*/

  /* Initialize the pFEA pointer to first FEA2 entry @WOA*/
  pFEA = (PFEA2)(&(Eaop.fpFEA2List->list[0]));

  /* The local i variable will govern the WHILE loop which @WOA*/
  /* follows. It counts the number of bytes remaining in @WOA*/
  /* the FEA2LIST. When the last FEA2 list entry is @WOA*/
  /* encountered, it is set to zero to stop the WHILE loop. @WOA*/
  /* If it goes negative, something went wrong while @WOA*/
  /* navigating around the FEA2 entries, stop the WHILE loop. @WOA*/

  i = (Eaop.fpFEA2List->cbList - sizeof(Eaop.fpFEA2List->cbList));
  while ( i > 0)
  { /* while more FEA2 list entries to process @WOA*/

    /* Temporarily set pAttValue to beginning szName which @WOA*/
    /* is the .DDMxxx attribute name. @WOA*/
    pAttValue = (PDDMOBJECT)((PBYTE)&pFEA->szName);

    /* Now move pAttValue past the .DDMxxx attribute to @WOA*/
    /* the value field by adding the number given in cbName@WOA*/
    /* plus one for the string terminating null character. @WOA*/
    pAttValue = (PDDMOBJECT)((PBYTE)pAttValue +
                           pFEA->cbName + 1); /*@WOA*/

    /* The value field is in PDDMOBJECT format. @WOA*/
    /* The PDDMOBJECT: OBJLENGTH-cbObject (len obj-4 byte) @WOA*/
    /* : CODEPOINT-cpObject (codept-2 byte) @WOA*/
    /* : BYTE-pData[1] (data value) @WOA*/

    /* We are expecting only 2 specific attributes back @WOA*/
    if (!memcmp(&(pFEA->szName[0]),FILCLS_NAME,
               sizeof(FILCLS_NAME))) /*@WOA*/
    { /* yes, this is the returned FILCLS attribute @WOA*/

```

Figure 86 (Part 6 of 11). Example of C Program using Extended Attributes

```

/* DUBCODPT.H (inc by DUB.H) defines SEQFIL, etc. @W0A*/
if (pFEA->cbValue == 0)
{ /* a cbValue of zero means value field is null @W0A*/
  printf("The .DDM_FILCLS attribute "
        "for %s is null. \n",      /*@W0C*/
        SeqFN);
} /* a cbValue of zero means value field is null @W0A*/
else
{ /* value returned @W0A*/
  switch (*(PCODEPOINT)pAttValue->pData) /*@W0C*/
  { case SEQFIL:
    printf("The .DDM_FILCLS attribute "
          "for %s is SEQFIL. \n",SeqFN); /*@W0C*/
    break;
    case DIRFIL:
    printf("The .DDM_FILCLS attribute "
          "for %s is DIRFIL. \n",SeqFN); /*@W0C*/
    break;
    case KEYFIL:
    printf("The .DDM_FILCLS attribute "
          "for %s is KEYFIL. \n",SeqFN); /*@W0C*/
    break;
    case ALTINDF:
    printf("The .DDM_FILCLS attribute "
          "for %s is ALTINDF. \n",      /*@W0C*/
          SeqFN);
    break;
    default:
    printf("The .DDM_FILCLS attribute "
          "for %s is invalid. \n",      /*@W0C*/
          SeqFN);
    break;
  }
  /* end switch @W0C*/
  /* value returned @W0A*/
} /* yes, this is the returned FILCLS attribute @W0A*/
else if (!memcmp(&(pFEA->szName[0]),DELCP_NAME,
                sizeof(DELCP_NAME)) ) /*@W0A*/
{ /* yes, this is the returned DELCP attribute @W0A*/
  if (pFEA->cbValue == 0)
  { /* a cbValue of zero means value field is null @W0A*/
    printf("The .DDM_DELCP attribute "
          "for %s is null. \n",      /*@W0C*/
          SeqFN);
  } /* a cbValue of zero means value field is null @W0A*/
  else
  { /* value returned @W0A*/
    if (*(PCODEPOINT)pAttValue->pData == 0xf1) /*@W0C*/
    printf("The .DDM_DELCP attribute for %s is TRUE. \n",
          SeqFN); /*@W0C*/
    else
    printf("The .DDM_DELCP attribute for %s is FALSE.\n",
          SeqFN); /*@W0C*/
  } /* value returned @W0A*/
} /* yes, this is the returned DELCP attribute @W0A*/
else /*@W0A*/
  printf("unexpected EA returned for %s \n",
        SeqFN);

```

Figure 86 (Part 7 of 11). Example of C Program using Extended Attributes

```

        /* Now move to the next entry in the FEA                                @WOA*/
        if (pFEA->oNextEntryOffset > 0)
        { /* the next entry is not the last entry                            @WOA*/
            i = i - pFEA->oNextEntryOffset; /*@WOA*/
            pFEA = (PFEA2)((PBYTE)pFEA + pFEA->oNextEntryOffset); /*@WOA*/
        } /* the next entry is not the last entry                            @WOA*/
        else
        { /* this was last entry-terminate WHILE                             @WOA*/
            i = 0; /*@WOA*/
        } /* this was last entry-terminate WHILE                             @WOA*/
    } /* while more FEA entries to process                                  @WOA*/
}
else
{ printf("Error in DDMQueryPathInfo call to file %s\n",SeqFN);
  printf("Severity code = %u\n",SevCode);
  ReplyMsg();
  Cleanup(SeqFN,DirFN,KeyFN,AltFN,KeyFN2);
  return(SevCode);
}
/***** STEP 7 *****/@WOA*/
/* 7. Free the GEA2LIST and FEA2LIST areas                                @WOA*/
/***** STEP 7 *****/@WOA*/
free(Eaop.fpFEA2List);
free(Eaop.fpGEA2List);
/*****@WOA*/
/* Prepare and execute a DDMSetPathInfo call to set a file's @WOA*/
/* Extended Attribute. @WOA*/
/*****@WOA*/
/* The DDM call to set a file's extended attributes is @WOA*/
/* based on the OS/2 extended attributes model. As such, the @WOA*/
/* calls to DDMSetPathInfo must pass a pointer to an EAOP2 @WOA*/
/* structure which, in turn, contains a pointer to the @WOA*/
/* FEA2LIST area which contains the attributes values. @WOA*/
/* @WOA*/
/* The EAOP2, FEA2LIST, GEA2LIST, FEA2 and GEA2 are defined @WOA*/
/* in DUBDEFS.H which is included by DUB.H. The format of the @WOA*/
/* values which can be returned are documented in the VSAM @WOA*/
/* API Reference manual in the "VSAM API Common Parameters" @WOA*/
/* chapter. @WOA*/
/* @WOA*/
/* Steps: @WOA*/
/* 1. Calculate the size of the required FEA2LIST area @WOA*/
/* 2. Do the FEA2LIST malloc and set the EAOP2 pointer @WOA*/
/* 3. Fill in the FEA2LIST area @WOA*/
/* 4. Issue the DDMSetPathInfo @WOA*/
/* 5. Free the FEA2LIST area @WOA*/
/*****@WOA*/
-- Set up for DDMSetPathInfo:
--
-- Build an extended attribute FEA2LIST area with one FEA2 @WOC
-- list entry to specify the TITLE extended attribute. @WOC
/******/

```

Figure 86 (Part 8 of 11). Example of C Program using Extended Attributes

```

/***** STEP 1 *****/@WOA*/
/* 1. Calculate the size of the required FEA2LIST area @WOA*/
/***** STEP 1 *****/@WOA*/
/* Now calculate size of FEA2 list area to hold returned info.@WOA*/
/* The FEA2LIST header: ULONG-Length of FEA2LIST area @WOA*/
/* (pointed to by fpFEA2LIST in EAOP2) @WOA*/
/* The FEA2 list entry: ULONG-oNextEntryOffset, @WOA*/
/* : UCHAR-fEA (flag) @WOA*/
/* : UCHAR-cbName (len of name) @WOA*/
/* : USHORT-cbValue (len of value) @WOA*/
/* : CHAR-szName[1] (char .DDM_XXX) @WOA*/
/* : followed by DDMOBJECT encoded value @WOA*/
/* Note 1: FEA2 list entries start on 4 byte boundaries @WOA*/
/* 2: The cbName does not count null string terminator @WOA*/
/* 3: Last entry is identified by oNextEntryOffset=0 @WOA*/
/* Calculate the FEA2LIST area size @WOA*/

/* FEA2LIST area begins with the FEA2LIST header @WOA*/
FEA2Size = sizeof(Eaop2.fpFEA2List->cbList); /*@WOA*/

/* Now add on an FEA2 list entry - .DDM_TITLE @WOA*/
/* TITLE_NAME is defined as: ".DDM_TITLE" @WOA*/
/* TitleString if defined as "Title String" @WOA*/
/* TITLE_SIZE is defined as: @WOA*/
/* sizeof(OBJLENGTH) + sizeof(CODEPOINT) @WOA*/
/* + strlen(TitleString); @WOA*/
FEA2Size = FEA2Size
+ sizeof(Eaop2.fpFEA2List->list[0].oNextEntryOffset)
+ sizeof(Eaop2.fpFEA2List->list[0].fEA)
+ sizeof(Eaop2.fpFEA2List->list[0].cbName)
+ sizeof(Eaop2.fpFEA2List->list[0].cbValue)
+ strlen(TITLE_NAME)
+ 1 /* + null string terminator @WOA*/
+ TITLE_SIZE; /*@WOC*/

/***** STEP 2 *****/@WOA*/
/* 2. Do the FEA2LIST malloc and set the EAOP2 pointer @WOA*/
/***** STEP 2 *****/@WOA*/
/* @WOA*/
/* The call to DDMSetPathInfo will include a pointer to the @WOA*/
/* EAOP2 structure (locally defined as Eaop2) and it has @WOA*/
/* pointers to the GEA2LIST and FEA2LIST areas. @WOA*/
/* @WOA*/
/* The EAOP2 struct: PGEA2LIST-fpGEA2List (ptr to GEA2LIST) @WOA*/
/* : PFEA2LIST-fpFEA2List (ptr to FEA2LIST) @WOA*/
/* : ULONG-oError @WOA*/
if ((Eaop2.fpFEA2List = (PFEA2LIST)malloc(FEA2Size)) == NULL)
{ printf("Out of memory\n");
Cleanup(SeqFN,DirFN,KeyFN,AltFN,KeyFN2);
return(1);
}
/* Since this is a DDMSetPathInfo call, there is no GEA2LIST @WOA*/
Eaop2.fpGEA2List = NULL; /*@WOM*/
Eaop2.oError = 0L; /*@WOA*/

```

Figure 86 (Part 9 of 11). Example of C Program using Extended Attributes

```

/***** STEP 3 *****/@WOA*/
/* 3. Fill in the FEA2LIST area @WOA*/
/***** STEP 3 *****/@WOA*/

/* Initialize the FEA2LIST area @WOA*/
memset(&(Eaop2.fpFEA2List->cbList),'\0',FEA2Size); /*@WOA*/

/* Fill in the FEA2LIST header which has list area length @WOA*/
Eaop2.fpFEA2List->cbList = FEA2Size;

/* The pFEA pointer will point to the specific FEA2 entry @WOA*/
/* on which we are working. Use the FEA2LIST structure @WOA*/
/* definition to orient to the first list entry. @WOA*/
pFEA = Eaop2.fpFEA2List->list;

/* Fill out the first and only FEA2 list entry - TITLE @WOA*/
pFEA->fEA = 0;
pFEA->cbName = (CHAR)(strlen(TITLE_NAME)); /*@WOC*/
pFEA->cbValue = TITLE_SIZE; /*@WOA*/
strcpy(pFEA->szName,TITLE_NAME); /*@WOC*/
/* Now set pAttValue ptr past the .DDMxxx attribute to @WOA*/
/* the value field by adding the number given in cbName @WOA*/
/* plus one for the string terminating null character. @WOA*/
pAttValue = (PDDMOBJECT)((PBYTE)&pFEA->szName +
pFEA->cbName + 1); /*@WOA*/

/* The value field is in PDDMOBJECT format. @WOA*/
/* The PDDMOBJECT: OBJLENGTH-cbObject (len obj-4 byte) @WOA*/
/* CODEPOINT-cpObject (codept-2 byte) @WOA*/
/* BYTE-pData[1] (data value) @WOA*/

pAttValue->cbObject = TITLE_SIZE; /*@WOA*/

/* DUBCODPT.H (included by DUB.H) TITLE codepoint=0x0045 @WOA*/
pAttValue->cpObject = TITLE;
strcpy(pAttValue->pData,TitleString);

/* This FEA2 entry is the last in this request. So set the @WOA*/
/* NextEntryOffset to 0 to indicate this is the last entry. @WOA*/
pFEA->oNextEntryOffset = 0L; /*@WOA*/

```

Figure 86 (Part 10 of 11). Example of C Program using Extended Attributes

```

/***** STEP 4 *****/
/* 4. Issue the DDMSetPathInfo @WOA*/
/***** STEP 4 *****/
SevCode = DDMSetPathInfo
    (SeqFN, /* PathName */
     1UL, /* PathInfoLevel */
     (PBYTE)&Eaop2, /* PathInfoBuf */
     (ULONG)sizeof(EAOP2) /* PathInfoBufSize */
    );
if (SevCode == SC_NO_ERROR)
    printf("\nSuccessful DDMSetPathInfo call to file %s\n",SeqFN);
else
{ printf("Error in DDMSetPathInfo call to file %s\n",SeqFN);
  printf("Severity code = %u\n",SevCode);
  ReplyMsg();
  Cleanup(SeqFN,DirFN,KeyFN,AltFN,KeyFN2);
  return(SevCode);
}
/***** STEP 5 *****/
/* 5. Free the FEA2LIST area @WOA*/
/***** STEP 5 *****/
free(Eaop2.fpFEA2List); /*@WOA*/
.
.
.

```

Figure 86 (Part 11 of 11). Example of C Program using Extended Attributes

Glossary

This glossary defines many of the terms and abbreviations used in this manual. If you do not find the term you are looking for, refer to the index or to the *Dictionary of Computing*, SC20-1699.

abend. Abnormal end of task.

access method. The part of the DDM architecture which accepts commands to access and process the records of a file.

ADL. A Data Language

ADSM. ADSTAR Distributed Storage Manager.

alternate index file. A file that has a different key path over a base file. The base file can be a keyed, direct, or sequential file.

API. Application Programming Interface

CCS. Common Communication Support.

CCSID. Coded character set identifier.

CDRA. Character Data Representation Architecture.

CM. Communications Manager

complete path name. The specifications for a file which includes the drive, directories, filename and file extension.

data conversion. A set of programs that convert data according to defined data descriptions. For example, characters can be converted from EBCDIC to ASCII, and numeric data can be converted from System /370 packed decimal to IEEE floating point or ASCII character (or vice versa).

data description. Specification of the layout of data. The data description of data stored in a file can be viewed as a file attribute.

data security. The protection of data against unauthorized disclosure, transfer, modifications or destruction, whether accidental or intentional.

data set. The major unit of data storage and retrieval. It consists of a collection of data in one of several pre-

scribed arrangements which is described by control information that the system has access to.

data stream. All data transmitted through a data channel in a single read or write operation.

DD&C. Data Description and Conversion; architecture extension to DDM.

DDM. Distributed Data Management; an SAA CCS architecture. A set of interfaces that gives users access to data files that reside on remote systems connected by a communication network. The DDM interfaces enable an application program to retrieve, add, update and delete data records in a file existing on a remote system. The DDM interfaces can be used to communicate between systems that have different architectures.

deadlock. Unresolved contention for the use of a resource. Each element in a process is waiting for an action by, or a response from, the other.

DFM. Distributed FileManager.

DFM client. Translates requests from the source system for access to file data on a remote system into a standard architected DDM request.

DFM server. A DFM component that accepts a remote request to access data and translates this request into a data management request on the target system.

direct file. A file that contains records that have a relationship between the contents of the record and the record position at which the record is stored.

distributed data management (DDM). Architecture for accessing distributed data located in files and distributed relational databases.

distributed file management (DFM). Strategy for a set of programming facilities that implement the file aspects of the DDM architecture on those systems which represent the SAA environments.

DRBA. Distributed relational data base access.

FSD. File System Driver.

HPFS. High Performance File System.

IFS. Installable File System.

independent LU. A logical unit (LU) that is not controlled by a System Network Architecture (SNA) host system.

intersystem communication. Communication between different systems by means of SNA facilities.

keyed file. A file organization that supports keyed forms of access to the records of the file.

LAN. Local area network.

Local area network. LAN

LDM. Local data management.

LDMI. Local data management interface.

local file. A file that resides on the same system as the application program that is accessing it.

LU. Logical unit.

protocol. A set of rules to be followed by communication systems.

RACF. Resource Access Control Facility. An external security management facility.

record. The basic unit of data stored in a file and transferred between DDM source and target servers.

record file. Record files consist of data fields organized into records that can be accessed as a set of bytes.

remote file. A file that resides on a system other than the system where the application program requesting access to the file resides.

Remote Record Access Support. DFM function that allows applications to access remote file data. function is to allow byte stream applications to access remote file data.

SAA. Systems Application Architecture

SAA data. Data on SAA systems that is subject to remote access and management using SAA DDM protocols.

SCM. Source communications manager. The DDM layer responsible for interfacing with the local communications facilities. It coordinates the sending and receiving of data on the source system.

sequential file. A file in which records are arranged in exactly the same sequence as they were stored into the file.

SNA. Systems Network Architecture.

source system. A system that requests access to data on another system. It is the "source" of the request.

Stream Agent. The DDM program responsible for transformation of data between the stream oriented API requests and the DDM byte requests.

stream file. Stream files contain strings of bytes that can be accessed according to their relative position within the file.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through and controlling the configuration and operation of networks.

target system. The system that contains data that is being accessed by another system.

target system data. Data considered to be owned and maintained according to the rules and functions prescribed by the data manager on the target system.

TP. Transaction Program

user exit. A point in an IBM-supplied program at which a user-exit routine may be given control.

Index

A

ACCATHRM (not authorized to use access method)
 reply message 415

access capability
 allow get record capability, create flag 407
 allow insert record capability, create flag 407
 allow modify record capability, create flag 407

access damage severity code 395

access intent
 list error, reply message 415
 list, parameter 361

access method 7, 18
 class, parameter 362
 CMBACCAM (combined) 20
 CMBKEYAM (combined key) 19
 CMBRNBAM (combined record number) 19
 invalid, reply message 416
 list, parameter 362
 not authorized to use, reply message 415
 promotions 20, 28
 RELKEYAM (relative by key) 19
 RELRNBAM (relative by record number) 18
 RNDKEYAM (random by key) 19
 RNDRNBAM (random by record number) 19

access method, definition 481

access order (records), copy flag

AccessFlags 399

accessing
 a file, date of 383

ACCINTLS (access intent list) parameter 361
 DELAI (delete record access intent) 361
 GETAI (get record access intent) 361
 INSAI (insert record access intent) 361
 MODAI (modify record access intent) 361

ACCINTRM (access intent list error) reply
 message 415

ACCMTHCL (access method class) parameter 362

ACCMTHLS (access method list) parameter 362

ACCMTHRM (invalid access method) reply
 message 416

address error, reply message 417

ADDRM (address error) reply message 417

AGNPRMRM (permanent agent error) reply
 message 418

ALCINISZ (allocate initial extent) parameter 363

allocate initial extent 363

allocating initial file size, parameter 375

allow
 cursor to be set to inactive record, access flag 401
 duplicate keys, create flag 409
 get record capability, create flag 407
 insert record capability, create flag 407
 modify record capability, create flag 407
 record deletion, create flag 408

alternate index file 16
 base file 364, 365

ALTINDLS (alternate index list) parameter 364

archiving
 a file, date of 384

assigning a new name to a file 386

attribute
 file hidden 375
 File system 378

B

base file 13, 16

base file name parameter 365

base management class name, parameter 365

base storage class name, parameter 366

BASFILNM (base file name) parameter 365

BASMGNM (base management class name) param-
 eter 365

BASNAMRM (invalid base file name) reply
 message 418

BASSTGNM (base storage class name) parameter 366

BOF (beginning of file), definition 15

bypassing
 all records, active and inactive, access flag 402
 damaged records, access flag 401
 damaged records, copy flag 405
 inactive records, copy flag 405

byte count parameter, file 373

C

calendar, Gregorian 367

changing
 a file, date of 373

class name
 base management 365

- class name (*continued*)
 - base storage 366
 - data 370
 - management 385
 - storage 393
- clock, date and time parameter 367
- closing a file 44
- CLSDMGRM (file closed with damage) reply message 419
- CMBACCAM (combined) access method 20
- CMBKEYAM (combined key) access method 19
- CMBRNBAM (combined record number) access method 19
- CMDCHKRM (command check) reply message 419
- Code Point
 - attribute parameter 366
- CODPNT (code point) parameter 366
- command check, reply message 419
- COMMRM (communications error) reply message 421
- composite key, definition 379
- concurrency protection 22
- concurrent opens exceeds maximum, reply message 453
- CopyFlags 404
- copying a file 46
- CreateFlags 405
- creating
 - a file, date of 374
 - alternate index file 50
 - record file 57
- CSRPOSST (cursor position status) parameter 366
- CSTNSARM (cursor not selecting a record position) reply message 424
- cursor
 - allow to be set to inactive record, access flag 401
 - general information 21
 - hold cursor indicator 22
 - hold position, access flag 401
 - moving EOF to current position 337
 - not selecting a record position, reply message 424
 - setting by key value 159
 - setting to BOF 141
 - setting to first record in key sequence 177
 - setting to first record of the file 148
 - setting to last record 233
 - setting to last record in key sequence 186
 - setting to minus the number of record positions in CsrDisp 243
 - setting to next record 269
 - setting to next record in key sequence 202

- cursor (*continued*)
 - setting to next record with equal key 253
 - setting to plus the number of record positions in CsrDisp 291
 - setting to previous record 301
 - setting to previous record in key sequence 220
 - setting to record number 314
 - status of, parameter 366
 - update error, reply message 465
 - update, access flag 400
- CVTNFNRM (conversation table not found) reply message 425

D

- damage
 - access damage severity code 395
 - permanent damage severity code 395
 - session damage severity code 395
- damaged
 - bypass damaged records, access flag 401
 - file, reply message 436
 - record, bypassing 455
 - record, reply message 455
- data
 - class name, parameter 370
 - lock status, parameter 371
 - stream syntax error, reply message 463
- data stream
 - description 1
 - parsing, terminating 396
- date
 - and time parameter 367
 - file access date, parameter 383
 - file access, parameter 383
 - file archived date, parameter 384
 - file change, parameter 373
 - file creation, parameter 374
 - Gregorian calendar 367
- DATE (date and time) parameter 367
- DDFNFNRM (data description file not found) reply message 425
- DDM
 - lock management 22
- DDM (Distributed Data Management)
 - data stream 1
 - records 4
- DDM definition 481
- DDM_ACCORD (access order) flag 405

DDM_ALDUPKEY (allow duplicate keys) flag 409
DDM_ALLREC (all records, active 485 inactive) flag 402
DDM_ALWINA (allow cursor set to inactive record) flag 401
DDM_BYPDMG (bypass damaged records) flag 401, 405
DDM_BYPINA (bypass inactive records) flag 405
DDM_DELC (allow record deletion) flag 408
DDM_FILHDD (hidden file) flag 407
DDM_FILPRT (protected file) flag 406
DDM_FILSYS (system file) flag 406
DDM_GETCP (allow get record capability) flag 407
DDM_HLDCSR (hold cursor position) flag 401
DDM_HLDUPD (hold update intent) flag 400
DDM_INHMODKY (inhibit modified keys) flag 400
DDM_INIEX (inhibit initial extent) flag 408
DDM_INSCP (allow insert record capability) flag 407
DDM_KEYVALFB (key value feedback) flag 402
DDM_MODCP (allow modify record capability) flag 407
DDM_NODATA (no record data returned) flag 402
DDM_RECNRFB (record number feedback) flag 403
DDM_RTININA (return inactive record) flag 402
DDM_TMPFIL (temporary file) flag 408
DDM_UPDCSR (update cursor) flag 400
DDM_UPDINT (update intent) flag 403
DDMClose 44
DDMCopyFile 46
DDMCreateAltIndex 50
DDMCreateRecFile 57
DDMDelete 64
DDMDeleteRec 66
DDMForceBuffer 70
DDMGetRec 72
DDMGetReplyMessage 81
DDMInsertRecEOF 83
DDMInsertRecKey 93
DDMInsertRecNum 98
DDMLoadFileFirst 106
DDMLoadFileNext 115
DDMModifyRec 122
DDMOpen 127
DDMQueryFileInfo 133
DDMQueryPathInfo 135
DDMRename 138
DDMSetBOF 141
DDMSetEOF 144
DDMSetFileInfo 146
DDMSetFirst 148
DDMSetKey 159
DDMSetKeyFirst 177
DDMSetKeyLast 186
DDMSetKeyLimits 195
DDMSetKeyNext 202
DDMSetKeyPrevious 220
DDMSetLast 233
DDMSetMinus 243
DDMSetNextKeyEqual 253
DDMSetNextRec 269
DDMSetPathInfo 288
DDMSetPlus 291
DDMSetPrevious 301
DDMSetRecNum 314
DDMSetUpdateKey 321
DDMSetUpdateNum 330
DDMTruncFile 337
DDMUnLoadFileFirst 339
DDMUnLoadFileNext 349
DDMUnLockRec 358
default
 record error, reply message 426
 record, parameter 369
defining a key field 382
DELA (delete record access intent) 361
DELC (record deletion capability) parameter 369
deleting
 a file 64
 a record 66
deletion (record) capability, parameter 369
DFTREC (default record) parameter 369
DFTRECRM (default record error) reply message 426
diagnostic information, server 392
direct file 10, 12
directory reply messages
 full 427
 not authorized to (access or update) 426
DRCATHRM (not authorized to directory) reply message 426
DRCFULRM (directory full) reply message 427
DTACLSNM (data class name) parameter 370
DTALCKST (data lock status) parameter 371
DTARECRM (invalid data record) reply message 427
DUPFILRM (duplicate file name) reply message 429
DUPKDIRM (duplicate key different index) reply message 429
DUPKSIRM (duplicate key same index) reply message 430
duplicate
 file name, reply message 429

duplicate (*continued*)

- key
 - allow, create flag 409
 - capability, parameter 381
 - different index, reply message 429
 - same index, reply message 430
 - record number, reply message 432
- DUPRNBRM (duplicate record number) reply message 432

E

- EAs (extended attributes) 5
- end of file
 - definition 15
 - record number, parameter 372
 - reply message 433
- ENDFILRM (end of file) reply message 433
- EOF (end of file), definition 15
- EOFNBR (end of file record number) parameter 372
- ERRFILNM (error file name) parameter 372
- error code
 - key definition, parameter 380
 - syntax 396
- error file name, parameter 372
- error severity code 394
- error, reply message 419
- existing condition, reply message 435
- EXSCNDRM (existing condition) reply message 435
- extended attributes (EAs) 5

F

- field length error, reply message 451
- FILATHRM (not authorized to file) reply message 435
- FILBYTCN (file byte count) parameter 373
- FILCHGDT (file change date) parameter 373
- FILCLS (file class) parameter 374
- FILCRTDT (file creation date) parameter 374
- FILDMGRM (file damaged) reply message 436
- file
 - access intent list parameter 361
 - access method class parameter 362
 - allocating storage 363
 - alternate index 16
 - base 13, 16
 - base file name, parameter 364, 365
 - byte count, parameter 373
 - causing an error 372
 - change date 373

file (*continued*)

- change date, parameter 373
- closed with damage, reply message 419
- closing 44
- concurrent opens exceeds maximum, reply message 453
- copying 46
- creating
 - an alternate index 50
 - record 57
- creation date 374
- creation date, parameter 374
- damaged, reply message 436
- deleting 64
- direct 10, 12
- duplicate keys capability 381
- file class, parameter 374
- file hidden attribute, parameter 375
- get capability, parameter 378
- handle
 - not found, reply message 443
- hidden 407
- in use, reply message 439
- index 14
- initial size, parameter 375
- insert capability, parameter 379
- invalid base file name, reply message 418
- invalid name, reply message 440
- is full, reply message 438
- keyed 13, 14
- last access date 383
- last access date, parameter 383
- last archived date 384
- last archived date, parameter 384
- length classes 7
- limits 433
- locked 439
- locking 23, 24
- modify capability, parameter 385
- name, parameter 376
- naming 386
- new name, parameter 386
- not authorized to file, reply message 435
- not found, reply message 440
- opening 127
- protected
 - flag 406
 - parameter 377
- quasi byte stream 9
- record-oriented, description 4

- file (*continued*)
 - renaming 138
 - retention class, parameter 392
 - sequential 8, 12
 - size, parameter 377
 - space not available, reply message 441
 - system 406
 - temporarily not available, reply message 442
 - unload records from 339, 349
- file information
 - getting 133
 - setting 146
- file name
 - duplicate, reply message 429
 - error 372
 - invalid new, reply message 452
 - invalid, reply message 440
 - parameter 376
 - validating 376
- file space not available, reply message 441
- files
 - maximum number opened, parameter 384
 - opening not authorized 444
 - permanent 408
- FILFULRM (file is full) reply message 438
- FILHDD (file hidden) parameter 375
- FILINISZ (initial file size) parameter 375
- FILIUSR (file in use) reply message 439
- FILNAM (file name) parameter 376
- FILNAMRM (invalid file name) reply message 440
- FILNFNR (file not found) reply message 440
- FILPRT (file protected) parameter 377
- FILSIZ (file size) parameter 377
- FILSNARM (file space not available) reply message 441
- FILSYS (system file) parameter 378
- FILTNR (file temporarily not available) reply message 442
- fixed-length records 17
- flag (invalid), reply message 444
- flags
 - access 399
 - all records, active and inactive 402
 - allow cursor to be set to inactive record 401
 - bypass damaged records 401
 - hold cursor position 401
 - hold update intent 400
 - inhibit modified keys 400
 - key value feedback 402
 - no record data returned 402
 - record number feedback 403

- flags (*continued*)
 - access (*continued*)
 - return inactive record 402
 - update cursor 400
 - update intent 403
 - copy 404
 - access order 405
 - bypass damaged records 405
 - bypass inactive records 405
 - create 405
 - allow duplicate keys 409
 - allow get record capability 407
 - allow insert record capability 407
 - allow modify record capability 407
 - allow record deletion 408
 - hidden file 407
 - inhibit initial extent 408
 - protected file 406
 - system file 406
 - temporary file 408
 - FUNATHRM (not authorized to function) reply message 442
 - function not supported, reply message 443
 - FUNNSPRM (function not supported) reply message 443

G

- get capability, file 378
- GETAI (get record access intent) 361
- GETCP (file get capability) parameter 378
- GETGETLK (get, reference only) lock 24
- GETMODLK (get, change) lock 24
- GETNONLK (get, no sharing) lock 24
- getting
 - a record 72
 - a reply message 81
 - file information 133
 - path information 135
- Gregorian calendar 367

H

- HDLNFNR (file handle not found) reply message 443
- hidden file
 - create flag 407
 - parameter 375
- hold
 - cursor indicator 22
 - cursor position, access flag 401

hold (*continued*)
update intent, access flag 400

I

inactive
inserting inactive records 428
record bypass, copy flag 405
record, parameter 389
record, reply message 457
information only severity code 394
inhibit initial extent, create flag 408
inhibit modified keys, access flag 400
initial file size, parameter 375
initially-variable-length records 17
INSAI (insert record access intent) 361
INSCP (file insert capability) parameter 379
insert capability, file 379
inserting
a record at EOF 83
a record by key value 93
a record by record number 98
INTATHRM (not authorized for open intent) reply message 444
invalid
base file name, reply message 418
data record, reply message 427
file name, reply message 440
flag, reply message 444
key definition, reply message 446
key length, reply message 447
key value, reply message 450
new file name, reply message 452
request, reply message 445
INVFLGRM (invalid flag) reply message 444
INVRQSRM (invalid request) reply message 445

K

key definition
error code, parameter 380
invalid, reply message 446
parameter 379
key field definition, parameter 382
key length
invalid, reply message 447
key update
not allowed by different index, reply message 448
not allowed by same index, reply message 449

key value
feedback, access flag 402
inhibit modified keys, access flag 400
inserting records by 93
invalid, reply message 450
parameter 383
setting limits 195
setting the cursor by 159
setting the update intent by 321
KEYDEF (key definition) parameter 379
KEYDEFCD (key definition error code) 380
KEYDEFRM (invalid key definition) reply message 446
KEYDUPCP (duplicate keys capability) parameter 381
keyed file 13, 14
example of fixed-length records 16
KEYFLDDF (key field definition) parameter 382
KEYLENRM (invalid key length) reply message 447
keys
capability of duplicates 381
duplicate 429, 430
KEYUDIRM (key update not allowed by different index)
reply message 448
KEYUSIRM (key update not allowed by same index)
reply message 449
KEYVAL (key value) parameter 383
KEYVALRM (invalid key value) reply message 450

L

LENGTHRM (field length error) reply message 451
loading records into a file 106, 115
locked file 439
locking
data lock status, parameter 371
files 23, 24
promotion rules 27
records 25
LSTACCDT (last access date) parameter 383
LSTARCDT (last archived date) parameter 384

M

management class
name parameter 385
naming 365, 366
MAXARNB (maximum active record number)
parameter 384
maximum
active record number, parameter 384
number of files opened, parameter 384

MAXOPN (maximum number of files opened)
 parameter 384
 message
 access intent list error 415
 address error 417
 command check 419
 communications error 421
 concurrent opens exceeds maximum 453
 conversational protocol error 454
 cursor not selecting a record position 424
 damaged file 436
 data description file not found 425
 data stream syntax error 463
 default record error 426
 directory full 427
 duplicate file name 429
 duplicate key different index 429
 duplicate key same index 430
 duplicate record number 432
 end of file 433
 error 419
 existing condition 435
 field length error 451
 file closed with damage 419
 file handle not found 443
 file in use 439
 file is full 438
 file not found 440
 file space not available 441
 file temporarily not available 442
 function not supported 443
 inactive record 457
 invalid access method 416
 invalid base file name 418
 invalid data record 427
 invalid file name 440
 invalid flag 444
 invalid key definition 446
 invalid key length 447
 invalid key value 450
 invalid new file name 452
 invalid request 445
 key update not allowed by different index 448
 key update not allowed by same index 449
 mismatched record length 458
 no update intent on record 466
 not authorized to (access or update) directory 426
 not authorized to file 435
 not authorized to function 442
 not authorized to open for intent 444

message (*continued*)
 not authorized to use access method 415
 object not supported 452
 parameter not supported 455
 parameter not supported error 464
 parameter value not supported 466
 permanent agent error 418
 record damaged 455
 record in use 457
 record not available 459
 record not found 461
 record number out of bounds 460
 resource limit reached in source system 463
 resource limits reached on target system 462
 severity code parameter 393
 translation error 467
 update cursor error 465
 MGMCLSNM (management class name)
 parameter 385
 MODAI (modify record access intent) 361
 MODCP (file modify capability) parameter 385
 MODGETLK (change, reference only) lock 24
 modified keys
 inhibited 400
 modifying
 a file, capability parameter 385
 a record 122
 MODMODLK (change, change) lock 24
 MODNONLK (change, no sharing) lock 24
 moving EOF to current cursor position 337

N

naming
 a data class 370
 a file 376, 386
 a management class 365
 new file name, parameter 386
 NEWFILNM (new file name) parameter 386
 NEWNAMRM (invalid new file name) reply
 message 452
 no record data returned, access flag 402
 not authorized
 to (access or update) directory, reply message 426
 to access method, reply message 415
 to file, reply message 435
 to function, reply message 442
 to open for intent, reply message 444
 to use access method, reply message 415

O

object not supported, reply message 452
OBJNSPRM (object not supported) reply message 452
opening a file 127
OPNMAXRM (concurrent opens exceeds maximum)
reply message 453

P

parameter not supported, reply message 455
parameter value not supported, reply message 466
path information
 getting 135
 setting 288
permanent
 agent error, reply message 418
 damage severity code 395
 file 408
PRCCNVRM (conversational protocol error) reply
message 454
PRMNSPRM (parameter not supported) reply
message 455
problem determination
 reply message diagnostic information 392
promotions
 access method 20, 28
 file and record locks 27
 record length class 390
protected file
 create flag 406
 parameter 377

Q

quasi byte stream file 9
querying
 file information 133
 path information 135

R

RECAL (record attribute list) 5
RECAL (record attribute list) parameter 386
RECCNT (record count) parameter 388
RECDMGRM (record damaged) reply message 455
RECFIX (fixed-length record) 4
RECINA (inactive record) parameter 389
RECINA (inactive records) 4

RECINARM (record inactive) reply message 457
RECIUSRM (record in use) reply message 457
RECIVL (initially-variable-length record) 4
RECLEN (record length) parameter 389
RECLENCL (record length class) parameter 390
RECLENRM (record length mismatch) reply
message 458
RECNAVRM (record not available) reply message 459
RECNR (record number) parameter 391
RECNRBRM (record number out of bounds) reply
message 460
RECNRFRM (record not found) reply message 461
record
 allow get record capability, create flag 407
 allow insert record capability, create flag 407
 allow record deletion, create flag 408
 attribute list (RECAL) 5
 attribute list, parameter 386
 bypass damaged records, access flag 401
 bypassing all, active and inactive, access flag 402
 bypassing damaged, copy flag 405
 bypassing inactive, copy flag 405
 count, parameter 388
 damaged, bypassing 455
 damaged, reply message 455
 default, parameter 369
 deleting 66
 deletion capability, parameter 369
 getting 72
 in use, reply message 457
 inactive, reply message 457
 inserting at EOF 83
 inserting by key value 93
 loading into a file 106, 115
 locking 25
 releasing 400
 modify capability, create flag 407
 modifying 122
 no data returned, access flag 402
 no update intent, reply message 466
 not available, reply message 459
 not found, reply message 461
 number feedback, access flag 403
 number out of bounds, reply message 460
 parameter 391
 return inactive, access flag 402
 unlock all implicit locks 358
 update intent, access flag 403
RECORD (record) parameter 391

- record key
 - defining a key field 382
 - definition 379
- record length
 - class
 - parameter 390
 - promotions 390
 - classes 8, 17
 - mismatch, reply message 458
 - parameter 389
- record number
 - duplicate, reply message 432
 - end of file, parameter 372
 - feedback, access flag 403
 - inserting a record by 98
 - maximum active record number, parameter 384
 - out of bounds, reply message 460
 - parameter 391
 - setting the update intent by 330
- records
 - active 391
 - basic description 4
 - inactive 389
 - inactive, description 4
 - inserting inactive 428
 - unload from file 339, 349
- RECVAR (variable-length record) 4
- releasing
 - record lock 400
 - update intent 400
- RELKEYAM (relative by key) access method 19
- RELRNBAM (relative by record number) access method 18
- renaming a file 138
- reply message
 - access intent list error 415
 - address error 417
 - command check 419
 - communications error 421
 - concurrent opens exceeds maximum 453
 - conversational protocol error 454
 - conversion table not found 425
 - cursor not selecting a record position 424
 - damaged file 436
 - data description file not found 425
 - data stream syntax error 463
 - default record error 426
 - directory full 427
 - duplicate file name 429
 - duplicate key different index 429
 - duplicate key same index 430
 - duplicate record number 432
 - end of file 433
 - error 419
 - existing condition 435
 - field length error 451
 - file closed with damage 419
 - file handle not found 443
 - file in use 439
 - file is full 438
 - file not found 440
 - file space not available 441
 - file temporarily not available 442
 - function not supported 443
 - getting 81
 - inactive record 457
 - invalid access method 416
 - invalid base file name 418
 - invalid data record 427
 - invalid file name 440
 - invalid flag 444
 - invalid key definition 446
 - invalid key length 447
 - invalid key value 450
 - invalid new file name 452
 - invalid request 445
 - key update not allowed by different index 448
 - key update not allowed by same index 449
 - mismatched record length 458
 - no update intent on record 466
 - not authorized to directory 426
 - not authorized to file 435
 - not authorized to function 442
 - not authorized to open for intent 444
 - not authorized to use access method 415
 - object not supported 452
 - parameter not supported 455
 - parameter not supported error 464
 - parameter value not supported 466
 - permanent agent error 418
 - record damaged 455
 - record in use 457
 - record not available 459
 - record not found 461
 - record number out of bounds 460
 - resource limit reached in source system 463
 - resource limits reached on target system 462
 - server diagnostic information 392
 - translation error 467

- reply message (*continued*)
 - update cursor error 465
- requesting storage size 375
- resource limit reached in source system, reply message 463
- resource limits reached on target system, reply message 462
- retention class, file 392
- return inactive record, access flag 402
- RNDKEYAM (random by key) access method 19
- RNDRNBAM (random by record number) access method 19
- RSCLMTRM (resource limits reached on target system) reply message 462
- RTNCLS (file retention class) parameter 392

S

- SCM 482
- sequential file 8, 12
- server diagnostic information, parameter 392
- session
 - damage severity code 395
- setting
 - file information 146
 - key value limits 195
 - path information 288
- setting the cursor
 - by key value 159
 - to BOF 141
 - to EOF 144
 - to first record in key sequence 177
 - to first record of the file 148
 - to last record 233
 - to last record in key sequence 186
 - to minus the number of record positions in CsrDisp 243
 - to next record 269
 - to next record in key sequence 202
 - to next record with equal key 253
 - to plus the number of record positions in CsrDisp 291
 - to previous record 301
 - to previous record in key sequence 220
 - to record number 314
- setting the update intent
 - by key value 321
 - by record number 330
- severe error severity code 395

- severity code, parameter 393
- severity codes
 - access damage 395
 - error 394
 - information only 394
 - permanent damage 395
 - session damage 395
 - severe error 395
 - warning 394
- size of file 377
- source system, description 1
- SRCLMTRM (resource limit reached in source system) reply message 463
- SRVDGN (server diagnostic information) parameter 392
- status
 - of cursor position, parameter 366
 - of data lock, parameter 371
- STGCLSNM (storage class name) parameter 393
- storage
 - class name, parameter 393
 - inhibit initial extent, create flag 408
 - size, requesting 375
- SVRCOD (severity code) parameter 393
- SYNERRCD (syntax error code) parameter 396
- syntax error code, parameter 396
- SYNTAXRM (data stream syntax error) reply message 463
- system file 406
 - File system attribute, parameter 378
- system file, create flag 406

T

- target system, description 1
- temporary file
 - create flag 408
- terminating data stream parsing 396
- time and date, parameter 367
- TITLE (title) parameter 397
- TP 482
- TRGNSPRM (Parameter not supported on target system) reply message 464
- truncating a file 337

U

- unload records from file 339, 349
- unlock all implicit record locks 358

- update cursor
 - access flag 400
 - error reply message 465
- update intent
 - none on record, reply message 466
 - on inactive record 401
 - releasing 400
 - setting by key value 321
 - setting by record number 330
- UPDCSRRM (update cursor error) reply message 465
- UPDINTRM (no update intent on record) reply message 466

V

- validating a file name 376
- VALNSPRM (parameter value not supported) reply message 466
- variable-length records 17
- VSAM
 - architecture 1
 - record files 6
 - technical considerations 29

W

- warning severity code 394

X

- XLATERM (translation error) reply message 467

We'd Like to Hear from You

SMARTdata UTILITIES
VSAM Application Programming
Interface Reference
Publication No. SC26-7133-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
 - IBMMail: USIB2VVG at IBMMAIL

Be sure to include the following with your comments:

- Title and publication number of this book
- Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

SMARTdata UTILITIES
VSAM Application Programming
Interface Reference
Publication No. SC26-7133-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				
Grammatically correct and consistent	<input type="checkbox"/>				
Graphically well designed	<input type="checkbox"/>				
Overall satisfaction	<input type="checkbox"/>				

May we contact you to discuss your comments? Yes No

Would you like to receive our response by E-Mail?

Your E-mail address

Name

Address

Company or Organization

Phone No.

Readers' Comments
SC26-7133-00



Cut or Fold
Along Line

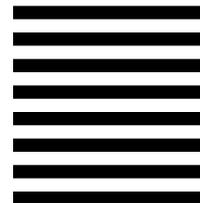
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department HHX/H3
PO Box 49023
San Jose, CA 95161-9945



Fold and Tape

Please do not staple

Fold and Tape

SC26-7133-00

Cut or Fold
Along Line



Program Number: 5765-548
5765-549
5622-793
5622-794
5639-B92



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-7133-00

