60459510

CONTROL DATA CORPORATION

TAF/CRM DATA MANAGER
VERSION 1
REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS 2

REVISION RECORD

REVISION	DESCRIPTION
A (04-26-82)	Manual released. This manual reflects NOS 2.0 at PSR level 562. Includes documentation of TAF/CRM support of MIP and AK files, the TAF/CRM automatic recovery feature, the TAF/CRM batch concurrency feature, and the TAF/CRM batch recovery feature.
B (01-27-83)	This revision reflects changes made to NOS 2.1 at PSR level 580. Includes documentation changes to the xxJ file where RMKDEF statements must be used for MIP files. Shows change in sequence of events to reconstruct the BRF(s). This edition obsoletes all previous editions.
C (03-22-85)	Manual updated to NOS 2.4 at PSR level 630. This revision reflects the addition of error processing for batch concurrency requests. Various technical and editorial changes have been made.

Publication No.
60459510

REVISION LETTERS I, O, Q, S, X AND Z ARE NOT USED.

Address comments concerning this manual to:

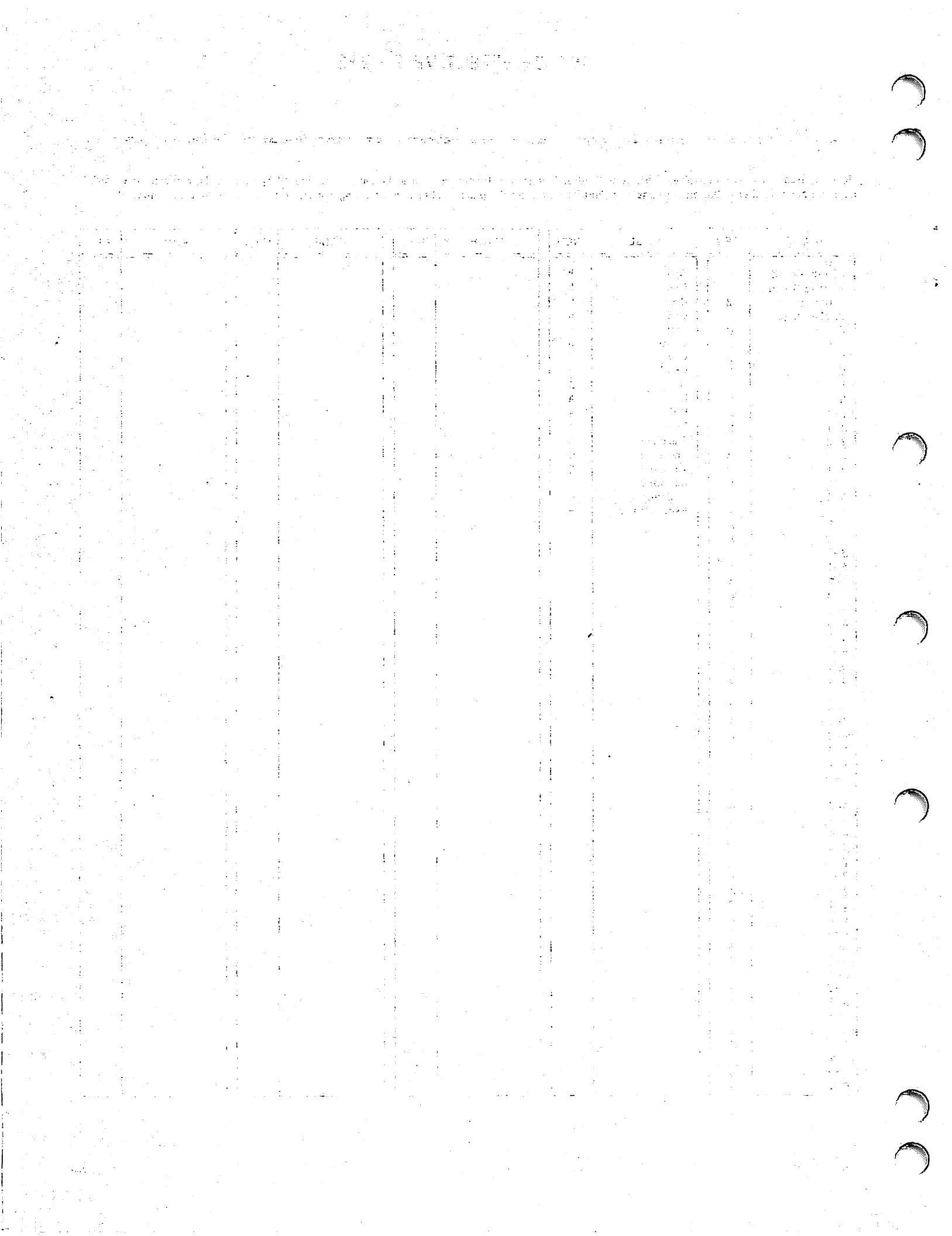
Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Front Cover	-
Inside Front Cover	B
Title Page	-
2	C
3/4	C
5	C
6	C
7	C
8	C
1-1	A
2-1	B
2-2	B
2-3	B
3-1	A
3-2	C
3-3	A
3-4	A
3-5	A
3-6	A
3-7	A
4-1	A
4-2	A
4-3	B
4-4	B
4-5	A
4-6	A
4-7	A
5-1	A
5-2	B
5-3	C
5-4	C
6-1	A
6-2	B
6-3	A
6-4	B
6-5	A
6-6	A
6-7	C
6-8	C
6-9	C
7-1	A
7-2	B
7-3	B
7-4	A
8-1	C
8-2	C
8-3	C
8-4	B
9-1	C
A-1	A
A-2	A
A-3	B
B-1	A
B-2	A
C-1	B
C-2	B
C-3	B
C-4	B
C-5	B
C-6	B
C-7	C
C-8	B
C-9	B
C-10	B
C-11	C
C-12	B
D-1	A
E-1	A
E-2	A
E-3	A
Index-1	C
Index-2	C
Index-3	C
Comment Sheet	C
Back Cover	-



PREFACE

This manual describes the CDC® Transaction Facility (TAF) Version 1.3 interface to the CONTROL DATA® CYBER Record Manager (CRM) Advanced Access Methods (AAM). It does not explain CYBER Record Manager. This manual assumes that CRM files already exist. Refer to the CRM AAM 2 Reference Manual for further information.

AUDIENCE

This manual is intended for the data administrator and applications programmers writing tasks in FORTRAN Extended Version 4, FORTRAN Version 5, or COBOL Version 5. A knowledge of one of these languages is required. TAF does not provide COMPASS macros for CRM; the COMPASS user must use the FORTRAN calling sequence.

The user must also be familiar with basic NOS operations such as running a compilation job. Refer to Related Publications for the manuals containing this information. Knowledge of CRM is also assumed.

The user is required to have knowledge of the TAF executive requests, which are described in the TAF Version 1 Reference Manual.

ORGANIZATION

Section 1 describes the features and capabilities of TAF/CRM. Section 2 explains the file organizations available, file locking, and record locking. Sections 3 and 4 give the COBOL and FORTRAN interfaces, respectively.

Sections 5 through 8 document the recovery features built into TAF/CRM. Section 5 describes automatic data base recovery. Section 6 describes batch recovery. Section 7 provides a description of the requests available to system tasks. Section 8 discusses various recovery situations and makes suggestions to the data administrator for dealing with them.

Section 9 describes the TAF/CRM batch concurrency feature.

TERMINOLOGY

References in this publication to FORTRAN mean FORTRAN Extended Version 4 or FORTRAN Version 5. References to COBOL mean COBOL Version 5.

The term alphanumeric refers to any combination of letters A through Z and numbers 0 through 9. Special characters are not included.

Models 815, 825, 835, 845, and 855 of the CDC CYBER 170 Computer System share many of the functional and architectural attributes of the CDC CYBER 180 Computer System (models 810, 830, 835, 845, and 855). The term CYBER 180-class machines describes these similar models collectively.

Extended memory for model 176 is large central memory extended (LCME). Extended memory for models 865, 875, and CYBER 180-class machines is unified extended memory (UEM). Extended memory for models 865 and 875 may also include either extended core storage (ECS) or extended semiconductor memory (ESM). Extended memory for all other NOS computer systems is either ECS or ESM. ECS and ESM are the only forms of extended memory that can be shared in a linked shared device multimainframe complex and can be accessed by a distributive data path (DDP).

In this manual extended memory refers to all forms of extended memory unless otherwise noted.

RELATED PUBLICATIONS

The NOS Manual Abstracts is a pocket-sized manual containing brief descriptions of the contents and intended audience of all NOS and NOS product manuals. The abstracts can be useful in determining which manuals are of greatest interest to a particular user.

Control Data also publishes a Software Publications Release History of all software manuals and revision packets it has issued. This history report lists the revision level of a particular manual that corresponds to the level of software installed at the site.

These manuals are available through Control Data sales offices or Control Data Literature Distribution Services (308 North Dale, St. Paul, Minnesota 55103).

The following publications provide information on related topics.

Control Data Publication	Publication Number
COBOL Version 5 Reference Manual	60497100
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
FORTRAN Extended Version 4 Reference Manual	60497800
FORTRAN Version 5 Reference Manual	60481300
NOS Version 2 Installation Handbook	60459320

Control Data Publication	Publication Number
NOS Version 2 Manual Abstracts	60485500
NOS Version 2 Operations Handbook	60459310
NOS Version 2 Reference Set, Volume 2, Guide to System Usage	60459670
NOS Version 2 Reference Set, Volume 3, System Commands	60459680
NOS Version 2 Reference Set, Volume 4, Program Interface	60459690
TAF Version 1 Reference Manual	60459500
TAF Version 1 User's Guide	60459520
Software Publications Release History	60481000
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
Network Products Network Access Method Version 1/ Communications Control Program Version 3 Terminal Interfaces Reference Manual	60480600

DISCLAIMER

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION TO TAF/CRM 1

The transaction facility provides COBOL and FORTRAN interfaces to CRM AAM indexed sequential, direct access, and actual key files. CRM AAM Multiple Index Processing (MIP) is also supported. COMPASS programmers must use the FORTRAN calling sequence, described in the FORTRAN reference manuals (refer to preface for the publication numbers).

TAF/CRM requests correspond to existing CRM requests but differ in format. Additional requests lock records or files for use in a multiuser environment and provide for recovery. The requests allow the task to:

- Read records sequentially and by primary or alternate key.
- Define begin-commit sequences for recovery purposes.
- Lock and unlock records and files.
- Write, update, and delete records.
- Rewind a file.
- Up, down, and status data buses.
- Skip logical records of a file forward and backward.

Only those AAM capabilities described in this manual are supported. The user cannot issue CRM AAM macros in the TAF environment.

The following sequence is recommended for implementing an application using TAF/CRM in a transaction environment.

1. Read the TAF Reference Manual.
2. Read the CRM AAM Reference Manual.
3. Read this manual.
4. Design the data base by choosing file types, record and block sizes, key types and sizes, and any other data base parameters.
5. Create the data base using batch CRM.
6. Test the transaction tasks using batch CRM requests. This step is optional if the tasks can be written immediately for the transaction environment.
7. Convert or write the transaction tasks to use TAF/CRM requests.
8. Build a task library by following the instructions given in the TAF Reference Manual.
9. Install or update the TAF configuration file (TCF) and xxJ file.
10. Test the application using TAF.

This section discusses some of the characteristics of indexed sequential files, direct access files, actual key files, multiple key files, file locking, and record locking. It also examines the problem of deadlock avoidance and how TAF/CRM handles the problem.

FILE CHARACTERISTICS

TAF/CRM supports extended indexed sequential files, extended direct access files, and extended actual key files. All three types allow the task to specify a key to access a record. The value of the key determines the location of the record. A key has any of the following forms.

- A 60-bit integer number (10 digits).
- An integer key assigned by AAM (AK files).
- A symbolic string (1 to 255 contiguous alphanumeric characters).

For indexed sequential files, CRM collates the key before using it. Characters in a symbolic string are collated according to the Control Data or ASCII collating sequence or to a user-supplied collating sequence. Numeric keys are collated by value.

There are three types of keys: primary key, alternate key, and major key.

A primary key is the entire key of a record. The task may specify the primary key to access a record in a file.

An alternate key is a portion of the record, separate from the primary key, that can also be used to access the record in multiple key (MIP) files.

A major key is the leading portion of a primary key. It may be used to access a record in an indexed sequential file and to establish position in a multiple key file. Major keys are useful when the user wants to read only those records whose leading portion contents are greater than or equal to the contents of the specified major key.

Actual key and direct access files, being accessed by primary key, require one disk access to obtain a record if the record is not in an overflow block, whereas indexed sequential files might require more than one disk access. For this reason, record access is usually faster on direct access and actual key files than on indexed sequential files.

The FLSTAT utility gives statistical information about the file since creation; this helps the data base administrator determine if the file needs reorganization.

The CRM AAM Reference Manual provides more information about these files.

INDEXED SEQUENTIAL FILES

Indexed sequential files are mass storage files in which CRM stores records in sorted order by primary key. The task can access records individually by key or sequentially. Therefore, the advantages of random access are combined with those of sorted sequential access.

CRM facilitates random access by creating and maintaining an index that links the key of each record with a location in a file. CRM ensures that the physical order of the records is the same as the logical key order, thereby allowing sorted sequential access.

CRM maintains the index in fixed-length index blocks, which are separate from the fixed-length data blocks. Both types of blocks are part of the same file. The index blocks contain the primary keys, and the data blocks contain the records.

The user must create indexed sequential files in a batch environment before using them in a batch or a transaction environment. The FLBLOK utility gives suggested MBL values for a given file. The CRM AAM Reference Manual provides more information about this utility and indexed sequential files.

DIRECT ACCESS FILES

Direct access files are mass storage files in which CRM stores records by using a randomizing (hashing) algorithm on the primary key. In this type of file, the physical order of the records is different from the logical key order.

CRM does not use an index for the keys. The task specifies the primary key, and CRM uses the hashing algorithm to locate the record.

Direct access files contain fixed-length blocks. The user must create direct access files in a batch environment before using them in a batch or a transaction environment.

CRM provides two utilities for use with direct access files. The key analysis utility tests hashing algorithms for effectiveness in providing uniform distribution of record keys. The CREATE utility creates direct access files more rapidly than the conventional method of reading input and calling CRM to write each record; Sort/Merge is required when using this utility.

ACTUAL KEY FILES

Actual key files contain records whose key values specify the actual storage location of a record in the file. In contrast to the keys for direct access or indexed sequential files that require the system

to translate a key value to a file position, the keys of actual key files provide the system with file position. Actual key files use a block number and a record slot number in that block as the key for a given record.

An actual key has no logical connection with the record it identifies. This use of keys differs from that in direct and indexed sequential file organizations. In these file organizations, keys are provided by the user; typically they contain information that has external meaning, such as employee name or inventory part number. Actual keys, on the other hand, have no purpose other than unique identification and location of records. However, alternate keys can be defined for actual key files thus providing the ability to access the file by externally meaningful keys. The user must create actual key files before using them under TAF.

MULTIPLE KEY FILES

Multiple key files are files of one of the preceding types associated with an index file. The index file describes alternate keys, which must be embedded in the data record. For each alternate key defined, the index file establishes a correspondence between each alternate key value on one or several primary key values. The index file has a three-level indexed sequential structure. Alternate key access to a record is done through the index file by first finding the corresponding primary key, then retrieving the record from the data file. Successive sequential accesses retrieve primary keys associated with an alternate key value and its successors. It is the user's responsibility to determine the alternate key value from the record returned since the key is embedded in the record. An end of key status is returned before each alternate key change. Accessing a file by alternate key establishes that alternate key as the current key for further accesses. Operations that write on the file affect the current key or the position in terms of alternate key and associated primary key in that the key access will be set to the primary key after the operation is completed. However, changing the alternate key currently in use to a different alternate key sets the alternate key position to the first alternate key value satisfying the request. If the positioning request cannot be satisfied, the file is positioned at EOI.

ALTERNATE KEY USAGE

The TAF user specifies an alternate key using the key-identifier parameter on an appropriate request. The key-identifier specifies which alternate key description statement (AKY) in the xxJ file is to be used. An alternate key can be deleted from the index file by using the MIPGEN utility. To maintain the correct ordinals for the other alternate keys, the data base administrator must define a deleted alternate key in xxJ file.

FILE AND RECORD LOCKING

A task is a program with a TAF interface that performs a specific function. A task or several tasks perform a sequence of events called a transaction.

In a transaction environment, several transactions might attempt to update the same record simultaneously. TAF/CRM requests provide file and record locking capabilities to ensure that only one transaction updates a record at a time.

A simultaneous update occurs when two transactions read the same record and update it. Because there might have been a delay between the time the first transaction reads and updates a record, the second transaction might have read the record before the update occurred. If the second transaction makes a decision based on the contents of the record before the update occurred, an incorrect decision might be made.

To avoid this situation, the first transaction must lock the record, update it, and unlock it. The second transaction cannot lock the record until the first transaction unlocks it, thereby eliminating the update problem.

TAF/CRM keeps track of locks by transaction. Therefore, tasks in a transaction can access files and records locked by other tasks in that transaction.

DEADLOCKS

TAF/CRM also prevents deadlocks from occurring. A deadlock occurs when two transactions have locked different records or files and now want access to the record or file held by the other transaction. As a result, both transactions could end up in a waiting state. TAF/CRM returns an error status code and releases all record and file locks held by a transaction when the transaction is unable to lock a record or file. The transaction must reestablish all necessary locks.

Appendix C shows the error status codes returned to a transaction after attempting to perform a request on a file or record held by another transaction.

FILE ACCESS CONFLICTS

Another problem can occur when a transaction attempts to access a file. Each file allows a certain number of transactions to access it concurrently; the user's parameter in the xxJ file specifies this number. Refer to the TAF Reference Manual, System Files section. If a transaction tries to access a file and causes the user's parameter for the file to be exceeded, TAF/CRM returns an error status code and closes all files the transaction has opened. Therefore, it is usually better to OPEN all files before issuing any requests that manipulate (that is, read, write, rewrite, and so on) records in these files. Furthermore, transactions should unlock records and files as soon as possible to allow other transactions to access them.

NOTE

File locking is not normally used because of possible performance degradation. Locking at the record level is preferable.

This section describes the COBOL interface to TAF/CRM. Included are definitions of the request parameters, explanations of each request, and the calling formats for COBOL.

REQUEST PARAMETER DEFINITIONS

The parameters that appear in the requests described in this section are listed in table 3-1. All data names must conform to the rules for naming user-defined words (refer to the COBOL 5 Reference Manual).

REQUESTS

The remainder of this section explains each request and gives the calling format for COBOL. Optional parameters are enclosed in brackets. Refer to table 3-1 for descriptions of each parameter.

Errors in the calling format abort the job.

The task should examine the taf-status and crm-status parameter fields whenever applicable. The taf-status error status codes are listed in appendix A. The crm-status error status codes appear in the CRM AAM 2 Reference Manual.

Appendix B gives the following programming aids.

- A list of the error status codes that can be returned for each request.
- A table that gives the calling formats in COBOL under the batch and transaction environments.

OPEN REQUEST

The OPEN request allows access to a file by a transaction. The transaction must issue this request before issuing any other request on the file.

The format is:

```
ENTER "OPEN" USING xxpfni,taf-status,crm-status.
```

If the taf-status field indicates that not enough table space exists (error status code 6), TAF/CRM closes all files that the transaction has opened. If this happens within a begin-commit sequence, a DBFREE request is processed before returning. The transaction must reopen all required files.

If the taf-status field indicates that a CRM error has occurred, the file is not opened. The file may be defective and cannot be accessed.

The data administrator specifies the file name and other file characteristics when creating the xxJ file. Refer to the TAF 1 Reference Manual, System Files section.

OPEN requests may be issued before or after a DBEGIN request; however, it is better to open files before a DBEGIN request to eliminate the possibility of DBFREE processing.

CLOSE REQUEST

The CLOSE request terminates processing of a file by a transaction, thereby allowing an additional transaction to access the file.

The format is:

```
ENTER "CLOSE" USING xxpfni,taf-status,crm-status.
```

The transaction should close any file not in use because only a limited number of transactions can access a file simultaneously. The TAF CEASE request closes all files (refer to the TAF 1 Reference Manual).

A recoverable file may not be closed from within a begin-commit sequence.

DBEGIN REQUEST

The DBEGIN request designates the start of a begin-commit sequence (refer to section 5). A task must issue a DBEGIN request before it modifies any data base file(s) designated as recoverable in a CRM statement in the xxJ file. Only one begin-commit sequence is allowed at one time from a transaction. This means that a DBEGIN request must be closed by a corresponding DBFREE or DBCOMIT request before another DBEGIN can be issued. However, multiple nonnested begin-commit sequences are allowed in a single transaction. After a DBEGIN request, any lock conflict causes TAF/CRM to process a DBFREE request before returning.

The format is:

```
ENTER "DBEGIN" USING begin-id, taf-status.
```

Refer to section 5 for more information on using this request.

DBCOMIT REQUEST

The DBCOMIT request designates the end of a begin-commit sequence. A successful completion of this request means that all updates to recoverable files made by this transaction between the last DBEGIN request and this DBCOMIT request are permanently made to the data base; these changes cannot be rolled back. All records locked by the task are unlocked when this request is executed. All files locked by the task remain locked. This request has no effect on updates made to nonrecoverable files.

The format is:

```
ENTER "DBCOMIT" USING taf-status.
```

Refer to section 5 for more information on the use of this request.

DBFREE REQUEST

The DBFREE request terminates a begin-commit sequence and, in effect, cancels all updates made to recoverable files since the preceding DBEGIN request. All records locked by this task will be unlocked when this request is executed. All files locked by this task will remain locked.

LOCK, FLOCK, WRITE, REWRITE, and DELETE requests are processed like a DBFREE request if they cause a lock conflict. This is called internal DBFREE processing. If this happens, all file and record locks are released (unlocked). Internal DBFREE processing is performed on an OPEN request if a taf-status error 6 (not enough table space) occurs.

The format is:

```
ENTER "DBFREE" USING taf-status.
```

Refer to section 5 for more information on the use of this request.

DBSTAT REQUEST

The DBSTAT request returns the begin-commit identifiers for the current begin-commit sequence and the most recent successfully completed begin-commit sequence. This request may be used at any time by the task; however, its primary purpose is to enable restarted multiple begin-commit sequence transactions to determine the appropriate restart point. When used for this purpose, it must be executed upon restart prior to any other data manager requests.

The format is:

```
ENTER "DBSTAT" USING new-id, taf-status, old-id.
```

Refer to section 5 for more information on the use of this request.

READ REQUEST

The READ request reads a record into a working storage area. This request reads a specific record.

The format is:

```
ENTER "READ" USING xxpfni,taf-status,crm-status,
wsa-name,wsa-length,record-length,key-name,
key-offset[,key-status,key-identifier,key-
area,key-area-length,lock-status].
```

The READ request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transactions.

READN REQUEST

The READN request reads the next sequential record into a working storage area. The READN request returns the record whose key is next in the keylist being accessed. For direct access files being accessed by primary key, the READN request reads the next record regardless of the primary key.

The format is:

```
ENTER "READN" USING xxpfni,taf-status,crm-status,
wsa-name,wsa-length,record-length,key-area,
key-area-length[,key-status,lock-status].
```

The READN request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transaction.

READM REQUEST

The READM request reads a record into a working storage area using the major key. This record is the first record that has a primary key greater than or equal to the specified major key.

When access is by primary key, the READM request is valid only for indexed sequential files.

The format is:

```
ENTER "READM" USING xxpfni,taf-status,crm-status,
wsa-name,wsa-length,record-length,key-area,key-
area-length,key-name,key-offset,major-key-
length[,key-status,key-identifier,lock-status].
```

This request is useful when processing a group of records whose primary keys begin with the same characters.

The READM request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transaction.

Table 3-1. Request Parameters (Sheet 1 of 3)

Parameter	Type	Definition
xxpfni	01-level	A two- to seven-character file name specified in the xxJ file.
taf-status	01-level,computational-1	A data name that specifies the field to contain the error status code returned by TAF. Refer to appendix A for a list of the error status codes.
crm-status	01-level,computational-1	A data name that specifies the field to contain the error status code returned by CRM. This parameter has meaning only when the taf-status parameter is 8. Refer to the CRM AAM Reference Manual for a list of CRM error status codes. The CRM AAM Reference Manual lists the CRM error status codes as octal values; the programmer must convert them to decimal.
wsa-name	01-level	A data name that specifies the area to contain the record after issuing a read request. It is also the name of the area from which a write request obtains the record.
wsa-length	01-level,computational-1	A data name that specifies the length (in characters) of the area identified by the wsa-name parameter. This parameter cannot be less than the value of the mrl parameter in the xxJ file.
record-length	01-level,computational-1	A data name that specifies the length (in characters) of a record returned by TAF/CRM after issuing a read request. It is also the length of the record to be written to the data base from the area identified by the wsa-name parameter. This parameter cannot exceed the value of the mrl parameter in the xxJ file.
key-area	01-level	A data name that specifies the field to contain the primary key returned after issuing a read request on an indexed sequential file. This field must be large enough to contain the primary key.
key-area-length	01-level,computational-1	A data name that specifies the length (in characters) of the field identified by the key-area parameter. The value of this parameter cannot be less than the value of the kl parameter in the xxJ file.
key-name	01-level	A data name that contains the key currently being used to access the record (primary or alternate). The key need not be a 01-level data name. Refer to the key-offset parameter for an example.
key-offset	01-level,computational-1	A data name that specifies the beginning character position of the key (key-name parameter).

Table 3-1. Request Parameters (Sheet 2 of 3)

Parameter	Type	Definition
		For example, the following portion of the working-storage section contains the first part of a record called RECORD-1, and a three-character key (primary or alternate) called KEY-1. 01 RECORD-1. 02 FIELD-1 PICTURE 9(6). 02 KEY-1 PICTURE XXX. 02 FIELD-2 PICTURE 9(5). The primary key occupies character positions 7, 8, and 9 of the record. The key-offset parameter must be assigned the value 7. The key-name parameter must be RECORD-1, even though the name of the key is KEY-1.
major-key-length	01-level,computational-1	A data name that specifies the length (in characters) of the major key. The major key is the leading portion of the key. The length of the major key must be less than or equal to the length of the key. This parameter applies only to the READM and START requests.
count	01-level,computational-1	A data name that specifies the number of records to skip. This value must be positive. The request specifies either forward or backward.
begin-id	01-level	The begin-commit identifier. It is a task-supplied value, five characters in length, left-justified, with blank fill.
new-id	01-level	A data item to which the current begin-commit identifier is returned. It must be 10 characters (60 bits) in length. The identifier is returned to the leftmost five characters (upper 30 bits) of the word. If no current begin-commit identifier exists, blanks are returned. In either case, the rightmost five characters (lower 30 bits) are not changed.
old-id	01-level	A data item to which the most recent successfully completed begin-commit sequence identifier is returned. It must be 10 characters (60 bits) in length. The identifier is returned to the leftmost five characters (upper 30 bits) of the word. If no begin-commit sequence has successfully completed, blanks are returned. In either case, the rightmost five characters (lower 30 bits) are not changed.
key-status	01-level,computational-1	A data item to which TAF/CRM returns the key status code. 0 On key. 1 Not on key.† 2 End of key. 4 End of information for this key.

†For the key-relation value GE, the key-status is 1 if an equal key value is not in the index file.

Table 3-1. Request Parameters (Sheet 3 of 3)

Parameter	Type	Definition
key-identifier	01-level,computational-1	A data item containing a value that specifies the key (alternate or primary) for the access. < 0 No change in key access. 0 Primary key. N Value associated with the key by an AKY† statement and assigned by the data base administrator. The AKY statement describes the alternate key.
key-relation	01-level	A data item containing a value that specifies the position of the file, relative to the given key, after the operation. EQ On the key. GE On or after the key. GT After the key.
lock-status	01-level,computational-1	The data item to which TAF/CRM returns the lock status of the record just read. 0 Neither the record nor the file was locked by another transaction. 2 The file containing the record was locked by another transaction. 3 The record was locked by another transaction.

† Refer to the description of the xxJ file in the TAF Reference Manual.

READL REQUEST

The READL request reads a record into a working storage area and locks the record.

The format is:

```
ENTER "READL" USING xxpfni,taf-status,crm-status,
  wsa-name,wsa-length,record-length,key-name,
  key-offset[,key-status,key-identifier,key-area,
  key-area-length].
```

The READL request permits other transactions to access the record only for reading (no locking is allowed). The transaction issuing the READL request should unlock the record when the record is no longer needed (refer to UNLOCK Request).

If the lock cannot be granted, a taf-status error occurs, and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed. If a crm-status error occurs, the record remains locked; this allows the transaction to modify the record (using the REWRITE request) without allowing another transaction to modify it.

READNL REQUEST

The READNL request reads the next sequential record into a working storage area and locks the record. The READNL request reads the record having the next highest key. For direct access files being accessed by primary key, the READNL request reads the next record regardless of the primary key.

The format is:

```
ENTER "READNL" USING xxpfni,taf-status,
  crm-status,wsa-name,wsa-length,record-length,
  key-area,key-area-length[,key-status].
```

The READNL request permits other transactions to access the record only for reading (no locking is allowed). The transaction issuing the READNL request should unlock the record when the record is no longer needed (refer to UNLOCK Request).

If the lock cannot be granted, a taf-status error occurs, and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed. If a crm-status error occurs, the record remains locked.

WRITE REQUEST

The WRITE request writes a new record.

The format is:

```
ENTER "WRITE" USING xxpfni,taf-status,crm-status,
  wsa-name,record-length,key-name,key-offset
  [,key-area,key-area-length].†
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record on the file if the record is not already locked by this transaction.
3. Write an empty before image of the record to the before image recovery file.
4. Write the record on the file.
5. Write the after image of the record to the after image recover file.
6. Return control to the calling task.

If the lock can be granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

REWRITE REQUEST

The REWRITE request replaces an existing record.

The format is:

```
ENTER "REWRITE" USING xxpfni,taf-status,
  crm-status,wsa-name,record-length,key-name,
  key-offset.
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record, if it is not already locked by this transaction.
3. Write the before image of the record to the before image recovery file.
4. Update the record.
5. Write the after image of the record to the after image recovery file.
6. Return control to the calling task.

If the lock is granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

†Optional parameters are for actual key files.

DELETE REQUEST

The DELETE request deletes a record.

The format is:

```
ENTER "DELETE" USING xxpfni,taf-status,
  crm-status,key-name,key-offset.
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record if it is not already locked by this transaction.
3. Write the before image of the record to the before image recovery file.
4. Delete the record.
5. Write an empty after image record to the after image recovery file.

If the lock can be granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

LOCK REQUEST

The LOCK request locks a record.

The format is:

```
ENTER "LOCK" USING xxpfni,taf-status,key-name,
  key-offset.
```

The LOCK request permits a transaction to lock a group of records before performing any requests on them, thereby saving time if TAF/CRM cannot grant one of the locks.

The transaction issuing the LOCK request should unlock the record when the record is no longer needed (refer to UNLOCK Request).

UNLOCK REQUEST

The UNLOCK request unlocks a record.

The format is:

```
ENTER "UNLOCK" USING xxpfni,taf-status,key-name,
  key-offset.
```

This request is not allowed within a begin-commit sequence for records in recoverable files. A taf-status error (29) is returned.

FLOCK REQUEST

The FLOCK request locks a file.

The format is:

```
ENTER "FLOCK" USING xxpfni,taf-status.
```

The FLOCK request is useful when not enough record lock table entries exist or when certain operations (such as a dump) are required on an entire file or when there is a need to retain the locks after a DBCOMIT or DBFREE request.

The transaction issuing the FLOCK request should unlock the file when the file is no longer needed (refer to UNFLOCK Request).

If the lock cannot be granted, a taf-status error occurs, and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed.

UNFLOCK REQUEST

The UNFLOCK request unlocks a file.

The format is:

```
ENTER "UNFLOCK" USING xxpfni,taf-status.
```

This request is not allowed within a begin-commit sequence for recoverable files. A taf-status error (29) is returned.

REWIND REQUEST

The REWIND request rewinds a file to the beginning of information or the beginning of the key sequence.

The format is:

```
ENTER "REWIND" USING xxpfni,taf-status,
  crm-status.
```

SKIPFL REQUEST

The SKIPFL request positions a file forward the number of records specified by the count parameter. This is done according to the current key sequence except for direct access files being accessed by primary key.

The format is:

```
ENTER "SKIPFL" USING xxpfni,taf-status,
  crm-status,count.
```

This request is not applicable to non-MIPPED direct access files.

SKIPBL REQUEST

The SKIPBL request positions a file backward the number of records specified by the count parameter.

The format is:

```
ENTER "SKIPBL" USING xxpfni,taf-status,
  crm-status,count.
```

This request is not applicable to non-MIPPED direct access files.

START REQUEST

The START request positions a file at or after a given key.

The format is:

```
ENTER "START" USING xxpfni,taf-status,crm-status,
  key-relation,key-name,key-offset[,key-status,
  key-identifier,major-key-length].
```

This section describes the FORTRAN interface to TAF/CRM. Included are definitions of the request parameters, explanations of each request, and the calling formats for FORTRAN.

REQUEST PARAMETER DEFINITIONS

The parameters that appear in the requests described in this section are listed in table 4-1. All data names must conform to the rules for naming variables and arrays (refer to the FORTRAN Reference Manuals).

REQUESTS

The remainder of this section explains each request and gives the calling format for FORTRAN. Optional parameters are enclosed in brackets. Refer to table 4-1 for descriptions of each parameter.

Errors in the calling format abort the job.

The task should examine the taf-status and crm-status parameter fields whenever applicable. The taf-status error status codes are listed in appendix A. The crm-status error status codes appear in the CRM AAM Reference Manual.

Appendix B gives the following programming aids.

- A list of the error status codes that can be returned for each request.
- A table that gives the calling formats in FORTRAN under the batch and transaction environments.

OPEN REQUEST

The OPEN request allows access to a file by a transaction. The transaction must issue this request before issuing any other request on the file.

The format is:

```
CALL OPEN(xxpfni,taf-status,crm-status)
```

If the taf-status field indicates that not enough table space exists (error status code 6), TAF/CRM closes all files that the task has opened. If this happens within a begin-commit sequence, a DBFREE request is processed before returning. The transaction must reopen all required files.

If the taf-status field indicates that a CRM error has occurred, the file is not rewound. The file is defective and cannot be accessed.

The data base administrator specifies the file name and other file characteristics when creating the xxJ file. Refer to the System Files section of the TAF Reference Manual.

OPEN requests may be issued before or after a DBEGIN request; however, it is better to open files before a DBEGIN request to eliminate the possibility of DBFREE processing.

CLOSE REQUEST

The CLOSE request terminates processing of a file by a transaction, thereby allowing an additional transaction to access the file.

The format is:

```
CALL CLOSE(xxpfni,taf-status,crm-status)
```

The transaction should close any file not in use because only a limited number of transactions can access a file simultaneously. The TAF CEASE request closes all files (refer to the TAF Reference Manual).

A recoverable file may not be closed from within a begin-commit sequence.

DBEGIN REQUEST

The DBEGIN request designates the start of a begin-commit sequence (refer to section 5). A task must issue a DBEGIN request before it modifies any data base file(s) designated as recoverable in a CRM statement in the xxJ file. Only one begin-commit sequence is allowed at one time from a transaction. This means that a DBEGIN request must be closed by a corresponding DBFREE or DBCOMIT request before another DBEGIN can be issued. However, multiple nonnested begin-commit sequences are allowed in a single transaction. After a DBEGIN request, any lock conflict causes TAF/CRM to process a DBFREE request before returning.

The format is:

```
CALL DBEGIN (begin-id,taf-status)
```

Refer to section 5 for more information on the use of this request.

DBCOMIT REQUEST

The DBCOMIT request designates the end of a begin-commit sequence. A successful completion of this request means that all updates to recoverable

files made by this transaction between the last DBEGIN request and this DBCOMIT request are permanently made to the data base; these changes cannot be rolled back. All records locked by the task are unlocked when this request is executed. All files locked by the task remain locked. This request has no effect on updates made to nonrecoverable files.

The format is:

```
CALL DBCOMIT (taf-status)
```

Refer to section 5 for more information on the use of this request.

DBFREE REQUEST

The DBFREE request terminates a begin-commit sequence and, in effect, cancels all updates made to recoverable files since the preceding DBEGIN request. All records locked by this task will be unlocked when this request is executed. All files locked by this task will remain locked.

LOCK, FLOCK, WRITE, REWRITE, and DELETE requests are processed like a DBFREE request if they cause a lock conflict. This is called internal DBFREE processing. If this happens, all file and record locks are released (unlocked). Internal DBFREE processing is performed on an OPEN request if a taf-status error 6 (not enough table space) occurs.

The format is:

```
CALL DBFREE (taf-status)
```

Refer to section 5 for more information on the use of this request.

DBSTAT REQUEST

The DBSTAT request returns the begin-commit identifiers for the current begin-commit sequence and the most recent successfully completed begin-commit sequence. This request may be used at any time by the task; however, its primary purpose is to enable restarted multiple begin-commit sequence transactions to determine the appropriate restart point. For this purpose, it must be executed upon restart prior to any other recovery requests.

The format is:

```
CALL DBSTAT (old-id,taf-status,new-id)
```

READ REQUEST

The READ request reads a record into an array. This request reads a specific record.

The format is:

```
CALL READ(xxpfni,taf-status,crm-status,wsa-name,
  wsa-length,record-length,key-name,key-offset
  [,key-status,key-identifier,key-area,
  key-area-length,lock-status])
```

The READ request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transaction.

READN REQUEST

The READN request reads the next sequential record into an array. The READN request returns the record whose key is next in the keylist currently being accessed. For direct access files being accessed by primary key, the READN request reads the next record regardless of the primary key.

The format is:

```
CALL READN(xxpfni,taf-status,crm-status,wsa-name,
  wsa-length,record-length,key-area,key-area-
  length[,key-status,lock-status])
```

The READN request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transaction.

READM REQUEST

The READM request reads a record into an array. This record is the first record that has a primary key greater than or equal to the specified major key. When access is by primary key, the READM request is valid only for indexed sequential files.

The format is:

```
CALL READM(xxpfni,taf-status,crm-status,wsa-name,
  wsa-length,record-length,key-area,key-area-
  length,key-name,key-offset,major-key-length
  [,key-status,key-identifier,lock-status])
```

This request is useful when processing a group of records whose keys begin with the same characters.

The READM request permits a transaction to read a record that another transaction has locked. The lock status of the record is available to the reading transaction.

Table 4-1. Request Parameters (Sheet 1 of 2)

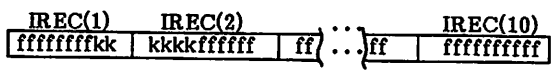
Parameter	Type	Definition
xxpfni	integer (FTN4) character (FTN5)	A two- to seven-character file name specified in the xxJ file. This field must be declared as Hollerith (in the form nHf or nLf).
taf-status	integer	A data name that specifies the field to contain the error status code returned by TAF. Refer to appendix A for a list of error status codes.
crm-status	integer	A data name that specifies the field to contain the error status code returned by CRM. This parameter has meaning only when the taf-status parameter is 8. Refer to the CRM AAM Reference Manual for a list of CRM error status codes.
wsa-name	integer	A data name that specifies the integer area to contain the record after issuing a read request. It is also the name of the area from which a write request obtains the record.
wsa-length	integer	A data name that specifies the length (in characters) of the area identified by the wsa-name parameter. This parameter cannot be less than the value of the mrl parameter in the xxJ file.
record-length	integer	A data name that specifies the length (in characters) of a record returned by TAF/CRM after issuing a read request. It is also the length of the record to be written to the data base from the area identified by the wsa-name parameter. For write operations, this parameter cannot exceed the value of the mrl parameter in the xxJ file.
key-area	integer	A data name that specifies the field to contain the primary key returned after issuing a read request on an indexed sequential file. This field must be large enough to contain the primary key.
key-area-length	integer	A data name that specifies the length (in characters) of the field identified by the key-area parameter. The value of this parameter cannot be less than the value of the kl parameter in the xxJ file.
key-name	integer	A data name that contains the key currently being used to access the record (primary or alternate). The key need not begin on a word boundary and can span a word boundary. Refer to the key-offset parameter for an example.
key-offset	integer	A data name that specifies the beginning character position of the key (key-name parameter). For example, the following 10-word array is a record called IREC with a six-character key (kkkkkk).  The key (primary or alternate) occupies character positions 9 and 10 of IREC(1) and positions 1 through 4 of IREC(2). The key-offset parameter must be assigned the value 9 because the first character of the key is character position 9 of IREC(1). IREC(1) is also the value of the key-name parameter, even though the key spans more than one word.
major-key-length	integer	A data name that specifies the length (in characters) of the major key. The major key is the leading portion of the primary key. The length of the major key must be less than or equal to the length of the primary key. This parameter applies only to the READM and START request.

Table 4-1. Request Parameters (Sheet 2 of 2)

Parameter	Type	Definition
count	integer	A data name that specifies the number of records to skip. This value must be positive. The request specifies either forward or backward.
begin-id	integer (FTN4) character (FTN5)	The begin-commit identifier. It is a task-supplied value, five characters in length, left justified, with blank fill.
new-id	integer (FTN4) character (FTN5)	A variable to which the current begin-commit identifier is returned. It must be 10 characters (60 bits) in length. The identifier is returned to the leftmost five characters (upper 30 bits) of the word. If no current begin-commit identifier exists, binary zeros are returned. In either case the rightmost five characters (lower 30 bits) are not changed.
old-id	integer (FTN4) character (FTN5)	A variable to which the identifier of the most recent successfully completed begin-commit sequence is returned. It must be 10 characters (60 bits) in length. The identifier is returned to the leftmost five characters (upper 30 bits) of the word. If no begin-commit sequence has successfully completed, binary zeros are returned. In either case, the rightmost five characters (lower 30 bits) are not changed.
key-status	integer	A variable to which TAF/CRM returns the key status code. 0 On key. 1 Not on key.† 2 End of key. 4 End of information for this key.
key-identifier	integer	A variable containing a value that specifies the key (alternate or primary) for the access. < 0 No change in key access. 0 Primary key. N Value associated with the key by an AKY†† statement and assigned by the data base administrator. The AKY statement describes the alternate key.
key-relation	integer (FTN4) character (FTN5)	A variable containing a value that specifies the position of the file, relative to the given key, after the operation. EQ On the key. GE On or after the key. GT After the key.
lock-status	integer	The variable to which TAF/CRM returns the lock status of the record just read. 0 Neither the record nor the file was locked by another transaction. 2 The file containing the record was locked by another transaction. 3 The record was locked by another transaction.

†For the key-relation value GE, the key-status is 1 if an equal key value is not in the index file.
††Refer to the description of the xxJ file in the TAF Reference Manual.

READL REQUEST

The READL request reads a record into an array and locks the record.

The format is:

```
CALL READL(xxpfni,taf-status,crm-status,wsa-name,
  wsa-length,record-length,key-name,key-offset
  [,key-status,key-identifier,key-area,
  key-length])
```

The READL request permits other transactions to access the record only for reading (no locking is allowed). The transaction issuing the READL request should unlock the record when the record is no longer needed (refer to UNLOCK Request later in this section).

If the lock cannot be granted, a taf-status error occurs and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed. If a crm-status error occurs, the record remains locked; this allows the transaction to modify the record (using the REWRITE request) without allowing another transaction to modify it.

READNL REQUEST

The READNL request reads the next sequential record into an array and locks the record. The READNL request reads the record having the next highest key. For direct access files being accessed by primary key, the READNL request reads the next record regardless of the primary key.

The format is:

```
CALL READNL(xxpfni,taf-status,crm-status,
  wsa-name,wsa-length,record-length,key-area,
  key-area-length[,key-status])
```

The READNL request permits other transactions to access the record only for reading (no locking is allowed). The transaction issuing the READNL request should unlock the record when the record is no longer needed (refer to UNLOCK Request).

If the lock cannot be granted, a taf-status error occurs and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed. If a crm-status error occurs, the record remains locked.

WRITE REQUEST

The WRITE request writes a new record.

The format is:

```
CALL WRITE(xxpfni,taf-status,crm-status,wsa-name,
  record-length,key-name,key-offset[,key-area,
  key-area-length])†
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record on the file if the record is not already locked by this transaction.
3. Write an empty before image record to the before image recovery file if the data file is designated recoverable.
4. Write the record on the file.
5. Write the after image of the record to the after image recovery file if the data file is designated recoverable.
6. Return control to the calling task.

If the lock can be granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

REWRITE REQUEST

The REWRITE request replaces an existing record.

The format is:

```
CALL REWRITE(xxpfni,taf-status,crm-status,
  wsa-name,record-length,key-name,key-offset)
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record if it is not already locked by this transaction.
3. Write the before image of the record to the before image recovery file if the data file is designated recoverable.
4. Update the record.
5. Write the after image of the record to the after image recovery file if the data file is designated recoverable.
6. Return control to the calling task.

If the request is granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

†Optional parameters for use with actual key files.

DELETE REQUEST

The DELETE request deletes a record.

The format is:

```
CALL DELETE(xxpfni,taf-status,crm-status,
  key-name,key-offset)
```

Internally, the sequence for this request is:

1. If the request pertains to a recoverable file and no DBEGIN is outstanding, return with a taf-status error of 30; otherwise, continue with the following steps.
2. Lock the record if it is not already locked by this transaction.
3. Write the before image of the record to the before image recovery file if the data file is designated recoverable.
4. Delete the record.
5. Write an empty after image record to the after image recovery file if the data file is designated recoverable.
6. Return control to the calling task.

If the lock can be granted, the record remains locked after the request is processed. If the lock cannot be granted, a taf-status error occurs and TAF/CRM performs a DBFREE before returning control to the task.

LOCK REQUEST

The LOCK request locks a record.

The format is:

```
CALL LOCK(xxpfni,taf-status,key-name,key-offset)
```

The LOCK request permits a transaction to lock a group of records before performing any requests on them, thereby saving time if TAF/CRM cannot grant one of the locks.

The transaction issuing the LOCK request should unlock the record when the record is no longer needed (refer to UNLOCK Request).

If another transaction has locked the record or if not enough record lock table entries exist, TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed.

UNLOCK REQUEST

The UNLOCK request unlocks a record.

The format is:

```
CALL UNLOCK(xxpfni,taf-status,key-name,
  key-offset)
```

This request is not allowed within a begin-commit sequence for records in recoverable files. A taf-status error (29) is returned.

FLOCK REQUEST

The FLOCK request locks a file.

The format is:

```
CALL FLOCK(xxpfni,taf-status)
```

The FLOCK request is useful when not enough record lock table entries exist, when certain operations (such as a dump) are required on an entire file, or when there is a need to retain the locks after a DBCOMIT or DBFREE request.

The transaction issuing the FLOCK request should unlock the file when the file is no longer needed (refer to UNFLOCK Request).

If the lock cannot be granted, a taf-status error occurs and TAF/CRM releases all record and file locks held by the transaction. If a begin-commit sequence is active when this happens, a DBFREE request is performed.

UNFLOCK REQUEST

The UNFLOCK request unlocks a file.

The format is:

```
CALL UNFLOCK(xxpfni,taf-status)
```

This request is not allowed within a begin-commit sequence for recoverable files. A taf-status error (29) is returned.

REWIND REQUEST

The REWIND request rewinds a file to the beginning of information.

The format is:

```
CALL REWIND(xxpfni,taf-status,crm-status)
```

SKIPFL REQUEST

The SKIPFL request positions a file forward the number of records specified by the count parameter.

The format is:

```
CALL SKIPFL(xxpfni,taf-status,crm-status,count)
```

This request is not applicable to non-MIPPED direct access file.

SKIPBL REQUEST

The SKIPBL request positions a file backward the number of records specified by the count parameter.

The format is:

```
CALL SKIPBL(xxpfni,taf-status,crm-status,count)
```

This request is not applicable to non-MIPPED direct access files.

START REQUEST

The START request positions a file at or after a given key.

The format is:

```
CALL START (xxpfni,taf-status,crm-status,key-
  relation,key-name,key-offset[,key-status,
  key-identifier,major-key-length])
```

TAF/CRM automatic recovery maintains data bases in a consistent state across failures. It does this by introducing the concept of a begin-commit sequence and ensuring that either all or none of the data base updates within a begin-commit sequence are performed. Transactions are given a means to determine the last successful begin-commit sequence and can thereby always determine the state of the data base. An overview of the automatic recovery scheme is presented in the TAF Reference Manual. The recovery sections in that manual should be read prior to reading this section.

BEGIN-COMMIT SEQUENCE

A begin-commit sequence is a series of data base updates preceded by a begin (DBEGIN) request and terminated by a commit (DBCOMIT), free (DBFREE), or a transaction CEASE request.

A transaction (recoverable or nonrecoverable) may logically group one or more updates into a single entity, the begin-commit sequence. The TAF/CRM data manager then ensures that in most cases either all of these updates or none of these updates are applied to the data base. The former happens only on a commit request, the latter on a free, transaction CEASE (without a prior commit) or after a system or hardware failure.

A recoverable transaction can update both recoverable and nonrecoverable files. Recoverable files can be modified only from within a begin-commit sequence. This restriction does not apply for modifying nonrecoverable files.

A begin-commit sequence containing no updates to recoverable files serves no useful purpose but does not result in an error condition.

RECOVERY REQUESTS

The DBEGIN and the DBCOMIT requests are explicit requests made by the application program. The DBFREE may be application program requested or invoked by the TAF/CRM data manager. In either case a DBFREE request terminates a begin-commit sequence, as does a transaction CEASE request. The DBEGIN request is the only request that initiates a begin-commit sequence. Another recovery related request, DBSTAT, is discussed in the following subsection.

All TAF/CRM requests except CLOSE, UNLOCK, UNFLOCK (for recoverable files), and another DBEGIN request are allowed inside a begin-commit sequence. As a result, a transaction may open or lock files and records as desired. However, if a lock conflict occurs as a result of these requests, an implicit free will take place. If no errors are detected during freeing, the lock error is returned to the task.

TAF/CRM imposes DBFREE processing (forced freeing) on a transaction when certain errors occur for the requests listed at the end of this paragraph. This is done so that resources held by the transaction are released as soon as possible. The forced freeing process is completed prior to informing the transaction of the requests failure. If rollback of the transaction updates (during the forced freeing process) is accomplished without another error occurring, the error code that caused TAF/CRM to impose forced freeing (initial error code) is reported to the transaction. If an error does occur during the forced freeing process, that error code (subsequent error code) is reported to the transaction.

Forced Freeing for Request	Initial Error Code	Subsequent Error Code
OPEN	6	8 or 32
LOCK,FLOCK, READL,READNL, DELETE,WRITE, REWRITE	2, 3, or 7	8 or 32

An implicit DBFREE is processed during recovery for each transaction active at the time of failure. This means that no locks or files are retained; therefore, after establishing appropriate restart point, the application program must open all necessary files, lock records, and perform a DBEGIN request before updating files.

BEGIN-COMMIT IDENTIFIER

Every begin-commit sequence has a five character (30-bit) identifier called the begin-commit identifier or the begin-id. A begin-commit identifier is supplied by the application program on every DBEGIN request. This application program-supplied string of characters becomes the begin-id of the current begin-commit sequence. It remains the current begin-id until a DBCOMIT request is processed, at which time it becomes the previous begin-id and the current begin-id is set to binary zero. DBFREE processing, whether imposed or requested, does not change either the current begin-id or the previous begin-id.

The TAF/CRM data manager retains the begin-ids of the previous and the current begin-commit sequence across a system failure. The begin-ids are written on the data base before image log (called the before image recovery file) along with each before image for the begin-commit sequence. If the before image recovery file (BRF) becomes unreadable, then the begin-ids on that BRF are lost.

The current and previous begin identifiers of a transaction are saved until the transaction CEASE request is processed by the data manager.

The DBSTAT request may be used at any time by a transaction to obtain the begin-ids of the previous and the current begin-commit sequences. The request should normally be used after recovery to establish the begin-id of the last completed begin-commit sequence. The recovered transaction can then transfer processing to the appropriate place within the transaction.

The TAF Reference Manual, section 7, includes examples showing the function described here for the DBSTAT request being performed by the TAF TSTAT request. When issued prior to any other recovery requests, the current and previous begin-ids returned by the TAF TSTAT and the TAF/CRM DBSTAT requests are identical. A TSTAT request issued at any time during the transaction returns the current and previous begin-id that existed at the start of the run unit. A DBSTAT request returns the values reflecting the current condition.

The begin-ids are affected by transaction requests as follows:

Request	Effect on begin-id
DBEGIN	Establish current begin-id as specified.
DBCOMIT	The current begin-id becomes the previous begin-id, current begin-id is set to zero.
DBFREE	Has no effect on the current or previous begin-id.
RSTDBI	Establish current and previous begin-ids as specified in the request. (System tasks only. Refer to section 7.)

Applications programmers must ensure that begin-commit identifiers are nonzero and unique within a recoverable transaction in order for the transaction restart point to be unambiguous. TAF/CRM makes no check for this.

DESIGNATING RECOVERABLE FILES

With the TAF/CRM automatic recovery feature, recovery can be selected on a file by file basis. Within a single TAF/CRM data base some files may be recoverable while others are not.

Recoverable files are indicated by the recovery parameter in the CRM statement in the xxJ file (refer to the TAF Reference Manual). However, a data base may have recoverable files only if there is one or more before image recovery files (BRF) specified for that data base. This is done by including a BRF,n. statement in the xxJ file, with n being greater than zero. If no before image recovery files are specified for a data base, then no recovery is available for files belonging to this data base, even if CRM statements have recovery specified for one or more files. Refer to the System Files section of the TAF Reference Manual for detailed information regarding the format of the xxJ file.

RECOVERY FILES

To support automatic recovery, before images of modified records are stored on a before image recovery file (BRF). To support batch recovery, after images of modified records are stored on the after image recovery files (ARF).

Before and after image recovery files are associated with TAF/CRM data bases. There may be 1 to 63 BRFs per data base, there are always exactly two ARFs per data base. All recovery files are preallocated direct access permanent files. They may not reside on private packs. They must be defined under the user name given in the xxJ file for the data base to which they apply. These files are created automatically during TAF initialization for each recoverable data base for which they do not already exist. Refer to the CREATE directive in section 6 for information on manual creation of these files.

No before or after images are kept for nonrecoverable files, so no recovery is supported, even if the nonrecoverable file is modified inside a begin-commit sequence.

BEFORE IMAGE RECOVERY FILE

All before images for a given transaction are written in a contiguous space on the BRF called a segment. A segment of the BRF is assigned to the transaction on the first DBEGIN request. After a DBCOMIT or DBFREE, this segment is reused to store before images for the transaction's next begin-commit sequence. The segment is released for use by other transactions after the transaction CEASE request.

Each before image recovery file defined in TAF/CRM will have a CIO buffer assigned to it.

Before image recovery files are always preallocated by TAF during normal initialization.

The before image write immediately transfers the before image to disk.

Installation parameter CRMUPM specifies the number of before images that a given begin-commit sequence can write to the BRF. When a transaction holds the lock for a record on a recoverable file and modifies that record more than once, only the first before image is written to disk. Therefore, if the transaction holds only record locks, it can update CRMUPM unique records within a begin-commit sequence; however, if the transaction holds the file lock, all before images are written to the BRF, including multiple updates of the same record. Consequently, if the transaction holds a file lock, the number of unique records it can update in a begin-commit sequence may be less than CRMUPM. Refer to the NOS Installation Handbook for more information regarding the installation parameter CRMUPM.

AFTER IMAGE RECOVERY FILE

After image writes on the ARF are buffered. There is one CIO buffer per data base for the after image recovery files. The size of this buffer is set by an installation parameter. The buffer is flushed to

AFTER IMAGE RECOVERY FILE

After image writes on the ARF are buffered. There is one CIO buffer per data base for the after image recovery files. The size of this buffer is set by an installation parameter. The buffer is flushed to disk when it becomes full, or on a DBCOMIT or DBFREE request.

Another installation parameter sets the maximum length of the ARFs. These files are preallocated to this size by TAF/CRM or DMREC, and the file will not be extended beyond this value by TAF/CRM.

CRMTASK

CRMTASK is a routine that enables the data administrator to determine and/or change the status of all TAF/CRM data bases or any particular data base or data file. Only those files and data bases described to TAF in the xxJ files can be specified on these commands.

CRMTASK CONSOLE COMMANDS

The full range of CRMTASK options is available from the console K display. CRMTASK is invoked by the system console command:

```
K.DIS,CRMTASK
```

When the K display is assigned, the following commands are available:

```
K.CRMSTAT
K.CRMSTAT,xx
K.CRMSTAT,xxpfn
K.DBUP,xx
K.DBUP,xxpfn
K.DBDOWN,xx
K.DBDOWN,xxpfn
K.END
K.MENU
K.+
K.-
```

The displays associated with each command are presented in the NOS 2 Analysis Handbook.

CRMSTAT Command

The CRMSTAT command enables the data administrator to view the status of data bases and/or data files on the operator console.

The format of the command is:

K.CRMSTAT	To view the status of all TAF data bases.
K.CRMSTAT,xx	To view the status of the indicated data base.
K.CRMSTAT,xxpfn	To view the status of the indicated file.

xx Data base identifier.

xxpfn File name within data base xx.

The status information is displayed on the left screen only. The DOWN status may appear as any of the following:

DOWN	The file or data base is down for batch recovery.
E-DOWN	The file or data base is down because of CRM errors.
O-DOWN	The file or data base is down because of an operator command.

DBUP and DBDOWN Commands

The DBUP and DBDOWN commands enable the data administrator to change the status of data bases and/or data files from the operator console.

The format of the command is:

```
K.DBUP,[xx or xxpfn]

K.DBDOWN,[xx or xxpfn]
```

xx Data base identifier. The command applies to all files in the indicated data base.

xxpfn File name within data base xx. The command applies only to the specific file indicated.

After DBDOWN is executed, no new users will be allowed to access downed data base files, but all existing users of such files will be allowed to finish.

K.DBUP makes the file or the data base available to TAF users for processing. A file can be upped only if the data base is in the UP state. A data base can be upped if all of the following are true.

- At least one data file can be upped.
- All BRFs are UP.
- The active ARF is UP.

CRMTASK TERMINAL COMMANDS

The data administrator can make the following CRMTASK commands available at the terminal by defining a transaction named CRMSTAT, DBUP, and/or DBDOWN, which declares CRMTASK as its only task.

```
CRMSTAT,xx
CRMSTAT,xxpfn
DBUP,xx
DBUP,xxpfn
DBDOWN,xx
DBDOWN,xxpfn
END
```

xx Data base identifier. The terminal must be logged in under the user name specified in the xxJ file for this data base.

xxpfn File name within data base xx. The terminal must be logged in under the user name specified in the xxJ file for data base xx.

The status displays sent to the terminal differ somewhat from the K display format and are illustrated in figures 5-1 and 5-2.

CRMTASK expects to find the CRMTASK command as the first characters in the message area of its communication block.

After CRMTASK sends its output to the terminal, it issues a WAITINP request, expecting another CRMTASK command. If none is entered within 480 seconds, it will timeout (480 is the release default; it can be changed by installation parameter DWITL).

All lines output by the release version of CRMTASK are 72 columns or less.

NOTE

If these commands are allowed from a terminal, any user could bring up or down the data base. It is recommended that a task (with the above commands placed in the communication block) be scheduled which does the security checking before calling CRMTASK.

IDLE STATUS

The IDLE status for a data base or file indicates a state of transition between UP and DOWN. It means that the data base or file is going down but still has active users. No new users are allowed to access it. The existing users are allowed to terminate their activities in an orderly fashion.

DATA BASE IDLE

All requests except OPEN and DBEGIN are allowed for an IDLE data base. OPEN and DBEGIN requests for an IDLE data base are terminated with a taf-status value of 28.

All requests for a DOWN data base will be terminated with a taf-status value of 32.

FILE IDLE

All requests except OPEN are allowed for an IDLE file. OPEN requests for an idle file are terminated with a taf-status value of 28. All requests for a DOWN file are terminated with a taf-status value of 32.

```
------------------- CRM DATA BASE STATUS -------------------

DATA BASE NAME =   RR              AFTER IMAGE FILE = ZZRRA01
DATA BASE STATUS = UP              PRU-S REMAINING +  XXX

-------------- BEFORE IMAGE FILES STATUS --------------------

  FILE   STATUS       FILE   STATUS       FILE   STATUS

ZZRRB01  UP         ZZRRB02  DOWN       ZZRRB03  UP
ZZRRB04  DOWN       ZZRRB05  UP

----------------- DATA BASE FILE STATUS --------------------

  FILE   STATUS       FILE   STATUS       FILE   STATUS

RRPFN01  UP         RRPFN02  E-DOWN     RRPFN03  DOWN
RRPFN04  0-DOWN     RRPFN05  IDLE
```

Figure 5-1. Terminal Output for CRMSTAT,xx Command

```
                    CRM FILE STATUS

              FILE NAME =   RRPFN01
              FILE STATUS = UP
              RECOVERABLE = YES

              PACK NAME =   PACKNAM
              DEVICE TYPE = DJ3
              ATTACH MODE = RM

              SIZE OF PRIMARY KEY =      80
              NUMBER OF ALTERNATE KEYS = 3

              ACTIVE USERS = 4
              ACTIVE LOCKS = 8
```

Figure 5-2. Terminal Output for CRMSTAT,xxpfn Command

Batch recovery provides a data base administrator with recovery capabilities beyond those provided by automatic recovery. It is used to recover defective data base files, dump and load data base files and after image recovery files, preallocate data base files, and perform general housekeeping procedures for backup tapes.

The batch recovery program is accessed with the command DMREC. DMREC can be called automatically by TAF or expressly by a user job. The circumstances under which TAF calls DMREC are presented later in this section under DMREC Calls From TAF.

Jobs not initiated by TAF can use DMREC to process only those data bases for which the user name specified in the xxJ file matches the user name under which the job is being run.

DMREC directives are provided to perform the following functions:

COMMENT	Inserts a comment into the input stream.
CREATE	Generates initial copies of log files ARF and BRF.
DUMP	Dumps a data base file within a TAF/CRM data base or dumps an after image recovery file (ARF) onto a backup medium (tape).
EDIT	Edits the backup directory (ZZxxDIR).
EXPAND	Expands (preallocates) a data base file by a given percentage.
LIST	Lists the contents of the backup directory (ZZxxDIR).
LOAD	Loads a data base file from the backup medium (tape) back to mass storage.
RECOVER	Recovers a data base file within a TAF/CRM data base.
UPDATE	Applies updates (after images) from log file dump tapes to a given data base file.

These functions are covered in detail under DMREC Directives, later in this section.

DMREC COMMAND

The DMREC command for non-TAF origin jobs has the following format.

DMREC(I=lfn$_1$,L=lfn$_2$)

or

DMREC(L=lfn$_2$,Z);d$_1$;d$_2$;...;d$_n$

In the second format, d is any input directive statement (refer to DMREC Directives, later in this section). The separator, shown here as a semicolon (;), can be any character that does not occur in the directive list.

All parameters are optional.

Parameter	Description
I	Specifies the file containing the input directives. The file is not rewound before processing. The options are:
	I — File is COMPILE.
	I = lfn — File is specified by user.
	omitted — File is INPUT.
L	Specifies the file to receive listed output. The options are:
	L — File is LIST.
	L = lfn — File is specified by the user.
	omitted — File is OUTPUT.
Z	Specifies that input directives are on the DMREC statement. When Z is used, the I parameter must be omitted or set equal to zero.

Users can issue the DMREC command from both interactive and batch jobs. For interactive jobs, DMREC reformats the output lines to 72 or fewer columns.

FILES

Batch recovery (DMREC) uses the files listed as follows. The xxJ file resides under TAF's user name. The user named in an xxJ file must be able to read the xxJ file; thus, the xxJ file must be public or permission to read it must be given to the specified user. All the other files must reside under the user name specified in the xxJ file. The job using DMREC must originate from this user.

- TAF/CRM data files

  Direct access files specified in the xxJ file that contain the data for a particular application.

- TAF/CRM index files

  Direct access files for applications using MIPPED data files.

- TAF/CRM backup directory

  A direct access file automatically allocated and maintained by DMREC. It contains information about the data base, index, and after image recovery files for the application. The file name is ZZxxDIR where xx is the application identifier.

- xxJ file

  An indirect access file describing the data base and providing an association between a data base and the user name under which it resides. Refer to the TAF Reference Manual.

- ZZZZZDG file

  An indirect access file containing FILE statements that further describe the characteristics of the data files.

- CRM owncode file

  An optional indirect access file containing the user-provided hashing routine for a particular data file.

- Tape files

  Labeled tapes containing backup copies of data files, index files, or after image recovery files.

DMREC DIRECTIVES

Directives for DMREC can be in a directive file (specified with the I parameter on the DMREC command) or in the DMREC commands. All directives begin with an asterisk (*). Fields within directives are delimited by a comma and/or one or more spaces. Directives can begin in any column and can be continued on up to 10 subsequent lines. Each line may extend to column 160. Individual fields within a directive cannot be split between two lines. Each directive in a directive file must start on a new line. Directives on the DMREC command are separated by any character that does not appear in the directive list. COMMENT directives cannot have continuation lines.

In all of the following directive descriptions the lowercase characters given below have the indicated meaning.

xx	Application (data base) identifier.
xxpfn	Data file name.
vsn	Volume serial number for a specific tape.
hhmmss	Time, using two digits each for hours, minutes, and seconds.
yymmdd	Date, using two digits each for year, month, and day.

COMMENT DIRECTIVE

The COMMENT directive allows a statement to be ignored by DMREC to be included in the job. The format is:

```
*.

   or

*COMMENT
```

The desired comment follows the directive on the same line. Continuation lines are not allowed. For comments longer than one line, each new line must begin with a COMMENT directive.

CREATE DIRECTIVE

The CREATE directive allows the data base administrator to create after image recovery files or before image recovery files on mass storage. The format is:

```
*CREATE,ZZxxAnn,LENGTH=dddd

   or

*CREATE,ZZxxBnn
```

nn	File ordinal.
dddd	Length of file in PRUs (decimal).

The first format creates an after image recovery file (ARF). The second format creates a before image recovery file (BRF).

The optional length parameter on the after image recovery file creation allows the data base administrator to specify the length for the after image recovery file. If the length parameter is not specified, the default value, defined by the installation parameter CRMARFN, is assumed.

DUMP DIRECTIVE

The DUMP directive dumps one or more data or after image recovery files to tape. The format is:

*DUMP,xxpfn$_1$ type, xxpfn$_2$ type,...,xxpfn$_n$ type,VSN=vsn$_1$/vsn$_2$/.../vsn$_n$.

type An optional parameter specifying the type of dump desired. When this parameter is used, the slash character (/) immediately follows the last character in the data file name. Type options are:

/BLOCK Specifies a block dump. All active CRM blocks of the data file and associated index file are dumped. If the data file has alternate keys, the multiple index file is also dumped.

/RECORD Specifies a record dump. A record dump copies all active logical records from the file. If the data file has alternate keys, the alternate key descriptions and the FSTT of the multiple index file are also dumped.

omitted If DMREC can attach the data file (and index file if applicable) in write mode, it does a block dump. If not, it attempts to attach the file in read-modify mode. If this attach is successful, DMREC then does a record dump.

NOTE

Block dumps are much faster than record dumps. Also, subsequent recovery processing is simpler with block dumps than it is with record dumps. The record dump capability is provided for use in situations in which a dump must be taken while the data file is available for transaction processing. This implies that the file can be actively modified during a record dump. For this reason, the time stamp given to a record dump is the time at the beginning of the dump, whereas the time stamp given to a block dump is the time at the completion of the dump. When restoring a file from a record dump, it is necessary to apply after images beginning with the time of the start of the dump until some time at or after the completion of the dump.

A single file cannot appear more than once in the directive statement. The number of data files in a single dump directive cannot exceed 100. If more than one VSN is specified, the additional VSNs indicate which tape reels to use for multireel dumps. If the VSN parameter is omitted, DMREC requests a tape. It is up to the user in this case to ensure that a blank-labeled tape to be used for the dump is mounted.

After DMREC processes a DUMP directive, the tape(s) used for the dump are unloaded. A new DUMP directive causes DMREC to request a new tape.

The DUMP directive is also used to dump after image recovery files. When the active after image recovery file is full, TAF automatically switches after image logging to the other after image recovery file and submits a job to dump the full after image recovery file to tape. The user can dump an after image recovery file to tape by using the DUMP directive specifying only the after image recovery file name (ZZxxAnn) and the VSN of the dump tape. After the after image recovery file is dumped, the file is logically cleared. Thus, information on the file prior to the dump is available only from the dump tape.

EDIT DIRECTIVE

The EDIT directive edits the backup directory for a specified application. There are three types of EDIT:

A. *EDIT,xx

B. *EDIT,xxpfn$_1$,...,xxpfn$_n$

C. Same as type A or B immediately followed on the same or succeeding lines by one or more subdirectives.

When type A or B is used, DMREC discards all entries for dumps of the data base (type A) or the specified data file (type B) except the two most recent dumps. The number of dumps to be retained is determined by the value of the COPYCNT installation parameter. The release value of this parameter is two. The user can use the *CYCLE subdirective to select another value for the number of dumps to be retained.

In the case of a data file EDIT (type B), DMREC then determines if any dump tapes now contain only inactive dumps. If so, entries for those tapes are discarded. For both types of EDIT (types A and B) DMREC then discards entries for any after image recovery file tapes that are older than the oldest remaining dump tape.

When type C is used, the edit processing to be done is explicitly stated by the user via the subdirectives that immediately follow the EDIT directive. (The absence of any subdirectives indicates a type A or B EDIT.)

NOTE

DMREC performs the editing specified by the EDIT subdirectives regardless of the impact such editing may have on future recovery.

EDIT Subdirectives

The subdirectives available for use with the EDIT directive are CYCLE, DELETE, and ADD.

CYCLE

The CYCLE subdirective specifies the number of data base file dumps to retain. Any future type A or B EDITs will use this as the default until it is changed again by the user.

The format is:

```
*CYCLE,CYCL=n

    n = 0, 1, 2,...,9
```

The CYCLE subdirective makes a change on the ZZxxDIR file for the entire data base, following a type A EDIT, or for the individual file(s) specified, following a type B EDIT. If the ZZxxDIR file is purged and re-created, the default cycle specified on the COPYCNT installation parameter will again be in effect for all files in data base xx. Otherwise, the cycle specified via a CYCLE subdirective remains in effect until explicitly changed by another CYCLE subdirective.

DELETE

The DELETE subdirective has two formats:

```
*DELETE,VSN=vsn

    or

*DELETE,TIME=hhmmss,DATE=yymmdd
```

The first format deletes the directory entry that refers to the backup tape named on the subdirective. The user must ensure that such deletions will not harm possible future recovery; DMREC makes no such check. If the VSN specified is the first of a multireel set of dumps, entries for the entire dump set will be deleted.

The second format specifies deletion of entries prior to the date and time specified. If the date is omitted, today's date is assumed. If the time is omitted, the beginning of the specified date (00:00:00) is assumed.

The DELETE subdirective can follow a data base or a data file EDIT directive. Following a data base EDIT directive (*EDIT,xx), the deletions apply to dump entries for all files in the data base. Following a data file EDIT directive (*EDIT,xxpfn), the deletions apply only to dump entries for the file(s) specified.

ADD

The ADD subdirective causes the addition of entries from the file backup directory on the specified tape to the file backup directory for this data base. In this way, entries previously deleted from the backup directory can be restored.

The format is:

```
*ADD,VSN=vsn
```

When DMREC encounters this subdirective, it requests the tape. It locates the last file on a dump tape, which is a copy of the entire file backup directory at the time the dump was taken. DMREC copies this directory to a local disk file and compares it to the permanent file backup directory. Entries in the local directory whose VSN fields differ from those of entries already in the permanent directory are then copied to the permanent directory.

The ADD subdirective can follow either type (A or B) of the EDIT directive and works identically in both cases.

EXPAND DIRECTIVE

The EXPAND directive is used to preallocate free space on data files (and associated index files, if applicable). The format is:

```
*EXPAND,xx,PERCENT=n

    or

*EXPAND,xxpfn,PERCENT=n
```

n The percentage of preallocated space desired. The permitted values are 0 through 99.

The first format expands all files in the indicated data base to the specified percent, except as noted in the following. The second format expands only the indicated data file (and its associated index file, if applicable) to the specified percent. No expansion takes place for any file whose percentage of free space already equals or exceeds the amount specified.

The percent parameter is optional. If it is omitted, DMREC uses the percent in the preallocation field of the ZZxxDIR file. Originally, this is the value specified by the EXPCT installation parameter (release default is 10). Use of the EXPAND directive with the percent parameter changes this field to the value given by the user. This new value then becomes the default for subsequent expands until again changed by the user.

The preallocation percent is recorded on the ZZxxDIR file for the data base and for each individual data file. When a data base EXPAND is used, the following occurs:

- The preallocation percent for all files whose present preallocation percent matches the present data base preallocation percent is changed to the new value specified in the directive.
- The preallocation percent for the data base is changed to the new value specified in the directive.
- The preallocation percent for any other file is not changed.
- Each file is expanded to the preallocation percent for that file if necessary.

DMREC must be able to attach the affected files in write or modify mode in order for expansion to take place. If the EXPAND directive is used and DMREC cannot attach an affected file in write or modify mode, the preallocation field for that file will be changed on the ZZxxDIR file but the file will not be expanded. The following diagnostic message is issued:

```
ATTACH ERROR ON PF filename.
```

If the xxJ file CRM statement for this data file specifies any mode other than M (modify) or W (write), a diagnostic is issued and the file and directory are unaltered.

The preallocation percent refers to the difference between the physical and logical length of the file expressed as a percentage of the logical length. When the EXPAND directive is used, DMREC gets the logical length of the file from the file statistics table (FSTT). This is the number of PRUs occupied by active or logically deleted records. (Logically deleted records can be physically deleted by re-creating the file using the FORM utility.) DMREC then calculates the physical length necessary to satisfy the preallocation percent specified on the directive or in the ZZxxDIR file and compares this length to the actual physical length of the file. If the actual physical length is less than required, the physical length is increased.

LIST DIRECTIVE

The LIST directive can be used to display the contents of the backup directory (ZZxxDIR) or to list the after image record headers on an after image dump tape. It has three formats:

```
*LIST,xx,TIME=hhmmss,DATE=yymmdd

        or

*LIST,xxpfn1,...,xxpfnn,TIME=hhmmss,DATE=yymmdd

        or

*LIST,xx,VSN=vsn,TIME=hhmmss
```

The first two formats of the LIST directive cause DMREC to display the contents of the backup directory (ZZxxDIR) for an entire data base, or for one or more files within a data base. This includes entries for dumps of data base files and after image recovery file dumps.

In the first two formats, TIME and DATE are optional parameters. When used, they indicate that all entries entered before the particular time and/or date should be listed. If the DATE parameter is omitted, today's date is assumed. If the TIME parameter is omitted, 00:00:00 is assumed.

Figure 6-1 shows a sample of the output that results from using the first format of the LIST directive.

The third format of the LIST directive causes DMREC to display the after image record headers on the specified after image dump tape. The optional TIME parameter in this format specifies that all entries (headers) starting from this particular time should be listed. If the TIME parameter is omitted, all headers on this tape will be listed.

LOAD DIRECTIVE

The LOAD directive causes DMREC to load a data base file (and its associated index file, if applicable) from tape to mass storage. It has two formats:

```
*LOAD,xxpfn type,TIME=hhmmss,DATE=yymmdd

        or

*LOAD,xxpfn,VSN=vsn
```

type — An optional parameter specifying the type of load desired. When this parameter is used, the slash character (/) immediately follows the last character in the data file name. Type options are:

- /BLOCK — Load from a block dump. If the data file has alternate keys, the associated multiple index file is also loaded.
- /RECORD — Load from a record dump. If the data file has alternate keys, the associated multiple index file is re-created from the alternate key descriptions and the FSTT of the multiple index file on the dump.
- omitted — The latest dump satisfying the TIME and DATE parameters is loaded regardless of type.

NOTE

A file loaded from a record dump cannot be presumed to be in a correct state until after images taken during the dump are applied. Refer to the note in the description of the DUMP directive, earlier in this section.

The optional TIME and DATE parameters indicate that the last (newest) dump taken before the given time and date and satisfying the type parameter, if specified, should be loaded. If the TIME parameter is omitted, 23:59:59 is assumed. If the DATE parameter is omitted, today's date is assumed.

The VSN parameter in the second format specifies the VSN of a backup tape from which to load. Only the VSN of the first tape in a multireel dump set can be specified; DMREC will automatically locate the rest of the VSNs in the set. No other parameters are allowed when a VSN is specified.

```
DMREC  - TAF/CRM BATCH RECOVERY PROGRAM.                                   82/02/10. 16.30.45. PAGE    1

  >>>>>       *LIST,DG.

        FULL LIST OF THE BACKUP DIRECTORY FOR THE DATA BASE - DG

              (1)                 (2)           (3)   (4)    (5)     (6)
           CREATION          BRF UNUSABLE     BRF-S PREA.  BACKUP  FIRST ARF
        DATE      TIME       DATE       TIME  DOWN  PERC.  DUMPS     VSN

      81/08/28. 14.44.10.                       0     8      2      TOTL

                       (7)       (8)               (9)
      FILE HEADER :   FILE  PREALLOCATION     BACKUP DUMPS

                     DGFILE        8                2

     (7)   (10)  (11)        (11)     (12)  (13)  (14)  (13)   (15)     (16)
     FILE  TYPE  DATE        TIME     FMT   ORD  INDEX  ORD  AI RECS    VSN

    DGFILE  D  82/02/09. 17.00.28. B     2                              DGDMP3
    DGFILE  D  82/02/10. 15.41.19. R     0                              IRRFEX

    FILE HEADER :  FILE  PREALLOCATION   BACKUP DUMPS

                  DGISF1       8               3

     FILE  TYPE  DATE        TIME     FMT   ORD  INDEX  ORD  AI RECS  VSN

    DGISF1  D  82/02/09. 16.58.14. B     0   DGINDX  1                 DGDMP1
    DGISF1  D  82/02/09. 16.59.28. B     0   DGINDX  1                 DGDMP2
    DGISF1  D  82/02/09. 17.00.28. R     0   DGINDX  1                 DGDMP3
    DGISF1  D  82/02/10. 15.41.19. B     1   DGINDX  2                 IRRFEX
    DGISF1  A1 82/02/09. 13.25.17.                          2997       TOTL

                      (16)     (17)       (18)       (19)       (20)
    VSN ENTRIES :     VSN    NEXT VSN    FILES    ACT.FILES   NEXT ARF

                  DGDMP1                   2          2
                  DGDMP2                   2          2
                  DGDMP3                   3          3
                  IRRFEX                   3          3
                  TOTL                   *ARF*        Z

    CHRONOLOGICAL LIST OF THE DUMPS TAKEN

       DATE       TIME       FILE      TYPE       VSN     FMT

    82/02/09. 16.58.14.   DGISF1   FILE DUMP  DGDMP1   B
    82/02/09. 16.59.28.   DGISF1   FILE DUMP  DGDMP2   B
    82/02/09. 17.00.28.   DGISF1   FILE DUMP  DGDMP3   R
    82/02/09. 17.00.28.   DGFILE   FILE DUMP  DGDMP3   B
    82/02/10. 15.41.19.   DGISF1   FILE DUMP  IRRFEX   B
    82/02/10. 15.41.19.   DGFILE   FILE DUMP  IRRFEX   R
```

1. Date and time this directory file was created.
2. Not used.
3. Not used.
4. Default data base preallocation percent. Refer to the EXPAND directive.
5. Data base backup dumps entry. This is the minimum value that any file backup dumps entry may have. It is also the default number of backup dumps assigned to new files added to this directory.
6. Self explanitory.
7. Preallocation percent for this file. Refer to the EXPAND directive.
8. Number of dumps to be retained on a type A or B EDIT.
9. Self explanitory.
10. Type of dump. D - data file dump; A - after image dump.
11. Date and time of dump.
12. Dump format. B - block dump; R - record dump.
13. Position of the file on the dump tape. The first file occupies position zero.
14. Associated index file.
15. Number of after image records on an A type dump.
16. Self explanitory.
17. For multitape dumps, the next tape in the set.
18. Number of files in this dump set.
19. Number of files on this VSN.
20. Next tape in a multitape after image dump.

Figure 6-1. *LIST Directive Output

RECOVER DIRECTIVE

The RECOVER directive causes DMREC to recover a data base file by loading the file from a backup dump and applying after image records for this file. It has two formats:

*RECOVER,xxpfn type,TIME=hhmmss,DATE=yymmdd

or

*RECOVER,xxpfn,VSN=yyyyyy,TIME=hhmmss,DATE=yymmdd

type — An optional parameter specifying the type of load desired. When this parameter is used, the slash character (/) immediately follows the last character in the data file name. Type options are:

- /BLOCK — Specifies that the latest block dump be used for the load. If the data file has alternate keys, the associated multiple index file is also loaded.
- /RECORD — Specifies that the latest record dump be used for the load. If the data file has alternate keys, the associated multiple index file is re-created from the alternate key descriptions and the FSTT of the multiple index file on the dump.
- omitted — The latest dump is loaded regardless of type.

The processing initiated by the RECOVER directive has two distinct phases: the load phase and the update phase. The type parameter (first format) or the VSN parameter (second format) apply to the load phase of the recovery. The TIME and DATE parameters apply to the update phase of the recovery.

In the second format, the VSN parameter indicates which tape should be used in the loading process. This VSN must be the VSN of a tape containing a dump of the data base file. For multireel dumps, it is the first tape within the dump set.

After the loading completes, after images are applied; starting with the first after image recovery file dump taken after the data file was dumped, and proceeding through the specified time and date. If no time is specified, 23:59:59 is assumed. If no date is specified, today's date is assumed. Thus, the omission of the DATE and TIME parameters results in a recovery up to the latest after image recovery file.

The optional IGNORE subdirective can be specified for the RECOVER directive. When used, this subdirective must immediately follow the RECOVER directive. Its format is:

*IGNORE,TS=$tseq_1$/$tseq_2$/.../$tseq_n$,
TN=$taskname_1$/$taskname_2$/.../$taskname_n$

tseq	Transaction sequence number of a transaction whose after image recovery file entries for this data base file should be ignored.
taskname	Name of a task whose after image recovery file entries for this data base file should be ignored.

The TS and TN parameters may be used in the same IGNORE subdirective. Updates for all tasks and transactions specified will be ignored regardless of any interrelationships among them.

Caution must be exercised in using the IGNORE function. Using the IGNORE function, in effect, rolls back any updates made by transaction $tseq_i$ and/or task $taskname_i$ but does not affect other transactions or tasks that may have used the results of those updates. This may leave the data base in an illogical state.

The RECOVER directive is also used by TAF to recover defective BRF files. Details are given under DMREC Calls from TAF, later in this section.

UPDATE DIRECTIVE

The UPDATE directive applies after images from after image recovery file tapes to a particular data base file. The format is:

*UPDATE,xxpfn,VSN=vsn,TIME=hhmmss/hhmmss,
DATE=yymmdd/yymmdd

When the VSN parameter is used, it refers to the data base dump tape. The TIME, DATE, and VSN parameters are used to indicate a time window. After images for all begin-commit sequences whose BEGIN requests occurred during the time window will be applied to the data file. The first after image recovery file dump tape used is the first dump that occurred after the opening of the time window.

The time window is determined as follows:

DMREC checks the following in the order given to determine the opening time and date for the time window.

1. The opening time and date specified in the directive. This is the time and date given before the slant in the TIME and DATE parameters. If an opening time is specified but no opening date, the data file dump date is assumed. If an opening date is specified but no opening time, 00:00:00 is assumed.

2. If neither the opening time nor the opening date was specified but a VSN was specified, the opening of the time window is the time that xxpfn was dumped to tape VSN.

3. If neither the opening time nor the opening date was specified and no VSN was specified, the opening of the time window is the time in the backup directory entry for the most recent dump of xxpfn to tape.

DMREC checks the following in the order given to determine the closing time and date for the time window. If the closing time can be determined in step 1, step 2 is skipped.

1. The closing time and date specified in the directive. This is the time and date given after the slant in the TIME and DATE parameters. If a closing time is specified but no closing date, the data file dump date is assumed. If a closing date is specified but no closing time, 23:59:59 is assumed. A closing time and/or date can be specified even if no opening time and/or date was specified.

2. The time stamp in the backup directory entry for the most recent after image recovery file dump.

The time stamp on the backup directory for a data file dump is the time of the start of the dump if the dump type is /RECORD. The time stamp on the backup directory for a data file dump is the time at the end of the dump if the dump type is /BLOCK.

A diagnostic message will be issued if no commit is found for a particular begin-commit sequence after searching the entire set of after image dumps.

DMREC CALLS FROM TAF

TAF submits a batch job containing a call to DMREC when any of the following situations occur.

- An after image recovery file must be dumped to tape because of an I/O error or when it becomes full.
- TAF detects a defective data base file.
- TAF detects a BRF DOWN condition.

In any of these cases, TAF sets up the following job file:

```
DMREC,T37777.                          } At least one tape
USER(username,password,family)         } is required for
RESOURC (tape drive type = 1)          } TAF-initiated
                                         batch recovery.

DMREC(I=0,Z,TT=nnn) d1,d2,...,dn
```

username — The user name from the xxJ file for this data base.

password — The password from the xxJ file for this data base.

family — The optional family name from the xxJ file for this data base.

tape drive type — This value is determined by the TAF installaion parameters TDEN and DTTP. The default is PE=1.

nnn — A TAF-assigned identifier that will be returned to TAF after completion of DMREC.

d_i — Specific DMREC directives as shown in the following subsections.

During successful execution of the batch job calling DMREC that was submitted by TAF, the following console operator interaction occurs.

1. The following message appears on the B-display at the job's control point.

   TAF TAPE REQUEST DB=xx DUMP

   xx is the two-character data base name. The operator responds by entering the following where jsn is the job sequence number.

   CRO,jsn.GO.

2. A tape request appears. The operator responds by assigning a blank labeled tape.

If errors are encountered during execution of this batch job submitted by TAF, the following message appears on the B-display at the job's control point.

SEE JOB DAYFILE

This specific error message precedes the following dayfile messages.

```
DMREC FAILED - directive xx.
NOTE FAILURE, THEN TYPE IN CFO,JSN.GO
```

directive xx is the directive and data base that DMREC was processing when the error occurred.

The data base administrator should direct the correction of the error conditions. After errors have occurred, the data base must be brought up manually using the K-display commands.

Error conditions during the dump process are described in section 8 and under File Error Conditions, later in this section.

NOTE

DMREC jobs initiated by TAF must not be terminated manually either by an operator DROP command or a system deadstart, as this may leave a data base in an inconsistent state.

AFTER IMAGE LOG DUMP

When an after image recovery file becomes full, TAF switches logging to the other ARF and submits the previously shown job with the directive:

*DUMP,ZZxxAnn

ZZxxAnn is the permanent file name of the after image recovery file.

xx The data base identifier.

nn The ARF number (01 or 02).

DMREC attaches file ZZxxAnn in write mode and requests a tape upon which to dump the file.

The after image recovery file is then dumped to tape. During the dump, a table is constructed containing the names of all data base files having after image records on this log tape. This table is added, one record per data base file, to the backup directory at the end of the dump. TAF is notified upon completion of the process.

Refer to section 8 for a discussion of the input/output (I/O) error condition on an ARF.

DEFECTIVE DATA BASE FILE

When TAF detects a defective data base file, it downs the file and submits the previously shown job with the directives:

*DUMP,ZZxxAnn;*RECOVER,xxpfn

ZZxxAnn is the permanent file name of the after image recovery file previously described. This is the ARF that was active at the time the fault was detected. xxpfn is the defective data base file.

Refer to section 8 for a more detailed discussion of this topic.

BRF DOWN CONDITION

When TAF detects a BRF DOWN condition, it downs the affected data base. Then it submits the previously shown job with the directives:

*DUMP,ZZxxAnn;*RECOVER,ZZxxBpp,TIME=hhmmss,
DATE=yymmdd

ZZxxAnn is the permanent file name of the after image recovery file active at the time the defective BRF was detected. ZZxxBpp is the permanent file name of the defective BRF file.

xx The data base identifier.

nn The ARF number (01 or 02).

pp The BRF number (01 - 77_8).

The TIME and DATE parameters indicate the moment when TAF wrote the earliest unresolved BRF DOWN stamp to the after image recovery file.

Refer to section 8 for a more detailed discussion of this topic.

FILE ERROR CONDITIONS

Errors can occur on any of the files processed by DMREC. In some cases, DMREC can correct the situation; in others, manual intervention is necessary. This subsection presents possible error conditions and the response by DMREC, and recommends action to be taken by the data administrator. In all cases, if the job issuing the DMREC directive was initiated by TAF, DMREC will notify TAF of successful completion only if the error condition could be corrected by DMREC.

Irrecoverable Error on an After Image Log Tape During Recovery

If DMREC detects an irrecoverable error during the dump of an after image recovery file to tape, it starts the dump over by returning the defective tape and requesting another tape. The following message appears:

```
TAPE VSN = xxxxxx IS BAD, PLEASE REPLACE.
```

The console operator responds by entering the following:

```
CFO,jsn,GO
```

jsn is the job sequence number. The VSN entry on the file backup directory is changed accordingly.

Irrecoverable Error on a Backup Dump Tape During a File Load

If DMREC detects an irrecoverable tape error during a file load, a diagnostic message is issued and DMREC terminates. The data administrator or file owner must:

- Delete this tape from the file backup directory (ZZxxDIR).
- Restart the process that initiated the file load.

Irrecoverable Error on a Mass Storage Data File

If an irrecoverable error is detected on mass storage during the recovery of a data base file, the data administrator or file owner must:

- Purge the defective file.
- Reallocate the file on a different mass storage device.
- Restart the file recovery process.

Irrecoverable Error on a Tape During an After Image Dump

If another copy of the tape is available, DMREC requests it and continues with recovery. If no other copy is available, DMREC issues dayfile and operator messages indicating that the data base file cannot be recovered correctly. An installation parameter within DMREC (NUMARF) governs the number of copies of the same mass storage after image recovery file that is to be retained. The data administrator should try the following.

- Clean the tape and try again.
- Clean the tape drive and try again.
- Try another tape drive.
- Rerun the affected transactions using input from JOURO.

Consideration should be given to getting multiple copies of future dump tapes by increasing NUMARF or manually copying ARF dump tapes.

Irrecoverable Error on a Directory or CRM Owncode File

If DMREC detects that a backup directory file or CRM owncode file is defective, it issues a dayfile and console message. The data administrator should:

- Down the affected data base if TAF is active.
- Perform manual repair of the file. The extent of this repair depends upon the type of error and the file affected.

Irrecoverable Error on an After Image Recovery File During a Dump

If DMREC detects an error on an after image recovery file during the dump process, the data base administrator should perform the following actions:

- Bring the data base down if TAF is active.
- Take a complete dump of the data base.
- Allocate a new ARF using the DMREC CREATE directive.

The TAF/CRM requests described in this section are available only to tasks written in COMPASS and loaded from the system task library. These requests are used by CRMTASK and CTASK to perform the functions described in section 5. They are defined in deck COMKCRM.

DBUP REQUEST MACRO

The DBUP macro is used to place a data base or file in an UP status. The format is:

```
DBUP   listaddr
```

listaddr — The FWA of a list of parameter addresses, terminated by a zero word.

listaddr+0 = The address of the word containing the left-justified, zero-filled data base or file name.

listaddr+1 = The address of the word containing the TAF status. The possible TAF status values are:

- 0 The request completed successfully.
- 1 The file or data base is not available in the xxJ file.
- 33 Not all the data base files were upped.
- 34 The file or data base cannot be upped.

listaddr+2 = 0

DBDOWN REQUEST MACRO

The DBDOWN macro is used to place a data base or file in a DOWN status. The format is:

```
DBDOWN   listaddr
```

listaddr — The FWA of a list of parameter addresses, terminated by a zero word.

listaddr+0 = The address of the word containing the left-justified, zero-filled data base or file name.

listaddr+1 = The address of the word containing the TAF status. The possible TAF status values are:

- 0 The request completed successfully.
- 1 The file or data base is not available in the xxJ file.
- 35 The data base or file is already down or idle.

listaddr+2 = 0

CRMSTAT REQUEST MACRO

The CRMSTAT format is as follows:

```
CRMSTAT   listaddr
```

listaddr — The FWA of a list of parameter addresses.

listaddr+0 = The address of the word containing the status function desired. The request codes are:

- 0 For CRM status.
- 1 For getting all data base tables (TDRFs) and recovery file tables.
- 2 For data base status. 42/0Ldb,18/2
- 3 For getting all logical name tables (TLNTs) for the specified data base. 42/0Ldb,18/3
- 4 For file status. 42/0Ldbpfnxx,18/4

listaddr+1 = The address of the word containing the TAF status. The possible TAF status values are:

- 0 The request completed successfully.
- 1 The data base or file is not available in the xxJ file.
- 35 The table area supplied by user is not large enough.

listaddr+2 = First word address of table where CRM returns status information.

request code 0 For CRM status (request code 0) this buffer must be large enough to hold the following tables in the order listed:

1. TAF/CRM transaction sequence table (TSEQ) in its entirety as it exists at the time the request is processed.

   The table is defined in COMKCRM with FIELD macros. This deck also defines the length of each entry of this table (TSEQE symbol). The number of entries in this table is CMDM, which is an installation parameter defined in COMKIPR.

   The first word of each entry of this table is zero if the entry does not contain a transaction entry. For a detailed handling of this table refer to subroutine FTS of AAMI.

   The length of this table in 60-bit CM words is (CMDM*RMDM+1)*TSEQE.

2. Binary zero word.

3. TAF/CRM data manager input queue FET. This is the FET for TAF/CRM queue where AAM accepts transaction requests. The length of this FET is defined by AAMQFL in COMKCRM. This length is in 60-bit words.

4. TAF/CRM data manager input queue. This is the input queue already referred to in item 3. The length of this queue, in 60-bit CM words, may be obtained from the FET (LIMIT-FIRST) or the symbol AIBFL in COMKIPR.

5. TAF/CRM data manager output queue FET. This is the FET for TAF/CRM queue where it places all completed AAM requests. The length of this FET, in 60-bits words, is AAMQFL defined in COMKCRM.

6. TAF/CRM data manager output queue. This is the output queue already referred to in item 5. The length of this queue, in 60-bit words, may be obtained from the FET (LIMIT-FIRST) or the symbol AOBFL in COMKIPR.

7. Binary zero word.

   The total length of the buffer required must be at least as large as:

   ```
   RMDM*CMDM*TSEQE+
   TSEQE+1+2*AAMQFL+
   AIBFL+AOBFL+1
   ```

   Where this length is in 60-bit CM words.

request code 1 For getting all TDRFs known to AAMI (request code 1), this buffer must be large enough to hold at most CMAXDB TDRFs; each TDRF is TDRFE words long. The last TDRF will be followed by a zero word.

request code 2 For data base status (request code 2) this buffer must be large enough to hold the TAF/CRM tables for the data base and files (data base and recovery) for the data base.

The tables will be arranged as follows in the buffer:

1. TDRF: data base table. Length=TDRFE.

2. Binary zero word.

3. TARF: after image recovery file table if the data base has recovery defined (TDQN of TDRF = 0). Length = TARFE.

4. Binary zero word.

5. TQRF: before image recovery file table(s). There will be TDQN, TQRFs returned. Length=TQRFE.

6. Binary zero word.

request code 3 For request code 3, the buffer must be large enough to hold all TLNT entries. Each TLNT entry is TLNTE words long. The last TLNT entry is followed by a zero word.

request code 4 For file status (request code 4) the buffer must be large enough to hold the TAF/CRM tables for the file (TLNT and a count of active TFCB and TKOK).

The tables will be arranged as follows in the buffer:

1. TLNT: (logical name table for file). Length=TLNTE.

2. Binary zero word.

3. Count of active TFCBs; 60-bit integer.

4. Count of active TKOKs; 60-bit integer.

5. Binary zero word.

listaddr+3 = The address of the word containing the length of the table space specified by the previous parameter.

listaddr+4 = 0

TRMREC REQUEST MACRO

The TRMREC request is processed as a transaction CEASE request. It causes the following events to occur in the order given.

1. A DBFREE is processed.

2. A CEASE stamp is written on the BRF.

3. All resources held by task are released.

A separate TRMREC request must be issued for each transaction for which this processing is desired.

The format of the TRMREC macro is as follows:

```
TRMREC   listaddr
```

listaddr = The FWA of a list of parameter addresses terminated by a zero word.

listaddr+0 = Contains address of a word, which is zero.

listaddr+1 = The address of the word containing the TAF status. The possible TAF status values are:

- 0 The request completed successfully.
- 24 There is no outstanding DBEGIN request.
- 25 An error was encountered on a data base or recovery file.
- 32 The data base or TAF/CRM is down.

listaddr+2 = 0

CRMSIC REQUEST MACRO

The CRMSIC request passes information from TAF to TAF/CRM about a DMREC job that has completed successfully. The format is:

```
CRMSIC   listaddr
```

listaddr = The FWA of a list of parameter addresses terminated by a zero word.

listaddr+0 = The address of the word containing the left-justified, zero-filled data base or file name.

listaddr+1 = The address of the word containing the TAF status. The possible TAF status values are:

- 0 The request completed successfully.
- 1 The file or data base is not available in the xxJ file (should never happen).
- 35 No batch job is outstanding for data base or file.

listaddr+2 = Batch job sequence number in octal. Right justified and zero filled in the word.

listaddr+3 = The address of the word containing the function completed by batch recovery. The possible values are:

- 1 The ARF dump is complete.
- 2 The data base file dump is complete.
- 3 The ARF or BRF preallocation is complete.

listaddr+4 = 0

RSTDBI REQUEST MACRO

This request is used to restore the data base begin identifiers for a recovered transaction.

The macro format is:

```
RSTDBI   listaddr
```

listaddr = FWA of a list of parameter addresses terminated by a zero word.

listaddr+0 = Address of word containing current begin identifier in the lower 30 bits of word.

listaddr+1 = The address of the word containing the TAF status.The possible TAF status values are:

- 0 The request completed successfully.
- 27 No recovery file is assigned to the data base.
- 32 The file or data base is down.

listaddr+2 = Address of word containing previous begin identifier in the lower 30 bits of word.

listaddr+3 = 0

The data base ids are not saved by AAMI if error code 27 is returned.

The data base ids are saved by AAMI if error code 32 is returned.

This section discusses automatic and batch recovery processing for various types of failures. In some cases the entire recovery process occurs without manual assistance other than to respond to requests to mount tapes. In others, analysis by the data administrator is necessary prior to proceeding with recovery.

RECOVERY HANDLED AUTOMATICALLY

DATA FILE FAILURE

When TAF/CRM detects a fatal error on a data base file, it declares the file DOWN. All requests issued specifically for this file, except CLOSE, return a fatal error code to the requesting task. However, DBCOMIT can be successfully executed even if the begin-commit sequence involved includes the DOWN file. This is because a fatal error may have occurred on a read and the begin-commit sequence might not include any writes to the damaged file. However, a DBFREE is normally preferable to a DBCOMIT.

When DBCOMIT is issued for a begin-commit sequence that includes a flawed file, the condition of the file is ambiguous. If the file is in forced write mode, all the previous updates to it, except for the one that received an error, are already on the disk. The update that received the error is not on the disk and its corresponding after image is not on the ARF.

If the file with an error is not in forced write mode, the after image of the record that caused the error is not on the ARF and, since the buffer with the updates for this file cannot be flushed, some of the previous updates for this file in this begin-commit sequence might not be on the disk.

A DBFREE request for a begin-commit sequence that involves the DOWN file returns an error code to the user but the DBFREE process completes. This means that the before image records for files other than the DOWN one are restored on the data base and that all of the before image records, including those belonging to the problem file, are written on the ARF along with a DBFREE stamp.

When there are no more active users on the file, TAF/CRM declares the file DOWN and submits a DMREC job to recover the file. Refer to DMREC Calls from TAF in section 6 for information on the submitted job. When DMREC has completed its processing (dumping the ARF, reloading the file from the dump tape, reapplying the after image records for the DBCOMITed updates and the before image records for the DBFREEed updates), DMREC notifies TAF/CRM that the file should be declared UP. The transaction processing of the file can now resume.

A file may also be declared DOWN during the automatic recovery processing after the system or TAF abort. During recovery, CTASK issues TRMREC requests for the transactions in the system at the time of failure. TRMREC is a combination of DBFREE and CEASE requests; thus, it receives an error code if a data base file fails during the DBFREE processing. The recovery process continues in spite of the error. When the recovery completes, the data base file is declared DOWN and TAF/CRM submits a DMREC job to recover the file in batch mode. In the meantime, TAF/CRM can continue to process transactions that do not use the file in question.

TAF/CRM declares a file DOWN when an OPEN request cannot be executed properly during TAF recovery. This is most likely to occur on a multiple index file. As previously described, the downed file is recovered by a DMREC batch job while TAF/CRM is processing other transactions not using this file.

ARF FAILURE

When TAF/CRM detects a fatal error on an ARF, it declares the associated data base IDLE. The begin-commit sequences, active in TAF, are allowed to proceed until DBCOMIT, DBFREE, or CEASE (implied DBFREE) requests are issued but no after image records are logged. When no users with opened files remain for the data base, the data base is declared DOWN. TAF/CRM then submits a DMREC job to dump and reallocate the defective ARF.

The data base files are consistent at this point. The committed updates have been applied to the data base files and the freed updates have been rolled back using the BRF. It is advisable to dump the data base files as soon as the data base is downed because the ARF has been corrupted. If this is not done, a problem may occur if subsequent recovery is necessary. This is because the ARF dump tapes contain a gap among the after image records and recovery will leave the data base inconsistent.

DMREC tries to dump the ARF. If the dump is not successful or if the error was detected by DMREC and not by TAF, DMREC issues an error message. The data administrator should then perform the following actions:

1. Take a complete dump to the data base.
2. Allocate a new ARF using the DMREC CREATE directive.

If the dump is successful, DMREC issues a message to the operator and reallocates the ARF, but does not notify TAF. This is still an error condition and the data administrator should be notified for possible action as previously described for an unsuccessful dump.

BRF FAILURE

Failure During a Write Operation

When TAF/CRM detects an error during a write operation on a BRF, it declares the data base IDLE. This means that new DBEGIN and file OPEN requests are not permitted (both return an error code to the requesting task). TAF/CRM issues an appropriate dayfile message and writes a BRF DOWN stamp on the ARF. All of the other data base requests from the active users are permitted. TAF/CRM processing of these requests is altered as follows.

- There is no before image record logging of images that would ordinarily go to the failing BRF.
- The DBFREE requests that reference this BRF will return an error code to the user.
- DBFREE requests that are affected by the bad BRF are not logged on the ARF, thus leaving the begin-commit sequence incomplete.

When there are no more active data base users, TAF/CRM declares the data base DOWN and submits a DMREC job to dump the active ARF and to recover the data base and reallocate the BRF. The structure of this job is given in DMREC Calls from TAF in section 6.

The following sequence of events is used to reconstruct the BRF(s):

1. DMREC locates the correct after image log tape.
2. DMREC searches the after image log tape(s) for the BRF DOWN stamp.
3. DMREC searches the tape forward from that point, assembling a list of the files affected by uncompleted updates and noting all begin-commit sequences terminated by a FREE (the FREE records should be excluded from any subsequent update process).
4. DMREC reallocates one or more BRF file(s) according to the BRF DOWN stamps on the after image dump tapes.
5. If after the BRF went down, all transactions were successfully committed, DMREC notifies TAF/CRM that normal processing can continue.
6. If after the BRF went down, any transaction active at that time did not later commit, the transaction sequence numbers are listed in the DMREC job dayfile for all such transactions and DMREC notifies TAF/CRM that processing cannot continue. This information should be given to the data administrator.

   If the data administrator wishes to rerun the uncommitted transactions, the original input for the transaction is recorded in the JOURO file. If the data administrator wishes to call back those transactions, a DMREC RECOVER may be done.

Failure During a Read Operation

This error can occur under two circumstances: during the normal TAF/CRM operation when DBFREE is being processed, and during the recovery initialization, when TAF has been brought up in recovery mode after a failure. In the latter case, manual intervention is required; refer to BRF Failure and TAF or System Failure, later in this section.

When an error occurs during DBFREE processing, TAF/CRM returns the error code to the requesting task and declares the data base IDLE. TAF/CRM continues to process requests as described in the preceding subsection.

RECOVERY REQUIRING ANALYSIS AND MANUAL INTERVENTION

Following most failures, the CRM data base files are recovered by TAF/CRM and DMREC without the data administrator's intervention. There are situations, however, when intervention is necessary. Such situations generally involve the failure of a recovery file (ARF or BRF) and the failure of another system component.

In the following discussions, several references are made to the ARF header state. This state is specified in the first word of the ARF header and has one of the following values.

0 The file is inactive. It contains no errors that TAF/CRM is aware of and is available for use.

1 The file is the active ARF. It contains no errors that TAF/CRM is aware of and is available for use.

2 The file is attached by DMREC. DMREC sets this header state when it gets the file for dumping if the previous state was 0 or 1.

3 This is the state set by TAF when the file is selected for use upon TAF initialization or ARF switching. TAF changes the state to 1 upon normal termination. Thus, if TAF is not active, an ARF header state of 3 indicates either abnormal TAF termination or CIO errors on this ARF.

TAF FAILURE

When TAF fails, the TAF procedure calls TAF2 into execution at the procedure's EXIT statement. TAF2 makes the recovery request to TAF/CRM which then closes all of the data base files and flushes the file buffers to disk. If the active ARF is attached and not down (header state 1 or 3), the after image buffer is flushed. If the data base was in the process of being idled due to an error on a BRF or ARF, TAF/CRM submits a batch job to allow DMREC to recover, as described previously. If the ARF and BRF are both up, the ARF header state is set to 1. The names of all BRFs and data base files down for recovery are reported in dayfile messages.

After the cause of the TAF failure has been diagnosed and repaired, TAF should be brought up in recovery mode. This causes a rollback of the active begin-commit sequences (from the BRF) and records the pertinent before image records on the ARF together with the DBFREE stamps. Care should be taken to examine TAF's dayfile for any abnormal messages (such as BRF DOWN) before TAF is brought up again or the recovery may not be correct.

LEVEL 3 DEADSTART

The TAF action in this situation is the same as that described for TAF Failure. Level 3 deadstart restarts the TAF procedure at the EXIT statement and causes TAF2 to execute.

LEVEL 0 DEADSTART

When the operating system fails with TAF in execution, an active ARF may have state 3 in its header. It is not possible to initiate TAF when one of its ARFs has state 3. This is because such a state is ambiguous. It may have been caused by the system crash alone, or a disk error may have caused TAF to return this file just before the system crashed. The dayfiles and the error log must be examined before any attempt is made to bring TAF up. If the dayfile indicates that a disk error occurred on any of the ARFs before the system crashed, determining how to recover must be done manually.

If no ARF is defective, DMREC should be initiated manually to dump the active ARF and reallocate it. Then TAF should be brought up in recovery mode to roll back the active begin-commit sequences from BRFs.

TAF OR SYSTEM FAILURE WHILE DMREC IS ACTIVE

If an ARF is found, indicating that DMREC is active (header status of 2), TAF issues a message to the dayfile and terminates processing. This condition occurs if the system crashes with DMREC actively processing a TAF-submitted job. The data administrator must examine system dayfiles and the error log and call DMREC to complete processing.

ARF FAILURE AND TAF OR SYSTEM FAILURE

If during TAF initialization (after TAF or system failure) neither ARF is available for a data base, automatic recovery of that data base is suspended until the operator types K.GO at the console. The reason for ARF absence should be examined by the data administrator and appropriate action taken. This situation is unusual and it is possible that more than one of the system components have failed.

BRF FAILURE AND TAF OR SYSTEM FAILURE

When the BRF for a data base goes down, normally TAF is able to gracefully idle down the transaction activity and inform DMREC of the situation with a BRF DOWN stamp in the ARF. DMREC can then roll back the data base by examining the ARF and selectivly applying the after image records if such an action is appropriate. However, if TAF or the system fails before the transactions are idled, the data base situation is ambiguous.

In this case, there may be updates on the data base file that have not been committed by transactions and DMREC cannot determine which begin-commit sequences to roll back. Under no circumstances should TAF be brought up in a recovery mode with any outstanding BRF errors. The data administrator should submit a batch job requesting DMREC to dump the ARF and to recover the defective BRF. The data administrator should then perform recovery of the data base files by using the DMREC RECOVER directive with appropriate IGNORE subdirectives. The data administrator can decide which transactions to roll back using the printout of the TAF communication recovery file (CRF) files or JOURO, which provides the list of transactions active at a time of failure, and a printout of ARF to correlate these transactions to their begin-commit sequences and ids. TAF may then be brought up in the recovery mode to recover other data bases and files.

NOTE

When DMREC scans the ARFs to determine the files affected by incomplete begin-commit sequences, it begins with the ARF dump tape marked as the beginning of the most recent TAF session. If data base files have been recovered prior to this TAF session but were not archived through DMREC, the DMREC recovery may lead to inconsistent files. Therefore, immediately after DMREC's recovery of the data base files due to a BRF error, the affected data base files should be dumped to tape using DMREC; otherwise, recovery after a subsequent BRF error may be difficult.

If the error is discovered during the recovery initialization, recovery will be terminated immediately. The data base must be recovered manually using DMREC. Neither the BRFs nor the ARFs are altered during the recovery initialization, hence they will be in the same condition as they were when the failure occurred.

When the recovery initialization is complete, TAF schedules CTASK to issue TRMREC requests for all transactions active during the time of failure. TRMREC is equivalent to a DBFREE request followed by the transaction CEASE request; thus, those transactions with open begin-commit sequences are

rolled back. If any errors occur during this process, CTASK continues issuing TRMREC requests until it exhausts the list of transactions; then CTASK brings TAF down. TAF/CRM processing during this time does not differ from its processing during normal TAF operation (data base updates are rolled back, before image records are recorded on ARF as after images, and so on). TAF/CRM reacts to any error condition as it would in normal operation. For example, if a data base file goes down, it is IDLEd; if a BRF goes down, a BRF DOWN stamp is written on the ARF. The batch recovery jobs are submitted as appropriate, but it is recommended that a more thorough analysis of the situation be made by the data administrator.

BRF FAILURE AND ARF FAILURE

If TAF detects a defective BRF, it writes pertinent information on the ARF, idles the transactions down, and submits a DMREC job for possible recovery processing. If DMREC is unable to read the ARF, the data administrator must intervene.

As in the preceding discussion, the printout of ARF is needed to determine which data base updates to roll back. The TAF JOURO file may have to be examined before any decision is made.

DEFECTIVE ARF AND DATA BASE FILE

TAF/CRM submits a batch job for DMREC to recover an individual file that has an I/O error. If an ARF cannot be read from disk or tape, the file must be reallocated and restored from a dump tape and whatever updates are readable may be restored. The objective is to restore the file to a consistent state as of some previous time.

The TAF/CRM batch concurrency feature allows users to access data files held by TAF from a batch or interactive job.† This can be done regardless of the mode in which TAF has the file(s) attached. TAF/CRM batch concurrency is available to all users who are validated to use TAF interactively and have system control point/user control point (SCP/UCP) permission (CUCP bit) specified in their access word.

RESTRICTIONS

As with interactive access to TAF, only one batch job can be active under a given user name at a given time. Also, an interactive user and a batch job cannot run simultaneously under the same user name.

TBCON STATEMENT

TAF/CRM batch concurrency is enabled by the TBCON statement in the TAF configuration file. The format is:

```
TBCON,n.
```

n The number of jobs that may access TAF simultaneously.

If n is zero or if the TBCON statement does not appear in the TCF, batch concurrency is disabled.

The maximum number of TAF/CRM jobs allowed can be changed from the operator console by a run time K display command. The format is:

```
K.TBCON,n.
```

n The number of jobs that may access TAF simultaneously. n can range from zero to the number specified in the TCF.

TASK REQUIREMENTS

TAF/CRM jobs use the requests described in this manual in exactly the same way as interactive transactions. The structure of the batch or interactive job can be identical to that of an interactive task that performs the same function; however, batch jobs cannot use the TAF WAITINP or SEND requests. They may use COBOL or FORTRAN I/O requests to read input and print output.

Batch concurrency jobs must satisfy externals using library BCLIB. This can be done by including the command

```
LDSET,LIB=BCLIB.
```

immediately following the compiler call in the job command list.

RECOVERY

The data base recovery requests (DBEGIN, DBCOMIT, DBSTAT, DBFREE) provide jobs with the same data base recovery capabilities available to interactive transactions. Run unit recovery for batch concurrency jobs is not supported by TAF.

If a user submits several jobs to be processed consecutively and one of them uses begin-commit sequences, all of the jobs not using begin-commit sequences should initially issue a DBSTAT request to determine if recovery is in progress. If it is, any job not using begin-commit sequences should terminate.

If several jobs use begin-commit sequences, each should supply unique begin-commit identifiers. Then, if a job receives an unrecognizable identifier on a DBSTAT request, recovery is in progress for one of the other jobs. The latter should terminate so as to avoid interfering with the recovery information for the former.

ERROR PROCESSING

Errors during processing of batch concurrency requests can be grouped into three categories. Dayfile messages can be found in appendix B of the TAF Reference Manual. Request status can be found in appendix A of this manual.

- Non-zero status

  The TAF/CRM status and TAF status for each request returns to the job and if their values are non-zero, they are put into the job's dayfile.

- Job suspension

  The job is suspended pending operator action if TAF is not up or if TAF is idling down.

- Job termination

  The job terminates and a dayfile message appears for the following reasons:

  - The user name used in this batch job does not have proper validation.
  - There is a problem with the request. For example, there may be a bad function code or data not in field length.
  - A fatal TAF/CRM error occurs. For example, there may be an invalid parameter list or the user is not validated for the data base.

† In the following discussion, the word job refers to either a batch or interactive job.

Table A-1 lists the error status codes returned in the taf-status parameter field.

Table A-1. TAF/CRM Error Status Codes (Sheet 1 of 3)

Error Status Code	Significance	Action
0	No error.	None.
1	The file or data base is not in the xxJ file.	Terminate the task or open another file. Consult the data base administrator.
2	Another transaction has locked the file. TAF/CRM unlocks all records and files held by this transaction.	Reestablish all necessary locks.
3	Another transaction has locked the record. TAF/CRM unlocks all records and files held by this transaction.	Reestablish all necessary locks.
4	Reserved for Control Data.	
5	Reserved for Control Data.	
6	TAF/CRM could not grant the OPEN request and closes all files held by the transaction.	Reopen all necessary files. Consult the data base administrator.
7	There is no table space available for record or file locks. TAF/CRM unlocks all records and files held by the transaction.	Reestablish all necessary locks. Consult the data base administrator to increase the number of locks. If this occurred in a begin-commit sequence, it is necessary to reopen this begin-commit sequence because an internal DBFREE has been performed.
8	A CRM error status code exists in the crm-status parameter field.	Examine the crm-status parameter field and consult the CRM AAM Reference Manual.
9	The transaction tried to unlock a record that was not locked.	Correct the transaction to unlock the correct record.
10	The transaction tried to unlock a file that was not locked.	Correct the transaction to unlock the correct file.
11	The transaction issued a TAF/CRM request on a file that was not opened.	Correct the transaction to open the file before issuing any TAF/CRM requests on it.
12	The transaction has exceeded the maximum number of locks.	Reduce the number of record lock requests in the transaction, use the FLOCK request, or increase the locks parameter in the xxJ file.

Table A-1. TAF/CRM Error Status Codes (Sheet 2 of 3)

Error Status Code	Significance	Action
13	The wsa-length parameter is not large enough.	Increase the wsa-length parameter.
14	The key-area-length parameter is not large enough.	Increase the key-area-length parameter.
15	The record length is less than one or greater than the maximum length specified at the time the file was created.	Correct the record-length parameter.
16	The key-offset parameter is zero, negative, or not in the task field length.	Correct the key-offset parameter.
17	The transaction tried to open a file that was already opened. The file remains opened.	Correct the transaction to avoid issuing duplicate OPEN requests on the same file.
18	The major-key-length parameter is zero, negative, or greater than the key length specified when the file was created.	Correct the major-key-length parameter.
19	The kl parameter in the xxJ file is less than the key length specified when the file was created.	Correct the kl parameter.
20	The mrl parameter in the xxJ file is less than the record length specified when the file was created.	Correct the mrl parameter.
21	The file is positioned at end-of-information.	None.
22	The key-relation parameter contains a value other than EQ, GT, or GE.	Correct the key-relation parameter.
23	The key-identifier is nonnumeric or has no corresponding AKY statement.	Correct the key-identifier.
24	Task has an incorrect recovery unit.	Correct the task to issue a DECOMIT, DBFREE, or DBEGIN request as required.
25	Batch job number returned by DMREC not recognized by TAF/CRM.	Call the data base administrator.
26	Begin-id not found.	The task has made no begin-commit requests, or the task was not recovered.
27	No recovery files not assigned to the data base.	No recovery is possible for files belonging to this data base. TAF data base administrator must specify recovery files in the BRF statement in the xxJ FILE.
28	The task is making a request when the data base or the file is IDLE.	Transaction should cease and be resubmitted at a later time, when the data base is UP.
29	The task is attempting a request not allowed within a begin-commit sequence.	Recoverable files may be closed, and record/file locks released only after a DBCOMIT or DBFREE request.
30	The task is attempting to update a recoverable file without a DBEGIN request.	Modify the task to issue a DBEGIN request before updating a recoverable file.

Table A-1. TAF/CRM Error Status Codes (Sheet 3 of 3)

Error Status Code	Significance	Action
31	The task has exceeded the number of updates allowed inside a begin-commit sequence.	Change the task to do fewer updates within a begin-commit sequence.
32	The data base or file is DOWN.	Rerun the task when the data base and file are both UP.
33	TAF/CRM could not UP all the files in the data base on a DBUP command.	Check affected file(s) for errors. The file(s) may be attached by another job.
34	TAF/CRM could not UP the file specified on a DBUP command.	Check the data file for errors. Data file may be attached by another job.
35	Insufficient buffer size specified for a CRMSTAT function.	Transaction must specify a buffer large enough to hold the status information to be returned (refer to section 7).

Table B-1 lists the error status codes that can be returned for each request. Applications programmers can use this table to decide which error status codes to check.

Table B-2 is a conversion aid that gives the calling formats for COBOL, FORTRAN, and COMPASS under the batch and transaction environments. The languages and environments are listed horizontally, and the calling formats are listed vertically.

Table B-1. Error Status Codes for Each Request

TAF/CRM Request	Error Status Codes
OPEN	0,1,6,8,17,19,20,27,28,32
CLOSE	0,1,8,11,29,32
READ	0,1,8,11,13,16,22,32
READN	0,1,8,11,13,14,21,32
READM	0,1,8,11,13,14,16,18,22,32
READL	0,1,2,3,7,8,11,12,13,16,22,32
READNL	0,1,2,3,7,8,11,12,13,14,21,32
WRITE	0,1,2,3,7,8,11,12,15,16,30,31,32
REWRITE	0,1,2,3,7,8,11,12,15,16,30,31,32
DELETE	0,1,2,3,7,8,11,12,16,30,31,32
LOCK	0,1,2,3,7,11,12,16,32
UNLOCK	0,1,9,11,16,20,32
FLOCK	0,1,2,3,7,11,12,32
UNFLOCK	0,1,10,11,29,32
REWIND	0,1,8,11,32
SKIPFL	0,1,8,11,21,32
SKIPBL	0,1,8,11,32
START	0,1,8,16,18,21,23
DBEGIN	0,24,27,28,32
DBCOMIT	0,24,32
DBFREE	0,24,32
DBSTAT	0,26,32

Table B-2. Conversion Aids

Batch Environment			Transaction or Batch Concurrency Environment		
COBOL	FORTRAN	COMPASS	COBOL	FORTRAN	COMPASS
OPEN	CALL OPENM	OPENM	ENTER "OPEN" USING	CALL OPEN	RJXOPEN
CLOSE	CALL CLOSEM	CLOSEM	ENTER "CLOSE" USING	CALL CLOSE	RJXCLOSE
READ INVALID KEY	CALL GET	GET	ENTER "READ" USING	CALL READ	RJXREAD
READ NEXT	CALL GETN	GETN	ENTER "READN" USING	CALL READN	RJXREADN
READ MAJOR INVALID KEY	CALL GET	GET	ENTER "READM" USING	CALL READM	RJXREADM
None	None	None	ENTER "READL" USING	CALL READL	RJXREADL
None	None	None	ENTER "READNL" USING	CALL READNL	RJXREADNL
WRITE INVALID KEY	CALL PUT	PUT	ENTER "WRITE" USING	CALL WRITE	RJXWRITE
REWRITE INVALID KEY	CALL REPLC	REPLACE	ENTER "REWRITE" USING	CALL REWRITE	RJXREWRITE
DELETE INVALID KEY	CALL DLTE	DELETE	ENTER "DELETE" USING	CALL DELETE	RJXDELETE
None	None	None	ENTER "LOCK" USING	CALL LOCK	RJXLOCK
None	None	None	ENTER "UNLOCK" USING	CALL UNLOCK	RJXUNLOCK
None	None	None	ENTER "FLOCK" USING	CALL FLOCK	RJXFLOCK
None	None	None	ENTER "UNFLOCK" USING	CALL UNFLOCK	RJXUNFLOCK
None	CALL REWND	REWINDM	ENTER "REWIND" USING	CALL REWIND	RJXREWIND
None	CALL SKIP	SKIPFL	ENTER "SKIPFL" USING	CALL SKIPFL	RJXSKIPFL
None	CALL SKIP	SKIPBL	ENTER "SKIPBL" USING	CALL SKIPBL	RJXSKIPBL
START	None	START	ENTER "START" USING	CALL START	RJXSTART
None	None	None	ENTER "DBEGIN" USING	CALL DBEGIN	RJXDBEGIN
None	None	None	ENTER "DBCOMIT" USING	CALL DBCOMIT	RJXDBCOMIT
None	None	None	ENTER "DBFREE" USING	CALL DBFREE	RJXDBFREE
None	None	None	ENTER "DBSTAT" USING	CALL DBSTAT	RJXDBSTAT

AAM

Refer to Advanced Access Methods.

ABH

Refer to Application Block Header.

ABN

Refer to Application Block Number.

Absolute Code

Code in which all required external references have been satisfied. All addresses refer to specific locations within the job field length.

ABT

Refer to Application Block Type.

ACN

Refer to Application Connection Number.

ACT

Refer to Application Character Type.

Actual Key

The primary key for a record in a file with actual key organization; indicates the storage location of the record.

Actual Key (AK) File

A mass storage file in which each record is stored at the location indicated by the primary key. For initial actual key files, the primary key must specify the block and record slot number in which the record is stored. For extended actual key files, the primary key is a record number that AAM converts to the storage location of the record. Access is random or sequential.

Address

The location of a word in memory. The location is designated by number or symbolic name.

Advanced Access Methods (AAM)

A file manager that processes indexed sequential, direct access, and actual key file organizations, and supports the Multiple-Index Processor. Refer to CYBER Record Manager.

After Image

A copy of a DMS-170 data base record after it has been modified.

After Image Recovery File (ARF)

A file upon which TAF writes after images of records in recoverable files when a request to update such a record is processed within a begin-commit sequence.

AK File

Refer to Actual Key File.

Alphabetic Character

A character belonging to the set of letters A through Z and the space.

Alphanumeric

Consisting of alphabetic and/or numeric characters only.

Alternate Key

A key, other than the primary key, by which an indexed sequential, direct access, or actual key file can be accessed.

ANSI

American National Standards Institute. An organization that establishes standards for the benefit of its member organizations.

Application

In TAF, a set of data files and the set of tasks associated with those files.

Application

A program resident in a host computer that provides an information storage, retrieval, and/or processing service to a remote user via the network.

Application Block Header (ABH)

A single 60-bit word description accompanying every block passing between an application program and NAM.

Application Block Number (ABN)

A field in the application block header. An application-assigned number used to identify a particular data message block.

Application Block Type (ABT)

A field in the application block header defining the accompanying block as either data or supervisory, null or not null, and indicating if this is the last block of a message.

Application Character Type (ACT)

A field in the application block header defining the byte size of text characters.

Application Connection Number (ACN)

A number assigned by the Communications Supervisor program to identify a particular logical connection within an application program.

Application Switching

The process of leaving the control of one application and entering the control of another, without going through another complete login sequence.

ARF

Refer to After Image Recovery File.

ASCII

American National Standard Code for Information Interchange. The standard character set and code used for information interchange between systems.

ASCII Mode

Use of the American National Standard Code for Information Interchange; 128-character set. It includes both uppercase and lowercase letters.

Assemble

To transform a COMPASS language program into a form that the computer can execute directly. During assembly, machine language operation codes are substituted for COMPASS codes and machine addresses for symbolic addresses. COMPASS performs the assembly.

Attach

The process of making a permanent file accessible to a job by specifying the proper permanent file identification and passwords.

Auto Mode

Mode of entering information into a primary file where the system automatically generates line numbers. This allows the user to correct, insert, and delete lines easily because the user can reference each line by its line number.

Automatic Login

The process whereby one or more of the Network Validation Facility login dialog parameters is supplied to NVF from the local configuration file. Parameters supplied through automatic login configuration of a terminal suppress prompting for the corresponding dialog entries and override any entries made from the terminal. Automatic login is required for terminals (passive devices) incapable of conducting dialog with NVF.

Backup Directory

A direct access file automatically allocated and maintained by DMREC. It contains information about the data base, index, and after image recovery files for an application. The file name is ZZxxDIR where xx is the application identifier.

Backup Dump

A copy of all or selected portions of a data base, which is produced on a regularly scheduled basis for the explicit purpose of data base recovery.

Batch Job

Instructions and data submitted as a complete unit without further user intervention. The job can be punched on cards or created and submitted from a terminal.

Batch Mode

The state of a mode 4A terminal during which batch data is transmitted from the terminal's card reader to central files and/or from central files to the terminal's line printer. Also, the state of an asynchronous terminal during which input transmission occurs on a block-by-block basis.

Before Image

A copy of a DMS-170 data base record before it has been modified.

Before Image Recovery File (BRF)

A file to which TAF/CRM writes the before images of records in recoverable files when a request to read such a record is processed from within a begin-commit sequence.

Begin-Commit Sequence

A series of data base updates beginning with a data base begin request and ending with a data base commit request or a data base free request.

Beginning-of-Information (BOI)

The start of the first user record in a file. Tape labels and system-supplied information, such as an index block or control word, do not affect beginning-of-information.

Binary File

Usually a noneditable file containing a precompiled program.

Bit

An abbreviation of binary digit. It is a single digit, 0 or 1, in a binary number. Also used to represent the smallest unit of information. A central memory word (one storage location) contains 60 bits.

Block

Blocking is the grouping of user records for efficiency in transfer between memory and storage devices. For magnetic tapes, it is the information between interrecord gaps.

Block

In the context of network communications, a portion or all of a message. A message is divided into blocks to facilitate buffering, transmission, error detection, and correction for variable length data streams.

During input from a terminal, a block is a single transmission comprising one or more lines of one or more messages.

During input to an application program, a block is a single line comprising part or all of a message. Terminal transmission blocks are divided into as many application program input blocks as needed to provide one block of input per terminal input line, until the message is completed.

During output to a terminal, a block is one or more lines comprising one message.

BOI

Refer to Beginning-of-Information.

BRF

Refer to Before Image Recovery File.

Called Task

A task that is the object of a CALLTSK or a CALLRTN request.

Calling Task

A task that executes a CALLSTK or a CALLRTN request.

Carriage Control

The control exercised over the format of printed output. The leftmost character of the data line to be printed is the carriage control character. Carriage control is also called format control.

CCL

Refer to CYBER Control Language.

Central Memory (CM)

The main storage device whose storage cells (words) can be addressed by a computer program and from which instructions and data can be loaded directly into registers from which the instructions can be executed or from which data can be manipulated.

Character

Any alphabetic, numeric, or special symbol that can be encoded. This term applies to the graphic characters for an input or output device, and to uniquely encoded control characters used by a terminal.

Charge Number

An alphanumeric identifier the installation uses to allocate charges to individual users for system usage.

Close

A set of terminating operations performed on a file when input and output operations are complete. All files processed by CRM must be closed.

CM

Refer to Central Memory.

COBOL

COmmon Business Oriented Language. This higher-level language simplifies the programming of business data applications.

Command

A sequence of words and characters that call a system routine to perform a job step. The command must conform to format specifications and end with either a period or a right parenthesis. A command is sometimes called a control statement.

Compile

To translate a program from a higher-level programming language (for example, FORTRAN or BASIC) into machine instructions called object code.

Concurrency

Simultaneous access to the same data in a data base by two or more applications programs during a given span of time.

Console

A terminal devoted to network control processing. There are three such terminals: the Network Operator's (NOP) terminal, the Local Operator's (LOP) terminal, and the NPU console.

Control Statement

Refer to Command.

CRM

Refer to CYBER Record Manager.

CYBER Control Language (CCL)

A group of commands that enable a user to control the order of execution of commands within a batch job or procedure. With CCL, the user can conditionally skip or process commands, process and reprocess a group of commands, and process commands in a file other than the job file.

CYBER Loader

The utility that prepares programs for execution by placing program instruction and data blocks in central memory.

CYBER Record Manager (CRM)

A software product that allows a variety of record types, blocking types, and file organizations to be created and accessed. The execution time input/output of COBOL 4, COBOL 5, FORTRAN Extended 4, FORTRAN 5, Sort/Merge 4, ALGOL 4, and the DMS 170 products is implemented through CYBER Record Manager. Neither the input/output of the operating system nor any of the system utilities such as COPY or SKIPF is implemented through CYBER Record Manager. All CYBER Record Manager file processing requests ultimately pass through the operating system input/output routines.

Data Base

A systematically organized, central pool of information.

Data Base Administrator

The individual who leads the design, programming, implementation, and maintenance efforts associated with a data base.

Data Block

A block in which user records are stored in an indexed sequential or actual key file. Data block structure is defined by the user, or AAM defaults are accepted. Contrast with Index Block for indexed sequential files.

Data Management System

Refer to Database Management System.

Database Management System

A generalized computer program that handles the mechanics of storing, updating, and accessing data for multiple applications.

Dayfile

A chronological file created during job execution which forms a permanent accounting and job history record. Dayfile messages are generated by operator action or when commands are processed. A copy of the dayfile is printed with the output for each job.

Deadlock

A situation that arises in concurrent data base access when two or more applications programs are contending for a resource that is locked by one of the other ones, and none of the programs can proceed without that resource.

Deadstart

The process of initializing the system by loading the operating system library programs and any of the product set from magnetic tape or disk. Deadstart recovery is reinitialization after system failure.

Default

A value supplied by the system when the user omits its specification from a parameter list.

Delimiter

A character used to separate statement elements, such as words and literal constants, or other strings of text.

Device

The physical recording medium of random mass storage, such as a disk pack.

Device Set

A group of rotating mass storage devices. No device can belong to more than one device set. Every file must be contained within one device set, but can be on different devices in that device set.

Direct Access File

A CYBER Record Manager file containing records stored randomly in home blocks according to the hashed value of the primary key in each record. Files must be mass storage resident. All allocation for home blocks occurs when the file is opened on its creation run. Access is random or sequential.

Direct Access File

An NOS permanent mass storage file that can be attached to the user's job. All changes to this file are made on the file itself rather than a temporary copy of the file (contrast with Indirect Access File).

Directives

The instructions that supplement processing defined by a command or by a program call for execution of a utility function or member of a product set. Directives do not appear in the command record; they are usually in a separate record of the file INPUT or a file referenced in a command call. Directives are required for execution of FORM, the CREATE utility, and EDITLIB among others.

Disk

A unit composed of one or more flat, circular plates with magnetic material on both sides that is used to store large amounts of data or programs.

Display Code

A 6-bit character code set used to represent alphanumeric and special characters.

Displays

Two console screens or a split screen used to display system and job information, operator messages, and contents of central memory. Through the console keyboard, the operator can control the operation of the system. The displays are identified by alphabetic characters; some used frequently are: job status (B), system files (H), and dayfile messages (A).

Dump

A printed listing of the contents of the central processor registers and a predetermined number of words within central memory that pertain to a job.

Dynamic Access

Access mode that allows a nonsequential file on mass storage to be accessed randomly or sequentially depending on the format of the access statement.

ECS

Refer to Extended Core Storage.

End-of-File (EOF)

A boundary within a sequential file, but not necessarily the end of a file that can be referenced by name. The actual end of a named file is defined by EOI. In the product set manuals, an end-of-file is also referred to as an end-of-partition.

End-of-Information (EOI)

An indicator that marks the physical end of a named file.

End-of-Record (EOR)

The terminator of a logical record. On a PRU device, a short PRU or a zero length PRU with a level designator of 0 indicates EOR. On tapes that are not PRU devices, an interrecord gap indicates EOR.

EOF

Refer to End-of-File.

EOI

Refer to End-of-Information.

EOR

Refer to End-of-Record.

Exchange Package

A table that contains information used during job execution. The system prints the table as part of the standard output of an aborted job.

Extended

A term used in conjunction with indexed sequential, direct access, and actual key files to denote a specific type of internal processing by AAM and MIP. Processing is indicated by setting the ORG FIT field to NEW.

Extended Core Storage (ECS)

Optional additional memory. ECS contains 60-bit words; it has a large amount of storage and fast transfer rates. ECS can be used only for program and data storage, not for program execution. Special hardware instructions exist for transferring data between central memory and ECS.

Extended Memory

An extension to central memory that is physically located outside of the machine. Formerly referred to as Extended Core Storage (ECS) or Large Central Memory (LCM).

Family Name

Name of the permanent file device or set of devices on which all of the user's permanent files are stored. When you request a permanent file, the system looks for it on this family of devices. Usually a system has only one family of permanent file devices but it is possible to have alternate families in the system. The user specifies which family he or she uses to log in. This family name is provided by personnel at the user's installation.

Fast Dynamic Loader (FDL)

A facility that provides fast loading and unloading of specially formatted code called capsules. The amount of memory required for job execution can be greatly reduced because capsules can be easily loaded and unloaded as needed, freeing memory for other uses.

FDL

Refer to Fast Dynamic Loader.

FE

Refer to Format Effector.

FET

Refer to File Environment Table.

Field Length

The area in central memory allocated to a particular job; the only part of central memory that a job can directly access. Also, the number of central memory words required to process a job.

File

A set of information referenced by a file name (one- to seven-alphanumeric characters). A user can create files at the terminal and can retrieve stored files for use during a terminal session.

FILE Command

A command that supplies file information table values after a source language program is compiled or assembled but before the program is executed. Basic file characteristics such as organization, record type, and description can be specified in the FILE command.

File Environment Table (FET)

A table within a program's field length through which the program communicates with operating system input/output routines. One FET exists for each file in use by the program.

File Information Table (FIT)

A table through which a user program communicates with CYBER Record Manager. For direct processing through CRM, a user must initiate establishment of this table. All file processing executes on the basis of information in this table. The user can set FIT fields directly or use parameters in a file access call that sets the fields indirectly. Some product set members set the fields automatically for the user.

File Organization

Defined by ORGANIZATION clause. Can be sequential, indexed, relative, direct, actual-key, or word-address. Established at time file is created and cannot change as long as file exists. Affects access mode, open mode, and formats of statements that can be used to manipulate file records.

Header

A word or set of words at the beginning of a block, record, file, or buffer which contains control information for that unit of data.

Host

A computer that executes application programs.

IAF

Refer to Interactive Access Facility.

Index

In the context of AAM, a series of keys and pointers to records associated with the keys. The system creates an index for AAM files which relates alternate record keys to primary keys, using the file defined by the second implementor-name of an ASSIGN clause.

In general context, a computer storage area or register, the content of which represents the identification of a particular element.

Index Block

For an indexed sequential file, a block with ordered keys and pointers to data blocks and other index blocks, forming a directory of the records within a file.

Indexed Sequential (IS) File

A file organization in which the CYBER Record Manager maintains records in sorted order by use of a programmer-defined key, which need not be within the record. Keys may be integer, floating point, or symbolic; access is random or sequential. Files contain index blocks and data blocks.

Indirect Access File

A NOS permanent file that you access by making a temporary copy of the file (GET or OLD command). You create or alter it by saving or substituting the contents of an existing temporary file (REPLACE or SAVE command).

Input

Information flowing upline from terminal to host computer.

Interactive

A mode of job processing in which the user enters a command or input to an executing program and receives an immediate response. Contrast with Batch Mode.

Interactive Access Facility (IAF)

An application program that provides a terminal operator with interactive processing capability. The interactive access facility makes terminal input/output and file input/output appear the same to an executing program.

I/O

Input/Output.

IS File

Refer to Indexed Sequential File.

Itemize

A utility routine that produces a listing of the contents of a file or library.

IXGEN

The utility that adds or deletes an index from the index file for the initial MIP.

Job

All activity associated with a terminal session from login to logoff. A batch job consists of instructions and data that is submitted as a complete unit.

Key

A group of contiguous characters or numbers the user defines to identify a record in a CRM file.

Keylist

A list of one or more primary key values associated with a specified value of an alternate key.

LDSET

The loader directive used to control the load process under a variety of conditions.

LGO

The default name of the file to which language processors write executable code during program assembly or compilation.

Load Sequence

A sequence of load operations which encompasses all of the loader's processing from the time that nothing is loaded until the time execution begins. It includes initialization, specification of specified loader requests, and completion of the load.

Loader

A software product that prepares programs for execution by placing program instructions and data blocks in central memory and linking references in the program to the appropriate external routines.

Local

Refers to data that exists only during the processing of a single job and that can only be accessed by that job.

Local Operator (LOP)

The network operator who monitors line, terminal, and application activities in the network.

Login

The procedure used at an interactive terminal to gain access to the system.

Logout

The procedure by which you end a terminal session. You type BYE.

LOP

Refer to Local Operator.

Macro

A sequence of source commands that are saved and then assembled whenever needed through a macro call.

Major Alternate Key

The leading portion of an alternate key field.

Major Key

The leading characters of a symbolic key in an indexed sequential file.

Major Primary Key

The leading portion of a primary key field.

Mass Storage

Magnetic disk or extended memory that can be accessed randomly as well as sequentially.

Master Directory

A file containing information used by CDCS in processing. This information consists of schema and subschema tables, media parameters, and data base procedure library and logging specifications.

MIP

Refer to Multiple Index Processor.

MIPGEN

The utility that adds or deletes an index from the index file for extended MIP.

Multifile Set

A tape file set having more than one tape file.

Multimainframe Operation

Operation that provides mechanisms by which more than one computer can share mass storage devices.

Multiple Index File

An indexed sequential, direct access, or actual key file that has alternate keys defined.

Multiple Index Processor (MIP)

A processor that allows AAM files to be accessed by alternate keys.

NAM

Refer to Network Access Method.

NCTF

Refer to Network Description File.

NDL

Refer to Network Definition Language.

Network Access Method (NAM)

A software package that provides a generalized method of using a communications network for switching, buffering, queuing, and transmitting data. NAM is a set of interface routines used by a terminal servicing facility for shared access to a network of terminals and other application programs, so that the facility program does not need to support the physical structures and protocols of a private communication network.

Network Definition Language (NDL)

The compiler-level language used to define the network configuration file and local configuration file contents.

Network Description File (NCTF)

File that must be present if the transaction facility is used. The file is prepared by the site analyst.

Numeric Character

Digits 0 through 9.

Object Code

The machine language version of a program that has been translated (compiled) from source code written in a higher level language.

Open

A set of preparatory operations performed on a file before input and output can take place; required for all CRM files.

Operating System

The set of system programs that controls the execution of computer programs and provides scheduling, error detection, input/output control, accounting, compilation, storage assignment, and other related services.

Operation

A particular function performed on units of data; for instance, opening or closing an area, or storing or deleting a record.

Order Dependent

Used to describe items which must appear in a specific order.

Multiple-Index File

An indexed sequential, direct access, or actual key file that has alternate keys defined.

Multiple-Index Processor (MIP)

A processor that allows AAM files to be accessed by alternate keys.

NAM

Refer to Network Access Method.

NCTF

Refer to Network Description File.

NDL

Refer to Network Definition Language.

Network Access Method (NAM)

A software package that provides a generalized method of using a communications network for switching, buffering, queuing, and transmitting data. NAM is a set of interface routines used by a terminal servicing facility for shared access to a network of terminals and other application programs, so that the facility program does not need to support the physical structures and protocols of a private communication network.

Network Definition Language (NDL)

The compiler-level language used to define the network configuration file and local configuration file contents.

Network Description File (NCTF)

File that must be present if the transaction facility is used. The file is prepared by the site analyst.

Numeric Character

Digits 0 through 9.

Object Code

The machine language version of a program that has been translated (compiled) from source code written in an original higher-level language.

Open

A set of preparatory operations performed on a file before input and output can take place; required for all CRM files.

Operating System

The set of system programs that controls the execution of computer programs and provides scheduling, error detection, input/output control, accounting, compilation, storage assignment, and other related services.

Operation

A particular function performed on units of data; for instance, opening or closing an area, or storing or deleting a record.

Order Dependent

Items that must appear in a specific order.

Order Independent

Items that need not appear in any specific order. Parameters, particularly those with keywords, may be order independent.

Output

Information flowing downline from host to terminal.

Output File

The file on which the system writes information to the user. Unless another file name is specified, the OUTPUT file is printed at the terminal.

Pack Name

A one- to seven-character name that identifies the auxiliary device to be accessed in a permanent file request.

Parameter

A variable that is given a specific value for a particular purpose or process.

Parity

In writing data, an extra bit is either set or cleared in each byte so that every byte has either an odd number of set bits (odd parity) or an even number of set bits (even parity). Parity is checked on a read for error detection and possible recovery.

Password

A name or word entered during login to provide extra security for a user's user name. A unique password ensures that no one else can log into the system using another user's user name and access another user's files. Initially the password is given to a user by installation personnel. Depending on the privileges assigned to the user name, a user may be able to change his or her password. Also, a user can assign a password to any file being saved. Any other user who wants to use your file must specify that password when he or she accesses the file. The file password has no connection with the login password.

Permanent File

A file stored on mass storage. This file is cataloged by the system so that its location and identification are always known to the system. Permanent files cannot be destroyed accidentally during normal system operation. They are protected by the system from unauthorized access according to privacy controls specified when they are created.

Physical Record Unit (PRU)

The amount of information transmitted by a single physical operation of a specified device. For mass storage files, a PRU is 64 central memory words (640 characters); for magnetic tape files, the size of the PRU depends upon the tape format. A PRU that is not full of user data is called a short PRU; a PRU that has a level terminator but no user data is called a zero-length PRU (refer to Zero-Length PRU).

Primary Key

A field in a record whose value uniquely identifies a record and determines the location of the record in a file. One primary key field exists for a given file. A file must be updated by primary key values. Contrast with Alternate Key.

Procedure

A user-defined set of instructions and/or commands that are referenced by name.

Procedure File

A file containing one or more procedures.

PRU

Refer to Physical Record Unit.

Purge

To delete a permanent file from the system. This enables releasing its mass storage space, erasing its catalog entries, and so forth.

Queue

A sequence of blocks, tables, messages, and so forth. Most NPU queues are maintained by leaving the queued elements in place and using tables of pointers to the next queued element. Most queues operate on a first in, first out basis. A series of worklist entries for a specific terminal is an example of an NPU queue.

RA

Refer to Reference Address.

RAE

Refer to Reference Address.

Record

A unit of information. In CYBER Record Manager and its language processors, a record is a unit of information produced by a single read or write request. Eight different record types exist within CRM. The user defines the structure and characteristics of records within a file by declaring a record format.

Other parts of the operating system and its products might have additional or different definitions of records.

Record Slot Number

The position of a record within a block in an actual key file; specified by the low-order bits of the primary key.

Record Type

The term record type can have one of several meanings, depending on the context. CYBER Record Manager defines eight record types established by an RT field in the file information table. Tables output by the loader are classified as record types such as text, relocatable, or absolute, depending on the first few words of the tables.

Recovery

A process that makes a data base useful after some type of software or hardware failure has occurred.

Reference Address (RA and RAE)

RA is the absolute central memory address that is the starting, or zero, relative address assigned to a program. Addresses within the program are relative to RA. RA+1 is used as the communication word between the user program and Monitor. RE is the absolute extended core storage starting address assigned to file.

Subschema

A detailed description of the portion of a data base to be made available to one or more application programs.

Supervisory Message

A message block in the host not directly involved with the transmission of data, but which provides information for establishing and maintaining an environment for the communication of data between the application program and NAM, then through the network to a destination or from a source. Supervisory messages may be transmitted to an NPU in the format of a service message.

TAF Configuration File (TCF)

A file created and maintained by the data administrator specifying which data managers are active and the applications to be associated with each.

Task

A program to be executed under the transaction subsystem. Tasks are stored in absolute binary format on libraries that must be available to the transaction subsystem.

Task Library Directory (TLD)

The directory on each task library that contains information on the tasks in the library. TAF makes its own copy of each TLD at initialization and uses this copy to locate requested tasks.

TCF

Refer to TAF Configuration File.

Terminal

The equipment a person uses to communicate with the computer.

Terminal Class

An NDL parameter describing the physical attributes of a group of similar terminals, in terms of an archetype terminal for the group.

Terminal Session

Period between the time you physically connect the terminal to the system in preparation for login to the time you log out.

Text Mode

Mode of entering information into a primary file without specifying line numbers. It is usually used to create data files. If a file does not contain line numbers, you can change lines only by using a text editor such as the Full Screen Editor.

TLD

Refer to Task Library Directory.

Transaction

A task or a series of tasks executed as a unit under the transaction subsystem. All tasks in a transaction have the same transaction sequence number. In CDCS, the term data base transaction refers to the process which this manual calls a begin-commit sequence.

Transaction Library Directory (TRD)

The directory on each task library which contains information on the transactions in the library. TAF makes its own copy of each TRD at initialization and uses this copy to locate requested transactions.

Transaction Name

The one- to seven-alphanumeric-character name given to a transaction created with the /tname input directive from the LIBTASK utility.

Transaction Recovery File

Random permanent file used for before image records necessary for recovery by CDCS.

Transaction Sequence Number

A unique number assigned by TAF to each transaction. It is used by TAF for task identification.

Transparent Mode

A software feature provided by the Network Access Method and the network processing unit (NPU). When transparent mode transmission occurs between an application and a terminal, the Network Access Method does not convert data to or from 6-bit display code, and the NPU does not edit the character stream or convert the characters to or from 7-bit ASCII code. When no parity is in effect for the terminal and transparent mode transmission occurs, all seven bits of the character byte can be used to represent characters in character sets (such as EBCDIC).

UCP

User Control Point.

User Index

A unique 17-bit identifier that is associated with each user name. The user index is used by the permanent file manager to identify the device and catalog track for the user's files.

User Name

A system access word that must be supplied by the user for validation purposes at login.

Utility

A program designed to help the user perform specific functions such as permanent file maintenance, library maintenance, and file editing.

Virtual Terminal

A software concept for CCP that converts all types of upline messages to one of two formats: batch virtual terminal (BVT) or interactive virtual terminal (IVT). By this method, application programs in the host need only to be able to process data in IVT or BVT format rather than in the multiplicity of formats which real terminals use. Downline messages from the host to real terminals are converted from IVT or BVT to real terminal format. The IVT/BVT processors are a part of the NPU's network communications software.

Volume Serial Number (VSN)

A one- to six-character identifier that identifies the volume of magnetic tape for the operator.

VSN

Refer to Volume Serial Number.

Word

A group of bits (or 6-bit characters) between boundaries imposed by the computer system. A word is 60 bits in length. The bits are numbered 59 through 0, starting from the left. It is also composed of five 12-bit bytes, numbered 0 through 4, from the left.

Working Storage Area

An area within the user's field length intended for receipt of data from a file or transmission of data to a file.

Write Mode

Allows a user to write, modify, append, read, execute, or purge the file (modify permission applies only to direct access files).

XEDIT

A text editor available for use on NOS.

xxJ Files

Files created and maintained by the data administration which associates a particular user with a particular application.

Zero-Length PRU

A PRU that contains system information, but no user data. Under the CYBER Record Manager, a zero-length PRU of level 17 is a partition boundary. Under NOS, a zero-length PRU defines EOF.

XEDIT

A text editor available for use on NOS.

Zero-Length PRU

A PRU that contains system information, but no user data. A zero-length PRU defines an end-of-file.

Table D-1 shows the error status codes returned to transaction T2 after attempting to perform a request on a file or record held by transaction T1. Refer to appendix A for more information about these error status codes.

Error Status Codes	Significance
0	No error.
2	Another transaction (in this case, T1) has the file locked; TAF/CRM releases all record and file locks held by the transaction (in this case, T2).
3	Another transaction (in this case, T1) has the record locked; TAF/CRM releases all record and file locks held by the transaction (in this case, T2).

Table D-1. Errors Returned in Conflicting Requests

T2 \ T1	READ	READN	READM	READL	READNL	WRITE	REWRITE	DELETE	LOCK	FLOCK
READ	0	0	0	0	0	0	0	0	0	0
READN	0	0	0	0	0	0	0	0	0	0
READM	0	0	0	0	0	0	0	0	0	0
READL	0	0	0	3	3	3	3	3	3	2
READNL	0	0	0	3	3	3	3	3	3	2
WRITE	0	0	0	3	3	3	3	3	3	2
REWRITE	0	0	0	3	3	3	3	3	3	2
DELETE	0	0	0	3	3	3	3	3	3	2
LOCK	0	0	0	3	3	3	3	3	3	2
FLOCK	0	0	0	3	3	3	3	3	3	2

TAF/CRM AAM PARAMETER CORRESPONDENCE

E

When an AAM file is created, many parameters can be specified either on the FILE command or placed directly in the FIT by the program to control various attributes of the file being created. RT (record type), FO (file organization), and FL (fixed record length) are specific examples. Under TAF/CRM, however, FILE command processing is disabled and application transactions do not have access to the files FIT. As a result, the only parameters supported under TAF/CRM are those which:

- Are placed in the FIT by TAF/CRM,

  or

- Become a part of the file at creation and need not be specified during file processing.

Table E-1 contains AAM parameters supported under TAF/CRM and their usage. This is not a complete list of AAM parameters supported. Other AAM parameters (not included in the table) that meet the following criteria also work under TAF/CRM.

- The parameter needs to be specified only during file creation.
- The parameter does not need to be specifed in the FIT or elsewhere during file processing.
- When used during file creation, the parameter does not cause AAM to expect some other parameter(s) to be specified during file processing.

For example, if a file is created with RT=R, then AAM expects FIT field RMK to be set to the record mark character during file processing. Since this violates the third criterion above, RT=R type files cannot be used under TAF. On the other hand, if the FLM parameter is used when a file is created, the file can be used under TAF since all of the preceding criteria are met.

Table E-1. AAM Parameters Supported under TAF/CRM (Sheet 1 of 3)

AAM Keyword	Values Supported by TAF/CRM	Meaning	Where Specified in AAM Environment	Where Specified in TAF/CRM - xxJ File	Comments
FO	IS,DA,AK	Specifies file organization.	FILE command/FIT	CRM statement	
RT	F, W, Z, U	Specifies record type.	FILE command/FIT		Specified only when the file is created.
MRL	†	Specifies maximum record length.	FILE command/FIT	CRM statement	
PD	INPUT,IO	Specifies processing directive.	FILE command/FIT	CRM statement mode parameter R = INPUT M, RM, W = IO	An AAM file can not be used under TAF/CRM in OUTPUT mode. This means that a file cannot be created under TAF/CRM.
HRL	†	Specified hashing routine address. Direct access files only.	FIT	CRM statement parameter HASH specifies the name of an indirect access permanent file containing relocatable binary of hashing routines	The entry point name must be the same as the HASH file name.

† The possible values are the same as for AAM files.

Table E-1. AAM Parameters Supported under TAF/CRM (Sheet 2 of 3)

AAM Keyword	Values Supported by TAF/CRM	Meaning	Where Specified in AAM Environment	Where Specified in TAF/CRM - xxJ File	Comments
FWI	YES,NO	Forced write indicator.	FILE command/FIT	CRM statement	With FWI=YES, AAM writes all blocks to disk as soon as core image is modified. With FWI=NO, AAM writes changed blocks to disk only when it is necessary. This parameter has significant effect on performance. For recoverable files, FWI=NO is recommended. In this case, all blocks modified in core are flushed to disk on an DBCOMIT or DBFREE request.
LFN	†	Local file name of AAM file.	FILE command/FIT	CRM statement	
KL	†	Specifies the length of primary key.	FILE command/FIT	CRM statement	
EMK	YES,NO	Specifies if the primary key is pacer of the record.	FILE command/FIT		Specified only when the file is created.
XN	†	Index file name for MIP files only.	FILE command/FIT	IXN statement	
ORG	NEW	Specifies OLD or NEW file organization.	FILE command/FIT	TAF/CRM initialization	Only files of ORG=NEW are supported under TAF/CRM.
MKL	†	Specifies the major key length.	FIT	Call to TAF/CRM	Major key access is supported under TAF/CRM via the appropriate call.
RKW		Specifies word within record where alternate starts, counting from zero.	RMKDEF/FIT	AKY statement	TAF/CRM sets the FIT fields RKW and RKP from the value specified for the ako parameter on the AKY statement. The ako parameter specifies the beginning character of the alternate key. The first character of the record is position 1. The relationship is: ako=10*rkw+rkp+1.
RKP		Beginning character position of key 0 to 9 counting left to right in the word (rkw) containing the key.	RMKDEF/FIT		
KL	†	Length of key (alternate) in characters.	RMKDEF	AKY statement	TAF/CRM sets the FET field KL from the value specified by akl parameter on the AKY card for the alternate key.

† The possible values are the same as for AAM files.

Table E-1. AAM Parameters Supported under TAF/CRM (Sheet 3 of 3)

AAM Keyword	Values Supported by TAF/CRM	Meaning	Where Specified in AAM Environment	Where Specified in TAF/CRM - xxJ File	Comments
KS	U,I,F	Specifies the storing order of of primary keys for MIP files.	RMKDEF	AKY statement	The value specified at file creation time stays in effect during processing under TAF/CRM.
REL	EQ,GT,GE	Specifies the relation operator on the START request.	FIT	Call to TAF/CRM	

INDEX

COMMENT SHEET

MANUAL TITLE: CDC TAF/CRM Data Manager Version 1 Reference Manual

PUBLICATION NO.: 60459510 **REVISION:** C

NAME: _____

COMPANY: _____

STREET ADDRESS: _____

CITY: _____ **STATE:** _____ **ZIP CODE:** _____

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 8241 MINNEAPOLIS, MINN.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION

Publications and Graphics Division
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD