MDBS APPLICATION PROGRAMMING REFERENCE MANUAL

The MDBS DMS MANUAL

Version 3.08

Micro Data Base Systems, Inc.

P. O. Box 248

Lafayette, Indiana 47902

USA

Telex: 209147 ISE UR

(312) 303-6300 (in Illinois)

December 1985

Rev. 85A

3250332

NEW RELEASES, VERSIONS, AND A WARNING

Any programming endeavor of the magnitude of the MDBS software will necessarily continue to evolve over time. Realizing this, Micro Data Base Systems, Inc., vows to provide its users with updates to **this version** for a nominal handling fee.

New versions of MDBS software will be considered as separate products. However, bona fide owners of previous versions are generally entitled to a preferential rate structure.

Finally, each copy of our software is personalized to identify the licensee. There are several levels of this personalization, some of which involve encryption methods guaranteed to be combinatorially difficult to decipher. Our products have been produced with a very substantial investment of capital and labor, to say nothing of the years of prior involvement in the data base management area by our principals. Accordingly, we are seriously concerned about any unauthorized copying of our products and will take any and all available legal action against illegal copying or distribution of our products.

PREFACE

By the mid-1960s, application developers were well aware of the data handling limitations of programming languages and file management systems. To overcome these limitations, data base management systems began to appear. By 1971, four major approaches to data base management had taken shape: the hierarchical, shallow-network, relational, and CODASYL-network approaches to logical data structuring and manipulation. Each represented an advance over the old file-oriented data handling methods and the latter two approaches offered advances over the former two. By the mid-1970s, data base management software was well established as the cornerstone for application development on mainframes and some mini computers.

Near the end of the decade, microcomputers -- with their phenomenal computing power on a per dollar basis -- began to proliferate. The acceptance of mainframe data base management systems coupled with the rise of microcomputers led to the formation of Micro Data Base Systems Incorporated by a group with expertise in both areas. The initial objective was to make genuine data base management tools available in the micro realm. This objective was fulfilled in 1979 with the release of MDBS I -- the first authentic and viable data base management system (dbms) for microcomputers. Over the years, this has evolved into the present MDBS III which operates not only on many microcomputers but on larger machines as well. The evolution of MDBS III is highlighted with many firsts: first micro dbms with built-in logging and recovery, first full implementation of a postrelational dbms, first multiuser micro dbms, first dbms to run under PCDOS, MSDOS and UNIX.

Today, MDBS III offers professional application developers a degree of power and flexibility unavailable with any other data base management software -- be it on micros, minis or mainframes. This is partially due to the highly efficient, proprietary implementation techniques. MDBS III is not a mainframe retread shoehorned into a microcomputer. It is also due to the innovative data modeling features that MDBS III provides. Because these features go far beyond those of the older data base management approaches, MDBS III is variously referred to as postrelational or multiarchical or extended-network. The emphasis in this approach to data base management is on direct, natural representation of the application world. The result is a tremendous increase in developer productivity. As stated in the authoritative Database - the 2nd Generation: State of the Art Report (ed., D. J. L. Gradwell, Pergamon Press, Oxford, England, 1982):

> **"The data modelling capability of MDBS III is superior to any other commercially available DBMS." MDBS III is "... a product that is, in many ways, ahead of mainframe DBMSs." (D. J. L. Gradwell)**

All of this translates into convenience for application system developers and administrators.

The MDBS R&D Lab's expertise in the areas of decision support systems and artificial intelligence has resulted in two unique environments for processing MDBS III data bases. One is a decision support environment called KnowledgeMan. It functions as a universal knowledge management system, allowing users to represent and process knowledge in many different ways -- including spreadsheets, text, graphics, forms, procedural models, relational data bases, and postrelational MDBS III data bases. The second is a revolutionary artificial intelligence environment called Guru. It makes the benefits of both expert system technology and natural language processing easily accessible to business users, without sacrificing familiar business computing capabilities. MDBS III data base contents are directly accessible within the Guru environment.

This manual provides details on the features and utilization of the MDBS Data Manipulation Language. Companion manuals discuss the MDBS Data Description Language and various optional modules. This manual is not intended to be a tutorial. For a tutorial treatment of data base management (including its many advantages over file management), the reader is advised to consult such suitable references as:

1. Micro Database Management - Practical Techniques for Application Development by R. H. Bonczek, et. al, 536 pages, Academic Press, New York, 1984.

2. "A Perspective on Data Models," PC Tech Journal, Vol. 2, No. 1, 1984.

3. "Micros Get Mainframe Data Scheme," Systems and Software, Vol. 3, No. 5, 1984.

4. "Uniting Relational and Postrelational Database Management Tools," Systems and Software, Vol. 3, No. 11, 1984.

The first reference provides the most definitive coverage to date of the postrelational approach, comparing and contrasting it with the four older data base management approaches. That book also includes extensive examples of MDBS III usage. It is available through your local bookstore, from the publisher, or from MDBS, Inc. For many years, the Company has offered practical MDBS III training seminars in major cities and at customer sites.

The MDBS, Inc. commitment to customer success does not stop with software innovation, quality and support. The Company offers a full range of consulting and application development services to clients through its regional offices in the Dallas its regional Chicago and New York areas. Services include:

1. Customized application systems design

2. Customized application systems development using Guru, KnowledgeMan and/or the MDBS III tools

3. Communications interfaces including mainframe-micro links

4. Special training on a wide variety of topics including specific application systems, C programming, MDBS III usage, and Guru usage

5. General consulting including feasibility studies and hardware/software recommendations

The experienced, professional staff handles consulting and application development needs of some of the world's largest corporations and governments. It has developed very extensive micro application systems in such diverse areas as cash management, strategic planning, human resources administration, waste disposal management, and distributed service support. In many cases, these application systems have involved multiuser processing or mainframe-micro links.

Table of Contents

COMMAND CLASSIFICATIONS

Page

DELETION COMMANDS

DISCONNECT COMMANDS

FIND COMMANDS

MODIFY COMMANDS

Page

MULTIUSER LOCKING COMMANDS

RECOVERY COMMANDS

RETRIEVAL COMMANDS

Page

SPECIAL COMMANDS

UTILITY COMMANDS

I. OVERVIEW

A. Organization

This is the **MDBS DMS Manual**. It is the second component in the set of **MDBS Reference Manuals** and assumes the reader is familiar with data description facilities presented in the **MDBS DDL Manual**. This document describes the MDBS III Data Manipulation Language (DML) and the effect that each DML command has on the behavior of the data base control system (MDBS.DMS). The **MDBS DMS Manual** is written on two levels: fundamental manipulation features and advanced manipulation features. An understanding of the fundamentals is sufficient for developing useful application systems. The advanced features are denoted by a vertical bar in the outside margin. These are perhaps less frequently needed, but they are valuable for certain special kinds of data manipulation.

The documentation in this manual is applicable to both Versions 3a and 3c. Appendix A describes those data manipulation features that are applicable only to Version 3a. It is also independent of the host language(s) selected to interface with MDBS.DMS. Host language-dependent aspects of the interface are described in appropriate **MDBS System Specific Manuals**.

The remainder of this chapter outlines the role of MDBS.DMS software and its associated Data Manipulation Language (DML), in application development. Chapter II provides the detailed background that is necessary to effectively use the MDBS Data Manipulation Language. Each of Chapters III through XV documents a class of DML commands. The individual commands in each class are presented alphabetically and the function of each command is described. When a DML command is executed, MDBS.DMS reports on the status of that command's execution. Each possible command status is explained in Chapter XVI.

B. The Role of DML in Application Development

The MDBS Data Manipulation Language consists of a group of commands, each of which performs some data manipulation task. A DML command is stated within an application program. The programming language used to write an application program is called the host language. A DML command is typically invoked as a function in the host language. When a DML command is given, the MDBS.DMS software carries out the actual data manipulation. This could involve storing data from a host program variable into the data base. Conversely, it could involve extracting data from the data base and depositing it in a host language variable. Many other kinds of data manipulation are supported. In effect, the DML extends the data handling capability of a programming language, so that programs can utilize MDBS data bases in addition to traditional files.

Usage of the MDBS DML in developing an application program depends only upon a knowledge of the data base's logical structure. The application programmer need not be concerned with using pointers, searching indices, disk I/O, file handling, free space management,

etc. All of these factors are automatically handled by the MDBS.DMS software. Furthermore, all data manipulation occurs subject to the security and integrity constraints defined in a data base's DDL specification. In a multiuser environment, MDBS.DMS manages record lockout to prevent such problems as one user attempting to modify a record that another user is reading. Special DML commands are available to the application programmer who (beyond the standard passive lockouts) desires to actively lock out records, record types, or sets.

A data manipulation language provides a procedural approach to data manipulation. It can therefore be contrasted with a query language, which provides a nonprocedural approach to data manipulation. From the standpoint of achieving the highest performance for application software, using a DML is inevitably superior to using a nonprocedural query language. On the other hand, the application developer who is not experienced in using DML can very likely achieve more rapid development by using a nonprocedural query language (at the expense of a sacrifice in per]Mrmance). In data base management systems that do not support both a DML and a query language, the application developer has no choice. Both kinds of languages are available with MDBS. It is recommended that the query language (see the **MDBS QRS Manual**) be used in situations where the need for rapid development outweighs the need for optimal performance. If optimal performance is the overriding consideration, then the MDBS DML is most appropriate. In addition to QRS, there are other optional access modules including IDML, IBS, RDL and BLF.

Although the MDBS DML makes many commands available, it is typically the case that an application developer will need to use only a few kinds of DML commands in any given application program. Across all application programs, there is a small group of about a dozen DML commands that tend to be heavily used. Varying degrees of selective MDBS.DMS linking are permitted, depending on the host language used (see MDBS System Specific Manuals for details). In other words, only that part of MDBS.DMS that is needed to execute an application program's DML commands is held in main memory. Since the entire MDBS.DMS software is not in main memory, there is more memory available for a larger application program or a larger page buffer region.

One component of MDBS.DMS performs virtual buffer paging, based on an enhanced least-recently-used page replacement algorithm. When a DML command is executed, MDBS.DMS determines whether images of all needed pages are in main memory. If needed pages are not resident, MDBS.DMS has them read into main memory. The larger the page buffer region in main memory, the more likely it is that needed pages are resident and the faster the processing will be. Thus it is important for the application developer to allow as large a page buffer region as possible. In a single-user environment, the page buffer region size is directly assigned by the programmer. In a multiuser environment, MDBS.DMS determines the size of the page buffer region that is shared by all users (i.e., simultaneously executing application programs).

II. GENERAL BACKGROUND

A. Data Transfer

Some DML commands pass data from a host language variable into the data base. Others retrieve data from the data base into a host language variable, from which the data can then be used for computation or program output. There are two basic approaches for allowing MDBS.DMS to accomplish the data transferral between a data base and an application program. The approach used depends on the nature of the host programming language. Non-record-oriented languages (such as BASIC) require the data block approach. Host languages with a facility for defining program record types (e.g., COBOL, PL/1, PASCAL) use the program record type approach. Full details of the approach used for a particular host language appear in the MDBS System Specific Manual for that host language.

1. Data Blocks. A data block is a named sequence of one or more host language variables. MDBS DML commands exist to define, extend, and undefine data blocks (see Chapter XIII). A given host language variable can participate in many data blocks. A DML command to put data into a data base has a data block name as one of its arguments. The value(s) of the host language variable(s) for that data block is(are) put into the data base. A DML command to retrieve data from a data base has a data block name as one of its arguments. The retrieved value(s) becomes the new value(s) for the data block's host language variable(s).

2. Program Record Types. A program record type plays the same role as a data block. It is a named sequence of one or more host language variables. It is not defined with DML commands, but is specified with the host language's facility for defining program record types (for example, a 01 level entry in a COBOL working storage section, or a record structure of C or PASCAL). A given host language variable usually cannot participate in more than one program record type. A DML command to put data into a data base has a program record type name as one of its arguments. The value(s) of the host language variable(s) for that program record type is(are) put into the data base. A DML command to retrieve data from a data base has a program record type name as one of its arguments. The retrieved value(s) becomes the new value(s) of the program record type's host language variable(s).

B. Currency Indicators

Currency indicators are used by an application programmer to keep track of which record occurrences in a data base are currently of interest. MDBS.DMS maintains two currency indicators for each set specified in a data base's schema: the current owner of the set and the current member of the set. There can be many owner record

occurrences in the data base for a given set. At any moment during the execution of an application program no more than one of those owner record occurrences is the current owner of the set. DML commands allow the application programmer to control which (if any) owner record occurrence is the current owner of a set. The SYSTEM record occurrence is always the current owner of every system-owned set (unless it is explicitly made to be null by the programmer). Similarly, there can be many member record occurrences in the data base for a given set. At any moment during the execution of an application program, no more than one of those member record occurrences is the set's current member. DML commands allow the application programmer to control which (if any) member record occurrence is the current member of a set.

As soon as a record occurrence becomes the current owner of a set, the DML can be used:

to find any of its related members through that set (Chapter III),
to modify its data values (Chapter V),
to retrieve its data values (Chapter IV),
to delete the record occurrence from the data base (Chapter X),
etc.

As soon as a record becomes the current member of a set, the DML can be used:

to find any of its related owners through that set (Chapter III),
to delete the record from the data base (Chapter IX),
to modify its data values (Chapter V),
to retrieve its data values (Chapter IV),
etc.

Another kind of currency indicator that can be used during data manipulation is called the current of the run unit. A run unit is an executing instance of an application program. There are many record occurrences in a data base. At any moment during the execution of an application program, no more than one record occurrence in the entire data base is the current record of that run unit. The current of a run unit is typically the record that was most recently found by the application program (i.e., by the run unit). DML commands allow the application programmer to control which (if any) data base record is the current of run unit. As soon as a record becomes the current of run unit, the DML can be used:

to delete the record from the data base (Chapter IX),
to modify its data values (Chapter V),
to retrieve its data values (Chapter IV).

Suppose a data base schema has five sets. These can be any mixture of 1:1, 1:N, N:1, N:M sets. An application program can make use of any of eleven currency indicators. There are five current owner indicators (one per set), five current member indicators (one per set), and the current of run unit. In a multiuser environment, each application program that is using this data base will have its own eleven currency indicators.

In addition to the three kinds of currency indicators already mentioned, MDBS allows an application programmer to optionally define additional currency indicators within the scope of an application program. No more than 255 user-defined currency indicators can be used within an application program. These additional currency indicators can be usefully employed to "remember" the current owner of some set, before processing causes that current owner to change. The same principle can be applied to current members and the current of run unit. Various DML commands can be used in subsequent processing to access the remembered record. DML commands are also provided for the user (i.e., application programmer) to define additional currency indicators and to make the record that is a set's current owner (or member or current of run unit) the current record for a user-defined currency indicator.

All currency indicators are null when an application program begins execution, except the current owner of each system-owned set and the current of run unit, which also has the SYSTEM record as its current record. A currency indicator remains null until a DML command is executed to change it.

C. Classes of DML Commands

The DML commands fall into thirteen classes. Each command class has one chapter devoted to it. The order of presentation for these classes is as follows:

Finding records (Chapter III)
Retrieving data from records (Chapter IV)
Modifying data in records (Chapter V)
Assigning currency indicators (Chapter VI)
Creating records (Chapter VII)
Connecting records (Chapter VIII)
Disconnecting records (Chapter IX)
Deleting records (Chapter X)
Utilities (Chapter XI)
Boolean operations (Chapter XII)
Special operators (Chapter XIII)
Multiuser locking (Chapter XIV)
Recovery operators (Chapter XV)

The DML commands within a given class are presented alphabetically in the chapter for that class. In becoming acquainted with this manual, it is recommended that these chapters be examined in sequence. Also, the reader may want to ignore the advanced manipulation features on the initial pass through this manual.

D. Stating a DML Command

Each DML command consists of a short mnemonic. For instance, the mnemonic FFM is used for finding the first member record connected to the current owner of a set. A list of zero, one, two, or three arguments is stated along with a DML command. An argument is either a

set name, record type name, area name, data item name, or data block/program record name. For example, FFM needs a single argument: a set name, to indicate for which set we want to find the first member. Correct usage of a DML command depends on a knowledge of the kinds of arguments required by that command. If one of the arguments is a data block (or program record), the application programmer must be aware of whether the command will use the data block's variables as inputs or outputs.

A DML command can use currency indicators and can change currency indicators. An application programmer must know which currency indicators a DML command can use and which currency indicators it can alter (and the nature of the alterations).

Command Status

The execution of a DML command always results in a command status. A command status is an integer in the range from 0 through 255. The command status number is returned as the value of a host language variable of the programmer's choosing. A command status of 0 means that the execution of a command was successfully completed. A command status of 255 means that the data manipulation system tried to find a record occurrence indicated by the command, but that the indicated record occurrence does not exist in the data base. Any other command status means that the command did not execute normally. Generally speaking, this is caused by either faulty logic of the programmer, a mis-statement of the command, an attempt to violate security or integrity constraints specified in the DDL, failure to have a needed disk on-line, or a hardware malfunction. A detailed description of each such command status error appears in Chapter XVI.

Command Form

The precise syntax employed for stating a DML command depends on the host language being used. The MDBS System Specific Manual for a given host language shows the syntax for using DML within that host language. There are four major categories of host language interfaces:

a) Data block oriented with direct DML invocation (e.g., some FORTRANs, some BASICs)
b) Data block oriented with indirect DML invocation (e.g., some BASICs)
c) Record oriented with direct DML invocation (e.g., C, some PASCALs, PL/1)
d) Record oriented with indirect DML invocation (e.g., some PASCALs, some COBOLs)

A host language is data block oriented if it does not allow program record types; otherwise, it is said to be record oriented. Invoking a DML command directly means that the host language allows a DML mnemonic to be treated as a host language function. This is not

allowed in the indirect case, where a DML mnemonic and its associated arguments are parameters of a function. For instance, in the direct case the FFM command has the following generic form:

```
E0 = FFM ("arguments")
```

where E0 is a host language variable that receives the command status value returned by the FFM function. In the indirect case, the FFM command has the following generic form:

```
E0 = DMS ("FFM,arguments")
```

where E0 is a host language variable that receives the command status returned by the DMS function as a result of executing the FFM command.

Command String

When a DML command is stated, the portion of that command within quotes is called the command string.* Two elements in a command string are separated by

a series of one or more blanks,
a comma, or
a comma embedded in a series of blanks.

Two consecutive commas, without an intervening nonblank character, indicate a missing element in a command string (e.g., "X,,Y"). A command string with one fewer than the permitted number of elements is typically treated as if the rightmost element is missing (e.g., if three elements are permitted, then the rightmost element is regarded as missing from "X,Y"). The exception is when the omitted element should have been a data block name (or program record type name) immediately following a data item or record type name (e.g., with the CRS command). In this case, the data block name is assumed to be the same as the record type or data item name (e.g., if Z is a data item name, then "X,Z" is equivalent to "X,Z,Z").

All elements are missing from a command string if there are no elements in the quotes. For some DML commands, it is permissible for an argument in a command string or the entire command string to be missing (e.g., FFS). Some DML commands have no command string (e.g., DBCLS).

Command Descriptions

As each DML command is described in the following chapters, examples of the four different categories of command usage are provided. Even within a category, the exact syntax for using a

---

*Some host languages do not use double quotes (""). The exact convention employed by a host language is described in its system specific manual.

command may differ slightly from one host language to another. To repeat, the appropriate system specific manual should be consulted for the exact syntax required by a given host language. The mnemonic, arguments, currency indicators used, currency indicators changed, and impact on the data base for each DML command are all independent of the host language used. All of these factors are described in subsequent chapters of the MDBS DMS Manual. Thus the logic employed in using the data manipulation language is the same, regardless of the host languages used.

In presenting the DML commands in subsequent Chapters, the following conventions are used:

EQ host language variable for the command status
itm the name of a data item defined with the DDL
rec the name of a record type defined with the DDL
set-1, set-2, set-3 the names of sets defined with the DDL
area the name of an area defined with the DDL
blk the name of a data block (or program record type)
iblk the name of a data block (or program record type) whose host language variables provide input to a DML command
oblk the name of a data block (or program record type) whose host language variables receive output from a DML command
CRU current of run unit indicator
CO(set) current owner of the indicated set
CM(set) current member of the indicated set
CU(i) current user-defined indicator i ($1 \leq i \leq 255$)
for example, CU(7) denotes the current record for the seventh user-defined indicator

Although the description for a DML command may refer to a data block, it should be understood that a program record type takes the place of a data block for record-oriented host languages.

E. Multiuser Environment

In a multiuser environment, many run units can simultaneously share the same copy of serially reentrant MDBS.DMS code. They also share a common page buffer area. Each run unit has its own program region, which contains the application program and program buffers. A run unit's program buffers are used by MDBS.DMS to manage the specific operations pertinent to that run unit. Each run unit has its own group of currency indicators, which are maintained independently of other run units' currency indicators.

A run unit can passively and actively lock various record occurrences to prevent them from being altered and/or read by other run units. The rules governing active and passive locking are presented in Chapter XIV. Also presented in that chapter are the DML commands for active locking and unlocking and for controlling the number of times (and the time interval length) that a DML command will re-try to access a previously locked record.

Built-in to the multiuser MDBS.DMS is a mechanism that allows an application developer to prevent both binary and n-ary deadlock situations. If a run unit attempts an access that will create a deadlock situation, a command status error to that effect is returned for the DML command that attempted the access. The run unit can use this information in its control structure to avoid a deadlock situation.

F. Protecting Data Base Consistency against External Factors

When a run unit opens a data base for processing, it can specify read-only processing. If this is the case, then page images in the page buffer region are always identical to their corresponding pages in the data base (i.e., in auxiliary memory). Thus data base consistency is not damaged by an abnormal interruption (e.g., power failure) of the run unit, which does not allow the run unit to close the data base.

If a run unit does not open the data base for read-only processing, then the run unit can alter the data base. Since these alterations are necessarily buffered through the page buffer region, it can happen that a page image in this region is not identical to the corresponding data base page. An abnormal interruption of the run unit could leave the data base inconsistent since the latest changes to page images in main memory might not have been written to the data base. When this occurs, subsequent efforts to open the data base yield a command status error of 15. A backup copy of the data base must be used in this event or the recovery utility furnished with the RTL form of MDBS can be used to restore the data base. There are two ways to drastically reduce the likelihood of such a situation.

One method makes use of a DML command (DBSAVE) that rewrites all changed page images into the data base. This memory flushing command can be used wherever and whenever desired in an application program. If, when a run unit abnormally terminates, no changes have been made to the page images in memory since the last flush, then the data base is consistent and error 15 will not occur in a subsequent attempt to open the data base for processing.

The second method is available with the RTL form of MDBS in single user situations. It allows the run unit to specify a page-image file via the PIFD command. This method of preserving consistency is particularly useful if the run unit carries out a complex kind of transaction that can change records on several pages. At some moment during the execution of a complex transaction, some changed page image may have been rewritten into the data base, while other altered images may still reside in the buffer region. An abnormal termination at this point would leave the data base inconsistent. If the data base could be restored to its former state, that existed at the start of the complex transaction, then that data base would be consistent and the interrupted transaction could be reinitiated. This is the purpose of the page image file.

At any juncture in a run unit, the TRBGN command can be used to declare the beginning of a complex transaction. When a commit (TRCOM) command is invoked, the entire complex transaction is committed to the data base. All changes made since TRBGN was last invoked are incorporated into the data base. If an abort (TRABT) command is invoked instead of TRCOM, no changes that were made since TRBGN was last invoked are incorporated into the data base. If the run unit abnormally terminates after a TRBGN and before a TRABT or TRCOM, then a subsequent command to open the data base finds the data base in the state that existed at the beginning of the complex transaction. With operating systems that do not dynamically allocate file sizes, it is important to create a page image file of sufficient size. Because page image posting is not meaningful in multiuser situations, it is available only for the single user version of MDBS. The PIFD, TRBGN, TRCOM and TRABT commands are described in Chapter XVI and in the **MDBS RTL Manual.**

Selective Rollback. The foregoing mechanisms do not help in the event of a hardware failure that physically destroys a portion of the data base. In this case the RTL transaction logging and recovery facilities are used to restore the data base. The DML commands for utilizing automatic transaction logging within a run unit are presented in Chapter XVI. The transaction log file (defined in the DDL specification) is distinct from the page-image file. Either, both, or neither can be used by a run unit. In the event of any type of inconsistency in the data base (due to abnormal termination, hardware failure, incorrect data entry, etc.) the transaction log file can automatically be applied to a data base back-up copy by using the RTL recovery utility (RCV). This recovery can be selective if desired, so that only certain logged transactions are used to recover. Bad transactions (e.g., incorrect data entry) can be discarded during the selective rollback.

Full details about the MDBS recovery facilities available with the RTL form of MDBS, including its use for surveillance of users' activities, are presented in the **MDBS RTL Manual.**

MDBS DMS MANUAL

III. FIND COMMANDS

A. Overview

The MDBS Data Manipulation Language supports twenty-four find commands, providing an application programmer with many ways to find a given record. The principal objective of any find command is to make a record the current owner of a set, the current member of a set, the current of the run unit, or some combination of these. The found record becomes one of the records that the application program is currently interested in; as soon as a record becomes current, it can be processed in various ways by other DML commands. The find commands fall into four main groups:

1. Find a member record that is related to a given owner record via some set:

   find the first member,
   next member,
   last member,
   prior member,
   member with a particular sort key value,
   member with a particular value for some data item,
   member with the same value for a data item or sort key as an already found member.

   Which member is first, next, last, or prior depends on the set's member order specified in the DDL. If the member order is sorted, members can quickly be found on the basis of their sort key values.

2. Find an owner record that is related to a given member record via some set:

   find the first owner,
   next owner,
   last owner,
   prior owner,
   owner with a particular sort key value,
   owner with a particular value for some data item,
   owner with the same value for a data item or sort key as an already found owner.

   Which owner is first, next, last, or prior depends on the set's owner order specified in the DDL. If the owner order is sorted, owners can quickly be found on the basis of their sort key values.

3. If a record type was declared in the DDL to have a calc key, then any of its records can be found "out of the blue," without reference to any set in which the record type participates. MDBS.DMS finds the desired record on the basis of a calc key value furnished by the application program. If duplicate calc keys are allowed, then all records with the same calc key value can be found.

4. The application program can sequentially find the records in a desired area or areas. This refers to the physical sequence of records.

If a find command returns a command status of zero, then the desired record was found. If the command status is 255, then the record requested through a find command does not exist in the data base. The branching and control structure of a host language program is typically based quite heavily on the command status returned by the various find commands used in the program.

Data security conditions, as defined in a DDL schema specification, are automatically enforced when a find command is invoked. If the user of an application program attempts to perform a "find" that is inconsistent with the user's read access codes, a command status will result which indicates that the find was not performed.

B. Command Details

FDRK Find Duplicate Record based on calc Key **FDRK**

Command and Arguments

FDRK,rec,iblk

Currency Indicators

Used: CRU Changed: CRU ← record having duplicate calc key value

Description

If the current of run unit is not null when FDRK is invoked: Find an occurrence of the same record type as the current of run unit, and having a calc key value that matches the calc key value held in iblk's variable(s). If such a record is found, it becomes the new current of run unit. This command is valid only if the record that is the current of run unit is an occurrence of rec and rec has been defined (in the DDL) as having a calc key. The calc key value indicated by iblk must duplicate the calc key value of the current of run unit, otherwise a command status error results (the current of run unit is not altered).

Use of FRK, followed by repeated use of FDRK, will find all records of a given record type having the same calc key value. The order in which these are found is not guaranteed. If an FDRK command is given when there are no more duplicates of a particular calc key value, then the current of run unit becomes null and the command status is 255. If a user does not have read access to all data items that form the calc key, then a command status error is returned.

If the current of run unit is null when FDRK is invoked: FDRK has exactly the same effect as FRK.

Examples of Command Usage

```
block/direct    ... E0 = FDRK ("rec,iblk")
block/indirect  ... E0 = DMS ("FDRK,rec,iblk")
record/direct   ... E0 = FDRK ("rec", iblk)
record/indirect... E0 = DMSD ("FDRK,rec", iblk)
```

FFM — Find First Member — FFM

Command and Arguments

FFM,set-1

Currency Indicators

Used: CO(set-1) Changed: CM(set-1)←——first member (set-1)
CRU ←——first member (set-1)

Description

The first member connected to the current owner of **set-1** becomes the current member of **set-1** and the current of the run unit. Which member is first depends on the **set-1** member order, as specified with the DDL. If the current owner of **set-1** has no member, then the command status is 255, the current of run unit becomes null, and the current member of **set-1** is null. If a user does not have read access to **set-1**, then a command status error is returned.

Examples of Command Usage

```
block/direct    ... E0 = FFM ("set-1")
block/indirect  ... E0 = DMS ("FFM,set-1")
record/direct   ... E0 = FFM ("set-1")
record/indirect... E0 = DMS ("FFM,set-1")
```


FFO — Find First Owner — FFO

Command and Arguments

FFO,set-1

Currency Indicators

Used: CM(set-1) Changed: CO(set-1)←——first owner (set-1)
CRU ←——first owner (set-1)

Description

The first owner connected to the current member of **set-1** becomes the current owner of **set-1** and the current of the run unit. Which owner is first depends on the **set-1** owner order, as specified with the DDL. If the current member of **set-1** has no owner, then the command status is 255, the current of run unit is null, and the current owner of **set-1** is null. If a user does not have read access to **set-1**, then a command status error is returned.

Examples of Command Usage

```
block/direct    ... E0 = FFO ("set-1")
block/indirect  ... E0 = DMS ("FFO,set-1")
record/direct   ... E0 = FFO ("set-1")
record/indirect... E0 = DMS ("FFO,set-1")
```

FFS — Find First Sequential record — FFS

Command and Arguments

FFS,area (the area argument is optional)

Currency Indicators

Used: none Changed: CRU←——first record in area

Description

The record that is physically first within the indicated **area** becomes the current of run unit. If the indicated **area** has no records, then current of run unit becomes null and the command status is 255. When the **area** argument is omitted, FFS operates on the main **area** and then on all subsequent **areas** until a record is found (i.e., the main area may have no data records).

FFS allows a user with any access code to find the physically first record of an **area** regardless of the access codes of that record's record type. However, the user will be unable to

retrieve data from the found record or modify its data unless the user's access authorization permits read or write access to occurrences of the record type.

Examples of Command Usage

```
block/direct   ... E0 = FFS ("area")      or E0 = FFS ("    ")
block/indirect ... E0 = DMS ("FFS,area") or E0 = DMS ("FFS")
record/direct  ... E0 = FFS ("area")      or E0 = FFS ("    ")
record/indirect... E0 = DMS ("FFS,area") or E0 = DMS ("FFS")
```

FLM — Find Last Member — FLM

Command and Arguments

FLM,set-1

Currency Indicators

Used: CO(set-1) Changed: CM(set-1) ←— last member (set-1)
CRU ←— last member (set-1)

Description

The last member connected to the current owner of **set-1** becomes the current member of **set-1** and the current of the run unit. Which member is last depends on the **set-1** member order, as specified with the DDL. If the current owner of **set-1** has no member, then the command status is 255, the current of run unit becomes null and the current member of **set-1** becomes null. If a user does not have read access to **set-1**, then a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = FLM ("set-1")
block/indirect ... E0 = DMS ("FLM,set-1")
record/direct  ... E0 = FLM ("set-1")
record/indirect... E0 = DMS ("FLM,set-1")
```

FLO — Find Last Owner — FLO

Command and Arguments

FLO,set-1

Currency Indicators

Used: CM(set-1) Changed: CO(set-1) ←— last owner (set-1)
CRU ←— last owner (set-1)

Description

The last owner connected to the current member of **set-1** becomes the current owner of **set-1** and the current of the run unit. Which owner is last depends on the **set-1** owner order, as specified with the DDL. If the current member of **set-1** has no owner, then the command status is 255, the current of run unit becomes null, and the current owner of **set-1** becomes null. If a user does not have read access to **set-1**, then a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = FLO ("set-1")
block/indirect ... E0 = DMS ("FLO,set-1")
record/direct  ... E0 = FLO ("set-1")
record/indirect... E0 = DMS ("FLO,set-1")
```

FMI — Find Member based on data Item — FMI

Command and Arguments

FMI,item,set-1,iblk

Currency Indicators

Used: CO(set-1) Changed: CM(set-1), CRU ← first member(set-1) having data item value

Description

The first member record connected to the current owner of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current member of **set-1**. It also becomes the current of run unit. If there are two or more member records for the current owner of **set-1**, all having the **iblk** value for **itm**, which one is first depends on the **set-1** member order. The command FNMI can be used to find the next member having the same **iblk** value for **itm**.

The host language variable of **iblk** must be consistent in size and type with the **itm** data item (as specified in the DDL). If the value of **iblk**'s host language variable does not match the **itm** value of any member record connected to the current owner of **set-1**, then the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to both **itm** and **set-1**.

Examples of Command Usage

```
block/direct    ... E0 = FMI ("itm,set-1,iblk")
block/indirect  ... E0 = DMS ("FMI,itm,set-1,iblk")
record/direct   ... E0 = FMI ("itm,set-1",iblk)
record/indirect... E0 = DMSD ("FMI,itm,set-1",iblk)
```

FMSK — Find Member based on Sort Key — FMSK

Command and Arguments

FMSK,set-1,iblk

Currency Indicators

Used: CO(set-1) Changed: CM(set-1), CRU ← first member (set-1) having sort key value

Description

The first member record that has the sort key value of **iblk** and that is connected to the current owner of **set-1** becomes the current member of **set-1**. It also becomes the current of the run unit. **Set-1** must have a sorted member order.

If several member records for the current owner of **set-1** have the same sort key value, the one that is first depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL). The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current owner of the **set-1** has no member whose sort key value matches the sort key value of **iblk**, then the command status is 255. The member whose sort key value immediately follows the **iblk** sort key value becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **iblk**'s sort key value is greater than the sort key value of the last member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key.

Examples of Command Usage

```
block/direct   ... E0 = FMSK ("set-1,iblk")
block/indirect ... E0 = DMS  ("FMSK,set-1,iblk")
record/direct  ... E0 = FMSK ("set-1",iblk)
record/indirect... E0 = DMSD ("FMSK,set-1",iblk)
```

FNM — Find Next Member — FNM

Command and Arguments

FNM,set-1

Currency Indicators

Used:	CO(set-1)	Changed:	CM(set-1) ←——next member(set-1)
	CM(set-1)		CRU ←——next member(set-1)

Description

If the current member of **set-1** is not null when FNM is invoked: The next member record (connected to the current owner of **set-1**) following the current member of **set-1** becomes the new current member of **set-1**. It also becomes the current of run unit. Which member record is logically next depends on the **set-1** member order, as specified with the DDL. If the current member of **set-1** is the last member for the current owner of **set-1**, then FNM cannot find a next member. When there is no next member for the current owner of **set-1**, the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to **set-1**.

If the current member of **set-1** is null when FNM is invoked: FNM has exactly the same effect as FFM.

Examples of Command Usage

```
block/direct   ... E0 = FNM ("set-1")
block/indirect ... E0 = DMS ("FNM,set-1")
record/direct  ... E0 = FNM ("set-1")
record/indirect... E0 = DMS ("FNM,set-1")
```

FNMI — Find Next Member based on data Item — FNMI

Command and Arguments

FNMI,itm,set-1,iblk

Currency Indicators

Used:		Changed:		
CO(set-1)		CM(set-1)	←	next member(set-1)
CM(set-1)		CRU		with item value

Description

If the current member of set-1 is not null when FNMI is invoked: The next member record connected to the current owner of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current member of **set-1**. It also becomes the current of run unit. The host language variable of **iblk** must be consistent in size and type with the **itm** data item (as specified in the DDL). If there are two or more member records following the current member of **set-1**, all having the **iblk** value for **itm**, which one is next depends on the **set-1** member order. If such a record does not exist, the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned.

If the current member of set-1 is null when FNMI is invoked: FNMI has exactly the same effect as FMI. The first member record connected to the current owner of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current member of **set-1**.

Examples of Command Usage

```
block/direct    ... E0 = FNMI ("itm,set-1,iblk")
block/indirect ... E0 = DMS ("FNMI,itm,set-1,iblk")
record/direct   ... E0 = FNMI ("itm,set-1",iblk)
record/indirect... E0 = DMSD ("FNMI,itm,set-1",iblk)
```

FNMSK — Find Next Member based on Sort Key — FNMSK

Command and Arguments

FNMSK,set-1,iblk

Currency Indicators

Used:		Changed:		
CO(set-1)		CM(set-1)	←	next member
CM(set-1)		CRU		(set-1) having
				sort key value

Description

The next member record (after the current member) that has the sort key value of **iblk** and that is connected to the current owner of **set-1** becomes the current member of **set-1**. It also becomes the current of the run unit. **Set-1** must have a sorted member order.

If **set-1** has no current member, then FNMSK behaves just as FMSK. If several member records for the current owner of **set-1** have the same sort key value, the one that is next depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL). The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current owner of the **set-1** has no member (after the current member) whose sort key value matches the sort key value of **iblk**, then the command status is 255. The next member immediately following the current member becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **iblk**'s sort key value is greater than the sort key value of the last member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key.

Examples of Command Usage

```
block/direct   ... E0 = FNMSK ("set-1,iblk")
block/indirect ... E0 = DMS   ("FNMSK,set-1,iblk")
record/direct  ... E0 = FNMSK ("set-1",iblk)
record/indirect... E0 = DMSD  ("FNMSK,set-1",iblk)
```

FNO — Find Next Owner — FNO

Command and Arguments

FNO,set-1

Currency Indicators

Used:	CM(set-1)	Changed:	CO(set-1) ←——next owner(set-1)
	CO(set-1)		CRU ←——next owner(set-1)

Description

If the current owner of **set-1** is not null when FNO is invoked: The next owner record (connected to the current member of **set-1**) following the current owner of **set-1** becomes the new current owner of **set-1**. It also becomes the current of run unit. Which owner record is logically next depends on the **set-1** owner order, as specified with the DDL. If the current owner of **set-1** is the last owner for the current member of **set-1**, then FNO cannot find a next owner. When there is no next owner for the current member of **set-1**, the command status is 255, the current owner of **set-1** becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to **set-1**.

If the current owner of **set-1** is null when FNO is invoked: FNO has exactly the same effect as FFO.

Examples of Command Usage

```
block/direct   ... E0 = FNO ("set-1")
block/indirect ... E0 = DMS ("FNO,set-1")
record/direct  ... E0 = FNO ("set-1")
record/indirect... E0 = DMS ("FNO,set-1")
```

FNOI — Find Next Owner based on data Item — FNOI

Command and Arguments

FNOI,itm,set-1,iblk

Currency Indicators

Used:	CM(set-1)	Changed:	CO(set-1) ←	next owner(set-1)
	CO(set-1)		CRU	with item value

Description

If the current owner of **set-1** is not null when FNOI is invoked: The next owner record connected to the current member of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current owner of **set-1**. It also becomes the current of run unit. The host language variable of **iblk** must be consistent in size and type with the **itm** data item (as specified in the DDL). If there are two or more owner records following the current owner of **set-1**, all having the **iblk** value for **itm**, which one is next depends on the **set-1** owner order. If such a record does not exist, the command status is 255, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned.

If the current owner of set-1 is null when FNOI is invoked: FNOI has exactly the same effect as FOI. The first owner record connected to the current member of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current owner of **set-1**.

Examples of Command Usage

```
block/direct    ... E0 = FNOI ("itm,set-1,iblk")
block/indirect  ... E0 = DMS ("FNOI,itm,set-1,iblk")
record/direct   ... E0 = FNOI ("itm,set-1",iblk)
record/indirect... E0 = DMS ("FNOI,itm,set-1",iblk)
```

FNOSK — Find Next Owner based on Sort Key — FNOSK

Command and Arguments

FNOSK,set-1,iblk

Currency Indicators

Used:		Changed:		
CO(set-1)		CO(set-1)		next owner(set-1)
CM(set-1)		CRU	←	having sort key value

Description

The next owner record (after the current owner) that has the sort key value of **iblk** and that is connected to the current member of **set-1** becomes the current owner of **set-1**. It also becomes the current of the run unit. **Set-1** must have a sorted owner order.

If **set-1** has no current owner, then FNOSK behaves just as FOSK. If several owner records for the current member of **set-1** have the same sort key value, the one that is next depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL). The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current member of the **set-1** has no owner (after the current owner) whose sort key value matches the sort key value of **iblk**, then the command status is 255. The next owner immediately following the current member becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the

current member of **set-1** has no owners or when **iblk**'s sort key value is greater than the sort key value of the last owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key.

Examples of Command Usage

```
block/direct    ... E0 = FNOSK ("set-1,iblk")
block/indirect  ... E0 = DMS   ("FNOSK,set-1,iblk")
record/direct   ... E0 = FNOSK ("set-1",iblk)
record/indirect... E0 = DMSD ("FNOSK,set-1",iblk)
```

FNS — Find Next Sequential record — FNS

Command and Arguments

FNS,area (the area argument is optional)

Currency Indicators

Used: CRU Changed: CRU←——next record in area

Description

The record that is physically next after the current of run unit within the indicated **area** becomes the current of run unit. A command status error is issued if the current of run unit is not in the indicated **area**; the current of run unit is not changed.

If the current of run unit is the physically last record in an **area** when FNS is invoked, then FNS will give a command status of 255 and the current of run unit is unchanged. If an **area** name is not indicated, then FNS will not give a 255 command status at the end of an **area**. It will proceed to make the physically first record of the next **area** the current of run unit. The sequence of areas is determined by the order in which areas are defined in the DDL **area** section. The main **area** is always the first **area**. If the current of run unit is the last record in the last **area**, invoking FNS will give a 255 command status and leave the current of run unit null.

FNS allows a user with any access code to find the physically next record of an **area** regardless of the access codes of that record's record type. However, the user will be unable to retrieve data from the found record or modify its data unless the user's access authorization permits read or write access to occurrences of the record type.

Examples of Command Usage

```
block/direct    ... E0 = FNS ("area")      or E0 = FNS ("    ")
block/indirect  ... E0 = DMS ("FNS,area") or E0 = DMS ("FNS")
record/direct   ... E0 = FNS ("area")      or E0 = FNS ("    ")
record/indirect... E0 = DMS ("FNS,area") or E0 = DMS ("FNS")
```

FOI — Find Owner based on data Item — FOI

Command and Arguments

FOI,item,set-l,iblk

Currency Indicators

Used: CM(set-l) **Changed:** CO(set-l) ← first owner(set-l) having data item value
CRU

Description

The first owner record connected to the current member of **set-l**, and having a value for **itm** that matches the value of **iblk**, becomes the current owner of **set-l**. It also becomes the current of run unit. If there are two or more owner records for the current member of **set-l**, all having the **iblk** value for **itm**, which one is first depends on the **set-l** owner order. The command FNOI can be used to find the next owner having the same **iblk** value for **itm**.

The host language variable of **iblk** must be consistent in size and type with the **itm** data item (as specified in the DDL). If the value of **iblk**'s host language variable does not match the **itm** value of any owner record connected to the current member of **set-l**, then the command status is 255, the current owner of **set-l** becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to both **itm** and **set-l**.

Examples of Command Usage

```
block/direct    ... E0 = FOI ("itm,set-l,iblk")
block/indirect  ... E0 = DMS ("FOI,itm,set-l,iblk")
record/direct   ... E0 = FOI ("itm,set-l",iblk)
record/indirect... E0 = DMSD ("FOI,itm,set-l",iblk)
```

FOSK — Find Owner based on Sort Key — FOSK

Command and Arguments

FOSK,set-1,iblk

Currency Indicators

Used: CM(set-1) Changed: CO(set-1) ← first owner (set-1) having sort key value
CRU

Description

The first owner record that has the sort key value of **iblk** and that is connected to the current member of **set-1** becomes the current owner of **set-1**. It also becomes the current of the run unit. **Set-1** must have a sorted owner order.

If several owner records for the current member of **set-1** have the same sort key value, the one that is first depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL). The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current member of the **set-1** has no owner whose sort key value matches the sort key value of **iblk**, then the command status is 255. The owner whose sort key value immediately follows the **iblk** sort key value becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the current member of **set-1** has no owners or when **iblk**'s sort key value is greater than the sort key value of the last owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key.

Examples of Command Usage

```
block/direct    ... E0 = FOSK ("set-1,iblk")
block/indirect  ... E0 = DMS  ("FOSK,set-1,iblk")
record/direct   ... E0 = FOSK ("set-1",iblk)
record/indirect... E0 = DMSD ("FOSK,set-1",iblk)
```


FPM — Find Prior Member — FPM

Command and Arguments

FPM,set-1

Currency Indicators

Used:		Changed:	
CO(set-1)		CM(set-1)	←——prior member(set-1)
CM(set-1)		CRU	←——prior member(set-1)

Description

If the current member of set-1 is not null when FPM is invoked: The prior member record (connected to the current owner of **set-1**) preceding the current member of **set-1** becomes the new current member of **set-1**. It also becomes the current of run unit. Which member record is logically prior depends on the **set-1** member order, as specified with the DDL. If the current member of **set-1** is the first member for the current owner of **set-1**, then FPM cannot find a prior member. When there is no prior member for the current owner of **set-1**, the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to **set-1**.

If the current member of set-1 is null when FPM is invoked: FPM has exactly the same effect as FLM.

Examples of Command Usage

```
block/direct   ... E0 = FPM ("set-1")
block/indirect ... E0 = DMS ("FPM,set-1")
record/direct  ... E0 = FPM ("set-1")
record/indirect... E0 = DMS ("FPM,set-1")
```

FPMI — Find Prior Member based on data Item — FPMI

Command and Arguments

FPMI,itm,set-1,iblk

Currency Indicators

Used:		Changed:	
CO(set-1)		CM(set-1)	←— prior member(set-1) with item value
CM(set-1)		CRU	←— prior member(set-1) with item value

Description

If the current member of **set-1** is not null when FPMI is invoked: The prior member record connected to the current owner of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current member of **set-1**. It also becomes the current of run unit. The host language variable of **iblk** must be consistent in size and type with the **itm** data item (as specified in the DDL). If there are two or more member records preceding the current member of **set-1**, all having the **iblk** value for **itm**, which one is prior depends on the **set-1** member order. If such a record does not exist, the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned.

If the current member of **set-1** is null when FPMI is invoked: The last member record connected to the current owner of **set-1**, and having a value for **itm** that matches the value of **iblk**, becomes the current member of **set-1**.

Examples of Command Usage

```
block/direct    ... E0 = FPMI ("itm,set-1,iblk")
block/indirect  ... E0 = DMS ("FPMI,itm,set-1,iblk")
record/direct   ... E0 = FPMI ("itm,set-1",iblk)
record/indirect... E0 = DMSD ("FPMI,itm,set-1",iblk)
```

FPMSK — Find Prior Member based on Sort Key — FPMSK

Command and Arguments

```
FPMSK,set-1,iblk
```

Currency Indicators

Used:		Changed:		
CO(set-1)		CM(set-1)		prior member
CM(set-1)		CRU	←	(set-1) having sort key value

Description

Set-1 must have a sorted member order. The prior member record (after the current member) that has the sort key value of **iblk** and that is connected to the current owner of **set-1** becomes the current member of **set-1**. It also becomes the current of the run unit. If **set-1** has no current member, then the last member having the sort key value is found. If several member records for the current owner of **set-1** have the same sort key value, the one that is prior depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL).

The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current owner of the **set-1** has no member (before the current member) whose sort key value matches the sort key value of **iblk**, then the command status is 255. The member immediately preceding the current member becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **iblk**'s sort key value is lower than the sort key value of the first member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key.

Examples of Command Usage

```
block/direct   ... E0 = FPMSK ("set-1,iblk")
block/indirect ... E0 = DMS ("FPMSK,set-1,iblk")
record/direct  ... E0 = FPMSK ("set-1",iblk)
record/indirect... E0 = DMSD ("FPMSK,set-1",iblk)
```

FPO — Find Prior Owner — FPO

Command and Arguments

FPO,set-1

Currency Indicators

Used:	CM(set-1)	Changed:	CO(set-1) ←——prior owner(set-1)
	CO(set-1)		CRU ←——prior owner(set-1)

Description

If the current owner of **set-1** is not null when FPO is invoked: The prior owner record (connected to the current member of **set-1**) preceding the current owner of **set-1** becomes the new current owner of **set-1**. It also becomes the current of run unit. Which owner record is logically prior depends on the **set-1** owner order, as specified with the DDL. If the current owner of **set-1** is the first owner for the current member of **set-1**, then FPO cannot find a prior owner. When there is no prior owner for the current member of **set-1**, the command status is 255, the current owner of

set-1 becomes null, and the current of run unit becomes null. A command status error is returned if a user does not have read access to set-1.

If the current owner of set-1 is null when FPO is invoked: FPO has exactly the same effect as FLO.

Examples of Command Usage

```
block/direct   ... E0 = FPO ("set-1")
block/indirect ... E0 = DMS ("FPO,set-1")
record/direct  ... E0 = FPO ("set-1")
record/indirect... E0 = DMS ("FPO,set-1")
```

FPOI — Find Prior Owner based on data Item — FPOI

Command and Arguments

FPOI,itm,set-1,iblk

Currency Indicators

Used:		Changed:		
CM(set-1)		CO(set-1)	←	prior owner(set-1)
CO(set-1)		CRU		with item value

Description

If the current owner of set-1 is not null when FPOI is invoked: The prior owner record connected to the current member of set-1, and having a value for itm that matches the value of iblk, becomes the current owner of set-1. It also becomes the current of run unit. The host language variable of iblk must be consistent in size and type with the itm data item (as specified in the DDL). If there are two or more owner records preceding the current owner of set-1, all having the iblk value for itm, which one is prior depends on the set-1 owner order. If such a record does not exist, the command status is 255, the current owner of set-1 becomes null, and the current of run unit becomes null. If a user does not have read access to both itm and set-1, then a command status error is returned.

If the current owner of set-1 is null when FPOI is invoked: The last owner record connected to the current member of set-1, and having a value for itm that matches the value of iblk, becomes the current owner of set-1.

Examples of Command Usage

```
block/direct   ... E0 = FPOI ("itm,set-1,iblk")
block/indirect ... E0 = DMS ("FPOI,itm,set-1,iblk")
record/direct  ... E0 = FPOI ("itm,set-1",iblk)
record/indirect... E0 = DMS ("FPOI,itm,set-1",iblk)
```


FPOSK — Find Prior Owner based on Sort Key — FPOSK

Command and Arguments

FPOSK,set-1,iblk

Currency Indicators

Used: CO(set-1), CM(set-1)

Changed: CO(set-1), CRU ← prior owner(set-1) having sort key value

Description

Set-1 must have a sorted owner order. The prior owner record (before the current owner) that has the sort key value of **iblk** and that is connected to the current member of **set-1** becomes the current owner of **set-1**. It also becomes the current of the run unit. If **set-1** has no current owner, then the last owner having the sort key value is found. If several owner records for the current member of **set-1** have the same sort key value, the one that is prior depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL).

The host language variable(s) of **iblk** must contain the desired sort key value. If there are multiple data items for the **set-1** sort key, the sort key value for **iblk** must be consistent with the sequence, types and sizes of the data items in the DDL sort key specification. If RECORD-TYPE is part of the sort key, then the name of the desired record type must appear as part of the **iblk** sort key value; the host language variable holding the record type name must be consistent with a string data item of 8 bytes in size.

If the current member of the **set-1** has no owner (before the current owner) whose sort key value matches the sort key value of **iblk**, then the command status is 255. The owner immediately preceding the current owner becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the current member of **set-1** has no owners or when **iblk**'s sort key value is less than the sort key value of the first owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key.

Examples of Command Usage

```
block/direct   ... E0 = FPOSK ("set-1,iblk")
block/indirect ... E0 = DMS   ("FPOSK,set-1,iblk")
record/direct  ... E0 = FPOSK ("set-1",iblk)
record/indirect... E0 = DMSD  ("FPOSK,set-1",iblk)
```


FRK — Find Record based on calc Key — FRK

Command Arguments

FRK,rec,iblk

Currency Indicators

Used: none Changed: CRU←— record having calc key value

Description

Find the occurrence of the indicated record type (**rec**) whose calc key has a value that matches the value of **iblk**'s host language variable(s). If such an occurrence is found, it becomes the current of run unit. This command is valid only if **rec** has a calc key clause in the DDL specification. Occurrences of **rec** with duplicate calc key values may exist in the data base, if the DDL specification declares that duplicates are allowed. If there is more than one occurrence of **rec** whose calc key value matches the **iblk** value, then FRK finds one of these duplicates and FDRK can be used repeatedly to find each of the other duplicates.

The sequence, size and type of **iblk**'s host language variables must be consistent with the sequence, size and type of data items that make up the calc key for **rec**. If the **iblk** value does not match the calc key value for any occurrence of the **rec** record type, then the command status is 255 and CRU becomes null. A command status error is returned if a user does not have read access to all data items that make up **rec**'s calc key.

Examples of Command Usage

```
block/direct    ... E0 = FRK ("rec,iblk")
block/indirect ... E0 = DMS ("FRK,rec,iblk")
record/direct   ... E0 = FRK ("rec", iblk)
record/indirect... E0 = DMSD ("FRK,rec", iblk)
```


This page intentionally left blank.

IV. RETRIEVAL COMMANDS

A. Overview

The MDBS Data Manipulation Language supports twenty-eight data retrieval commands. Each retrieval command has the effect of transferring data from a record to a host language variable(s) through the mechanism of a data block (or program record type). There are two major groups of retrieval commands: those that get data from a current record and those that obtain data from a record that may not have previously been current.

The first group is intended for situations where data need to be retrieved from a previously found record that is presently the current owner of a set, the current member of a set, or the current of run unit. This group of retrieval commands is composed of two subgroups.

1. Get the values of all data items (i.e., fields) from the record that is the current owner of an indicated set, the current member of an indicated set, or the current of run unit. These values are deposited in the host language variable(s) of a specified data block.
2. Get the value of a particular data item (i.e., field) from the record that is the current owner of an indicated set, the current member of an indicated set, or the current of run unit. This data value is deposited in the host language variable of a specified data block. Commands in this subgroup should be used where maximum data independence is desired.

The second major group of retrieval commands is intended for situations where data needs to be retrieved from a record that is not presently current. Each of these commands obtains a desired record by first making it the current owner of an indicated set, the current member of an indicated set, and/or the current of run unit. Then the values of all data items from that record are deposited in the host language variable(s) of a specified data block. This retrieval approach can result in faster processing than using a find command followed by a command to get data from the found record, particularly in multiuser environments.

For either retrieval approach, it is vital that the host language variable(s) receiving retrieved data is consistent with the type and size of the data item(s). Each data value is converted from the internal MDBS form for the data item type into the consistent host language form. If a host language variable is of insufficient size to hold the entire stored data value, a command status error results. However, MDBS.DMS does attempt to return as much of the data value as possible into the host language variable. Where all data values of a record are being retrieved at once, the host language variables defined for the receiving data block must also conform with the sequence of data items as defined for that record's type in the DDL.

Data security conditions, as defined in a DDL schema specification, are automatically enforced when a retrieval command is invoked. If the user of an application program attempts to perform a retrieval that is inconsistent with that user's read access codes, a command status will result which indicates that the retrieval was not performed.

If any command status other than zero is returned, then no data was retrieved by the command and the values of the data block's variables are unchanged.

B. Command Details

GETC — GET data from Current of run unit — GETC

Command and Arguments

GETC,oblk

Currency Indicators

Used: CRU Changed: none

Description

All data values in the record that is the current of run unit are returned in the host language variables of **oblk**. The types, sizes and sequence of these variables must be consistent with the data items that make up the record type for the current of run unit. A command status error is returned if a user does not have read access to all data items that make up the record type for the current of run unit.

Examples of Command Usage

```
block/direct   ... E0 = GETC ("oblk")
block/indirect ... E0 = DMS  ("GETC,oblk")
record/direct  ... E0 = GETC (oblk)
record/indirect... E0 = DMSD ("GETC",oblk)
```

GETM — GET data from Member — GETM

Command and Argument

GETM,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: none

Description

All data values in the current member of **set-1** are returned in the host language variables of **oblk**. The types, sizes and sequence of these variables must be consistent with the data items that make up the member record type of **set-1**. A command status error is returned if a user does not have read access to all data items of the member record type of **set-1**.

Examples of Command Usage

```
block/direct   ... E0 = GETM ("set-1,oblk")
block/indirect ... E0 = DMS ("GETM,set-1,oblk")
record/direct  ... E0 = GETM ("set-1",oblk)
record/indirect... E0 = DMSD ("GETM,set-1",oblk)
```


GETO — GET data from Owner — GETO

Command and Argument

GETO,set-1,oblk

Currency Indicators

Used: CO(set-1) Changed: none

Description

All data values in the current owner of **set-1** are returned in the host language variables of **oblk.** The types, sizes and sequence of these variables must be consistent with the data items that make up the owner record type of **set-1.** A command status error is returned if a user does not have read access to all data items of the owner record type of **set-1.**

Examples of Command Usage

```
block/direct    ... E0 = GETO ("set-1,oblk")
block/indirect  ... E0 = DMS ("GETO,set-1,oblk")
record/direct   ... E0 = GETO ("set-1",oblk)
record/indirect... E0 = DMSD ("GETO,set-1",oblk)
```

GFC — Get Field from Current of run unit — GFC

Command and Arguments

GFC,itm,oblk

Currency Indicators

Used: CRU Changed: none

Description

The value of the **itm** data item (field) in the record that is the current of run unit is returned in the host language variable of **oblk.** The type and size of this variable must be consistent with the type and size of **itm** in the DDL specification. A command status error is returned if a user does not have read access to the **itm** field.

Examples of Command Usage

```
block/direct    ... E0 = GFC ("itm,oblk")
block/indirect  ... E0 = DMS ("GFC,itm,oblk")
record/direct   ... E0 = GFC ("itm",oblk)
record/indirect... E0 = DMSD ("GFC,itm",oblk)
```


GFM — Get Field from Member — GFM

Command and Arguments

GFM,itm,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: none

Description

The value of the item data item (field) in the current member of **set-1** is returned in the host language variable of **oblk.** The type and size of this variable must be consistent with the type and size of **itm** in the DDL specification. A command status error is returned if a user does not have read access to the **itm** field.

Examples of Command Usage

```
block/direct    ... E0 = GFM ("itm,set-1,oblk")
block/indirect  ... E0 = DMS ("GFM,itm,set-1,oblk")
record/direct   ... E0 = GFM ("itm,set-1",oblk)
record/indirect... E0 = DMSD ("GFM,itm,set-1",oblk)
```

GFO — Get Field from Owner — GFO

Command and Arguments

GFO,itm,set-1,oblk

Currency Indicators

Used: CO(set-1) Changed: none

Description

The value of the item data item (field) in the current owner of **set-1** is returned in the host language variable of **oblk.** The type and size of this variable must be consistent with the type and size of **itm** in the DDL specification. A command status error is returned if a user does not have read access to the **itm** field.

Examples of Command Usage

```
block/direct    ... E0 = GFO ("itm,set-1,oblk")
block/indirect  ... E0 = DMS ("GFO,itm,set-1,oblk")
record/direct   ... E0 = GFO ("itm,set-1",oblk)
record/indirect... E0 = DMSD ("GFO,itm,set-1",oblk)
```

ODRK Obtain Duplicate Record based on calc Key ODRK

Command and Arguments

ODRK,rec,blk

Currency Indicators

Used: CRU Changed: CRU←——record having duplicate calc key value

Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items defined for **rec**. Before ODRK is invoked, the desired calc key value should be assigned to the **blk** variable(s) that corresponds to the calc key's data item(s).

If the current of run unit is not null when ODRK is invoked: Obtain an occurrence of the same record type as the current of run unit, and having a calc key value that matches the calc key value held in **blk**'s variable(s). If such a record exists, it becomes the new current of run unit and the obtained data values are returned in the host language variables of **blk**. This command is valid only if the record that is the current of run unit is an occurrence of **rec** and **rec** has been defined (in the DDL) as having a calc key. The calc key value indicated by **blk** must duplicate the calc key value of the current of run unit, otherwise a command status error results, no record is obtained and the current of run unit is not altered.

Use of ORK, followed by repeated use of ODRK, will obtain all records of a given record type having the same calc key value. The order in which these are obtained is not guaranteed. If an ODRK command is given when there are no more duplicates of a particular calc key value, then the command status is 255, no record is obtained, and the current of run unit becomes null.

If a user does not have read access to all of **rec**'s data items, then a command status error results. The record is not obtained, but it does become the current of run unit.

If the current of run unit is null when ODRK is invoked: ODRK has exactly the same effect as ORK.

Examples of Command Usage

```
block/direct   ... E0 = ODRK ("rec,blk")
block/indirect ... E0 = DMS ("ODRK,rec,blk")
record/direct  ... E0 = ODRK ("rec",blk)
record/indirect... E0 = DMSD ("ODRK,rec",blk)
```


OFM — Obtain First Member

Command and Arguments

OFM,set-1,oblk

Currency Indicators

Used: CO(set-1) Changed: CM(set-1) ←—first member (set-1)
CRU ←—first member (set-1)

Description

The first member connected to the current owner of **set-1** is obtained. It becomes the current member of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must conform to the data items existing for the member record type. Which member is first depends on the **set-1** member order, as specified with the DDL.

If the current owner of **set-1** has no member, then the command status is 255, no record is obtained, the current of run unit becomes null, and the current member of **set-1** is null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and the currency indicators are unchanged. If a user has read access to **set-1** but not to all data items of its member record type, then a command status error results. The currency indicators are changed, but no record is obtained.

Examples of Command Usage

```
block/direct    ... E0 = OFM ("set-1,oblk")
block/indirect ... E0 = DMS ("OFM,set-1,oblk")
record/direct   ... E0 = OFM ("set-1",oblk)
record/indirect... E0 = DMSD ("OFM,set-1",oblk)
```


OFO Obtain First Owner OFO

Command and Arguments

OFO,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: CO(set-1) ←—first owner (set-1)
CRU ←—first owner (set-1)

Description

The first owner connected to the current member of **set-1** is obtained. It becomes the current owner of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with the data items existing for the owner record type. Which owner is first depends on the **set-1** owner order, as specified with the DDL.

If the current member of **set-1** has no owner, then the command status is 255, no record is obtained, the current of run unit is null, and the current owner of **set-1** is null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and currency indicators are unchanged. If a user has read access to **set-1** but not to all data items of its owner record type, then a command status error results. The currency indicators are changed, but no record is obtained.

Examples of Command Usage

```
block/direct   ... E0 = OFO ("set-1,oblk")
block/indirect ... E0 = DMS ("OFO,set-1,oblk")
record/direct  ... E0 = OFO ("set-1",oblk)
record/indirect... E0 = DMSD ("OFO,set-1",oblk)
```


OLM — Obtain Last Member

Command and Arguments

```
OLM,set-1,oblk
```

Currency Indicators

Used: CO(set-1) Changed: CM(set-1) ←—last member (set-1)
CRU ←—last member (set-1)

Description

The last member connected to the current owner of **set-1** is obtained. It becomes the current member of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the member record type. Which member is last depends on the **set-1** member order, as specified with the DDL.

If the current owner of **set-1** has no member, then the command status is 255, no record is obtained, the current of run unit becomes null and the current member of **set-1** becomes null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and the currency indicators are unchanged. If a user has read access to **set-1** but not to all data items of its member record type, then a command status error results. The currency indicators are changed, but no record is obtained.

Examples of Command Usage

```
block/direct    ... E0 = OLM ("set-1,oblk")
block/indirect ... E0 = DMS ("OLM,set-1,oblk")
record/direct   ... E0 = OLM ("set-1",oblk)
record/indirect... E0 = DMSD ("OLM,set-1",oblk)
```


OLO — Obtain Last Owner

Command and Arguments

OLO,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: CO(set-1) ⟵ last owner (set-1)
CRU ⟵ last owner (set-1)

Description

The last owner connected to the current member of **set-1** is obtained. It becomes the current owner of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the owner record type. Which owner is last depends on the **set-1** owner order, as specified with the DDL.

If the current member of **set-1** has no owner, then the command status is 255, no record is obtained, the current of run unit becomes null, and the current owner of **set-1** becomes null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and no currency indicators are changed. If a user has read access to **set-1** but not to all data items of its owner record type, then a command status error results. The currency indicators are changed, but no record is obtained.

Examples of Command Usage

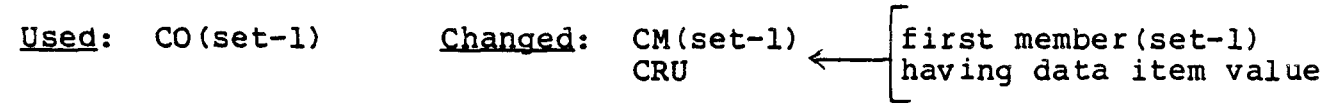
```
block/direct   ... E0 = OLO ("set-1,oblk")
block/indirect ... E0 = DMS ("OLO,set-1,oblk")
record/direct  ... E0 = OLO ("set-1",oblk)
record/indirect... E0 = DMSD ("OLO,set-1",oblk)
```


OMI — Obtain Member based on data Item

Command and Arguments

OMI,itm,set-1,blk

Currency Indicators



Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the member record type of **set-1**. Before OMI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**. The system obtains the first member record connected to the current owner of **set-1**, and having the indicated value for **itm**. It becomes the current member of **set-1** and the current of run unit. The obtained data values are returned in the host language variables of **blk**. If there are two or more member records for the current owner of **set-1**, all having the **blk** value for **itm**, which one is first depends on the **set-1** member order. The command ONMI can be used to obtain the next member having the same value for **itm**.

If the value of **blk**'s host language variable does not match the **itm** value of any member record connected to the current owner of **set-1**, then the command status is 255, no record is obtained, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to **set-1** and **itm**, then a command status error will result, no record will be obtained, and currency indicators will remain unchanged. If a user has read access to **set-1** and **itm** but not to all data items of the member record type, then a command status error results. The currency indicators are changed, but no record is obtained.

Examples of Command Usage

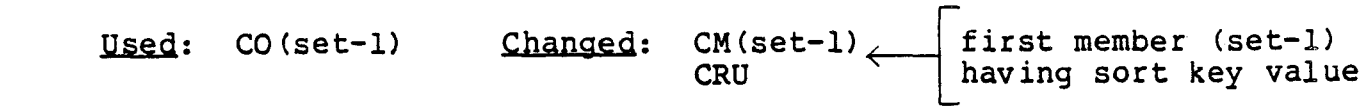
```
block/direct   ... E0 = OMI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("OMI,itm,set-1,blk")
record/direct  ... E0 = OMI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("OMI,itm,set-1",blk)
```


OMSK — Obtain Member based on Sort Key

Command and Arguments

OMSK,set-1,blk

Currency Indicators



Description

Set-1 must have a sorted member order. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the member record type of **set-1**. Before OMSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is part of the sort key, then OMSK cannot be used in place of FMSK and GETC. The system obtains the first member record connected to the current owner of **set-1** and having the indicated sort key value. This record becomes the current member of **set-1** and the current of run unit. Its data values are returned in the host language variables of **blk**.

If several member records for the current owner of **set-1** have the same sort key value, the one that is first depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL).

If the current owner of the **set-1** has no member whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The member whose sort key value immediately follows the **blk** sort key value becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **blk**'s sort key value is greater than the sort key value of the last member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key. No record is obtained and currency indicators are unchanged. If a user has read access to **set-1** and the sort key but not to all data items of the member record type, then a command status error is returned. Currency indicators are changed, but no record is obtained.

Examples of Command Usage

```
block/direct    ... E0 = OMSK ("set-1,blk")
block/indirect  ... E0 = DMS  ("OMSK,set-1,blk")
record/direct   ... E0 = OMSK ("set-1",blk)
record/indirect... E0 = DMSD ("OMSK,set-1",blk)
```


ONM — Obtain Next Member

Command and Arguments

ONM,set-1,oblk

Currency Indicators

Used:	CO(set-1)	Changed:	CM(set-1)←——next member(set-1)
	CM(set-1)		CRU ←——next member(set-1)

Description

If the current member of set-1 is not null when ONM is invoked: The next member record (connected to the current owner of **set-1**) following the current member of **set-1** is obtained. It becomes the new current member of **set-1** and the current of run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the member record type. Which member record is logically next depends on the **set-1** member order, as specified with the DDL.

If the current member of **set-1** is the last member for the current owner of **set-1**, then ONM cannot find a next member. When there is no next member for the current owner of **set-1**, the command status is 255, no record is obtained, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and the currency indicators remain unchanged. If a user has read access to **set-1** but not to all data items of its member record type, then a command status error results. The currency indicators are changed, but no record is obtained.

If the current member of set-1 is null when ONM is invoked: ONM has exactly the same effect as OFM.

Examples of Command Usage

```
block/direct   ... E0 = ONM ("set-1,oblk")
block/indirect ... E0 = DMS ("ONM,set-1,oblk")
record/direct  ... E0 = ONM ("set-1",oblk)
record/indirect... E0 = DMSD ("ONM,set-1",oblk)
```


ONMI Obtain Next Member based on data Item ONMI

Command and Arguments

ONMI,itm,set-1,blk

Currency Indicators

Used:	CO(set-1) CM(set-1)	Changed:	CM(set-1) CRU	← next member(set-1) with item value

Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the member record type of **set-1**. Before ONMI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**.

If the current member of **set-1** is not null when ONMI is invoked: The system obtains the next member record connected to the current owner of **set-1** and having the indicated value for **itm**. This record becomes the current member of **set-1** and the current of run unit. The obtained data values are returned in the host language variables of **blk**. If there are two or more member records following the current member of **set-1**, all having the **blk** value for **itm**, which one is next depends on the **set-1** member order. If there is no next member with the indicated value for **itm**, the command status is 255, no record is obtained, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned, no record is obtained, and currency indicators remain unchanged. If a user has read access to **set-1** and **itm** but not to all member data items, then a command status error results. Currency indicators are changed, but no data values are retrieved.

If the current member of **set-1** is null when ONMI is invoked: ONMI has exactly the same effect as OMI. The first member record connected to the current owner of **set-1**, and having a value for **itm** that matches the indicated **blk** value becomes the current member of **set-1**.

Examples of Command Usage

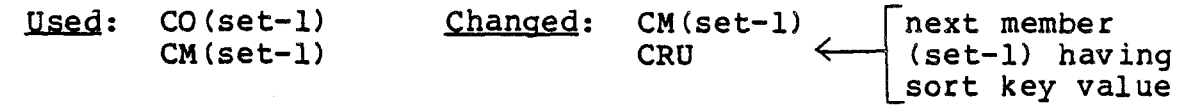
```
block/direct    ... E0 = ONMI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("ONMI,itm,set-1,blk")
record/direct   ... E0 = ONMI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("ONMI,itm,set-1",blk)
```


ONMSK — Obtain Next Member based on Sort Key — ONMSK

Command and Arguments

ONMSK,set-1,blk

Currency Indicators



Description

Set-1 must have a sorted member order. The type, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the member record type of **set-1**. Before ONMSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is part of the sort key, then ONMSK cannot be used in place of FNMSK and GETC. The system obtains the next member record (after the current member) that has the indicated sort key value and is connected to the current owner of **set-1**. This record becomes the current member of **set-1** and the current of run unit. Its data values are returned in the nost language variables of **blk**.

If **set-1** has no current member, then ONMSK behaves just as OMSK. If several member records for the current owner of **set-1** have the same sort key value, the one that is next depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL).

If the current owner of the **set-1** has no member (after the current member) whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The next member immediately following the current member becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **blk**'s sort key value is greater than the sort key value of the last member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key. No record is obtained and currency indicators are unchanged. If a user has read access to **set-1** and the sort key but not to all member data items, then a command status error results. The currency indicators are changed, but no data values are retrieved.

Examples of Command Usage

```
block/direct   ... E0 = ONMSK ("set-1,blk")
block/indirect ... E0 = DMS   ("ONMSK,set-1,blk")
record/direct  ... E0 = ONMSK ("set-1",blk)
record/indirect... E0 = DMSD  ("ONMSK,set-1",blk)
```


ONO Obtain Next Owner ONO

Command and Arguments

ONO,set-1,oblk

Currency Indicators

Used:		Changed:	
CM(set-1)		CO(set-1)	←——next owner(set-1)
CO(set-1)		CRU	←——next owner(set-1)

Description

If the current owner of **set-1** is not null when ONO is invoked: The next owner record (connected to the current member of **set-1**) following the current owner of **set-1** is obtained. This record becomes the new current owner of **set-1** and the current of run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the owner record type. Which owner record is logically next depends on the **set-1** owner order, as specified with the DDL.

If the current owner of **set-1** is the last owner for the current member of **set-1**, then ONO cannot find a next owner. When there is no next owner for the current member of **set-1**, the command status is 255, no record is obtained, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to **set-1**, then a command status error is returned, no record is obtained, and currency indicators remain unchanged. If a user has read access to **set-1** but not to all owner data items, then a command status error results. Currency indicators are changed, but no data values are retrieved.

If the current owner of **set-1** is null when ONO is invoked: ONO has exactly the same effect as OFO.

Examples of Command Usage

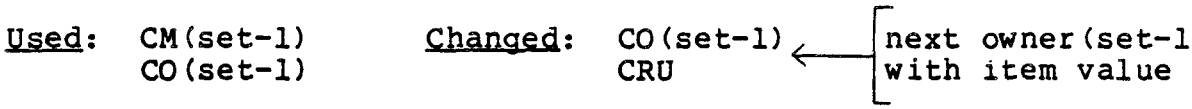
```
block/direct    ... E0 = ONO ("set-1,oblk")
block/indirect ... E0 = DMS ("ONO,set-1,oblk")
record/direct   ... E0 = ONO ("set-1",oblk)
record/indirect... E0 = DMSD ("ONO,set-1",oblk)
```


ONOI Obtain Next Owner based on data Item ONOI

Command and Arguments

ONOI,itm,set-1,blk

Currency Indicators



Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before ONOI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**.

If the current owner of **set-1** is not null when ONOI is invoked: The system obtains the next owner record connected to the current member of **set-1** and having the indicated value for **itm**. This record becomes the current owner of **set-1** and the current of run unit. The obtained data values are returned in the host language variable of **blk**. If there are two or more owner records following the current owner of **set-1**, all having the **blk** value for **itm**, which one is next depends on the **set-1** owner order. If there is no next owner with the indicated value for **itm**, then the command status is 255, no record is obtained, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned, no record is obtained, and currency indicators remain unchanged. If a user has read access to **set-1** and **itm** but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

If the current owner of **set-1** is null when ONOI is invoked: ONOI has exactly the same effect as OOI. The first owner record connected to the current member of **set-1**, and having a value for **itm** that matches the indicated **blk** value becomes the current owner of **set-1**.

Examples of Command Usage

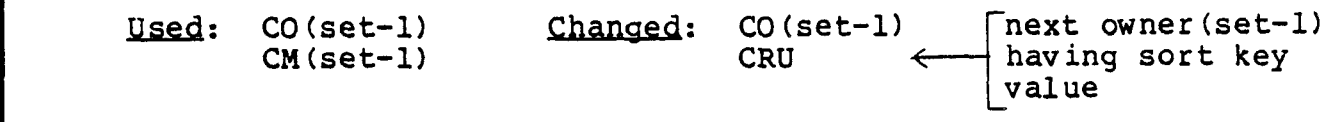
```
block/direct    ... E0 = ONOI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("ONOI,itm,set-1,blk")
record/direct   ... E0 = ONOI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("ONOI,itm,set-1",blk)
```


ONOSK — Obtain Next Owner based on Sort Key — ONOSK

Command and Arguments

ONOSK,set-1,blk

Currency Indicators



Description

Set-1 must have a sorted owner order. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before ONOSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is a part of the sort key, then ONOSK cannot be used in place of FNOSK and GETC. The system obtains the next owner record (after the current owner) that has the indicated sort key value and is connected to the current member of **set-1**. This record becomes the current owner of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **blk**.

If **set-1** has no current owner, then ONOSK behaves just as OOSK. If several owner records for the current member of **set-1** have the same sort key value, the one that is next depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL).

If the current member of the **set-1** has no owner (after the current owner) whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The next owner immediately following the current member becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the current member of **set-1** has no owners or when **blk**'s sort key value is greater than the sort key value of the last owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key. No record is obtained and currency indicators are unchanged. If a user has read access to **set-1** and the sort key but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

Examples of Command Usage

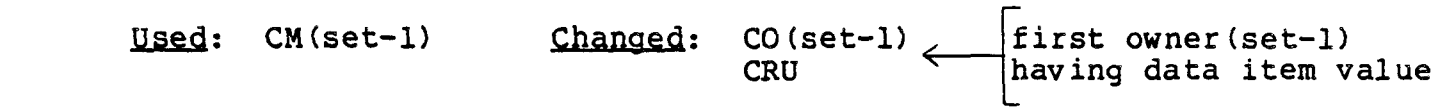
```
block/direct    ... E0 = ONOSK ("set-1,blk")
block/indirect  ... E0 = DMS   ("ONOSK,set-1,blk")
record/direct   ... E0 = ONOSK ("set-1",blk)
record/indirect... E0 = DMSD ("ONOSK,set-1",blk)
```


OOI — Obtain Owner based on data Item

Command and Arguments

OOI,itm,set-1,blk

Currency Indicators



Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before OOI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**. The system obtains the first owner record connected to the current member of **set-1** and having the indicated value for **itm**. This record becomes the current owner of **set-1** and the current of run unit. The obtained data values are returned in the host language variables of **blk**. If there are two or more owner records for the current member of **set-1**, all having the **blk** value for **itm**, which one is first depends on the **set-1** owner order. The command ONOI can be used to obtain the next owner having the same value for **itm**.

If the value of **blk**'s host language variable for **itm** does not match the **itm** value of any owner record connected to the current member of **set-1**, then the command status is 255, no record is obtained, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error will result, no record will be obtained, and currency indicators will remain unchanged. If a user has read access to **set-1** and **itm** but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

Examples of Command Usage

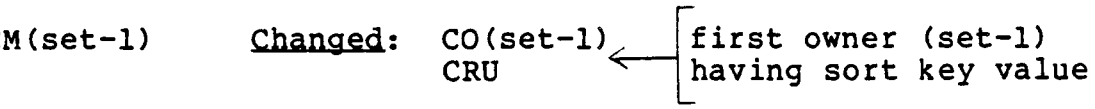
```
block/direct   ... E0 = OOI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("OOI,itm,set-1,blk")
record/direct  ... E0 = OOI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("OOI,itm,set-1",blk)
```


OOSK — Obtain Owner based on Sort Key — OOSK

Command and Arguments

```
OOSK,set-1,blk
```

Currency Indicators



Description

Set-1 must have a sorted owner order. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before OOSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is part of the sort key, then OOSK cannot be used in place of FOSK and GETC. The system obtains the first owner record connected to the current member of **set-1** and having the indicated sort key value. This record becomes the current owner of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **blk**.

If several owner records for the current member of **set-1** have the same sort key value, the one that is first depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL).

If the current member of the **set-1** has no owner whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The owner whose sort key value immediately follows the **blk** sort key value becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the current member of **set-1** has no owners or when **blk**'s sort key value is greater than the sort key value of the last owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key. No record is obtained and currency indicators are unchanged. If a user has read access to **set-1** and the sort key but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

Examples of Command Usage

```
block/direct    ... E0 = OOSK ("set-1,blk")
block/indirect  ... E0 = DMS  ("OOSK,set-1,blk")
record/direct   ... E0 = OOSK ("set-1",blk)
record/indirect... E0 = DMSD ("OOSK,set-1",blk)
```


OPM Obtain Prior Member OPM

Command and Arguments

OPM,set-1,oblk

Currency Indicators

Used:		Changed:	
CO(set-1)		CM(set-1)	←——prior member(set-1)
CM(set-1)		CRU	←——prior member(set-1)

Description

If the current member of **set-1** is not null when OPM is invoked: The prior member record (connected to the current owner of **set-1**) preceding the current member of **set-1** is obtained. This record becomes the new current member of **set-1** and the current of run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the member record type. Which member record is logically prior depends on the **set-1** member order, as specified with the DDL.

If the current member of **set-1** is the first member for the current owner of **set-1**, then OPM cannot find a prior member. When there is no prior member for the current owner of **set-1**, then the command status is 255, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to **set-1**, then a command status error is returned and the currency indicators remain unchanged. If a user has read access to **set-1** but not to all member data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

If the current member of **set-1** is null when OPM is invoked: OPM has exactly the same effect as OLM.

Examples of Command Usage

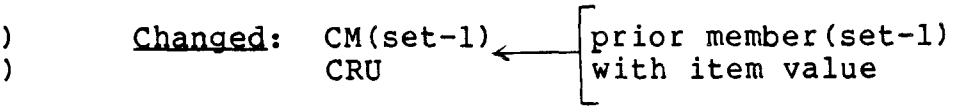
```
block/direct   ... E0 = OPM ("set-1,oblk")
block/indirect ... E0 = DMS ("OPM,set-1,oblk")
record/direct  ... E0 = OPM ("set-1",oblk)
record/indirect... E0 = DMSD ("OPM,set-1",oblk)
```


OPMI — Obtain Prior Member based on data Item

Command and Arguments

OPMI,itm,set-1,blk

Currency Indicators



Description

The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the member record type of **set-1**. Before OPMI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**.

If the current member of **set-1** is not null when OPMI is invoked: The system obtains the prior member record connected to the current owner of **set-1** and having the indicated value for **itm**. This record becomes the current member of **set-1** and the current of run unit. The obtained data values are returned in the host language variable of **blk**. If there are two or more member records preceding the current member of **set-1**, all having the **blk** value for **itm**, which one is prior depends on the **set-1** member order. If there is no prior member with the indicated value for **itm**, then the command status is 255, no record is obtained, the current member of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned, no record is obtained, and currency indicators remain unchanged. If a user has read access to **itm** and **set-1** but not to all member data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

If the current member of **set-1** is null when OPMI is invoked: The last member record connected to the current owner of **set-1**, and having a value for **itm** that matches the indicated **blk** value becomes the current member of **set-1**.

Examples of Command Usage

```
block/direct    ... E0 = OPMI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("OPMI,itm,set-1,blk")
record/direct   ... E0 = OPMI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("OPMI,itm,set-1",blk)
```


OPMSK Obtain Prior Member based on Sort Key OPMSK

Command and Arguments

```
OPMSK,set-1,blk
```

Currency Indicators

Used:		Changed:		
CO(set-1)		CM(set-1)		prior member
CM(set-1)		CRU	←	(set-1) having
				sort key value

Description

Set-1 must have a sorted member order. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing in the member record type for **set-1**. Before OPMSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is part of the sort key, then OPMSK cannot be used in place of FPMSK and GETC. The prior member record (after the current member) that has the sort key value of **blk** and that is connected to the current owner of **set-1** becomes the current member of **set-1** and the current of the run unit.

If **set-1** has no current member, then the last member having the sort key value is found. If several member records for the current owner of **set-1** have the same sort key value, the one that is prior depends on the DDL sort-clause specification for the **set-1** member order (either LIFO, FIFO, or IMMATERIAL).

If the current owner of the **set-1** has no member (before the current member) whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The member immediately preceding the current member becomes the current member of **set-1**, as well as the current of run unit. The exceptions are when the current owner of **set-1** has no members or when **blk**'s sort key value is lower than the sort key value of the first member of **set-1**'s current owner. In these cases, the current member of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the member sort key. No record is obtained and currency indicators remain unchanged. If a user has read access to **set-1** and the sort key but not to all member data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

Examples of Command Usage

```
block/direct   ... E0 = OPMSK ("set-1,blk")
block/indirect ... E0 = DMS   ("OPMSK,set-1,blk")
record/direct  ... E0 = OPMSK ("set-1",blk)
record/indirect... E0 = DMSD ("OPMSK,set-1",blk)
```


OPO — Obtain Prior Owner — OPO

Command and Arguments

OPO,set-1,oblk

Currency Indicators

Used:	CM(set-1)	Changed:	CO(set-1)←——prior owner(set-1)
	CO(set-1)		CRU ←——prior owner(set-1)

Description

If the current owner of **set-1** is not null when OPO is invoked: The prior owner record (connected to the current member of **set-1**) preceding the current owner of **set-1** is obtained. This record becomes the new current owner of **set-1** and the current of run unit. Its data values are returned in the host language variables of **oblk**. The types, sizes and sequence of variables designated for **oblk** must be consistent with data items existing for the owner record type. Which owner record is logically prior depends on the **set-1** owner order, as specified with the DDL.

If the current owner of **set-1** is the first owner for the current member of **set-1**, then OPO cannot find a prior owner. When there is no prior owner for the current member of **set-1**, then the command status is 255, no record is obtained, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to **set-1**, then a command status error is returned and currency indicators remain unchanged. If a user has read access to **set-1** but not to all its owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

If the current owner of **set-1** is null when OPO is invoked: OPO has exactly the same effect as *OLO*.

Examples of Command Usage

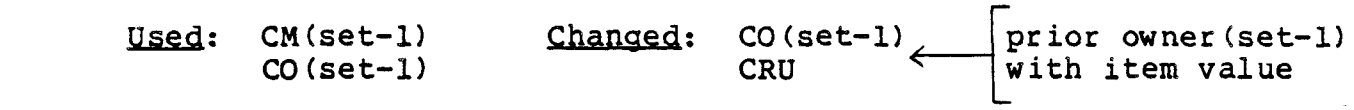
```
block/direct    ... EO = OPO ("set-1,oblk")
block/indirect  ... EO = DMS ("OPO,set-1,oblk")
record/direct   ... EO = OPO ("set-1",oblk)
record/indirect... EO = DMSD ("OPO,set-1",oblk)
```


OPOI — Obtain Prior Owner based on data Item — OPOI

Command and Arguments

OPOI,itm,set-1,blk

Currency Indicators



Description

The type, sizes and sequences of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before OPOI is invoked, the desired item value should be assigned to the **blk** variable that corresponds to **itm**.

If the current owner of **set-1** is not null when OPOI is invoked: The system obtains the prior owner record connected to the current member of **set-1** and having the indicated value for **itm**. This record becomes the current owner of **set-1** and the current of run unit. The obtained data values are returned in the host language variable of **blk**. If there are two or more owner records preceding the current owner of **set-1**, all having the **blk** value for **itm**, which one is prior depends on the **set-1** owner order. If there is no prior owner with the indicated value for **itm**, then the command status is 255, no record is obtained, the current owner of **set-1** becomes null, and the current of run unit becomes null. If a user does not have read access to both **itm** and **set-1**, then a command status error is returned, no record is obtained, and currency indicators remain unchanged. If a user has read access to **itm** and **set-1**, but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

If the current owner of **set-1** is null when OPOI is invoked: The last owner record connected to the current member of **set-1**, and having a value for **itm** that matches the indicated **blk** value becomes the current owner of **set-1**.

Examples of Command Usage

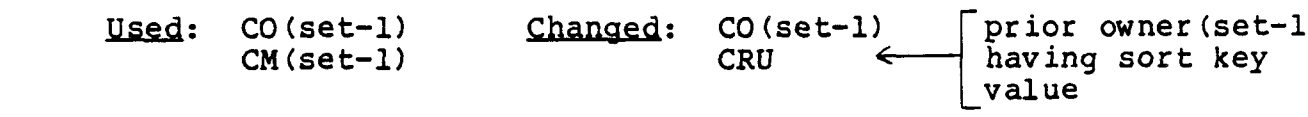
```
block/direct   ... E0 = OPOI ("itm,set-1,blk")
block/indirect ... E0 = DMS ("OPOI,itm,set-1,blk")
record/direct  ... E0 = OPOI ("itm,set-1",blk)
record/indirect... E0 = DMSD ("OPOI,itm,set-1",blk)
```


OPOSK — Obtain Prior Owner based on Sort Key — OPOSK

Command and Arguments

OPOSK,set-1,blk

Currency Indicators



Description

Set-1 must have a sorted owner order. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items existing for the owner record type of **set-1**. Before OPOSK is invoked, the desired sort key value should be assigned to the **blk** variable(s) that corresponds to the sort key's data item(s). If RECORD-TYPE is a part of the sort key, then OPOSK cannot be used in place of FPOSK and GETC. The system obtains the prior owner record (before the current owner) that has the indicated sort key value and is connected to the current member of **set-1**. This record becomes the current owner of **set-1** and the current of the run unit. Its data values are returned in the host language variables of **blk**.

If **set-1** has no current owner, then the last owner having the sort key value is found. If several owner records for the current member of **set-1** have the same sort key value, the one that is prior depends on the DDL sort-clause specification for the **set-1** owner order (either LIFO, FIFO, or IMMATERIAL).

If the current member of the **set-1** has no owner (before the current owner) whose sort key value matches the sort key value of **blk**, then the command status is 255 and no record is obtained. The owner immediately preceding the current member becomes the current owner of **set-1**, as well as the current of run unit. The exceptions are when the current member of **set-1** has no owners or when **blk**'s sort key value is less than the sort key value of the first owner of **set-1**'s current member. In these cases, the current owner of **set-1** becomes null and the current of run unit becomes null.

A command status error is returned if a user does not have read access to **set-1** and all data items that make up the owner sort key. No record is obtained and currency indicators are unchanged. If a user has read access to **set-1** and the sort key but not to all owner data items, then a command status error results. Currency indicators are changed, but no data are retrieved.

Examples of Command Usage

```
block/direct    ... E0 = OPOSK ("set-1,blk")
block/indirect  ... E0 = DMS   ("OPOSK,set-1,blk")
record/direct   ... E0 = OPOSK ("set-1",blk)
record/indirect... E0 = DMSD ("OPOSK,set-1",blk)
```


ORK — Obtain Record based on calc Key — ORK

Command Arguments

ORK,rec,blk

Currency Indicators

Used: none

Changed: CRU←——record having calc key value

Description

This command is valid only if **rec** has a calc key clause in the DDL specification. The types, sizes and sequence of host language variables designated for **blk** must be consistent with the data items defined for **rec**. Before ORK is invoked, the desired calc key value should be assigned to the **blk** variable(s) that corresponds to the calc key's data item(s). The system obtains an occurrence of **rec** having the indicated calc key value. This record becomes the current of run unit.

Occurrences of **rec** with duplicate calc key values may exist in the data base, if the DDL specification declares that duplicates are allowed. If there is more than one occurrence of **rec** whose calc key value matches the indicated value, then ORK obtains one of these duplicates and ODRK can be used repeatedly to obtain each of the other duplicates.

If there is no **rec** occurrence having the indicated calc key value, then the command status is 255, no record is obtained and CRU becomes null. A command status error is returned if a user does not have read access to all data items that make up **rec**'s calc key. No record is obtained and currency indicators remain unchanged. If a user has read access to the calc key but not to all **rec** data items, then a command status error results. The current of run unit changes, but no data are retrieved.

Examples of Command Usage

```
block/direct   ... E0 = ORK ("rec,blk")
block/indirect ... E0 = DMS ("ORK,rec,blk")
record/direct  ... E0 = ORK ("rec",blk)
record/indirect... E0 = DMSD ("ORK,rec",blk)
```


V. MODIFY COMMANDS

A. Overview

Six commands are available for modifying the data values of a record occurrence. Each command has the effect of transferring data from a host language variable(s) into a current record through the mechanism of a data block (or program record type). A record must be either the current owner of a set, the current member of a set, or the current of run unit before its data can be modified. In other words, the record must have been found before its data can be modified. Modify commands never alter any currency indicator. The modify commands fall into two groups.

1. Put new values into all data items (i.e., fields) for the record that is the current owner of an indicated set, the current member of an indicated set, or the current of run unit. MDBS.DMS obtains these new values from the host language variable(s) of a specified data block.

2. Put a new value into a particular data item (i.e., field for the record that is the current owner of an indicated set, the current member of an indicated set, or the current of run unit. MDBS.DMS obtains the new value from the host language variable(s) of a specified data block. Commands in this group should be used where maximum data independence is desired.

In either case, it is vital that the host language variables, containing data to be put into a record, are consistent with the type and size of the target data items. Where all data values of a record are being modified at once, the host language variables defined for the sending data block must also conform to the sequence of data items as defined for that record's type in the DDL.

If a data value being modified is part of a sort key, MDBS.DMS automatically maintains the sorted order.

A data item's value cannot be modified with the modify commands if this is part of a calc key. The modification can be accomplished by retrieving the record's contents, deleting the record from the data base, creating a new record having the old record's contents (except for the new calc key value), and re-establishing any relevant set connections.

Data security conditions, as defined in a DDL schema specification, are automatically enforced when a data modification command is invoked. If the user of an application program attempts to perform a "put" that is inconsistent with that user's write access codes, a command status will result which indicates that the modification was not performed.

If a data base has multiple areas, then all areas which permit pointer indices must be on-line during the execution of data modification commands

Data integrity conditions, as defined in a DDL schema specification, are automatically enforced when a data modification command is invoked. If an attempt is made to store a new data item value that is outside of the data item's feasible range of values, then a command status will result which indicates that the modification was not performed.

If any command status other than zero is returned, then no data in the record (current owner, current member, or current of run unit) was modified.

B. Command Details

PFC Put data into Field of Current of run unit **PFC**

Command and Arguments

PFC,itm,iblk

Currency Indicators

Used: CRU Changed: none

Description

The value of the **iblk** host language variable is put into the **itm** field of the record that is current of run unit. The type and size of this variable must be consistent with **itm**. A command status error is returned if the user does not have write access to **itm** and to the area containing the record to be changed. A command status error is also returned if the value of **iblk**'s variable is outside of **itm**'s feasibility range.

Examples of Command Usage

```
block/direct   ... E0 = PFC ("itm,iblk")
block/indirect ... E0 = DMS ("PFC,itm,iblk")
record/direct  ... E0 = PFC ("itm",iblk)
record/indirect... E0 = DMSD ("PFC,itm",iblk)
```

PFM Put data into Field of Member **PFM**

Command and Arguments

PFM,itm,set-1,iblk

Currency Indicators

Used: CM(set-1) Changed: none

Description

The value of the **iblk** host language variable is put into the **itm** field of the current member of **set-1**. A command status error is returned if the user does not have write access to **itm** and to the area containing the record to be changed. A command status error is also returned if the value of **iblk**'s variable is outside of **itm**'s feasibility range.

Examples of Command Usage

```
block/direct    ... E0 = PFM ("itm,set-1,iblk")
block/indirect  ... E0 = DMS ("PFM,itm,set-1,iblk")
record/direct   ... E0 = PFM ("itm,set-1",iblk)
record/indirect... E0 = DMSD ("PFM,itm,set-1",iblk)
```

PFO — Put data into Field of Owner — PFO

Command and Arguments

```
PFO,itm,set-1,iblk
```

Currency Indicators

Used: CO(set-1) Changed: none

Description

The value of the **iblk** host language variable is put into the **itm** field of the current owner of **set-1**. A command status error is returned if the user does not have write access to **itm** and to the area containing the record to be changed. A command status error is also returned if the value of **iblk**'s variable is outside of **itm**'s feasibility range.

Examples of Command Usage

```
block/direct    ... E0 = PFO ("itm,set-1,iblk")
block/indirect  ... E0 = DMS ("PFO,itm,set-1,iblk")
record/direct   ... E0 = PFO ("itm,set-1",iblk)
record/indirect... E0 = DMSD ("PFO,itm,set-1",iblk)
```

PUTC — PUT data into Current of run unit — PUTC

Command and Arguments

```
PUTC,iblk
```

Currency Indicators

Used: CRU Changed: none

Description

Data values of the **iblk** host language variables are put into the record that is the current of run unit. The types, sizes and sequence of these variables must be consistent with the data items that make up the record type for the current of run unit. A command status error is returned if the user does not have write access to all data items that make up the record type for the current of run unit and to the area containing the record to be changed. A command status error is also returned if the value of an **iblk** variable is outside of the feasibility range defined for the corresponding data item.

Examples of Command Usage

```
block/direct   ... E0 = PUTC ("iblk")
block/indirect ... E0 = DMS ("PUTC,iblk")
record/direct  ... E0 = PUTC (iblk)
record/indirect... E0 = DMSD ("PUTC",iblk)
```

PUTM — PUT data into Member — PUTM

Command and Arguments

```
PUTM,set-1,iblk
```

Currency Indicators

Used: CM(set-1) Changed: none

Description

Data values of the **iblk** host language variables are put into the current member of **set-1**. The types, sizes and sequence of these variables must be consistent with the data items that make up the member record type for **set-1**. A command status error is returned if the user does not have write access to all data items that make up the member record type and to the area containing the record to be changed. A command status error is also returned if the value of an **iblk** variable is outside the feasibility range defined for the corresponding data item.

Examples of Command Usage

```
block/direct   ... E0 = PUTM ("set-1,iblk")
block/indirect ... E0 = DMS ("PUTM,set-1,iblk")
record/direct  ... E0 = PUTM ("set-1",iblk)
record/indirect... E0 = DMSD ("PUTM,set-1,"iblk)
```


PUTO — PUT data into Owner — PUTO

Command and Arguments

PUTO,set-1,iblk

Currency Indicators

Used: CO(set-1) Changed: none

Description

Data values of the **iblk** host language variables are put into the current owner of **set-1**. The types, sizes and sequence of these variables must be consistent with the data items that make up the owner record type for **set-1**. A command status error is returned if the user does not have write access to all data items that make up the owner record type and to the area containing the record to be changed. A command status error is also returned if the value of an **iblk** variable is outside the feasibility range defined for the corresponding data item.

Examples of Command Usage

```
block/direct    ... E0 = PUTO ("set-1,iblk")
block/indirect ... E0 = DMS ("PUTO,set-1,iblk")
record/direct   ... E0 = PUTO ("set-1",iblk)
record/indirect... E0 = DMSD ("PUTO,set-1,"iblk)
```

This page intentionally left blank.

VI. ASSIGNMENT COMMANDS

A. Overview

Assignment commands allow an application programmer to assign a particular record (that has been previously found) to a currency indicator. Together with the find commands, the assignment commands form the basis for "navigating" through the record occurrences in a data base. As an example, suppose that a DEPT (department) record type owns an EMP (employee) record type through a 1:N set named CONTAINS. A find command can be used to find a particular occurrence of DEPT, making that occurrence the current of run unit. An assignment command (SOC) can then be used to set the current owner of CONTAINS to be the same record as the current of run unit. This leaves the program poised to find various employees (members of CONTAINS) that are related to a particular department (the current owner of CONTAINS). The assignment commands fall into four major groups.

1. Assign the new current of run unit to be the same as:

   the current owner of a particular set,
   the current member of a particular set,
   the current record of a particular user-defined indicator, or
   a previously saved current of run unit.

   The current of run unit can also be set to null and can be saved in the form of a data base key. It can also be set to null.

2. Assign the new current member of a particular set to be the same as:

   the current of run unit,
   the current member of a particular set,
   the current owner of a particular set, or
   the current record of a particular user-defined indicator.

   The current member of any set can also be set to null.

3. Assign the new current owner of a particular set to be the same as:

   the current of run unit,
   the current member of a particular set,
   the current owner of a particular set, or
   the current record of a particular user-defined indicator.

   The current owner of any set can also be set to null.

4. Assign the current record of a particular user-defined indicator to be the same as:

   the current of run unit,
   the current member of a particular set,
   the current owner of a particular set, or
   the current record of a particular user-defined indicator.

   A user-defined indicator can also be set to null.

With the exception of SOE and SON, any assignment command that sets the current owner of set-1 to be the same as some other currency indicator also automatically performs FFM, set-1. With the exception of SME and SMN, any assignment command that sets the current member of set-1 to be the same as some other currency indicator also automatically performs FFO, set-1. The DBENV command can be used to suppress the automatic FFM and FFO operations for assignment commands. Instead of FFM, the current member becomes null. Instead of FFO, the current owner becomes null. Setting a run unit's environment in this way may allow the program to be more compact if it needs to loop through owners or members of a set.

If a user does not have read access to all sets that are specified as arguments to an assignment command, then a command status error results and no currency indicator is altered.

B. Command Details

SCD — Set Current of run unit to Data base key — SCD

Command and Arguments

SCD,iblk

Currency Indicators

Used: none Changed: CRU ←—— data base key

Description

The value of **iblk**'s host language variable becomes the new current of run unit. Normally, the value will be a data base key that was previously saved by the SDC command. If the system detects that the value is not a valid data base key, then command status error 49 results for SCD and the current of run unit remains unchanged. The integrity of the data base itself will not be damaged.

In most environments, the **iblk** host language variable must be consistent with a four-byte binary data item. If some other type or size is necessary for a particular environment, then the difference is described in the pertinent system specific manual.

Examples of Command Usage

```
block/direct   ... EO = SCD ("iblk")
block/indirect ... EO = DMS ("SCD,iblk")
record/direct  ... EO = SCD (iblk)
record/indirect... EO = DMSD ("SCD",iblk)
```

SCM — Set Current of run unit based on Member — SCM

Command and Arguments

SCM,set-1

Currency Indicators

Used: CM(set-1) Changed: CRU ←—— CM(set-1)

Description

The record that is presently the current member of **set-1** becomes the new current of run unit. The current member of **set-1** remains unchanged. If the user does not have read access to **set-1**, a command status error is returned and the current of run unit is unchanged.

Examples of Command Usage

```
block/direct   ... E0 = SCM ("set-1")
block/indirect ... E0 = DMS ("SCM,set-1")
record/direct  ... E0 = SCM ("set-1")
record/indirect... E0 = DMS ("SCM,set-1")
```

SCN — Set Current of run unit to Null — SCN

Command and Arguments

SCN (no arguments)

Currency Indicators

Used: none Changed: CRU←——null

Description
The current of run unit becomes null.

Examples of Command Usage

```
block/direct   ... E0 = SCN (     )
block/indirect ... E0 = DMS ("SCN")
record/direct  ... E0 = SCN (     )
record/indirect... E0 = DMS ("SCN")
```

SCO — Set Current of run unit based on Owner — SCO

Command and Arguments

SCO,set-1

Currency Indicators

Used: CO(set-1) Changed: CRU←——CO(set-1)

Description
The record that is presently the current owner of **set-1** becomes the new current of run unit. The current owner of **set-1** remains unchanged. If the user does not have read access to **set-1**, a command status error is returned and the current of run unit is unchanged.

Examples of Command Usage

```
block/direct   ... E0 = SCO ("set-1")
block/indirect ... E0 = DMS ("SCO,set-1")
record/direct  ... E0 = SCO ("set-1")
record/indirect... E0 = DMS ("SCO,set-1")
```


SCU — Set Current of run unit based on User indicator — SCU

Command and Arguments

SCU,iblk

Currency Indicators

Used: CU(i) Changed: CRU< CU(i)

Description

The current record of a user-defined indicator becomes the new current of run unit. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The current record of the user-defined indicator remains unchanged. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct   ... E0 = SCU ("iblk")
block/indirect ... E0 = DMS ("SCU,iblk")
record/direct  ... E0 = SCU (iblk)
record/indirect... E0 = DMSD ("SCU",iblk)
```

SDC — Save Data base key for Current of run unit — SDC

Command and Arguments

SDC,oblk

Currency Indicators

Used: CRU Changed: none

Description

A data base key is a unique internal identifier for a record occurrence. The SDC command saves the data base key for the record that is the current of run unit. If the current of run unit is null, then a null data base key is saved. The data base key is saved in **oblk**'s host language variable. In most environments, this variable must be consistent with a four-byte binary data item. If some other type or size is necessary for a particular environment, then the difference is described in the pertinent system specific manual.

The SDC command can be used to set the current of run unit to be the same as a saved data base key.

Examples of Command Usage

```
block/direct   ... EO = SDC ("oblk")
block/indirect ... EO = DMS ("SDC,oblk")
record/direct  ... EO = SDC (oblk)
record/indirect... EO = DMSD ("SDC",oblk)
```

SMC — Set Member based on Current of run unit — SMC

Command and Arguments

SMC,set-1

Currency Indicators

Used:		Changed:		
CRU		CM(set-1)	<	CRU
		CO(set-1)	<	first owner
		CRU	<	first owner

Description

The record that is the current of run unit becomes the new current member of **set-1**. The first owner connected to the new current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. If the current member of **set-1** has no owners via **set-1**, then the current of run unit becomes null, the current owner of **set-1** becomes null, and a command status of 255 is returned. If the user does not have read access to **set-1**, a command status error is returned and no currency indicators are changed.

Examples of Command Usage

```
block/direct   ... E0 = SMC ("set-1")
block/indirect ... E0 = DMS ("SMC,set-1")
record/direct  ... E0 = SMC ("set-1")
record/indirect... E0 = DMS ("SMC,set-1")
```

SME — Set Member to current of run unit (Exception) — SME

Command and Arguments

```
SME,set-1
```

Currency Indicators

Used: CRU Changed: CM(set-1) ←—— CRU

Description

The record that is the current of run unit becomes the new current member of **set-1**. The current of run unit remains unchanged. If a user does not have read access to **set-1**, the current member of **set-1** does not change and a command status error is returned. If the record type of the current of run unit is not a member of **set-1**, then a command status error is returned and the current member of **set-1** is unchanged. If a current owner for **set-1** exists and the record indicated by the current of run unit is not connected to that owner through **set-1**, then a command status error of 12 is returned.

Examples of Command Usage

```
block/direct   ... E0 = SME ("set-1")
block/indirect ... E0 = DMS ("SME,set-1")
record/direct  ... E0 = SME ("set-1")
record/indirect... E0 = DMS ("SME,set-1")
```

SMM — Set Member based on Member — SMM

Command and Arguments

```
SMM,set-1,set-2
```

Currency Indicators

Used: CM(set-2) Changed: CM(set-1) ←—— CM(set-2)
CO(set-1) ←—— first owner
CRU ←—— first owner

Description

Set the current member of **set-1** to be the same as the current member of **set-2**. This command is valid only if **set-1** and **set-2** have the same member record type(s). The first owner of the new current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. If the new current member of **set-1** has no owners via **set-1**, then the current owner of **set-1** becomes null, the current of run unit becomes null and a command status of 255 is returned. If a user does not have read access to both **set-1** and **set-2**, then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct   ... E0 = SMM ("set-1,set-2")
block/indirect ... E0 = DMS ("SMM,set-1,set-2")
record/direct  ... E0 = SMM ("set-1,set-2")
record/indirect... E0 = DMS ("SMM,set-1,set-2")
```

SMN — Set Member to Null — SMN

Command and Arguments

```
SMN,set-1
```

Currency Indicators

Used: none Changed: CM(set-1)←——null

Description

The current member of **set-1** becomes null. The current owner of **set-1** is unaffected. If a user does not have read access to **set-1**, then the current member of **set-1** does not change and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = SMN ("set-1")
block/indirect ... E0 = DMS ("SMN,set-1")
record/direct  ... E0 = SMN ("set-1")
record/indirect... E0 = DMS ("SMN,set-1")
```

SMO — Set Member based on Owner — SMO

Command and Arguments

```
SMO,set-1,set-2
```

Currency Indicators

Used: CO(set-2) Changed: CM(set-1)←——CO(set-2)
CO(set-1)←——first owner
CRU ←——first owner

Description

Set the current member of **set-1** to be the same as the current owner of **set-2**. This command is valid only if the member record type(s) for **set-1** is the same as the owner record type(s) for **set-2**. The first owner of the new current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. If the new current member of **set-1** has no owners via set **set-1**, then the current owner of **set-1** becomes null, the current of run unit becomes null, and the command status is 255. If a user does not have read access to **set-1** and **set-2**, then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct    ... E0 = SMO ("set-1,set-2")
block/indirect  ... E0 = DMS ("SMO,set-1,set-2")
record/direct   ... E0 = SMO ("set-1,set-2")
record/indirect... E0 = DMS ("SMO,set-1,set-2")
```

SMU — Set Member based on User indicator — SMU

Command and Arguments

SMU,set-1,iblk

Currency Indicators

Used:	CU(i)	Changed:	CM(set-1) <	CU(i)
			CO(set-1) <	first owner
			CRU <	first owner

Description

Set the current member of **set-1** to be the same as the current record of a user-defined indicator. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The first owner of the new current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. If the new current member has no owners via **set-1**, then the current owner of **set-1** becomes null, the current of run unit become null, and the command status is 255. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct    ... E0 = SMU ("set-1,iblk")
block/indirect  ... E0 = DMS ("SMU,set-1,iblk")
record/direct   ... E0 = SMU ("set-1",iblk)
record/indirect... E0 = DMSD ("SMU,set-1",iblk)
```


SOC — Set Owner based on Current of run unit — SOC

Command and Arguments

SOC,set-1

Currency Indicators

Used:		Changed:		
CRU		CO(set-1)<	CRU	
		CM(set-1)<	first member	
		CRU <	first member	

Description

The record that is the current of run unit becomes the new current owner of **set-1**. The first member connected to the new current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. If the current owner of **set-1** has no members via **set-1**, then the current of run unit becomes null, the current member of **set-1** becomes null, and a command status of 255 is returned. If the user does not have read access to **set-1**, a command status error is returned and no currency indicators are changed.

Examples of Command Usage

```
block/direct   ... E0 = SOC ("set-1")
block/indirect ... E0 = DMS ("SOC,set-1")
record/direct  ... E0 = SOC ("set-1")
record/indirect... E0 = DMS ("SOC,set-1")
```

SOE — Set Owner to current of run unit (Exception) — SOE

Command and Arguments

SOE,set-1

Currency Indicators

Used:	Changed:	
CRU	CO(set-1)<	CRU

Description

The record that is the current of run unit becomes the new current owner of **set-1**. The current of run unit remains unchanged. If a user does not have read access to **set-1**, the current owner of **set-1** does not change and a command status error is returned. If the record type of the current of run unit is not an owner of **set-1**, then a command status error is returned and the current owner of **set-1** is unchanged. If a current member for **set-1** exists and the record indicated by the current of run unit is not connected to that member through **set-1**, then a command status error of 12 is returned.

Examples of Command Usage

```
block/direct   ... E0 = SOE ("set-1")
block/indirect ... E0 = DMS ("SOE,set-1")
record/direct  ... E0 = SOE ("set-1")
record/indirect... E0 = DMS ("SOE,set-1")
```

SOM — Set Owner based on Member — SOM

Command and Arguments

```
SOM,set-1,set-2
```

Currency Indicators

Used: CM(set-2) Changed: CO(set-1) ←—— CM(set-2)
CM(set-1) ←—— first member
CRU ←—— first member

Description

Set the current owner of **set-1** to be the same as the current member of **set-2**. This command is valid only if the owner record type(s) for **set-1** is the same as the member record type(s) for **set-2**. The first member of the new current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. If the new current owner of **set-1** has no members via set **set-1**, then the current member of **set-1** becomes null, the current of run unit becomes null and the command status is 255. If a user does not have read access to **set-1** and **set-2**, then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct   ... E0 = SOM ("set-1,set-2")
block/indirect ... E0 = DMS ("SOM,set-1,set-2")
record/direct  ... E0 = SOM ("set-1,set-2")
record/indirect... E0 = DMS ("SOM,set-1,set-2")
```

SON — Set Owner to Null — SON

Command and Arguments

```
SON,set-1
```

Currency Indicators

Used: none Changed: CO(set-1) ←—— null

Description

The current owner of **set-1** becomes null. The current member of **set-1** is unaffected. If a user does not have read access to **set-1**, then the current owner of **set-1** does not change and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = SON ("set-1")
block/indirect ... E0 = DMS ("SON,set-1")
record/direct  ... E0 = SON ("set-1")
record/indirect... E0 = DMS ("SON,set-1")
```

SOO — Set Owner based on Owner — SOO

Command and Arguments

SOO,set-1,set-2

Currency Indicators

Used: CO(set-2)

Changed: CO(set-1) ⟵ CO(set-2)
CM(set-1) ⟵ first member
CRU ⟵ first member

Description

Set the current owner of **set-1** to be the same as the current owner of **set-2**. This command is valid only if **set-1** and **set-2** have the same owner record type(s). The first member of the new current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. If the new current owner of **set-1** has no members via **set-1**, then the current member of **set-1** becomes null, the current of run unit becomes null, and a command status of 255 is returned. If a user does not have read access to both **set-1** and **set-2**, then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct   ... E0 = SOO ("set-1,set-2")
block/indirect ... E0 = DMS ("SOO,set-1,set-2")
record/direct  ... E0 = SOO ("set-1,set-2")
record/indirect... E0 = DMS ("SOO,set-1,set-2")
```

SOU — Set Owner based on User indicator — SOU

Command and Arguments

SOU,set-1,iblk

Currency Indicators

Used: CU(i) Changed: CO(set-1) ⟵ CU(i)
CM(set-1) ⟵ first member
CRU ⟵ first member

Description

Set the current owner of **set-1** to be the same as the current record of a user-defined indicator. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The first member of the new current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. If the new current owner has no members via **set-1**, then the current member of **set-1** becomes null, the current of run unit becomes null, and the command status is 255. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct   ... E0 = SOU ("set-1,iblk")
block/indirect ... E0 = DMS ("SOU,set-1,iblk")
record/direct  ... E0 = SOU ("set-1",iblk)
record/indirect... E0 = DMSD ("SOU,set-1",iblk)
```

SUC — Set User indicator to Current of run unit — SUC

Command and Arguments

SUC,iblk

Currency Indicators

Used: CRU Changed: CU(i) ⟵ CRU

Description

Set the current record of a user-defined indicator to be the same as the current of run unit. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct   ... E0 = SUC ("iblk")
block/indirect ... E0 = DMS ("SUC,iblk")
record/direct  ... E0 = SUC (iblk)
record/indirect... E0 = DMSD ("SUC",iblk)
```

SUM — Set User indicator to Member — SUM

Command and Arguments

```
SUM,set-1,iblk
```

Currency Indicators

Used: CM(set-1) Changed: CU(i)←——CM(set-1)

Description

Set the current record of a user-defined indicator to be the same as the current member of **set-1**. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

If a user does not have read access to **set-1**, no currency indicator changes and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = SUM ("set-1,iblk")
block/indirect ... E0 = DMS ("SUM,set-1,iblk")
record/direct  ... E0 = SUM ("set-1",iblk)
record/indirect... E0 = DMSD ("SUM,set-1",iblk)
```

SUN — Set User indicator to Null — SUN

Command and Arguments

```
SUN,iblk
```

Currency Indicators

Used: none Changed: CU(i)←——null

Description

Set the current record of a user-defined indicator to be null. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct   ... E0 = SUN ("iblk")
block/indirect ... E0 = DMS ("SUN,iblk")
record/direct  ... E0 = SUN (iblk)
record/indirect... E0 = DMSD ("SUN",iblk)
```

SUO — Set User indicator to Owner — SUO

Command and Arguments

```
SUO,set-1,iblk
```

Currency Indicators

Used: CO(set-1) Changed: CU(i)←——CO(set-1)

Description

Set the current record of a user-defined indicator to be the same as the current owner of **set-1**. The value of the host language variable for **iblk** determines which one of the user-defined indicators is involved in this command. The host language variable must be consistent with an unsigned data item that is one byte in size. The variable's value must be an integer between 1 and 255, inclusive. The AUI command (Chapter XI) is employed to allocate user-defined indicators.

If a user does not have read access to **set-1**, no currency indicator changes and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = SUO ("set-1,iblk")
block/indirect ... E0 = DMS ("SUO,set-1,iblk")
record/direct  ... E0 = SUO ("set-1",iblk)
record/indirect... E0 = DMSD ("SUO,set-1",iblk)
```


SUU — Set User indicator to User indicator — SUU

Command and Arguments

SUU,iblk

Currency Indicators

Used: CU(j) Changed: CU(i) ⟵ CU(j)

Description

Two host language variables, say i and j, are defined for **iblk**. Each must be consistent with an unsigned data item that is one byte in size. Each must have an integer value between 1 and 255, inclusive. This command sets the current record of a user-defined indicator (determined by i's value) to be the same as the current record of another user-defined indicator (determined by j's value). The AUI command (Chapter XI) is employed to allocate user-defined indicators.

Examples of Command Usage

```
block/direct   ... E0 = SUU ("iblk")
block/indirect ... E0 = DMS ("SUU,iblk")
record/direct  ... E0 = SUU (iblk)
record/indirect... E0 = DMSD ("SUU",iblk)
```

VII. CREATION COMMANDS

A. Overview

There are two MDBS DML commands that can be used to create a record occurrence. One forces MDBS.DMS to create the record in a particular desired area, indicated by a command argument. However, the record cannot be forced into an area it was not assigned to in the DDL specification. This command cannot be used with a record type that has a calc key.

The other creation command allows MDBS.DMS to place the new record in any area it was assigned to in the DDL specification. If a CALC or clustering record location mode was defined for the record in the DDL, then MDBS.DMS physically positions the new record accordingly. In the case of clustering, the new member (owner) record is clustered on the basis of the current owner (current member) of the set indicated in that record type's clustering clause (as specified in the DDL).

Data to be stored in the new record is transferred from the host language variables of a data block (or program record type) in to the data base. The types, sizes and sequence of these variables must be consistent with the types, sizes and sequences of the record type's data items as specified in the DDL. The values of these variables must conform to the feasibility ranges specified for the record type's data items in the DDL specification. If they do not, the record is not created and a command status error is returned. Data compression and encryption are automatic, where applicable.

A record can be created only if the user has write access to all data items of that record type, write access to the area in which that record is to be created, and write access to all sets in which that record type participates as an automatic owner or member. In the event that there is no space left, in the area(s) in which MDBS.DMS is permitted to create an occurrence of the record type, a command status error to that effect is returned.

If a data base has multiple areas, then all areas which permit pointer indices must be on-line during record creation.

B. Command Details

CRA — Create Record in Area — CRA

Command and Arguments

CRA,rec,area,iblk

Currency Indicators

Used:	Changed: CRU←—newly created record
...if rec is AUTO member of set	...if rec is AUTO member of set
CO (auto set)	CM (auto set) ←—newly created record
...if rec is AUTO owner of set	...if rec is AUTO owner of set
CM (autoset)	CO (autoset)←— newly created record

Description

An occurrence of the record type **rec** is created in the indicated **area**. The values of the host language variables for **iblk** are stored into that record occurrence. The sequence, types, and sizes of host language variables defined for **iblk** must correspond to the sequence, types, and sizes of data items defined for **rec** in the DDL specification. The values of the variables must conform to feasibility ranges specified for the record type's data items. If they do not, a command status error is returned, the record is not created, and no currency indicators change.

The physical placement of the newly created record is consistent with the DDL specification for **rec** (either clustered or system-determined). A record with a calc key cannot be created with CRA. In order to create a record with CRA, a user must have write access to **rec**, to the **area** in which the record is to be created, and to all sets in which the record type participates as an automatic owner or member. If a user does not have this security clearance, a command status error is returned, the record is not created, and no currency indicator is changed.

If **rec** has been declared to be the AUTO member of a set, then the newly created record is automatically connected to the current owner of that set and the new record becomes the current member of that set. The connection takes place according to the member order specified with the DDL (SORTED, FIFO, etc.). If the member order for the set is NEXT, then the new record is connected (i.e., logically inserted) immediately after the set's current member. If there is no current member, the record is inserted as the first member. For PRIOR member order, the new record is logically connected immediately before the set's current member. If there is no current member, the record is inserted as the last member.

If **rec** has been declared to be the AUTO owner of a set, then the newly created record is automatically connected to the current member of that set and the new record becomes the current owner of that set. The connection occurs according to the set's owner order, as specified with the DDL (SORTED, LIFO, NEXT, etc.). If the owner order for the set is NEXT (or PRIOR), then the new record is logically connected immediately after (or before) the set's current owner. If there is no current owner, the record is inserted as the first (last) owner.

Examples of Command Usage

```
block/direct   ... E0 = CRA ("rec,area,iblk")
block/indirect ... E0 = DMS ("CRA,rec,area,iblk")
record/direct  ... E0 = CRA ("rec,area",iblk)
record/indirect... E0 = DMSD ("CRA,rec,area",iblk)
```

CRS — Create Record and Store — CRS

Command and Arguments

CRS,rec,iblk

Currency Indicators

Used:	Changed: CRU←—newly created record
...if rec is AUTO member of set CO (auto set)	...if rec is AUTO member of set CM (auto set) ←—newly created record
...if rec is AUTO owner of set CM (autoset)	...if rec is AUTO owner of set CO (autoset)←— newly created record

Description

An occurrence of the record type **rec** is created in a permissible area. The values of the host language variables for **iblk** are stored into that record occurrence. The sequence, types, and sizes of host language variables defined for **iblk** must correspond to the sequence, types, and sizes of data items defined for **rec** in the DDL specification. The values of the variables must conform to feasibility ranges specified for the record type's data items. If they do not, a command status error is returned, the record is not created, and no currency indicators change.

The physical placement of the newly created record is consistent with the DDL specification for **rec** (either CALCed, clustered, or system-determined). In order to create the record, a user must have write access to **rec**, to the **area** in which the record is to be created, and to all sets in which the record type participates

as an automatic owner or member. If a user does not have this security clearance, a command status error is returned, the record is not created, and no currency indicator is changed.

If **rec** has been declared to be the AUTO member of a set, then the newly created record is automatically connected to the current owner of that set and the new record becomes the current member of that set. The connection takes place according to the member order specified with the DDL (SORTED, FIFO, etc.). If the member order for the set is NEXT, then the new record is connected (i.e., logically inserted) immediately after the set's current member. If there is no current member, the record is inserted as the first member. For PRIOR member order, the new record is logically connected immediately before the set's current member. If there is no current member, the record is inserted as the last member.

If **rec** has been declared to be the AUTO owner of a set, then the newly created record is automatically connected to the current member of that set and the new record becomes the current owner of that set. The connection occurs according to the set's owner order, as specified with the DDL (SORTED, LIFO, NEXT, etc.). If the owner order for the set is NEXT (or PRIOR), then the new record is logically connected immediately after (or before) the set's current owner. If there is no current owner, the record is inserted as the first (last) owner.

Examples of Command Usage

```
block/direct   ... E0 = CRS ("rec,iblk")
block/indirect ... E0 = DMS ("CRS,rec,iblk")
record/direct  ... E0 = CRS ("rec",iblk)
record/indirect... E0 = DMSD ("CRS,rec",iblk)
```


VIII. CONNECT COMMANDS

A. Overview

The MDBS connect commands allow two record occurrences to be connected to each other through a set relationship. The record that is the current of run unit becomes connected to the current owner of an indicated set or the current member of an indicated set, depending on which connect command is used. When a record is connected as a member (owner), it is inserted into a set relationship on the basis of that set's member (owner) order. During record connection, all areas allowing pointer indices must be on-line.

Connection commands are used to accomplish the manual insertion of a record into a set relationship. This is typically needed in cases where a) MANUAL set insertion has been defined for a set's owner or member record type, b) AUTO set insertion has been declared in the DDL for both the owner and member record types of an N:M set, c) it is necessary to reconnect a record (perhaps, to a different owner or member record) after it has been disconnected.

In the MANUAL case, the creation of a member (owner) record does not automatically connect it to an owner (member) record of the set. If and when such a connection is desired, it is accomplished with a connect command. In the second case, suppose that a member (owner) record of an N:M set is created. It is automatically connected to one owner (member) record. If it is desired to connect it to additional owner (member) records, a connect command is used. In the third case, it is sometimes important to disconnect a member (owner) from an owner (member) and later connect that member (owner) to the same or a different owner (member). Regardless of whether there is AUTO or MANUAL set insertion, this reconnection is accomplished by a connect command.

The integrity of all set relationships is preserved. A command status error results if an attempt is made to connect a member with more than one owner in a 1:1 or 1:N set, to connect an owner with more than one member of 1:1 or N:1 set, or to connect a member with an owner to which it is already connected.

If a user does not have write access to the set involved in a connect command, the connection does not take place and a command status error to that effect is returned.

B. Command Details

IMS Insert Member into Set **IMS**

Command and Arguments

IMS,set-1

Currency Indicators

Used: CO(set-1)
CRU

Changed: CM(set-1)←——CRU

Description

The record that is current of run unit becomes connected to the current owner of **set-1**. This new member for the current owner of **set-1** is made the current member of **set-1**. The insertion of this new member record among the existing member records, connected to the current owner of **set-1**, is made in accordance with the member order declared for **set-1** in the DDL specification. If the member order is NEXT (or PRIOR) then the new member is logically inserted after (or before) the set's current member; if the set has no current member, the record is logically inserted as the first (or last) member.

If multiple owners can exist for this new member, the record that is the current owner is logically inserted into those owners according to the owner order of **set-1**. The exception is if the owner order is NEXT (or PRIOR), in which case the record that is the current owner becomes the logically first (or last) owner of the set's new current member.

The current of run unit must be an occurrence of a record type that is a member of **set-1**. A user must have write access to **set-1**, in order to connect a member record to an owner record.

Examples of Command Usage

```
block/direct   ... E0 = IMS ("set-1")
block/indirect ... E0 = DMS ("IMS,set-1")
record/direct  ... E0 = IMS ("set-1")
record/indirect... E0 = DMS ("IMS,set-1")
```

IOS — Insert Owner into Set — IOS

Command and Arguments

IOS,set-1

Currency Indicators

Used: CM(set-1)
CRU

Changed: CO(set-1) ←—— CRU

Description

The record that is current of run unit becomes connected to the current member of **set-1**. This new owner for the current member of **set-1** is made the current owner of **set-1**. The insertion of this new owner record among the existing owner records, connected to the current member of **set-1**, is made in accordance with the owner order declared for **set-1** in the DDL specification. If the owner order is NEXT (or PRIOR) then the new owner is logically inserted after (or before) the set's current owner; if the set has no current owner, the record is logically inserted as the first (or last) owner.

If multiple members can exist for this new owner, the record that is the current member is logically inserted into those members according to the member order of **set-1**. The exception is if the member order is NEXT (or PRIOR), in which case the record that is the current member becomes the logically first (or last) member of the set's new current owner.

The current of run unit must be an occurrence of a record type that is a owner of **set-1**. A user must have write access to **set-1**, in order to connect an owner record to a member record.

Examples of Command Usage

```
block/direct   ... E0 = IOS ("set-1")
block/indirect ... E0 = DMS ("IOS,set-1")
record/direct  ... E0 = IOS ("set-1")
record/indirect... E0 = DMS ("IOS,set-1")
```


This page intentionally left blank.

IX. DISCONNECT COMMANDS

A. Overview

The MDBS DML disconnect commands allow an owner record and a member record that are connected via a set to be disconnected from each other. Either the current member or all members can be disconnected from a set's current owner. Similarly, either the current owner or all owners can be disconnected from a set's current member. Removing a set connection between two occurrences does not delete those occurrences from the data base.

If a set has fixed retention, no disconnections can be made for it; in this case, a record must be deleted to be disconnected. If a user does not have write access for a set, then that user cannot perform any disconnections on that set. During disconnection all areas allowing pointer indices must be on-line.

B. Command Details

RMS Remove Member from Set **RMS**

Command and Arguments

RMS,set-1

Currency Indicators

Used:	CM(set-1)	Changed:	CM(set-1) ←—— next member
	CO(set-1)		CRU ←—— next member

Description

Remove the connection between the current owner and current member of **set-1**. The next member (following the member that was removed) of the current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. Which member is logically next depends on the member order of **set-1**, as declared in the DDL specification. If there is no next member, the current member indicator for **set-1** becomes null, the current of run unit becomes null, and the command status is 255. If a user does not have write access to **set-1**, then the current member is not disconnected, no currency indicators change, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = RMS ("set-1")
block/indirect ... E0 = DMS ("RMS,set-1")
record/direct  ... E0 = RMS ("set-1")
record/indirect... E0 = DMS ("RMS,set-1")
```


ROS — Remove Owner from Set — ROS

Command and Arguments

ROS,set-1

Currency Indicators

Used:		Changed:	
CO(set-1)		CO(set-1)	←——next owner
CM(set-1)		CRU	←——next owner

Description

Remove the connection between the current member and current owner of **set-1**. The next owner (following the owner that was removed) of the current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. Which owner is logically next depends on the owner order of **set-1**, as declared in the DDL specification. If there is no next owner, the current owner indicator for **set-1** becomes null, the current of run unit becomes null, and the command status is 255. If a user does not have write access to **set-1**, then the current owner is not disconnected, no currency indicators change, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = ROS ("set-1")
block/indirect ... E0 = DMS ("ROS,set-1")
record/direct  ... E0 = ROS ("set-1")
record/indirect... E0 = DMS ("ROS,set-1")
```

RSM — Remove all Set Members — RSM

Command and Arguments

RSM,set-1

Currency Indicators

Used:	Changed:	
CO(set-1)	CM(set-1)	←——null

Description

Remove the connection between the current owner of **set-1** and each one of its members. The current member of **set-1** becomes null. If a user does not have write access to **set-1**, then the members are not disconnected, the current member of **set-1** does not change, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = RSM ("set-1")
block/indirect ... E0 = DMS ("RSM,set-1")
record/direct  ... E0 = RSM ("set-1")
record/indirect... E0 = DMS ("RSM,set-1")
```

RSO — Remove all Set Owners — RSO

Command and Arguments

```
RSO,set-1
```

Currency Indicators

Used: CM(set-1) Changed: CO(set-1)←——null

Description

Remove the connection between the current member of **set-1** and each one of its owners. The current owner of **set-1** becomes null. If a user does not have write access to **set-1**, then the owners are not disconnected, the current owner of **set-1** does not change, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = RSO ("set-1")
block/indirect ... E0 = DMS ("RSO,set-1")
record/direct  ... E0 = RSO ("set-1")
record/indirect... E0 = DMS ("RSO,set-1")
```


This page intentionally left blank.

X. DELETION COMMANDS

A. Overview

Each of the three MDBS DML deletion commands deletes a record occurrence from the data base. A record must have been found before it can be deleted. Thus there are commands to delete the record that is the

current of run unit,
current owner of an indicated set, or
current member of an indicated set.

A record that has been remembered in a user-defined indicator can also be deleted, after first making it the current of run unit, a current owner, or a current member (with an assignment command).

When a record is deleted, MDBS.DMS disconnects it from every record that owns it and from every member record that it owns. The space occupied by that record is nulled and is returned to the pool of free space, so that it can later be reused by MDBS.DMS. Deletion commands can have extensive effects on currency indicators.

A user is not allowed to delete a record without write access to all of the record type's data items, write access to all sets in which that record type participates, and write access to the area containing the record and all areas containing records connected to the record being deleted. An attempt to violate these security constraints does not affect the record and it results in a command status error.

In the case of multiple areas, all areas that can contain pointer indices must be on-line for record deletion.

B. Command Details

DRC Delete Record that is Current of run unit **DRC**

Command and Arguments

DRC (no argument)

Currency Indicators

Used: CRU Changed: CRU←——null
also see description

Description

The record that is the current of run unit is disconnected from all set relationships and is physically deleted from the data base. The current of run unit becomes null. If the deleted record is the current owner of any set, the current member of any set, or the current record of any user-defined indicator, then those indicators become null. If a user does not have write access to the current of run unit's record type, to the record's

area and to all sets in which that record type participates, then no deletion occurs, no currency indicator changes, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = DRC (     )
block/indirect ... E0 = DMS ("DRC")
record/direct  ... E0 = DRC (     )
record/indirect... E0 = DMS ("DRC")
```

DRM — Delete Record that is Member — DRM

Command and Arguments

DRM,set-1

Currency Indicators

Used:	CO(set-1)	Changed:	CM(set-1) ←——next member
	CM(set-1)		CRU ←——next member
			also see description

Description

The record that is the current member of **set-1** is disconnected from all set relationships and is physically deleted from the data base. The next member that is owned by the current owner of **set-1** becomes the current member of **set-1** and the current of run unit. Which member is next depends on the member order of **set-1**. If there is no next member, then the current of run unit becomes null the current member of **set-1** becomes null, and the command status is 255. If the record being deleted is the current owner of any set, the current member of any set other than **set-1**, or the current record of any user-defined indicator, then those indicators become null.

If the user does not have write access to **set-1**'s member record type, to the record's area and to all sets in which that record type participates, then no deletion occurs, no currency indicator changes, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = DRM ("set-1")
block/indirect ... E0 = DMS ("DRM,set-1")
record/direct  ... E0 = DRM ("set-1")
record/indirect... E0 = DMS ("DRM,set-1")
```


DRO — Delete Record that is Owner — DRO

Command and Arguments

DRO,set-1

Currency Indicators

Used:		Changed:	
CM(set-1)		CO(set-1)	←——next owner
CO(set-1)		CRU	←——next owner
		also see description	

Description

The record that is the current owner of **set-1** is disconnected from all set relationships and is physically deleted from the data base. The next owner that is owned by the current member of **set-1** becomes the current owner of **set-1** and the current of run unit. Which owner is next depends on the owner order of **set-1**. If there is no next owner, then the current of run unit becomes null, the current owner of **set-1** becomes null, and the command status is 255. If the record being deleted is the current member of any set, the current owner of any set other than **set-1**, or the current record of any user-defined indicator, then those indicators become null.

If the user does not have write access to **set-1**'s owner record type and to all sets in which that record type participates, then no deletion occurs, no currency indicator changes, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = DRO ("set-1")
block/indirect ... E0 = DMS ("DRO,set-1")
record/direct  ... E0 = DRO ("set-1")
record/indirect... E0 = DMS ("DRO,set-1")
```


This page intentionally left blank.

XI. UTILITY COMMANDS

A. Overview

Utility commands fall into the following groups:

1. Opening and closing a database or database area
   DBCLS,DBCLSA,DBOPN,DBOPNA

2. Flushing the page buffers
   DBSAVE

3. Allocating user-defined indicators
   AUI

4. Getting the owner (member) count for a member (owner)
   GMC,GOC

5. Getting the type of a record occurrence
   GTC,GTM,GTO

6. Testing for the type of a record occurrence and checking for the equality of currency indicators
   TCT,CCU,TMT,TOT

7. Acquiring database statistics
   DBSTAT

8. Nulling all currency indicators
   NCI

9. Customizing the data base control system's behavior
   DBCNV, DBENV

B. Command Details

AUI Allocate User Indicators AUI

Command and Arguments

AUI,iblk

Currency Indicators

Used: none Changed: see description

Description

One host language variable is defined for iblk. It must be consistent with a one byte unsigned data item. The value of that variable must be between 0 and 255, inclusive. This value tells MDBS.DMS how many user-defined indicators are to be allocated. If the value is 0, then no user-defined indicators are allocated.

If the value is 12, then user indicators 1 through 12 are allocated. If the value is 139, the user indicators 1 through 139 are allocated.

AUI can be invoked repeatedly within an application program to increase or decrease the number of user-defined indicators, as desired. When a data base is opened for data manipulation (with the DBOPN command), four user-defined indicators are automatically allocated and set to null. These first two user indicators (1,2) are needed when performing a Boolean DML command. If an application program does not use a Boolean command and does not need any user indicator, the AUI can be used to allocate zero user indicators. If more than four user indicators are needed, AUI is used to allocate the desired number. When AUI is employed to increase the number of user indicators, the records that were current for already allocated user indicators are still current. The new indicators are set to null. When AUI is employed to decrease the number of user indicators, only those records that are current for the resultant user indicators are still current. For instance, if ten user indicators are allocated and AUI is used to reduce this to five, then only those records that were current for user indicators 1 through 5 are still current after invoking AUI. Knowledge about which records were current for user indicators 6 through 10 is not maintained.

Examples of Command Usage

```
block/direct   ... E0 = AUI ("iblk")
block/indirect ... E0 = DMS ("AUI,iblk")
record/direct  ... E0 = AUI (iblk)
record/indirect... E0 = DMSD ("AUI",iblk)
```

CCU — Check Current of run unit against User indicator — CCU

Command and Arguments

CCU,iblk

Currency Indicators

Used: CRU
CU(i)

Changed: none

Description

Check to determine whether the record that is current for a user-defined indicator is the same record that is the current of run unit. The value of **iblk**'s host language variable determines the user indicator for which this check is made. That variable must be consistent with a one byte, unsigned data item. If the current of run unit is the same record as the specified user indicator, then the command status is 0. If it is not, then the command status is 255.

Examples of Command Usage

```
block/direct   ... E0 = CCU ("iblk")
block/indirect ... E0 = DMS ("CCU,iblk")
record/direct  ... E0 = CCU (iblk)
record/indirect... E0 = DMSD ("CCU",iblk)
```

DBCLS — Data Base CLoSe — DBCLS

Command and Arguments

DBCLS (no arguments)

Currency Indicators

Used: none Changed: all←——null

Description

In a single-user environment, DBCLS should be the last DML command invoked in an application program. DBCLS nulls all of that program's currency indicators, eliminates all of the program's user indicators, flushes the page buffers (i.e., rewrites all pages, that have changed since entering main memory, into auxiliary memory), and performs a TRCOM (for the RTL form of the system). No further data manipulation can be performed after DBCLS, until the data base is reopened (with DBOPN).

Failure to close a data base can leave its contents inconsistent, if the data base was opened for modification. This inconsistency will prevent subsequent efforts to open the data base. The data base can be restored to a consistent state with the MDBS RCV utility.

In a multi-user environment, many application programs share the same page buffers. Invoking DBCLS within an application program nulls out all of that program's currency indicators and eliminates all of the program's user indicators. It will also flush the page buffers if no other application program currently has the data base open for processing. In order to flush the page buffers, irrespective of whether other programs have the data base open for processing, the DBSAVE command should be used. An application program can perform no further data manipulation after it has invoked DBCLS, until it reopens the data base (with DBOPN).

Examples and Command Usage

```
block/direct   ... E0 = DBCLS (     )
block/indirect ... E0 = DMS ("DBCLS")
record/direct  ... E0 = DBCLS (     )
record/indirect... E0 = DMS ("DBCLS")
```

DBCLSA — Data Base CLoSe for Area — DBCLSA

Command and Arguments

DBCLSA,area

Currency Indicators

Used: none Changed: none

Description

DBCLSA flushes all page buffers that contain pages from the indicated **area**. "Flush" means to rewrite pages, that have changed since entering main memory, into auxiliary memory. A user must have either read or write access to the indicated **area** in order to use this command. The file on which the area resides is closed by the operating system.

Examples of Command Usage

```
block/direct   ... E0 = DBCLSA ("area")
block/indirect ... E0 = DMS ("DBCLSA,area")
record/direct  ... E0 = DBCLSA ("area")
record/indirect... E0 = DMS ("DBCLSA,area")
```

DBCNV — Data Base format CoNVersion — DBCNV

Command and Arguments

DBCNV,iblk

Currency Indicators

Used: none Changed: none

Description

A run unit can use the DBCNV command to cause the data base control system to automatically convert date values into an alternative format as they are transferred between that run unit and a data base. As a value is transferred to a data base, it is converted from the alternative format to the standard MDBS format (mm/dd/yyyy for date, prior to compression). Conversely, as a value is transferred to the run unit, it is converted from the standard MDBS format to the alternative format.

The desired alternative format is indicated by the value of **iblk**'s host language variable. This variable must be consistent with a one byte, unsigned data item. The permissible **iblk** variable values and their corresponding formats are:

1	mm/dd/yyyy (the standard MDBS date format)
2	dd/mm/yyyy
3	yyyy/mm/dd
4	yyyy/dd/mm

DBCNV can be invoked multiple times within a run unit to switch between various date formats.

DBCNV can also be used to allow (or disallow) null date or time values. The standard MDBS default condition is that null dates and null times are not allowed. Additional permissible **iblk** variable values and their effects are as follows:

5	null dates are allowed on input
6	null dates are not allowed on input (default condition)
7	null times are allowed on input
8	null times are not allowed on input (default condition)

Here, "input" refers to any attempt to create or alter the value of a date or time data item. If a null date or time is involved in a sorted relationship, null values are sequenced after non-null values.

Examples of Command Usage

```
block/direct     ... E0 = DBCNV ("iblk")
block/indirect   ... E0 = DMS ("DBCNV,iblk")
record/direct    ... E0 = DBCNV (iblk)
record/indirect  ... E0 = DMSD ("DBCNV",iblk)
```

DBENV — Data Base ENVironment options — DBENV

Command and Arguments

DBENV,iblk

Currency Indicators

Used: none Changed: none

Description

A run unit can use this command to customize its processing environment by specifying how the data base control system should behave when interacting with that run unit. The value of **iblk**'s host language variable governs which aspect of the environment is altered. If the value is 1, 2 or 3 then MDBS.DMS behavior is altered for all run units for as long as any of them still has the data base open. The host language variable for **iblk** must be consistent with a one byte unsigned data item. Since option 1 alleviates some of the checking that is otherwise performed by MDBS.DMS, it can increase the processing speed of an application program that modifies a data base. This option is typically used in an application program that is known to be "bug" free.

If the variable has a value of 1, then MDBS.DMS does not perform a check to determine whether a record being connected to an owner record for a set is already owned by that owner record via that set.

If the variable has a value of 2, then checksums on disk reads are not suppressed. If the variable has a value of 3, the MDBS.DMS automatically performs a DBSAVE whenever a DML command causes a change to a page(s) in memory. Although these two options provide additional integrity safeguards, they will result in slower processing.

If **iblk**'s variable has a value of 4, then no passive lock checking will be performed for the run unit that invokes DBENV. Though this option results in faster processing of the run unit's DML commands, it should be used only with extreme caution because it eliminates a significant portion of the built-in integrity checking. When its passive lock checking is disabled, the run unit will be allowed to alter records that are pasively locked by other run units. It will not be allowed to actively lock a record that is passively locked by others. The usual record deletion and set removal integrity restrictions are still enforced (see Table XIV-1).

If **iblk**'s variable has a value of 5, the run unit's attempts to find a record based on a sort key (e.g., with FMSK) will be processed in a slightly different way than usual. If a record with an indicated sort key value cannot be found (because it does not exist), then the pertinent currency indicators (CRU and either CO or CM) become null and the command status is 255.

If the variable has a value of 6, the automatic FFM or FFO that is normally carried out for assignment commands (e.g., SOM) is not performed. In some situations this allows the run unit's control logic to be more concise.

After DBENV has been invoked to alter some aspect of the environment, that aspect of the environment can be restored to its default behavior. This is accomplished by invoking DBENV again, using a variable value that is 100 greater than what was previously used. For instance, if **iblk**'s variable has a value of 105, then the effect of formerly invoking DBENV with a value of 5 is no longer in force. If DBENV is invoked with a value of 101, 102, or 103 then the normal MDBS.DMS behavior is restored for all run units.

Examples of Command Usage

```
block/direct   ... E0 = DBENV ("iblk")
block/indirect ... E0 = DMS ("DBENV,iblk")
record/direct  ... E0 = DBENV (iblk)
record/indirect... E0 = DMSD ("DBENV",iblk)
```


DBOPN — Data Base OPeN — DBOPN

Command and Arguments

DBOPN,iblk

Currency Indicators

Used: none

Changed:
CO(system-owned set) ←—system
CO(other sets) ←—null
CM(all sets) ←—null
CRU ←—system
CU(1),CU(2),CU(3),CU(4) ←—null

Description

DBOPN opens a data base for processing by an application program. An application program must invoke the DBOPN command before any data held in the data base can be accessed. There are four host language variables for the **iblk** data block (or program record type), which MDBS.DMS expects in the following sequence:

1. The first variable's value is a user's name that has been specified in the DDL. This variable must be consistent with a string data item of no more than 16 bytes.
2. The second variable's value is the user's password as specified in the DDL. This variable must be consistent with a string data item of no more than 12 bytes.
3. The third variable's value indicates the type of processing that the program will perform. This variable must be consistent with a string data item of no more than 4 bytes. If the variable's value begins with R, the program is allowed to perform retrieval only (i.e., read access only, regardless of a user's write access codes). If it begins with M, the program can modify (as well as retrieve) data. If the variable's value begins with E, the run unit has exclusive modification and retrieval access to a data base, subject to security constraints; in a multiuser environment this prohibits any other application program from accessing a data base, until the program with exclusive access invokes DBCLS. Since it is not sharing the data base with other run units, a run unit with exclusive access will execute faster than if it had R or M access.
4. The fourth variable's value is the fully qualified file name for the main **area** of the data base being opened. This variable must be consistent with a string data item; the maximum length is operating system dependent. It cannot exceed the length of a fully qualified file name permitted in the operating system (consult the appropriate system specific manual).

In block-oriented languages, the **iblk** data block must be the last data block defined (see DEFINE command) before invoking DBOPN. This block and all other data blocks defined before invoking DBOPN are inoperative after invoking DBOPN. (They must be defined again, after DBOPN, in order to be used after DBOPN.)

Upon opening a data base for processing the current of run unit and the current owners of all system-owned sets become the SYSTEM record occurrence. All other currency indicators are null. Four user-defined currency indicators are automatically allocated. If any page images exist on the page-image file, they are used to automatically restore data base consistency. The page-image file must have been delcared with the PIFD command (Chapter XV) prior to invoking DBOPN. For the RTL form, a command status warning of 74 is returned if no log file is present; this does not prevent the data base from being opened.

Although DBOPN opens only the main **area** of the data base, other **areas** are automatically opened by MDBS.DMS when the application program needs to access records in those **areas**. These are opened for the same type of processing as the main **area**. A command status error results if an **area** that needs to be automatically opened is not on-line. A command status error results if the user does not have read access or write access to the **area** that needs to be opened.

Examples and Command Usage

```
block/direct   ... E0 = DBOPN ("iblk")
block/indirect ... E0 = CALL DMS ("DBOPN,iblk")
record/direct  ... E0 = DBOPN (iblk)
record/indirect... E0 = CALL DMSD ("DBOPN",iblk)
```

DBOPNA Data Base OPeN Area **DBOPNA**

Command and Arguments

DBOPNA,area,iblk

Currency Indicators

Used: none Changed: none

Description

DBOPNA explicitly opens the indicated **area** for processing. MDBS.DMS expects to find this **area** on the file indicated by the value of **iblk**'s host language variable. This variable must be consistent with a string data item; its size is operating system dependent, but cannot exceed the maximum length of a fully qualified file name allowed by the host operating system (consult the pertinent system specific manual). The value of the variable must be the fully qualified name of a file that is on-line when DBOPNA is invoked. This name does not need to match the file name assigned to the **area** in the DDL specifications. If the indicated file is not on-line, a command status of 54 is returned. This fact can be used by the application developer to determine whether an end user of the application program has the appropriate disks on-line; if they are not, a corresponding prompt can be issued to the end user.

If **iblk**'s host language variable has a null value, then MDBS.DMS assumes that the **area** exists on the file specified for that **area** in the DDL specifications (or its default name, if none is specified).

Examples and Command Usage

```
block/direct   ...  E0 = DBOPNA ("area,iblk")
                         or E0 = DBOPNA ("area")
block/indirect ... E0 = DMS ("DBOPNA,area,iblk")
                         or E0 = DMS ("DBOPNA,area")
record/direct  ... E0 = DBOPNA ("area",iblk)
record/indirect... E0 = CALL DMSD ("DBOPNA,area",iblk)
```

DBSAVE Data Base SAVE **DBSAVE**

Command and Arguments

DBSAVE (no arguments)

Currency Indicators

Used: none Changed: none

Description

DBSAVE flushes the page buffers, regardless of whether the run unit is executing in a single-user or multiuser environment. This saves all changes that have been made to pages (in the page buffers) since they entered main memory. These changed pages are saved by rewriting them into the data base that exists in auxiliary memory. It is advisable to invoke DBSAVE after extremely crucial transactions that change data base contents. This guarantees that those transactions are immediately incorporated into the permanent copy of a data base. In the event of an abnormal interruption of a run unit, the following rule holds. If no changes have been made to the data base since the last DBSAVE, then the data base is consistent and can be re-opened

successfully (i.e., without a command status error of 15). In a multiuser environment, the data base control system may automatically invoke DBSAVE many times, independently of any executing run unit. Thus an abnormal interruption may still leave the data base in a consistent state so that it can be successfully re-opened.

Examples of Command Usage

```
block/direct    ... E0 = DBSAVE (     )
block/indirect  ... E0 = DMS ("DBSAVE")
record/direct   ... E0 = DBSAVE (     )
record/indirect... E0 = DMS ("DBSAVE")
```

DBSTAT Data Base STATistics **DBSTAT**

Command and Arguments

DBSTAT,oblk

Currency Indicators

Used: none Changed: none

Description

Statistics on data base utilization are returned as values of **oblk**'s five host langugage variables. Each of these variables must be consistent with a two byte, unsigned data item. The sequence of the five returned statistics is as follows:

1. the number of page buffers currently allocated in main memory,
2. the number of times (since opening the data base) that the most recent page access was to a different page than the last page access,
3. the number of read requests issued by MDBS.DMS since opening the data base,
4. the number of write requests issued since the data base was opened that were due to DBSAVE or to background processing in a multiuser environment,
5. the total number of write requests issued by MDBS.DMS since opening the data base.

Examples of Command Usage

```
block/direct    ... E0 = DBSTAT ("oblk")
block/indirect  ... E0 = DMS ("DBSTAT,oblk")
record/direct   ... E0 = DBSTAT (oblk)
record/indirect... E0 = DMSD ("DBSTAT",oblk)
```


GMC — Get Member Count — GMC

Command and Arguments

GMC,set-1,oblk

Currency Indicators

Used: CO(set-1) Changed: none

Description

The number of member records connected to the current owner of **set-1** is returned in the host language variable defined for **oblk.** This variable must be consistent with a four byte, unsigned data item (see system specific manuals). If a user does not have read access to **set-1**, the member count is not returned and a command status error is issued.

Examples of Command Usage

```
block/direct    ... E0 = GMC ("set-1,oblk")
block/indirect  ... E0 = DMS ("GMC,set-1,oblk")
record/direct   ... E0 = GMC ("set-1",oblk)
record/indirect... E0 = DMSD ("GMC,set-1",oblk)
```

GOC — Get Owner Count — GOC

Command and Arguments

GOC,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: none

Description

The number of owner records connected to the current member of **set-1** is returned in the host language variable defined for **oblk.** This variable must be consistent with a four byte, unsigned data item (see system specific manuals). If a user does not have read access to **set-1**, the owner count is not returned and a command status error is issued.

Examples of Command Usage

```
block/direct    ... E0 = GOC ("set-1,oblk")
block/indirect  ... E0 = DMS ("GOC,set-1,oblk")
record/direct   ... E0 = GOC ("set-1",oblk)
record/indirect... E0 = DMSD ("GOC,set-1",oblk)
```


GTC — Get Type of current of run unit — GTC

Command and Arguments

GTC,oblk

Currency Indicators

Used: CRU Changed: none

Description

The name of the record type for the record that is current of run unit is returned in **oblk**'s host language variable. That variable must be consistent with an eight byte, character data item.

Examples of Command Usage

```
block/direct   ... E0 = GTC ("oblk")
block/indirect ... E0 = DMS ("GTC,oblk")
record/direct  ... E0 = GTC (oblk)
record/indirect... E0 = DMSD ("GTC",oblk)
```

GTM — Get Type of Member — GTM

Command and Arguments

GTM,set-1,oblk

Currency Indicators

Used: CM(set-1) Changed: none

Description

The name of the record type for the record that is the current member of **set-1** is returned in **oblk**'s host language variable. That variable must be consistent with an eight byte, character data item. If a user does not have read access to **set-1**, a command status error is returned and the value of **oblk**'s host language variable is unchanged.

Examples of Command Usage

```
block/direct   ... E0 = GTM ("set-1,oblk")
block/indirect ... E0 = DMS ("GTM,set-1,oblk")
record/direct  ... E0 = GTM ("set-1",oblk)
record/indirect... E0 = DMSD ("GTM,set-1",oblk)
```

GTO — Get Type of Owner — GTO

Command and Arguments

GTO,set-1,oblk

Currency Indicators

Used: CO(set-1) Changed: none

Description

The name of the record type for the record that is the current owner of **set-1** is returned in **oblk**'s host language variable. That variable must be consistent with an eight byte, character data item. If a user does not have read access to **set-1**, a command status error is returned and the value of **oblk**'s host language variable is unchanged.

Examples of Command Usage

```
block/direct   ... E0 = GTO ("set-1,oblk")
block/indirect ... E0 = DMS ("GTO,set-1,oblk")
record/direct  ... E0 = GTO ("set-1",oblk)
record/indirect... E0 = DMSD ("GTO,set-1"oblk)
```

NCI — Null all Currency Indicators — NCI

Command and Arguments

NCI (no arguments)

Currency Indicators

Used: none Changed: all currency indicators null

Description

All currency indicators are made null, with the exceptions of current owners of system-owned sets. The active lock statuses are not affected. Thus, if MCP is in effect, NCI nulls the current of run unit; but as soon as the current of run unit again becomes non-null, its record is actively locked.

Examples of Command Usage

```
block/direct   ... E0 = NCI (     )
block/indirect ... E0 = DMS ("NCI")
record/direct  ... E0 = NCI (     )
record/indirect... E0 = DMS ("NCI")
```

TCN — Test Current of run unit for Null — TCN

Command and Arguments

TCN (no arguments)

Currency Indicators

Used: CRU Changed: none

Description

A command status of 255 is returned if the current of run unit is null. If it is not null, the command status is 0.

Examples of Command Usage

```
block/direct     ... E0 = TCN (     )
block/indirect   ... E0 = DMS ("TCN")
record/direct    ... E0 = TCN (     )
record/indirect  ... E0 = DMS ("TCN")
```

TCT — Test Current of run unit Type — TCT

Command and Arguments

TCT,rec

Currency Indicators

Used: CRU Changed: none

Description

The record type of the current of run unit is compared to rec. If they are the same, the command status is 0. If they are not the same, the command status is 3.

Examples of Command Usage

```
block/direct   ... E0 = TCT ("rec")
block/indirect ... E0 = DMS ("TCT,rec")
record/direct  ... E0 = TCT ("rec")
record/indirect... E0 = DMS ("TCT,rec")
```


TMN — Test Member for Null

Command and Arguments

TMN,set-1

Currency Indicators

Used: CM(set-1) Changed: none

Description

If the current member of **set-1** is null, a command status of 255 is returned. If it is not null, the command status is 0.

Examples of Command Usage

```
block/direct    ... E0 = TMN ("set-1")
block/indirect  ... E0 = DMS ("TMN,set-1")
record/direct   ... E0 = TMN ("set-1")
record/indirect ... E0 = DMS ("TMN,set-1")
```

TMT — Test Member Type

Command and Arguments

TMT,rec,set-1

Currency Indicators

Used: CM(set-1) Changed: none

Description

The record type of the current member of **set-1** is compared to **rec.** If they are the same, the command status is 0. If they are not the same, the command status is 3.

Examples of Command Usage

```
block/direct   ... E0 = TMT ("rec,set-1")
block/indirect ... E0 = DMS ("TMT,rec,set-1")
record/direct  ... E0 = TMT ("rec,set-1")
record/indirect... E0 = DMS ("TMT,rec,set-1")
```


TON — Test Owner for Null — TON

Command and Arguments

TON,set-1

Currency Indicators

Used: CM(set-1) Changed: none

Description

If the current owner of **set-1** is null, a command status of 255 is returned. If it is not null, the command status is 0.

Examples of Command Usage

```
block/direct    ... E0 = TON ("set-1")
block/indirect  ... E0 = DMS ("TON,set-1")
record/direct   ... E0 = TON ("set-1")
record/indirect ... E0 = DMS ("TON,set-1")
```

TOT — Test Owner Type — TOT

Command and Arguments

TOT,rec,set-1

Currency Indicators

Used: CO(set-1) Changed: none

Description

The record type of the current owner of **set-1** is compared to **rec.** If they are the same, the command status is 0. If they are not the same, the command status is 3.

Examples of Command Usage

```
block/direct   ... E0 = TOT ("rec,set-1")
block/indirect ... E0 = DMS ("TOT,rec,set-1")
record/direct  ... E0 = TOT ("rec,set-1")
record/indirect... E0 = DMS ("TOT,rec,set-1")
```


TUN — Test User indicator for Null

Command and Arguments

TUN,iblk

Currency Indicators

Used: CU(i) Changed: none

Description

The value of **iblk**'s host language variable determines the user indicator which is tested for being null. That variable must be consistent with a one byte, unsigned data item. A 255 command status is returned if the indicated user indicator is null. If it is not null, then a 0 command status is returned.

Examples of Command Usage

```
block/direct   ... E0 = TUN ("iblk")
block/indirect ... E0 = DMS ("TUN,iblk")
record/direct  ... E0 = TUN (iblk)
record/indirect... E0 = DMS ("TUN",iblk)
```


This page intentionally left blank.

XII. BOOLEAN COMMANDS

A. Overview

MDBS DML supports the Boolean intersect (i.e., logical AND) and difference operators on sets. Each Boolean DML command has three sets as arguments. The first two sets are used to specify inputs to the Boolean operation. The third set is a system-owned set whose member record occurrences, after the execution of a Boolean command, are the result of the Boolean operation. This system-owned set must be either $SYSSET or a set that owns all record types whose occurrences could be a result of the Boolean operation. The first two set names may or may not be the same ($SYSSET can be used as an input set). The third set can be the same as either of the first two sets. The user of a Boolean command must have read access to the first two sets and write access to the third set.

Each Boolean command utilizes the first two user-defined indicators. These must have been allocated prior to invoking the Boolean command. They are automatically allocated by the DBOPN command and can be reallocated by the AUI command, as needed.

There are four intersect commands. Take the intersection of

1. the members of the owner of the first set (as indicated by the first user indicator) and the members of the owner of the second set (as indicated by the second user indicator): AMM:,

2. the members of the owner of the first set (as indicated by the first user indicator) and the owners of the member of the second set (as indicated by the second user indicator): AMO,

3. the owners of the member of the first set (as indicated by the first user indicator) and the members of the owner of the second set (as indicated by the second user indicator): AOM,

4. the owners of the member of the first set (as indicated by the first user indicator) and the owners of the member of the second set (as indicated by the second user indicator): AOO.

Similarly, there are four difference commands. Exclude from

1. the members of the owner of the first set (as indicated by the first user indicator) those records that are also members of the owner of the second set (as indicated by the second user indicator): XMM,

2. the members of the owner of the first set (as indicated by the first user indicator) those records that are also owners of the member of the second set (as indicated by the second user indicator): XMO,

3. the owners of the member of the first set (as indicated by the first user indicator) those records that are also members of the owner of the second set (as indicated by the second user indicator): XOM,

4. the owners of the member of the first set (as indicated by the first user indicator) those records that are also owners of the member of the second set (as indicated by the second user indicator): XOO.

In the case of multiple areas, all areas allowing pointer arrays must be on-line during the execution of Boolean commands.

B. Command Details

AMM And of Members with Members AMM

Command and Arguments

AMM,set-1,set-2,set-3

Currency Indicators

Used:	CU(1)	Changed:	CM(set-3) ←——null
	CU(2)		CRU ←——null

Description

Take the intersection (logical AND) of the members of the owner of **set-1** (denoted by the first user indicator) and the members of the owner of **set-2** (denoted by the second user indicator). All records in the intersection are made members of **set-3**. Any member of **set-3** that is not in the intersection is disconnected from **set-3** during the execution of AMM. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after AMM is a function of the member order of **set-1** and the member order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use AMM unless the member record types of **set-1** and the member record types of **set-2** have at least one record type in common. Otherwise, the intersection will be empty. If **set-3** is not $SYSSET, then it must own each record type that is both a member of **set-1** and a member of **set-2**. If the intersection is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = AMM ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("AMM,set-1,set-2,set-3")
record/direct   ... E0 = AMM ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("AMM,set-1,set-2,set-3")
```


AMO And of Members with Owners AMO

Command and Arguments

AMO,set-1,set-2,set-3

Currency Indicators

Used:	CU(1)	Changed:	CM(set-3) ←—— null
	CU(2)		CRU ←—— null

Description

Take the intersection (logical AND) of the members of the owner of **set-1** (denoted by the first user indicator) and the owners of the member of **set-2** (denoted by the second user indicator). All records in the intersection are made members of **set-3**. Any member of **set-3** that is not in the intersection is disconnected from **set-3** during the execution of AMO. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after AMO is a function of the member order of **set-1** and the owner order and of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use AMO unless the member record types of **set-1** and the owner record types of **set-2** have at least one record type in common. Otherwise, the intersection will be empty. If **set-3** is not $SYSSET, then it must own each record type that is both a member of **set-1** and an owner of **set-2**. If the intersection is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = AMO ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("AMO,set-1,set-2,set-3")
record/direct   ... E0 = AMO ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("AMO,set-1,set-2,set-3")
```

AOM And of Owners with Members AOM

Command and Arguments

AOM,set-1,set-2,set-3

Currency Indicators

Used:	CU(1)	Changed:	CM(set-3) ←—— null
	CU(2)		CRU ←—— null

Description

Take the intersection (logical AND) of the owners of the member of **set-1** (denoted by the first user indicator) and the members of the owner of **set-2** (denoted by the second user indicator). All records in the intersection are made members of **set-3**. Any member of **set-3** that is not in the intersection is disconnected from **set-3** during the execution of AOM. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after AOM is a function of the owner order of **set-1** and the member order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use AOM unless the owner record types of **set-1** and the member record types of **set-2** have at least one record type in common. Otherwise, the intersection will be empty. If **set-3** is not $SYSSET, then it must own each record type that is both an owner of **set-1** and a member of **set-2**. If the intersection is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = AOM ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("AOM,set-1,set-2,set-3")
record/direct   ... E0 = AOM ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("AOM,set-1,set-2,set-3")
```

AOO — And of Owners with Owners — AOO

Command and Arguments

```
AOO,set-1,set-2,set-3
```

Currency Indicators

Used:		Changed:	
CU(1)		CM(set-3)	←—— null
CU(2)		CRU	←—— null

Description

Take the intersection (logical AND) of the owners of the member of **set-1** (denoted by the first user indicator) and the owners of the member of **set-2** (denoted by the second user indicator). All records in the intersection are made members of **set-3**. Any member of **set-3** that is not in the intersection is disconnected from **set-3** during the execution of AOO. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after AOO is a function of the owner order of **set-1** and the owner order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use AOO unless the owner record types of **set-1** and the owner record types of **set-2** have at least one record type in common. Otherwise, the intersection will be empty. If **set-3** is not $SYSSET, then it must own each record type that is both an owner of **set-1** and an owner of **set-2**. If the intersection is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = AOO ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("AOO,set-1,set-2,set-3")
record/direct   ... E0 = AOO ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("AOO,set-1,set-2,set-3")
```

XMM eXclude Members from Members XMM

Command and Arguments

XMM,set-1,set-2,set-3

Currency Indicators

Used:		Changed:	
CU(1)		CM(set-3)	←——null
CU(2)		CRU	←——null

Description

Exclude (logical difference) from the members of the owner of **set-1** (denoted by the first user indicator) those records that are also members of the owner of **set-2** (denoted by the second indicator). All records in the difference are made members of **set-3**. Any member of **set-3** that is not in the logical difference is disconnected from **set-3** during the execution of XMM. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after XMM is a function of the member order of **set-1** and the member order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use XMM unless the member record types of **set-1** and the member record types of **set-2** have at least one record type in common. If **set-3** is not $SYSSET, then it must own each record type that is both a member of **set-1** and a member of **set-2**. If the difference is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = XMM ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("XMM,set-1,set-2,set-3")
record/direct   ... E0 = XMM ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("XMM,set-1,set-2,set-3")
```

XMO eXclude Members from Owners XMO

Command and Arguments

XMO,set-1,set-2,set-3

Currency Indicators

Used:	CU(1)	Changed:	CM(set-3) ←—— null
	CU(2)		CRU ←—— null

Description

Exclude (logical difference) from the members of the owner of **set-1** (denoted by the first user indicator) those records that are also owners of the member of **set-2** (denoted by the second indicator). All records in the difference are made members of **set-3**. Any member of **set-3** that is not in the logical difference is disconnected from **set-3** during the execution of XMO. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after XMO is a function of the member order of **set-1** and the owner order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use XMO unless the member record types of **set-1** and the owner record types of **set-2** have at least one record type in common. If **set-3** is not $SYSSET, then it must own each record type that is both a member of **set-1** and an owner of **set-2**. If the difference is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = XMO ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("XMO,set-1,set-2,set-3")
record/direct   ... E0 = XMO ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("XMO,set-1,set-2,set-3")
```


XOM eXclude Owners from Members XOM

Command and Arguments

XOM,set-1,set-2,set-3

Currency Indicators

Used:		Changed:	
CU(1)		CM(set-3)	←—— null
CU(2)		CRU	←—— null

Description

Exclude (logical difference) from the owners of the member of **set-1** (denoted by the first user indicator) those records that are also members of the owner of **set-2** (denoted by the second indicator). All records in the difference are made members of **set-3**. Any member of **set-3** that is not in the logical difference is disconnected from **set-3** during the execution of XOM. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after XOM is a function of the owner order of **set-1** and the member order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use XOM unless the owner record types of **set-1** and the member record types of **set-2** have at least one record type in common. If **set-3** is not $SYSSET, then it must own each record type that is both an owner of **set-1** and a member of **set-2**. If the difference is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = XOM ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("XOM,set-1,set-2,set-3")
record/direct   ... E0 = XOM ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("XOM,set-1,set-2,set-3")
```


XOO eXclude Owners from Owners XOO

Command and Arguments

XOO,set-1,set-2,set-3

Currency Indicators

Used:	CU(1)	Changed:	CM(set-3) ←—— null
	CU(2)		CRU ←—— null

Description

Exclude (logical difference) from the owners of the member of **set-1** (denoted by the first user indicator) those records that are also owners of the member of **set-2** (denoted by the second indicator). All records in the difference are made members of **set-3**. Any member of **set-3** that is not in the logical difference is disconnected from **set-3** during the execution of XOO. The current member of **set-3** and the current of run unit become null and the command status becomes 255. If the member order of **set-3** is not sorted, then the ordering of members of **set-3** after XOO is a function of the owner order of **set-1** and the owner order of **set-2**. $SYSSET has FIFO ordering.

It is not reasonable to use XOO unless the owner record types of **set-1** and the owner record types of **set-2** have at least one record type in common. If **set-3** is not $SYSSET, then it must own each record type that is both an owner of **set-1** and an owner of **set-2**. If the difference is empty, then **set-3** will have no members.

Examples and Command Usage

```
block/direct    ... E0 = XOO ("set-1,set-2,set-3")
block/indirect  ... E0 = DMS ("XOO,set-1,set-2,set-3")
record/direct   ... E0 = XOO ("set-1,set-2,set-3")
record/indirect... E0 = DMS ("XOO,set-1,set-2,set-3")
```


XIII. SPECIAL COMMANDS

A. Overview

There are several MDBS DML commands that are used only within certain host languages or environments. These are referred to as special commands. For instance, there are some commands that are used only in block-oriented host languages in order to define, extend, and undefine data blocks. Another special command is employed in languages that do not allow DML commands to be used as functions that take on a command status value. This command is used to set up the host language variable that will hold the command status. Yet another special command is used to alter the end-of-set command status number (255). For instance, with FORTRAN, 255 can be altered to -1; this is useful in branching structures involving an arithmetic IF statement.

A very important special command, used with nearly every host language, is SETPBF. It is used at the start of a program to set up the program buffers that MDBS.DMS will utilize for that run unit. In a single user environment, the program buffers are used as the page buffer region. It is advisable to allocate as much space as possible for page buffers. In a multiuser environment, the page buffer region is shared by many users and is therefore not allocated by an individual run unit. Each run unit may however, have its own program buffers that are used by MDBS.DMS when interacting with the run unit. A run unit uses SETPBF to allocate its program buffers. In some multiuser environments there is yet another special DML command that allows each run unit to control its own throughput priority level.

The system specific manual for a host language and environment indicates which of these special commands are available in that setting.

B. Command Details

ALTEOS ALTer End Of Set **ALTEOS**

Command and Arguments

ALTEOS (no arguments)

Currency Indicators

Used: none Changed: none

Description

The command status value of 255 that is normally used to indicate an end-of-set is altered within the scope of the application program that uses ALTEOS. If ALTEOS is permitted within a given host language, the system specific manual for that language indicates the altered end-of-set command status value (usually -1).

Examples of Command Usage

```
block/direct   ... E0 = ALTEOS (     )
block/indirect ... E0 = DMS ("ALTEOS")
record/direct  ... E0 = ALTEOS (     )
record/indirect... E0 = DMS ("ALTEOS")
```

DBINIT Data Base control system INITialization DBINIT

Command and Arguments

DBINIT,host language variables

Currency Indicators

Used: none . Changed: none

Description

When DBINIT is available, it is invoked as the first DML command of a run unit. Two (or more) host language variables or constants are used as arguments. The nature of these arguments and the precise actions performed by DBINIT are operating system specific. See the appropriate system specific manual. In general, DBINIT initializes the data base control system(s) to be used by a run unit.

In run units that support DBINIT, a single run unit can use DBINIT to have multiple MDBS data bases open simultaneously.

Examples of Command Usage

See system specific manuals.

DBSEL Data Base SELection DBSEL

Command and Arguments

DBSEL,host language variable

Currency Indicators

Used: none Changed: none

Description

Where DBSEL is available, it can be used by a run unit to select any one of the concurrently open MDBS data bases for processing. The host language variable (or constant) is used to indicate which of the data bases is to be processed.

Examples of Command Usage

See system specific manuals.

DEFINE DEFINE data block DEFINE

Command and Arguments

DEFINE,blk,list

Currency Indicators

Used: none Changed: none

Description

This command is used with host languages that do not support program record types. It defines the **blk** data block to have the characteristics indicated in **list**. The composition of **list** is host language dependent. It typically consists of at least a list of the host language variables that make up the data block. Full details for each host language appear in the system specific manuals.

A data block must have been defined with the DEFINE command before it can be used as an argument in any other DML command. A given host language variable can appear more than once in the same or different data blocks. Many different data blocks can be used in a host language program. Only the last data block defined before using DBOPN will exist after invoking DBOPN. Any data block defined after the data base is opened will continue to exist until it is undefined (with the UNDEF command).

Examples of Command Usage

```
block/direct   ... E0 = DEFINE ("blk",list)
block/indirect ... E0 = DMSD ("DEFINE,blk",list)
```


DMSSJP — DMS Set JumP — DMSSJP

Command and Arguments

DMSSJP,host language jump description, command status array

Currency Indicators

Used: none Changed: none

Description

This special command pertains to C host languages and provides a concise error trapping mechanism. The indicated host language jump description is a pointer to a stack environment. The values of the indicated array specify which command status numbers are to be exempted from the error trapping. The last element of this array must have a value of 0.

When DMSSJP is invoked, its first argument indicates the environment to which control is transferred whenever a command status other than those specified in the array occurs. This condition remains in force until DMSSJP is re-invoked with a different argument (indicating a different error handling module or a different set of exempt command status numbers). If the first argument is 0, then standard error handling is resumed. If the second argument is 0 rather than an array, then all command status numbers other than 0 and 255 are trapped.

Example of Command Usage

See system specific manuals.

EXTEND — EXTEND data block — EXTEND

Command and Arguments

EXTEND,blk,list

Currency Indicators

Used: none Changed: none

Description

This command is used with host languages that do not support program record types. It is used to extend the previously defined blk data block by assigning more host language variables to it. These additional variables are indicated in list and they will follow the variables already defined for the data block. The composition of list is host language dependent. It typically consists of at least a list of the additional variables. Full details for each host language appear in the system specific manuals.

The **blk** data block must have been defined with the DEFINE command before it can be used in an EXTEND command (otherwise a command status of 7 is returned). A given host language variable can appear more than once in the same or different data blocks.

Examples of Command Usage

```
block/direct   ... E0 = EXTEND ("blk",list)
block/indirect ... E0 = DMSD ("EXTEND,blk",list)
```

MPL — Multiuser Priority Level — MPL

Command and Arguments

MPL,iblk

Currency Indicators

Used: none Changed: none

Description

The MPL command is of interest in multiuser processing situations. It enables different run units to have varying degrees of throughput priority. If a run unit invokes MPL, the value of **iblk**'s host language variable is used to set that run unit's priority. The **iblk** host language variable must be consistent with a two byte unsigned data item. Its value can be any integer from 1 through 10.

The highest possible priority level is 1. Selecting this level results in throughput that is at least as fast as level 2. Level 2 throughput, in turn, is at least as fast as level 3, and so forth. A run unit that does not require fastest throughput should use MPL to set its priority level lower than 1 (i.e., 2-10). A run unit that requires more than minimal throughput should use MPL to set its priority level higher than 10 (i.e., 9-1). A run unit can invoke MPL repeatedly to alter its own priority level at desired points in the program.

An interactive run unit has a priority level of 3 until it invokes MPL to get a different priority. A batch run unit has a priority level of 7 until it invokes MPL to get a different priority.

Examples of Command Usage

```
block/direct   ... E0 = MPL ("iblk")
block/indirect ... E0 = DMS ("MPL,iblk")
record/direct  ... E0 = MPL (iblk)
record/indirect... E0 = DMSD ("MPL",iblk)
```


SETPBF — SET Program BuFfer region — SETPBF

Command and Arguments

SETPBF,list

Currency Indicators

Used: none Changed: none

Description

This command allocates the region of main memory that is to be used for program buffers. If SETPBF is invoked, it must appear before any other DML command that needs to use program buffers. The composition of list is environment dependent and is fully discussed in the various system specific manuals. It typically consists of a starting address and a length (in bytes) for the page buffer area. For instance, with FORTRAN the starting address could be indicated by an array name and the length would be the number of bytes occupied by that array. In the C language, for example, the SBRK function is a dynamic memory allocation function; the value returned by SBRK can be used as an input to the SETPBF DML command.

In many environments the programmer can optionally omit SETPBF. If omitted in a single-user environment, MDBS.DMS itself sets up the largest program (i.e., page) buffer region possible. If omitted in a multiuser environment, MDBS.DMS allocates adequate program buffers. For a given operating environment, the system specific manual indicates whether SETPBF can be omitted.

Examples of Command Usage

See system specific manuals.

UNDEF — UNDEFine data blocks — UNDEF

Command and Arguments

UNDEF (no arguments)

Currency Indicators

Used: none Changed: none

Description

All data blocks previously defined in the program are undefined (i.e., they no longer exist). This is typically employed in host language programs that use chaining. UNDEF is invoked before the call to a chained subprogram and needed data blocks are redefined within the subprogram.

Examples of Command Usage

```
block/direct   ... E0 = UNDEF (     )
block/indirect ... E0 = DMS ("UNDEF")
```

VARCMD VARiable for CoMmanD string VARCMD

Command and Arguments

VARCMD,host language variable

Description

This special command is used in some host languages to specify a host language variable, whose value is treated as a DML command string. The application program sets the value of this variable to the desired command string (i.e., DML command and arguments) prior to each invocation of the data base control system.

Examples of Command Usage

See system specific manuals.

VARCS VARiable for Command Status VARCS

Command and Arguments

VARCS,host language variable

Currency Indicators

Used: none Changed: none

Description

The indicated host language variable will hold the command status that results from invoking a DML command. VARCS is needed only in those very few host languages that do not permit function subprograms. If VARCS is available for a host language (see system specific manuals), it should be the first DML command used in an application program. An exception to this is the case where DBINIT is also available, in which case VARCS is invoked after DBINIT.

Examples of Command Usage

See system specific manuals.

This page intentionally left blank.

XIV. MULTIUSER LOCKING COMMANDS

A. Overview

In a multiuser environment, MDBS III supports two kinds of record locking: passive and active. A record is passively locked if it is the current of any run unit, the current owner or current member of any set for any run unit, the current record occurrence of any record type for any run unit (for 3a only: see Appendix A), or the current record for any user-defined indicator in any run unit. A record can be actively locked by invoking a DML multiuser locking command (of which there are several). As long as a record is locked by one run unit, other run units are denied either read, write, or read and write access to that record. The type of access denied depends on whether the record is actively or passively locked.

Restrictions to access caused by multiuser locking do not affect security restrictions on a user's access. All security restrictions are enforced independently of whether an environment is single user or multiuser (though page image posting is unavailable for multiuser). Using a multiuser locking command in a single user environment has no effect.

In a multiuser environment, a record can be read if it is not locked or if it is passively locked. A record cannot be read if it is actively locked. A run unit can modify a record, if that record is not locked by any other run unit. As long as a record is actively or passively locked by one run unit, no other run unit can modify it.

The programmer of application systems for a multiuser environment should keep the following facts in mind:

1. If a record is not actively locked and a run unit has that record passively locked (i.e., it is a current record for that run unit), any run unit can read it.*

2. If a run unit has a record actively locked, no other run unit can read that record until the active lock is removed (with a DML command described in this chapter).

3. If a record is not actively locked and more than one run unit has that record passively locked, then no run unit can change or delete the record until all other run units have removed their passive locks on that record (e.g., with a DML assignment or find command).

4. If a run unit has a record actively locked, no other run unit can possibly change that record until the active lock is removed.

There are four types of active locking commands, which differ according to the scope of records that they actively lock. One actively locks the current of run unit, which is normally only

---

* MDBS QRS utilizes passive locking.

Table XIV-1: Multiuser Locking Contention Protocols

Run Unit A has \ Run Unit B attempts	Find or retrieve locked record	Modify locked record	Create record	Connect record into set-2	Disconnect a record from set-2	Disconnect all member (owner) records from set-2	Delete record
Passive lock on a record	permitted	not permitted*	not relevant	permitted	not permitted if A has the same CO and CM for set-2 as B	not permitted if A has the same CO(CM) for set-2 as B and CM(CO) of set-2 for A is non-null	not permitted
Active lock on a record	not permitted	not relevant	not relevant	not relevant	not relevant	permitted	not relevant
Active lock on all records of a record type	not permitted	not relevant	not permitted	not relevant	not relevant	permitted	not permitted
Active lock on all records of set-1	not permitted through set-1	not permitted if attempt to modify sort key value of set-1	not permitted if record is auto owner or member of set-1	not permitted if set-1 = set-2	not permitted if set-1 = set-2	not permitted if set-1 = set-2	not permitted if record is owner or member of set-1

* Modification is permitted if B's run unit environment has been altered by invoking DBENV with a host language variable value of 4.

passively locked. Another command allows a run unit to declare that one or more user-defined indicators are to be actively (rather than passively) locked. Another kind of active locking command places an active lock on all occurrences of a record type. The fourth actively locks a set, meaning that no other run unit can utilize that set and that the occurrences of that set's owner and member record types are actively locked. The effects of these locks on record retrieval, modification, creation, connection, disconnection and deletion are presented in Table XIV-1. A special DML command for setting run unit priority levels is available in some environments (see Chapter XIII).

If one run unit attempts to access a record occurrence (or set) that is locked in such a way that access is prohibited, then the access attempt will be re-tried periodically until either access is accomplished or some number of failed access tries have occurred. In the latter case, a command status message, indicating that a needed record is locked, is returned to the program. At any juncture in a program, the programmer can use a DML command (MCC) to specify the number of retries and the time units between tries. Defaults for these are operating system dependent (see MDBS System Specific Manuals).

As soon as a run unit invokes DBCLS, all records that it had locked are freed.

B. Command Details

MAU — Multiuser Active User indicators — MAU

Command and Arguments

MAU,iblk

Currency Indicators

Used: user indicators Changed: none

Description

One host language variable is defined for **iblk**. It must be consistent with a one byte unsigned data item. The value of that variable must be an integer between 0 and 255 inclusive. This value indicates how many user-defined indicators are to be made actively locked, rather than passively locked. If the value is 0, then all user-defined indicators revert to their normal state of being passively locked. If the value is 12, then user indicators 1 through 12 are actively locked and all other allocated user indicators are passively locked. MAU can be used repeatedly within a run unit to alter the number of user indicators that are actively locked.

A user indicator must have been allocated (with AUI) before it can be declared to be actively locked. A command status message of 62 or 63 results from MAU if any record that is the current

record of a user indicator is already actively or passively locked by another run unit. In this case, MAU will not affect the current active/passive status of user indicators.

Examples of Command Usage

```
block/direct   ... E0 = MAU ("iblk")
block/indirect ... E0 = ("MAU,iblk")
record/direct  ... E0 = MAU (iblk)
record/indirect... E0 = ("MAU",iblk)
```

MCC Multiuser Contention Count MCC

Command and Arguments

MCC,iblk

Currency Indicators

Used: none Changed: none

Description

This command sets the number of re-tries and the time units between tries that will be enforced by MDBS.DMS when an access attempt is prohibited due to multiuser locking. This is enforced for all DML commands that follow MCC, until another MCC is invoked. The first host language variable for iblk has an integer value that indicates the number of re-tries. The second variable defined for iblk has an integer value that indicates the number of time units between tries. The time unit is operating system dependent (see system specific manuals), but is never smaller than .01 seconds. Both of iblk's variables must be consistent with an unsigned data item that is two bytes in size.

Examples of Command Usage

```
block/direct   ... E0 = MCC ("iblk")
block/indirect ... E0 = DMS ("MCC,iblk")
record/direct  ... E0 = MCC (iblk)
record/indirect... E0 = DMSD ("MCC",iblk)
```

MCF Multiuser Current of run unit Free MCF

Command and Arguments

MCF (no arguments)

Currency Indicators

Used: CRU Changed: none

Description

If the record that is the current of run unit has been actively locked by MCP, invoking MCF will free that record so that it is only passively locked.

Examples of Command Usage

```
block/direct   ... E0 = MCF (     )
block/indirect ... E0 = DMS ("MCF")
record/direct  ... E0 = MCF (     )
record/indirect... E0 = DMSD ("MCF")
```

MCP — Multiuser Current of run unit Protect — MCP

Command and Arguments

MCP (no arguments)

Currency Indicators

Used: CRU Changed: none

Description

When this command is executed, the record that is the current of run unit is actively locked for as long as it is the current of run unit. After MCP has been invoked (and before invoking MCF), a record can become the current of run unit only if it is not actively or passively locked by some other run unit. When it does become the current of run unit, it becomes actively locked and the former current of run unit is no longer actively locked (by way of MCP).

To reinstate the current of run unit to its normal condition of being only passively locked, the MCF command is used.

Examples of Command Usage

```
block/direct   ... E0 = MCP (     )
block/indirect ... E0 = DMS ("MCP")
record/direct  ... E0 = MCP (     )
record/indirect... E0 = DMSD ("MCP")
```


MRTF — Multiuser Record Type Free — MRTF

Command and Arguments

MRTF,rec (rec is optional)

Currency Indicators

Used: none Changed: none

Description

All record occurrences of **rec** are freed. They are no longer actively locked by the run unit. The exception is that if one of those occurrences is actively locked through some other multiuser protect command, then it is still actively locked (until the corresponding multiuser free command is invoked). If no record type is specified with this command, then every record type that is actively locked by the run unit is freed.

Examples of Command Usage

```
block/direct   ... E0 = MRTF ("rec")
block/indirect ... E0 = DMS ("MRTF,rec")
record/direct  ... E0 = MRTF ("rec")
record/indirect... E0 = DMS ("MRTF,rec")
```

MRTP — Multiuser Record Type Protect — MRTP

Command and Arguments

MRTP,rec

Currency Indicators

Used: none Changed: none

Description

All occurrences of the record type **rec** are actively locked by the run unit, if no other run unit has locked any of those occurrences (either actively or passively). If any occurrence of **rec** is locked by another run unit, then the MRTP request is refused and a command status message to that effect is returned.

Examples of Command Usage

```
block/direct   ... E0 = MRTP ("rec")
block/indirect ... E0 = DMS ("MRTP,rec")
record/direct  ... E0 = MRTP ("rec")
record/indirect... E0 = DMS ("MRTP,rec")
```


MSF — Multiuser Set Free — MSF

Command and Arguments

MSF,set-1 (set-1 is optional)

Currency Indicators

Used: none Changed: none

Description

The indicated set-1 is freed if it has been actively locked earlier in the run unit. The record occurrences of the owner and member record types are no longer actively locked, unless there is also an active lock on one or more of those record types or on a record occurrence (until the corresponding multiuser free command is invoked). If no set is specified, all of the run units' actively locked sets are freed.

Examples of Command Usage

```
block/direct   ... E0 = MSF ("set-1")
block/indirect ... E0 = DMS ("MSF,set-1")
record/direct  ... E0 = MSF ("set-1")
record/indirect... E0 = DMS ("MSF,set-1")
```

MSP — Multiuser Set Protect — MSP

Command and Arguments

MSP,set-1

Currency Indicators

Used: none Changed: none

Description

An active lock is placed on set-1, if no other run unit has a lock (either active or passive) on both an owner record and a member record of that set. If any such occurrence is locked by another run unit, then the MSP request is refused and a command status message to that effect is returned.

Examples of Command Usage

```
block/direct   ... E0 = MSP ("set-1")
block/indirect ... E0 = DMS ("MSP,set-1")
record/direct  ... E0 = MSP ("set-1")
record/indirect... E0 = DMS ("MSP,set-1")
```


This page intentionally left blank.

XV. RECOVERY COMMANDS

A. Overview

The features and commands described in this chapter are available in various versions of the RTL form of the MDBS data base management system. The RTL form requires a larger minimum buffer size than the standard form. Invoking a recovery command follows the same conventions as invoking any other DML command. The recovery commands fall into two groups:

1. those that deal with page image posting of complex transactions (TRABT,TRBGN,TRCOM,PIFD), and
2. those that deal with the logging of transactions (LGFILE,LGFLSH,LGMSG,TRBGN,TRCOM).

Page image posting may be considered to form a first line of defense against data base inconsistency due to external factors such as power failures. The transaction logging commands provide the ultimate defense against failures of many kinds, including power failures, hardware malfunctions, and erroneous (although authorized) data modification. Either or both of the two recovery approaches can be used by an application program. However, page image posting is not meaningful in multiuser situations.

Page image posting allows the application developer to specify the beginning of a complex transaction sequence (with the TRBGN command). The data base changes caused by the transaction sequence are not incorporated into the data base until the commit command (TRCOM) is invoked. A transaction sequence can be aborted by invoking the abort (TRABT) command. An abnormal termination of the program in the midst of a complex transaction sequence will not leave the data base inconsistent. When the data base is re-opened, it is consistent. It is current up to the point of the last invocation of TRBGN before the interruption.

A page image preservation file is used by MDBS.DMS to provide this automatic recovery capability. The page image file must be declared with the PIFD command prior to opening the data base. If PIFD is not invoked, no posting occurs. If a page image file of insufficient size is used, then the data base could become inconsistent if a transaction sequence is abnormally interrupted. In this event, a command status error to that effect is returned when an attempt is made to re-open the data base. The log of transactions can be used to recover from such a situation.

Transaction logging allows an application program to make use of the log file defined in the DDL specification. The name of the file used for logging a program's transactions can be changed with the LGFILE command. All transactions since the last data base back-up are automatically logged onto the log file. The log file is used by the RCV utility (provided with the RTL form). RCV can be used to automatically re-apply transactions to an old back-up copy of the data base, thereby recreating an up-to-date data base. The user has extensive control over which transactions are re-applied.

The TRBGN command is used to indicate the start of a complex sequence of transactions, for the purpose of transaction logging. The TRCOM command terminates the complex transaction sequence started by TRBGN. RCV will ignore any logged transactions that occur after an TRBGN, if a subsequent TRCOM was not reached during a run unit. Other commands that can be used in conjunction with transaction logging are LGFLSH and LGMSG which flush the log file buffers and insert messages into the log file, respectively.

A complete description of the RTL form is presented in the MDBS-RTL Manual.

B. Command Details

LGFILE — LoG FILE specification — LGFILE

Command and Arguments

LGFILE,iblk

Currency Indicators

Used: none Changed: none

Description

The fully qualified name of a file to which transactions can be logged is specified by **iblk**'s host language variable. This variable must be consistent with a string data item, whose maximum length is operating system dependent (see the appropriate system specific manual for size information). If the file to which transactions are to be logged is the same as the log file specified in the DDL specification, this command need not be invoked. If it is invoked, this command must appear before the DBOPN command.

Examples of Command Usage

```
block/direct   ... E0 = LGFILE ("iblk")
block/indirect ... E0 = DMS ("LGFILE,iblk")
record/direct  ... E0 = LGFILE (iblk)
record/indirect... E0 = DMSD ("LGFILE",iblk)
```


LGFLSH — LoG file buffer FLuSH — LGFLSH

Command and Arguments

LGFLSH (no arguments)

Currency Indicators

Used: none Changed: none

Description

The buffers holding transactions to be logged are flushed to the log file when this command is invoked. LGFLSH is automatically invoked by DBSAVE. This is especially useful when an important transaction has been processed by the system. It can be written to the log file before the log file buffer is full.

Examples of Command Usage

```
block/direct   ... E0 = LGFLSH (     )
block/indirect ... E0 = DMS ("LGFLSH")
record/direct  ... E0 = LGFLSH (     )
record/indirect... E0 = DMS ("LGFLSH")
```

LGMSG — LoG file MeSsaGe — LGMSG

Command and Arguments

LGMSG,iblk

Currency Indicators

Used: none Changed: none

Description

The message specified in iblk's host language variable is written to the log file. This variable must be consistent with a string data item not exceeding 90 bytes. Messages written to the log file can be listed using the RCV utility program. The use of the LGMSG command is especially useful for surveillance.

Examples of Command Usage

```
block/direct   ... E0 = LGMSG ("iblk")
block/indirect ... E0 = DMS ("LGMSG,iblk")
record/direct  ... E0 = LGMSG (iblk)
record/indirect... E0 = DMSD ("LGMSG",iblk)
```


PIFD — Page Image File Declaration — PIFD

Command and Arguments

PIFD,iblk

Currency Indicators

Used: none Changed: none

Description

The fully qualified file name, indicated by iblk's host language variable, is declared to be a page image file for use by MDBS.DMS in page image posting. This command must be invoked prior to use of the TRBGN, TRCOM, and TRABT commands. The host language variable must be consistent with a string data item. The size (in bytes) is operating system dependent and is documented in the corresponding system specific manual. If PIFD is invoked, it must appear before the data base is opened.

Examples of Command Usage

```
block/direct   ... E0 = PIFD ("iblk")
block/indirect ... E0 = DMS ("PIFD,iblk")
record/direct  ... E0 = PIFD (iblk)
record/indirect... E0 = DMSD ("PIFD",iblk)
```

TRABT — TRansaction ABorT — TRABT

Command and Arguments

TRABT (no arguments)

Currency Indicators

Used: none Changed: most currency indicators ←—null

Description

Posting Effects (occur only if PIFD has been invoked):
This command aborts a complex transaction sequence that was initiated by the TRBGN command. A transaction cannot be aborted after it has already been committed (with the TRCOM command). This command invokes NCI internally to ensure that no inconsistent currency indicators will be present.

Logging Effects:
TRABT may be invoked only if PIFD has been invoked, in which case the transaction sequence that is aborted will be ignored by the RCV recovery processor.

Examples of Command Usage

```
block/direct    ... E0 = TRABT (     )
block/indirect  ... E0 = DMS ("TRABT")
record/direct   ... E0 = TRABT (     )
record/indirect... E0 = DMS ("TRABT")
```

TRBGN — TRansaction BeGiN — TRBGN

Command and Arguments

TRBGN (no arguments) or LGCPLX (no arguments)

Currency Indicators

Used: none Changed: none

Description

Posting Effects (occur only if PIFD has been invoked):
This command denotes the beginning of a complex transaction sequence, involving the use of several DML commands. The changes that these DML commands cause in the data base do not become permanent until the TRCOM command is invoked. TRCOM commits these changes to the data base. The TRABT command aborts a complex transaction. After TRCOM or TRABT has been invoked, TRBGN must be re-invoked to initiate another complex transaction. If TRBGN is re-invoked without having issued a TRABT or TRCOM command, a command status error is issued and the re-invocation of TRBGN is ignored by MDBS.DMS.

Logging Effects:

When this command is invoked, it indicates the beginning of a complex sequence of transactions. Although all transactions are automatically logged, RCV will ignore any transactions that follow TRBGN unless a subsequent TRCOM command is encountered. The TRBGN and TRCOM commands are used in tandem to ensure that either all or none (if a complex sequence is not completed) of the transactions, which form part of a complex sequence, are used by RCV during recovery.

Examples of Command Usage

```
block/direct    ... E0 = TRBGN (     )  or  E0 = LGCPLX (     )
block/indirect  ... E0 = DMS ("TRBGN")  or  E0 = DMS ("LGCPLX")
record/direct   ... E0 = TRBGN (     )  or  E0 = LGCPLX (     )
record/indirect... E0 = DMS ("TRBGN")  or  E0 = DMS ("LGCPLX")
```


TRCOM — TRansaction COMmit — TRCOM

Command and Arguments

TRCOM (no arguments) or **LGENDX** (no arguments)

Currency Indicators

Used: none Changed: none

Description

Posting Effects (occur only if PIFD has been invoked):
This command commits a complex transaction, that was initiated by the TRBGN command, to the data base. It has the added effect of clearing the contents of the page image file declared with the PIFD command.

Logging Effects:
This command indicates the end of a complex sequence of transactions. Any transactions logged since the last TRBGN command was invoked can now be processed by the RCV utility for data base restoration.

Examples of Command Usage

```
block/direct   ... E0 = TRCOM (     )  or  E0 = LGENDX (     )
block/indirect ... E0 = DMS ("TRCOM")  or  E0 = DMS ("LGENDX")
record/direct  ... E0 = TRCOM (     )  or  E0 = LGENDX (     )
record/indirect... E0 = DMS ("TRCOM")  or  E0 = DMS ("LGENDX")
```


XVI. COMMAND STATUS DESCRIPTIONS

A. Overview

This chapter describes the different command status messages that may result from the execution of a DML command. For each command status (other than 0 and 255), the possible reasons for which that command status could be obtained are described. In almost all cases, the reasons themselves suggest a solution to the problem.

B. Command Status Details

1 Invalid area name **1**

Possible Causes:
1. Typographical error in spelling area name.
2. Incorrect order in command string.
3. Command string not terminated properly.

2 Invalid set name **2**

Possible Causes:
1. Typographical error in spelling set name.
2. Incorrect order in command string.
3. Command string not terminated properly.

3 Invalid record type name **3**

Possible Causes:
1. Typographical error in spelling record type name.
2. Incorrect order in command string.
3. Command string not terminated properly.

4 Invalid data item for this record type **4**

Possible Causes:
1. Typographical error in spelling data item name.
2. Incorrect order in command string.
3. Command string not terminated properly.

5 Invalid owner record type for this set **5**

Possible Causes:
1. The record occurrence specified has not been defined as the owner for the set specified.
2. Typographical error in spelling set name.

6 Invalid member record type for this set **6**

Possible Causes:
1. The record occurrence specified has not been defined as the member for the set specified.
2. Typographical error in spelling set name.

7 Invalid data block name **7**

Possible Causes:
1. A data block specified in the command string has not been defined.
2. A blank data block name was specified for a "DEFINE" command.
3. An "EXTEND" command does not immediately follow a "DEFINE" command.

8 Invalid name **8**

Possible Causes:
1. There was a syntax error related to the filename specification in a DBOPN or DBOPNA command.
2. Invalid drive specification.

9 Invalid number **9**

Possible Causes:
1. An invalid number was passed to DBCNV or DBENV.

10 Syntax error on command line **10**

Possible Causes:
1. Self-explanatory.

11 Record already member/owner of set **11**

Possible Causes:
1. An attempt was made to connect a member record to an owner record for a set, when that member was already connected to that owner for that set.
2. An attempt was made to connect an owner record to a member record for a set, when that owner was already connected to that member for that set.

12 Record does not belong to set occurrence **12**

Possible Causes:
1. The record occurrence specified by the SME (or SOE) command is not a member or owner of the specified set.

13 Invalid area for this record type **13**

Possible Causes:
1. An area has been specified in the CRA command and the record type does not belong to this area.

14 Data base area already open **14**

Possible Causes:
1. The DBOPN or DBOPNA (for a given area) command was invoked twice.
2. For interpretive languages, DBCLS or DBCLSA was not called.

15 Data base not closed previously **15**

Possible Causes:
1. The data base may be inconsistent since it was not closed previously.

16 Area not consistent with main area **16**

Possible Causes:
1. A referenced area was not consistent with the main area.
2. An old copy of the area was on-line rather than a copy of the area that is up-to-date with the on-line main area.

17 No space available in data base **17**

Possible Causes:
1. No space is available in the data base or area for processing the requested transaction.

18 Set internally inconsistent **18**

Possible Causes:
1. This error is very rare and is returned from the following commands: CRA, CRS, IMS, IOS.
2. The error occurs due to insufficient space being available for pointers to be allocated.

19 No duplicate record allowed **19**

Possible Causes:
1. No duplicates have been allowed in the DDL specifications for the record type.

20 No current of run unit **20**

Possible Causes:
1. Self-explanatory.
2. The user should check the currency indicators changed by the previous DML command.

21 No current owner of set type **21**

Possible Causes:
1. The command SOC, SOE, SOM, SOO or SOU was not invoked for this set.
2. The command FFO, FLO, FNO, FPO, SMC, SMM, SMO or SMU returned command status 255 previously.

22 No current member of set type **22**

Possible Causes:
1. The command SMC, SME, SMM, SMO or SMU was not invoked for this set.
2. The command FFM, FLM, FNM, FPM, SOC, SOM, SOO or SOU returned command status 255 previously.

****23**** **No current record of record type** ****23****

Possible Causes:
1. The command CR, CRS,SRC,SRM or SRO was not previously invoked for this record type.
2. The command DRC, DRM, DRO, or DRR was invoked for the current record of this record type.

****24**** **No current user record** ****24****

Possible Causes:
1. There is no current record of a referenced user-defined indicator.

****25**** **User-defined currency indicator not allocated** ****25****

Possible Causes:
1. A referenced user-defined indicator has not been allocated.

****26**** **Current of run unit not in specified area** ****26****

Possible Causes:
1. The command FNS has been invoked with an area specified as an argument, and the current of run unit is not in the indicated area.

****27**** **Inappropriate set** ****27****

Possible Causes:
1. An inappropriate set appeared as an argument. For example, the last set specified in a Boolean command is not system-owned.
2. The first or second set specified with a Boolean command is N:1.

****28**** **Inappropriate record type** ****28****

Possible Causes:
1. An FRK command has been invoked on a record whose record type has not been declared as having a calc key.

****29**** **Cannot obtain based on record type** ****29****

Possible Causes:
1. An attempt was made to obtain a record based on a sort key involving the record type names. Use the corresponding find command and GETC instead.

****30**** **Data out of range** ****30****

Possible Causes:
1. The data being input is outside the range of possible values as indicated in the DDL specifications.
2. The value of an input argument to a DML command is not a permissible value for that argument.

****31**** **Insufficient room in memory** ****31****

Possible Causes:
1. For languages using the SETPBF command, not enough buffer space has been allocated.
2. For languages not using the SETPBF command, the DMS has been ORGed too high in memory, or the FIRST and LAST word in memory do not allow enough room.

****32**** **Data base opened for read access only** ****32****

Possible Causes:
1. Self-explanatory.
2. Invalid read/write specifications in OPEN command.

****33**** **Data conversion error** ****33****

Possible Causes:
1. An error has been obtained during data conversion by MDBS.DMS based on the DDL item specifications.

****34**** **No such routine** ****34****

Possible Causes:
1. Typographical error in spelling command name.
2. Logging commands have been used without the RTL form of MDBS.
3. A command was invoked using wrong entry point (DMS or DMSD).
4. A command was invoked that attempted to perform explicitly disabled processing (e.g., disabled by NOCALC or NOFLOAT).

35 Duplicate data block name **35**

Possible Causes:
1. The DEFINE command was invoked for a data block that already exists.

36 Invalid password **36**

Possible Causes:
1. User name/password does not exactly match DDL specifications.

37 Invalid number of arguments **37**

Possible Causes:
1. Too many arguments specified in command string.

38 Data base not open **38**

Possible Causes:
1. A DML command, needing to access the data base, has been invoked before the data base was opened.

39 Set not sorted **39**

Possible Causes:
1. The FOSK or FMSK command is invoked on a non-sorted set.

40 User may not read this set **40**

Possible Causes:
1. The user's read access codes have no common elements with the read access codes of the set being processed.

41 User may not write this set **41**

Possible Causes:
1. The user's write access codes have no elements in common with the write access codes of the set being processed.

42 User may not read this record **42**

Possible Causes:
1. The user's read access codes have no elements in common with the read access codes of the record being processed.

43 User may not write this record **43**

Possible Causes:
1. The user's write access codes have no common elements with the write access codes of the record being processed.

44 User may not read this item **44**

Possible Causes:
1. The user's read access codes have no elements in common with the read access codes of the item being processed.

45 User may not write this item **45**

Possible Causes:
1. The user's write access codes have no elements in common with the write access codes of the item being processed.

46 User may not read this area **46**

Possible Causes:
1. The user's read access codes have no elements in common with the read access codes of the specified area.

47 User may not write this area **47**

Possible Causes:
1. The user's write access codes have no elements in common with the write access codes of the specified area.

48 No value labels defined for this field **48**

Possible Causes:
1. The specified field presently has no value labels.

49 Invalid data base key **49**

Possible Causes:
1. The value input to the SCD command is not the data base key for a valid record occurrence in the data base.
2. A record whose data base key was formerly saved by SDC may have since been deleted or moved.

50 Checksum error **50**

Possible Causes:
MDBS.DMS automatically closes the data base when this error occurs.
1. An error was encountered when a checksum was being performed on the data base pages.

51 Disk read or write error **51**

Possible Causes:
This command status is returned as a result of any I/O error. MDBS.DMS automatically closes the data base when this error occurs. Some of the possible causes are:
1. Bad sector on disk.
2. Disk drive misaligned.
3. The operating system has detected an error when trying to read/write from/to disk.

52 Invalid data base format **52**

Possible Causes:
An error was encountered when the system performed a synchronization check on the data base pages. MDBS.DMS automatically closes the data base when this error occurs. This may mean that:
1. Disk is in the wrong drive.
2. The correct area disk is not on-line.

53 Data base not expanded **53**

Possible Causes:
1. The data base was initialized with the s or o option, but was not expanded with MDBS.DMU.

54 File not present **54**

Possible Causes:
1. Disk in wrong drive.
2. File name misspelled.

55 Log file I/O error **55**

Possible Causes:
1. An I/O error was encountered when trying to open the log file.
2. See also Command Status 51.

56 Log file not current **56**

Possible Causes:
1. The log file is not up-to-date.

57 Logging inhibited **57**

Possible Causes:
1. An I/O error was detected on the log file.
2. The log file may already be open.

59 Needed area not on-line **59**

Possible Causes:
1. As the result of a DML command, the data base control system attempted to reference an area that is currently off-line.
2. For record types, whose records can exist in any area, a CRS was invoked without having on-line the first (as specified in the DDL) extra area with sufficient space.
3. FFS or FNS was invoked for an off-line area.
4. All areas allowing pointer indices were not on-line for data base alteration.

60 Data base not available **60**

Possible Causes:
1. This command status is returned only when multiple users are accessing the data base.
2. The user is not allowed to open the data base because some other user has opened the data base on an exclusive basis.

****61**** Exclusive open refused ****61****

Possible Causes:
1. This command status is returned only when multiple users are accessing the data base.
2. An exclusive open was not granted as some other user had the data base already open.

****62**** Operation rejected due to active lock ****62****

Possible Causes:
1. This command status is returned only when multiple users are accessing the data base.
2. The operation requested was not performed since there was an active lock on the record or set specified.
3. The run unit is attempting to access a record when the access request, if granted, would cause a deadlock situation.
4. The error is issued after the waiting time specified with MCC is exhausted.
5. The run unit encountering this situation can invoke NCI (and TRABT, if applicable) and restart its processing. This is not advisable if the run unit is in the midst of logging a complex transaction.

****63**** Operation rejected due to passive lock ****63****

Possible Causes:
1. This command status is returned only when multiple users are accessing the data base.
2. The operation requested was refused as there was a passive lock on the record specified.
3. The run unit is attempting to access a record when the access request, if granted, would cause a deadlock situation.
4. The error is issued after the waiting time specified with MCC is exhausted.
5. The run unit encountering this situation can invoke NCI (and TRABT, if applicable) and restart its processing. This is not advisable if the run unit is in the midst of logging a complex transaction.
6. The SYSTEM record cannot be actively locked because it is always the current owner of every system-owned set (i.e., it is always passively locked).

****65**** Invalid use of CALC record ****65****

Possible Causes:
1. The CRA command was used for a record type having a calc key.

66 Record may not be removed from fixed set **66**

Possible Causes:
1. The retention for the set specified has been declared as fixed in the DDL specifications.

67 User count exceeded **67**

Possible Causes:
1. Too many run units attempted to concurrently access the data base.

70 No transaction in progress **70**

Possible Causes:
1. The TRABT, TRCOM, or LGENDX command was invoked when no complex transaction was in progress.
2. TRBGN or LGCPLX has not been invoked.

71 Transaction already in progress **71**

Possible Causes:
1. The TRBGN or LGCPLX command was invoked before the prior complex transaction was terminated by a TRABT, TRCOM or LGENDX command.

72 Posting not active **72**

Possible Causes:
1. The TRABT command was invoked without having previously invoked PIFD.
2. PIFD has not been invoked before opening the data base.

73 Posting table overflow **73**

Possible Causes:
1. This command status can be returned from TRCOM or TRABT when posting is active.
2. The transaction is committed to the data base, even though a TRABT was attempted.
3. Consider defining several shorter transaction sequences to replace the existing transaction sequence.

74 Log file not present **74**

Possible Causes:
1. Using the RTL form of MDBS, the data base was opened without the log file being present. Processing continues.

75 Unable to re-insert record into all sets **75**

Possible Causes:

1. The value of a sort key field changed and there was insufficient memory for resorting the affected set occurrences. The record is removed from all set occurrences for which there is insufficient resorting room.

80 Load error from DMSLDR **80**

Possible Causes:

1. An attempt was made to load a non-executable file.
2. There are presently too many operating system files open to allow another to be opened.

81 Communications Failure **81**

Possible Causes:

1. This command status can appear only in versions of the system that require interprocess communication (e.g., multiuser versions).
2. It indicates that an unexpected communication error was encountered during interprocess communication (e.g., a user tries to start a run unit while the data base control system is not executing).

100-199 Catastrophic errors **100-199**

Command status errors 100 through 199 are particular cases of a general "catastrophic" error, which would result in the integrity of the data base being destroyed. If any of these **extremely rare** command status errors is obtained, the user should contact the Micro Data Base Systems, Inc. technical support staff for aid in solving the problem. A few general notes on some of these errors are provided below.

100 Hole table inconsistency. The table which specifies how much room is on each page in this area lists more free bytes than the page itself lists. No workaround.

101 Invalid record occurrence. An invalid record id was encountered when trying to determine the record type of the record occurrence. If possible, remove this record from the set.

103 Cannot find a connected record when resorting. When reordering a sorted set, problems with set connections are encountered. This could happen only for a "put" command. No workaround.

104 Record not fully connected. When inserting a record into a set whose order is next or prior, a partially connected record was encountered.

106 Invalid record id encountered during a comparison operation. When comparing two item values, an invalid record occurrence was encountered.

107 Pointer not found in higher level index (DPA) during insertion. Invalid pointer encountered when moving a DPA. Possible workaround: Use Boolean commands to move all members into $SYSSET, remove all members from the original set, then reinsert the members back from $SYSSET.

108 Cannot determine item type. When trying to locate an item, the data base control system could not find an item type in the item descriptors.

109 Cannot determine record type during a "find" command. An invalid record id was found when building the sort key value.

110 Cannot find pointer in higher level during insertion. Invalid pointer encountered when splitting DPA. Possible workaround: Use Boolean commands to move all members into $SYSSET, remove all members from the original set, then reinsert the members back from $SYSSET.

111 Cannot find pointer in higher level during removal. Invalid pointer encountered when removing a record from a set. Possible workaround: Use Boolean commands to move all members into $SYSSET, remove all members from the original set, then reinsert the members back from $SYSSET.

112 SYSTEM is not owner of a set. This set is expected to be a system-owned set, but the set descriptor does not list SYSTEM as an owner.

113 Cannot find pointer in higher level during insertion. Invalid pointer encountered when updating higher DPAs after first element has changed. Possible workaround: Use Boolean commands to move all members into $SYSSET, remove all members from the original set, then reinsert the members back from $SYSSET.

114 Record reference not found in DPA during removal. The record for which removal was attempted is not completely connected in this set.

116 Record not found in set descriptor. When doing a "find" operation, a record was encountered that could not be found in the set descriptor for this set.

117 SYSTEM record DPA not found. When doing a "find" operation, a record was encountered that could not be found in the set descriptor for this set.

119 Invalid record type in set descriptor. When doing a "find" operation on a sorted set, a record was encountered that could not be found in the set descriptor for this set.

120 Record not found in set descriptor. When removing a record from a set, a record was encountered that could not be found in the set descriptor for this set.

121 Invalid record type encountered during Boolean operation. An invalid record id was encountered when trying to determine the record type of the record occurrence. If possible, remove the record from the set.

124 Record not found in set descriptor. When checking for duplicates, a record was encountered that could not be found in the set descriptor for this set.

126 Error in locating record. An invalid record id was encountered when trying to determine the record type of the record occurrence. If possible, remove the record from the set.

130 Master unable to deallocate user descriptor. The user descriptor was not found when processing user logout.

MDBS DMS MANUAL

Appendix A

Obsolete DML Commands Available with Version 3a

Appendix A

A. Introduction

For purposes of upward compatibility from MDBS Version 1 to MDBS Version 3a, additional commands and alternative command names are available in Version 3a. The additional commands primarily involve the use of a kind of currency indicator not described earlier in this manual. In Version 3a, every record type has a current record, provided the DDL Analyzer was invoked with the -v option. At any moment during the execution of a run unit, one occurrence of a record type is the current record occurrence (CRO) of that record type. If a schema has nine record types, there are nine currency indicators beyond those described earlier in this manual. Although these additional currency indicators exist in Version 3a, they are never required for processing; they exist solely to provide upward compatibility so that application programs written for Version 1 can be used by Version 3a with only minimal (if any) changes.

The effects of a few Version 3 DML commands differ slightly, in their effect on currency indicators, from Version 1 commands of the same name. This manual fully explains the effect of each Version 3 command. Most notable among the differences are those for the FMSK, FOSK, RMS, DRM and DRO. In Version 3, the FMSK (FOSK) "wildcard" feature leaves the current member (owner) indicator positioned at the next record rather than the prior record as is the case with the Version 1 FMSK (FOSK). The Version 3 remove and delete commands do not leave the set's current member (or current owner for DRO) indicator null as they do in Version 1. The next member (or owner) becomes current. Also note that the Version 3 SMM, SMO and SMC do not return a command status of 12 when there is no owner. They return a command status of 255 in this situation. Similarly, the Version 3 SMR (described in this appendix) will return a 255 command status, rather than a 12 as in Version 1.

B. Alternative Command Names

Alternative Command Name (Supported in Versions 1 and 3a)	Command Name Used in this Manual
ACS	IMS
CLOSE	DBCLS
CMT	TMT
COT	TOT
ERRSET	VARCS
FINDM	FMI
FINDO	FOI
OPEN	DBOPN
SFC	PFC
SFM	PFM
SFO	PFO
STAT	DBSTAT
TOGGLE	DBENV

The Version 3a TOGGLE, OPEN and STAT commands (while identical to DBENV, DBOPN and DBSTAT respectively) differ slightly from the Version 1 TOGGLE, OPEN and STAT commands. See the descriptions of DBENV, DBOPN and DBSTAT in this manual.

C. Additional Commands

AMS Add Member to Set **AMS**

Command and Arguments

AMS,rec,set-1

Currency Indicators

Used: CO(set-1)
CRO(rec)

Changed: CM(set-1)←——CRO(rec)

Description

The record that is current record occurrence of **rec** becomes connected to the current owner of **set-1**. This new member for the current owner of **set-1** is made the current member of **set-1**. The insertion of this new member record among the existing member records, connected to the current owner of **set-1**, is made in accordance with the member order declared for **set-1** in the DDL specification.

The current record occurrence of **rec** must be an occurrence of a record type that is a member of **set-1**. A user must have write access to **set-1**, in order to connect a member record to an owner record.

Examples of Command Usage

```
block/direct   ... E0 = AMS ("rec,set-1")
block/indirect ... E0 = DMS ("AMS,rec,set-1")
record/direct  ... E0 = AMS ("rec,set-1")
record/indirect... E0 = DMS ("AMS,rec,set-1")
```


CR — Create Record — CR

Command and Arguments

CR,rec

Currency Indicators

Used: none	Changed: CRU ←——newly created record CRO(rec) ←——newly created record
...if rec is AUTO member of set	...if rec is AUTO member of set
CO(autoset)	CM(autoset) ←——newly created record
...if rec is AUTO owner of set	...if rec is AUTO owner of set
CM(autoset)	CO(autoset) ←——newly created record

Description

An occurrence of the record type **rec** is created. This occurrence has no data. No range checking is performed.

The physical placement of the newly created record is consistent with the DDL specification for **rec** (either CALCed, clustered, or system-determined). In order to create the record, a user must have write access to **rec**, to the **area** in which the record is to be created, and to all sets in which the record type participates as an automatic owner or member. If a user does not have this security clearance, a command status error is returned, the record is not created, and no currency indicator is changed.

If **rec** has been declared to be the AUTO member of a set, then the newly created record is automatically connected to the current owner of that set and the new record becomes the current member of that set. The connection takes place according to the member order specified with the DDL (SORTED, FIFO, etc.). If the member order for the set is NEXT, then the new record is connected (i.e., logically inserted) immediately after the set's current member. If there is no current member, the record is inserted as the first member. For PRIOR member order, the new record is logically connected immediately before the set's current member. If there is no current member, the record is inserted as the last member.

If **rec** has been declared to be the AUTO owner of a set, then the newly created record is automatically connected to the current member of that set and the new record becomes the current owner of that set. The connection occurs according to the set's owner order, as specified with the DDL (SORTED, LIFO, NEXT, etc.). If the owner order for the set is NEXT (or PRIOR), then the new record is logically connected immediately after (or before) the set's current owner. If there is no current owner, the record is inserted as the first (last) owner.

Examples of Command Usage

```
block/direct   ... E0 = CR ("rec")
block/indirect ... E0 = DMS ("CR,rec")
record/direct  ... E0 = CR ("rec")
record/indirect... E0 = DMS ("CR,rec")
```

DRR Delete Record that is current Record occurrence DRR

Command and Arguments

```
DRR,rec
```

Currency Indicators

Used:		Changed:		
CRO(rec)		CRO(rec)	null
		CRU	null
		also see description	

Description

The record that is the current record occurrence of **rec** is disconnected from all set relationships and is physically deleted from the data base. The current record of **rec** and the current of run unit becomes null. If the deleted record is the current owner of any set, the current member of any set, or the current record of any user-defined indicator, then those indicators become null. If a user does not have write access to the current of run unit's record type, to the record's area and to all sets in which that record type participates, then no deletion occurs, no currency indicator changes, and a command status error is returned.

Examples of Command Usage

```
block/direct   ... E0 = DRR ("rec")
block/indirect ... E0 = DMS ("DRR,rec")
record/direct  ... E0 = DRR ("rec")
record/indirect... E0 = DMS ("DRR,rec")
```


GETR GET data from Record GETR

Command and Arguments

GETR,rec,oblk

Currency Indicators

Used: CRO(rec) Changed: none

Description

All data values in the current record occurrence of **rec** are returned in the host language variables **oblk.** The types, sizes and sequence of these variables must be consistent with the data items that make up the **rec** record type. A command status error is returned if a user does not have read access to all data items of the **rec** record type.

Examples of Command Usage

```
block/direct    ... E0 = GETR ("rec,oblk")
block/indirect  ... E0 = DMS ("GETR,rec,oblk")
record/direct   ... E0 = GETR ("rec",oblk)
record/indirect... E0 = DMSD ("GETR,rec",oblk)
```

GFR Get Field from Record GFR

Command and Arguments

GFR,itm,rec,oblk

Currency Indicators

Used: CRO(rec) Changed: none

Description

The value of the item data item (field) in the current record occurrence of **rec** is returned in the host language variable of **oblk.** The type and size of this variable must be consistent with the type and size of **itm** in the DDL specification. A command status error is returned if a user does not have read access to the **itm** field.

Examples of Command Usage

```
block/direct    ... E0 = GFR ("itm,rec,oblk")
block/indirect  ... E0 = DMS ("GFR,itm,rec,oblk")
record/direct   ... E0 = GFR ("itm,rec",oblk)
record/indirect... E0 = DMSD ("GFR,itm,rec",oblk)
```


PUTR PUT data into Record PUTR

Command and Arguments

PUTR,rec,iblk

Currency Indicators

Used: CRO(rec) Changed: none

Description

Data values of the **iblk** host language variables put into the current record occurrence of **rec.** The types, sizes and sequence of these variables must be consistent with the data items that make up the **rec** record type. A command status error is returned if the user does not have write access to all data items that make up the **rec** record type and to the area containing the record to be changed. A command status error is also returned if the value of an **iblk** variable is outside the feasibility range defined for the corresponding data item.

Examples of Command Usage

```
block/direct   ... E0 = PUTR ("rec,iblk")
block/indirect ... E0 = DMS ("PUTR,rec,iblk")
record/direct  ... E0 = PUTR ("rec",iblk)
record/indirect... E0 = DMSD ("PUTR,rec",iblk)
```

SCR Set Current of run unit based on Record SCR

Command and Arguments

SCR,rec

Currency Indicators

Used: CRO(rec) Changed: CRU←——CRO(rec)

Description

The record that is presently the current record occurrence of **rec** becomes the new current of run unit. The current record occurrence of **rec** remains unchanged. If the user does not have read access to **rec,** a command status error is returned and the current of run unit is unchanged.

Examples of Command Usage

```
block/direct   ... E0 = SCR ("rec")
block/indirect ... E0 = DMS ("SCR,rec")
record/direct  ... E0 = SCR ("rec")
record/indirect... E0 = DMS ("SCR,rec")
```


SFR Store Field into Record SFR

Command and Arguments

SFR,itm,rec,iblk

Currency Indicators

Used: CRO(rec) Changed: none

Description

The value of the **iblk** host language variable is put into the **itm** field of the current record of **rec.** A command status error is returned if the user does not have write access to **itm** and to the area containing the record to be changed. A command status error is also returned if the value of **iblk**'s variable is outside of **itm**'s feasibility range.

Examples of Command Usage

```
block/direct   ... E0 = SFR ("itm,rec,iblk")
block/indirect ... E0 = DMS ("SFR,itm,rec,iblk")
record/direct  ... E0 = SFR ("itm,rec",iblk)
record/indirect... E0 = DMSD ("SFR,itm,rec",iblk)
```

SMR Set Member based on Record SMR

Command and Arguments

SMR,rec,set-1

Currency Indicators

Used: CRO(rec) Changed: CM(set-1) ←—— CRO(rec)
CO(set-1) ←—— first owner
CRU ←—— first owner

Description

Set the current member of **set-1** to be the same as the current occurrence of **rec.** This command is valid only if **rec** is a member record type for **set-1.** The first owner of the new current member of **set-1** becomes the new current owner of **set-1** and the current of run unit. If the new current member of **set-1** becomes null, the current of run unit becomes null and a command status of 255 is returned. If a user does not have read access to both **set-1** and **rec,** then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct    ... E0 = SMR ("rec,set-1")
block/indirect  ... E0 = DMS ("SMR,rec,set-1")
record/direct   ... E0 = SMR ("rec,set-1")
record/indirect... E0 = DMS ("SMR,rec,set-1")
```

SOR Set Owner based on Record SOR

Command and Arguments

```
SOR,rec,set-1
```

Currency Indicators

Used: CRO(rec) Changed: CO(set-1) ←—— CRO(rec)
CM(set-1) ←—— first member
CRU ←—— first member

Description

Set the current owner of **set-1** to be the same as the current occurrence of **rec**. This command is valid only if **rec** is an owner record type for **set-1**. The first member of the new current owner of **set-1** becomes the new current member of **set-1** and the current of run unit. If the new current owner of **set-1** becomes null, the current of run unit becomes null and a command status of 255 is returned. If a user does not have read access to both **set-1** and **rec**, then a command status error is returned and no currency indicators change.

Examples of Command Usage

```
block/direct    ... E0 = SOR ("rec,set-1")
block/indirect  ... E0 = DMS ("SOR,rec,set-1")
record/direct   ... E0 = SOR ("rec,set-1")
record/indirect... E0 = DMS ("SOR,rec,set-1")
```

SRC Set Record based on Current of run unit SRC

Command and Arguments

```
SRC
```

Currency Indicators

Used: CRU Changed: CRO ←—— CRU

Description
The record that is the current of run unit becomes the new current record occurrence of its record type.

Examples of Command Usage

```
block/direct   ... E0 = SRC (     )
block/indirect ... E0 = DMS ("SRC")
record/direct  ... E0 = SRC (     )
record/indirect... E0 = DMS ("SRC")
```

SRM — Set Record based on Member — SRM

Command and Arguments

SRM,set-1

Currency Indicators

Used: CM(set-1) Changed: CRO←——CM(set-1)

Description
The record that is the current member of **set-1** becomes the new current record occurrence of its type.

Examples of Command Usage

```
lock/direct   ... E0 = SRM ("set-1")
block/indirect ... E0 = DMS ("SRM,set-1")
record/direct  ... E0 = SRM ("set-1")
record/indirect... E0 = DMS ("SRM,set-1")
```

SRN — Set Record to Null — SRN

Command and Arguments

SRN,rec

Currency Indicators

Used: none Changed: CRO(rec)←—— null

Description
The current record occurrence for the **rec** record type becomes null. If a user does not have read access to **rec**, then the current record occurrence is unchanged and a command status error is returned.

Examples of Command Usage

```
block/direct    ... E0 = SRN ("rec")
block/indirect  ... E0 = DMS ("SRN,rec")
record/direct   ... E0 = SRN ("rec")
record/indirect ... E0 = DMS ("SRN,rec")
```


SRO Set Record based on Owner SRO

Command and Arguments

SRO,set-1

Currency Indicators

Used: CO(set-1) Changed: CRO←— CO(set-1)

Description

The record that is the current owner of **set-1** becomes the new current record occurrence of its type.

Examples of Command Usage

```
block/direct   ... E0 = SRO ("set-1")
block/indirect ... E0 = DMS ("SRO,set-1")
record/direct  ... E0 = SRO ("set-1")
record/indirect... E0 = DMS ("SRO,set-1")
```


MDBS DMS MANUAL

Appendix B

DML Command Formats

This page intentionally left blank.

DOCUMENTATION COMMENT FORM

MDBS Document Title:_____

We welcome and appreciate all comments and suggestions that can help us to improve our manuals and products. Use this form to express your views concerning this manual.

Please do not use this form to report system problems or to request materials, etc. System problems should be reported to **MDBS** by phone or telex, or in a separate letter addressed to the attention of the technical support division. Requests for published materials should be addressed to the attention of the marketing division.

Sender:

_____ _____
(name) (position)

_____ _____
(company) (telephone)

_____
(address)

_____
(city, state, zip)

COMMENTS:

Areas of comment are general presentation, format, organization, completeness, clarity, accuracy, etc. If a comment applies to a specific page or pages, please cite the page number(s).

_____

_____

_____

_____

_____

_____

_____

_____

Continue on additional pages, as needed. Thank you for your response.