CA-MetaCOBOL™ +

Program Development Reference
CA-DATACOM / DB

Release 1.1

R203M+11DRP

Release 1.1, January, 1992
Updated March, 1992, May, 1992

Contents

1. About This Manual

This manual provides the syntax for all CA-DATACOM/DB Facility constructs and statements. The statements are organized alphabetically under the program division in which they are coded.

1.1 Purpose

The CA-MetaCOBOL+ CA-DATACOM/DB Facility extends the COBOL programming language by providing full integration with CA-DATADICTIONARY, a simple but powerful Data Manipulation Language (DML) to access CA-DATACOM/DB from COBOL programs, and complete support for set-at-a-time or record-at-a-time processing.

High-level statements request CA-DATACOM/DB services with statements that are easy to write and easy to read. The CA-MetaCOBOL+ Translator turns these statements into equivalent CA-DATACOM/DB parameters, work areas, request areas, and CALL statements. The CA-MetaCOBOL+ Translator produces conventional COBOL.

1.2 Organization

The CA-MetaCOBOL+ Program Development Guide - CA-DATACOM/DB is the companion to the CA-MetaCOBOL+ Program Development Reference - CA-DATACOM/DB.

Chapter	Description
1	Discusses the purpose of the manual, gives a list of CA-MetaCOBOL+ documentation, and explains notation conventions for CA-MetaCOBOL+.
2	Describes the PROGRAM-ID paragraph.
3	Explains the DATACOM section statement.
4	Describes DATAVIEW statements.
5	Describes the remaining statements.
Appendix A	Shows the relationships between CA-MetaCOBOL+ CA-DATACOM/DB Facility statements and CA-DATACOM/DB commands.
Appendix B	Lists generated data-names.
Appendix C	Lists reserved words.
Appendix D	Contains diagnostics.
Appendix E	Contains return codes.
Appendix F	Describes how to use the Dynamic RQA to build Request Qualification Areas based on user's runtime responses.
Index	Gives page references for important topics covered in this manual.

1.3 Publications

This manual assumes familiarity with the COBOL language, a working knowledge of CA-MetaCOBOL+ and its CA-DATACOM/DB Facility. You may want to refer to the following documentation supplied with CA-MetaCOBOL+. All manuals are updated as required. Instructions accompany each update packet.

Title	Contents
Introduction to CA-MetaCOBOL+	Introduces the CA-MetaCOBOL+ Work Bench, Structured Programming Facility, Quality Assurance Facility, CA-DATACOM/DB Facility, Macro Facility, Panel Definition Facility, the Online Programming Language and the Chart Writer Language.
Installation Guide - MVS	Explains how to install CA-MetaCOBOL+ in the MVS environment.
CA-ACTIVATOR Installation Supplement - MVS	Explains how to install CA-MetaCOBOL+ in the MVS environment using CA-ACTIVATOR.
Installation Guide - VSE	Explains how to install CA-MetaCOBOL+ in the VSE environment.
Installation Guide - CMS	Explains how to install CA-MetaCOBOL+ in the CMS environment.
User Guide	Explains how to customize, get started, and use CA-MetaCOBOL+. Includes information on keyword expansion, the CA-MetaCOBOL+ translator, and CA macro sets and programs.
Structured Programming Guide	Introduces the Structured Programming Facility. Includes information on creating, testing, and maintaining structured programs.
Macro Facility Tutorial	Introduces the Macro Facility. Includes information on writing basic macros, model programming, macro writing techniques, and debugging.
Macro Facility Reference	Includes detailed information on the program flow of the CA-MetaCOBOL+ macro translator, macro format, definition of comments, macro nesting, macro prototypes, symbolic words, and model programming.
Quality Assurance Guide	Introduces the Quality Assurance Facility. Includes all the necessary information to perform quality assurance testing on COBOL source programs.
Program Development Guide CA-DATACOM/DB	Includes all the information necessary to develop programs that make full use of the functions and features of the CA-DATACOB/DB environment.
Panel Definition Facility Command Reference	Contains all Panel Defintion Facility commands.

Panel Definition Facility User Guide	Includes all the information necessary to create, edit, duplicate, rename, delete, index, and print panel definitions and members. Also describes how to generate BMS source.
Online Programming Language Reference	Contains all Online Programming Language statements.
Online Programming Language Guide	Provides further instruction for using Online Programming Language statements.
PC User Guide	Explains how to use CA-MetaCOBOL+/PC. Includes information on the CA-MetaCOBOL+ translator and CA macro sets and programs. Also decribes the relationship between CA-MetaCOBOL+ and CA-MetaCOBOL+/PC.
Program Development Guide CA-DATACOM/PC	Describes how to develop programs that use the CA-DATACOM/PC environment.
String Manipulation Language Guide	Introduces the String Manipulation Language, which provides string handling and inspection capabilities unavailable in COBOL.

1.4 Notation Conventions

The following conventions are used in the command formats throughout this manual:

UPPERCASE	is used to display commands or keywords you must code exactly as shown.
lowercase italic	is used to display information you must supply. For example, DASD space parameters may appear as *xxxxxxx xxxxxxx xxxxxxx*.
<u>Underscores</u>	either show a default value or represent the highlighting of a word in a screen image.
Brackets [ ]	mean that you can select one of the items enclosed by the brackets; none of the enclosed items is required.
Braces { }	mean that you must select one of the items enclosed by the braces.
Vertical Bar |	separates options. One vertical bar separates two options, two vertical bars separate three options, and so on. You must select one of the options.
Ellipsis **. . .**	means that you can repeat the word or clause that immediately precedes the ellipsis.

2. Identification Division

PROGRAM-ID Paragraph;

During the translation step, CA-MetaCOBOL+ signs on to CA-DATADICTIONARY when the first CA-DATADICTIONARY DATAVIEW Statement is located. The program-name, which is coded following the PROGRAM-ID statement, must be defined as a program entity-occurrence in CA-DATADICTIONARY.

Format:

```
PROGRAM-ID. program-name.
```

program-name
: is any valid 1- to 8-character COBOL program name. The program-name must be defined to CA-DATADICTIONARY if the program uses CA-DATADICTIONARY dataviews.

Note: The DATE-PRECOMP and PRODUCT-ID fields of the CA-DATADICTIONARY entity-occurrence for the *program-name* are updated with the current date and $DL, respectively. When the entity-occurrence is in PROD status, these fields are not updated.

Example:

```
PROGRAM-ID. PAYROLL.
```

3. Environment Division

DATACOM SECTION

The DATACOM SECTION statement contains information relating to the translation of batch or online CA-METACOBOL+ CA-DATACOM/DB Facility programs. It also provides optional information concerning CA-METACOBOL+ CA-DATACOM/DB Facility listings.

Format:

```
DATACOM SECTION.
          [MONITOR IS CICS]

          [PRINT {GEN  } ]
          [      {NOGEN} ]

          [ID-AREA IS id-area-identifier].
```

MONITOR
: specifies the data communications monitor to be used with the application program. The MONITOR clause is invalid in a batch environment.

CICS
: specifies that IBM's CICS is the data communications monitor used by this application.

PRINT
: specifies the CA-METACOBOL+ CA-DATACOM/DB Facility listing option.

GEN
: specifies that generated Data Division code is to appear on the Input Listing.

NOGEN

specifies that generated code is not to appear on the Input Listing. This is the default.

ID-AREA

specifies the name of the user-defined program identification area. This clause is required for CA-DATACOM/DB 7.4 or later. It is also required if DBNTRY is used as the entry point for CA-DATACOM/DB 7.3.

id-area-identifier

is a 1-30 character COBOL data-name of the ID-AREA located in the Data Division.

Note: The following statements are invalid in a program when the MONITOR IS clause is coded in the DATACOM SECTION:

LOCATE/READ SEQUENTIAL
LOCATE/READ PHYSICAL.

Example:

```
ENVIRONMENT DIVISION.
DATACOM SECTION.
   ID-AREA IS PROGRAM-ID-AREA.
   . . .
DATA DIVISION.
WORKING-STORAGE SECTION.
01  PROGRAM-ID-AREA.
   02  FILLER VALUE 'DLPROB1' PIC X(7).
   02  FILLER PIC X(25).
```

4. Data Division

The DATAVIEW Statement

The DATAVIEW Statement generates data areas that contain the formatted request for CA-DATACOM/DB services and the data returned by CA-DATACOM/DB. For the DATADICTIONARY formats of the DATAVIEW Statement, CA-MetaCOBOL+ generates a request area, workarea, and element list. CA-MetaCOBOL+ uses the DATAVIEW statements to generate request areas and element lists with VALUE clauses in the Working-Storage Section. If the DATAVIEW Statement is coded in the Linkage Section, these areas are generated without VALUE clauses and are initialized prior to execution of a statement.

The DATAVIEW Statement has three formats. Each format corresponds to the type of dataview and type of access used.

Format 1: Selects a CA-DATADICTIONARY dataview for set-at-a-time processing. You identify the CA-DATADICTIONARY name of the dataview the program requires. This format requires CA-DATACOM/DB 7.4 or higher.

Format 2: Selects a CA-DATADICTIONARY dataview for record-at-a-time processing. You identify the CA-DATADICTIONARY dataview and the method used to search the tables. This format requires CA-DATACOM/DB 7.4 or higher.

Format 3: Selects a CA-DATADICTIONARY dataview for physical sequential processing. You specify the CA-DATADICTIONARY dataview used to sequentially read a data area in its physical sequence from block one through the end. This format requires CA-DATACOM/DB 7.4 or higher.

The clauses within the formats for CA-DATADICTIONARY dataviews may be entered in any order.

4.1 DATAVIEW Format 1 - Set-at-a-time

This format of the DATAVIEW Statement specifies the CA-DATADICTIONARY dataview used by the FOR Statement. Various data-areas are generated for the set-at-a-time dataview.

```
DATAVIEW dataview-name
   [DATADICTIONARY NAME IS dd-dataview-name]

   [TEST-VERSION IS {LAST          } ]
   [                {version-number} ]

   [PREFIX IS 'data-name-prefix-literal']
   [DATA-BASE-IDENTIFICATION [IS data-name]].
```

DATAVIEW
is the keyword that begins the DATAVIEW Statement.

dataview-name
is the 1-30 character valid COBOL name of the dataview. The first 15 characters of the dataview-name must be unique within a program.

DATADICTIONARY
begins the DATADICTIONARY clause. It can be abbreviated as DD or DICTIONARY.

dd-dataview-name
is a valid CA-DATADICTIONARY dataview entity-occurrence name.

TEST VERSION
specifies the dataview's version as TEST. When this clause is not used, the dataview defaults to PROD status.

LAST
is the last test version, i.e., the version with the highest number. This is the default. LAST can also be specified as LATEST.

version-number
is a 1-3 digit numeric identifying the TEST-VERSION.

PREFIX
is a data-name prefix to be used in the generation of uniquely named workareas for the dataview.

data-name-prefix-literal
The 1-5 character alphanumeric literal prefix. The literal value must be a valid COBOL data-name, with the exception that the last character of the prefix may be a hyphen.

DATA-BASE-IDENTIFICATION
supplies the database identification (DBID). It can also be abbreviated as DBID and DATA-BASE-ID.

When only the keyword DATA-BASE-IDENTIFICATION is coded, the DBID is retrieved from CA-DATADICTIONARY.

data-name
is an alphanumeric COBOL data-name identifying the desired database. When *data-name* is coded, it must contain a valid DBID at run time. (CA-MetaCOBOL+ generates the MOVE statement to move data-name to the request area.) The DBID is required if SYNONYM=YES is specified in the User Requirements Table.

Notes: If the DATADICTIONARY NAME IS clause is omitted, *dataview-name* must be the CA-DATADICTIONARY dataview name.

The *dataview-name* can be the *dd-dataview-name* or a programmer-chosen name. The advantages of using a name chosen by the programmer are:

1. It can be more meaningful in the context of the program.

2. Multiple DATAVIEW statements that actually reference the same CA-DATADICTIONARY dataview can be used.

   Therefore code *dataview-name* and *dd-dataview-name* if *dd-dataview-name* is insufficient or if more than one DATAVIEW Statement is required for a single CA-DATADICTIONARY dataview.

   If the first 15 characters of two *dd-dataview-names* are identical, select unique *dataview-names* and code the DATADICTIONARY NAME IS clause. This situation may occur when accessing CA-DATACOM/DB 7.5 or later.

Example: In the following example, ENROLLMENT is the dataview name used in the COBOL program. ENR-1 is the actual name of the dataview in the CA-DATADICTIONARY.

```
DATAVIEW ENROLLMENT
    DICTIONARY NAME IS ENR-1
    TEST-VERSION IS LAST
    DATA-BASE-IDENTIFICATION IS DBID-AREA.
```

4.2 DATAVIEW Format 2 - Record-at-a-time

The record-at-a-time DATAVIEW Statement is required for CA-MetaCOBOL+ statements that access a database using CA-DATACOM/DB record-at-a-time processing. It generates a request area, workarea, and element list, which are used in the access.

```
DATAVIEW dataview-name
   [DATADICTIONARY NAME IS dd-dataview-name]

   [TEST-VERSION IS {LAST          }]
   [                {version-number}]

   [PREFIX IS 'data-name-prefix-literal']

    ACCESS IS [GENERIC] {KEY IS  }
                        {KEYS ARE} dd-key-name [dd-key-name]...

   [DATA-BASE-IDENTIFICATION [IS data-name]].
```

DATAVIEW
is the keyword that begins the DATAVIEW Statement.

dataview-name
is the 1-30 character valid COBOL name of the dataview. The first 15 characters of the dataview-name must be unique within a program.

DATADICTIONARY
begins the DATADICTIONARY clause. It can also be abbreviated as DD or DICTIONARY.

dd-dataview-name
is a valid CA-DATADICTIONARY dataview entity-occurrence name.

TEST VERSION
specifies the dataview's version as TEST. When this clause is not used, the dataview defaults to PROD status.

LAST
is the last test version, i.e., the version with the highest number. This is the default. LAST can also be specified as LATEST.

version-number
is a 1-3 digit numeric identifying the TEST-VERSION.

PREFIX
is a data-name prefix to be used in the generation of uniquely named workareas for the dataview.

data-name-prefix-literal

is a data-name prefix to be used in the generation of uniquely named workareas for the dataview. The 1-5 character alphanumeric literal prefix. The literal value must be a valid COBOL data-name, with the exception that the last character of the prefix may be a hyphen.

ACCESS

specifies whether a specific table or multiple (generic) tables are searched. ACCESS IS GENERIC KEY IS *dd-key-name* means that access is by key without restriction to table. This form is used to establish position in the database index.

ACCESS KEY IS *dd-key-name* means that a specific table is searched.

DATA-BASE-IDENTIFICATION

supplies the database identification (DBID). It can also be specified as DBID and DATA-BASE-ID.

When only the keyword DATA-BASE-IDENTIFICATION is coded, the DBID is retrieved from CA-DATADICTIONARY. The DBID is required if SYNONYM=YES is specified in the User Requirements Table. It is also required for CICS programs.

data-name

is an alphanumeric COBOL data-name identifying the desired database. When *data-name* is coded, it must contain a valid DBID at run time. (CA-MetaCOBOL+ generates the MOVE statement to move data-name to the request area.)

Notes: The FOR Statement cannot specify a dataview with a record-at-a-time format.

The record-at-a-time DATAVIEW Statement can specify whether a single table (specific search) or multiple tables (generic search) are searched.

For a generic search, the presence or absence of the DBID clause affects which tables are searched. If the DBID clause is not specified, all table names for the first DBID specified in the URT are searched. If the DBID clause is specified, only the table names listed in the URT for the specified DBID are searched.

Examples: In the following example, STUDENT is the dataview name used in the COBOL program. STD-1 is the actual name of the dataview in the CA-DATADICTIONARY. Multiple tables are searched, and the search key is STDID.

```
DATAVIEW STUDENT
    DICTIONARY NAME IS STD-1
    ACCESS IS GENERIC KEY IS STDID.
```

In the following example, COURSE is the dataview name used in the COBOL program. CRS-1 is the actual name of the dataview in the CA-DATADICTIONARY. A specific table is searched, and the search key is COURSE-KEY.

```
DATAVIEW COURSE
    DICTIONARY NAME IS CRS-1
    ACCESS KEY IS COURSE-KEY.
```

4.3 DATAVIEW Format 3 - Physical Sequential

The physical sequential format is used with the LOCATE PHYSICAL and READ/OBTAIN PHYSICAL statements to access data by physical block reads. A key definition is not required for physical sequential access.

Format:

```
DATAVIEW dataview-name
   [DATADICTIONARY NAME IS dd-dataview-name]

   [TEST-VERSION IS {LAST          } ]
   [                {version-number} ]

   [PREFIX IS 'data-name-prefix-literal']
    ACCESS IS PHYSICAL
   [DATA-BASE-IDENTIFICATION [IS data-name]].
```

DATAVIEW
is the keyword that begins the DATAVIEW Statement.

dataview-name
is the 1-30 character valid COBOL name of the dataview. The first 15 characters of the dataview-name must be unique within a program.

DATADICTIONARY
begins the DATADICTIONARY clause. It can also be specified as DD or DICTIONARY.

dd-dataview-name
is a valid CA-DATADICTIONARY dataview entity-occurrence name.

TEST VERSION
specifies the dataview's version as TEST. When this clause is not used, the dataview defaults to PROD status.

LAST
is the last test version, i.e., the version with the highest number. This is the default. LAST can also be specified as LATEST.

version-number
is a 1-3 digit numeric identifying the TEST-VERSION.

PREFIX
is a data-name prefix to be used in the generation of uniquely named workareas for the dataview.

data-name-prefix-literal

is a data-name prefix to be used in the generation of uniquely named workareas for the dataview. The 1-5 character alphanumeric literal prefix. The literal value must be a valid COBOL data-name, with the exception that the last character of the prefix may be a hyphen.

ACCESS IS PHYSICAL

means that the dataview is used with the LOCATE PHYSICAL and READ PHYSICAL statements. A key definition is not required for physical access.

DATA-BASE-IDENTIFICATION

supplies the database identification (DBID). It can also be specified as DBID and DATA-BASE-ID.

When only the keyword DATA-BASE-IDENTIFICATION is coded, the DBID is retrieved from CA-DATADICTIONARY.

data-name

is an alphanumeric COBOL data-name identifying the desired database. When *data-name* is coded, it must contain a valid DBID at run time. (CA-MetaCOBOL+ generates the MOVE statement to move data-name to the request area.)

Notes: The physical sequential format cannot be referenced with the FOR Statement.

Refer to the LOCATE PHYSICAL and READ PHYSICAL statements for more information on physical sequential processing.

Example: In the following example the CA-DATADICTIONARY dataview STD-1 is designated for physical access in the application program. STUDENT is the dataview name coded in the application program.

```
DATAVIEW STUDENT
    DICTIONARY NAME IS STD-1
    ACCESS IS PHYSICAL.
```

5. Procedure Division

5.1 ABEND

The ABEND statement terminates the program by causing a user abend with a user-specified return code.

Format:

```
ABEND literal {DUMP  }
              {NODUMP}
```

literal
is a 5-character literal or 1-to-4-digit numeric literal that identifies the ABEND return code.

DUMP
produces a dump when the abend is generated.

NODUMP
does not produce a dump. This is the default.

Example:

```
PROCEDURE DIVISION.
      . . .
     ABEND 'U4095'.
     ABEND 4095 DUMP.
     ABEND 'ABND1' NODUMP.
```

5.2 CLOSE

The CLOSE statement is issued after processing is completed for the user requirements table. As with the OPEN statement, CLOSE is required when processing with a URT specifying OPEN=USER.

Format:

```
CLOSE
```

Example:

```
PROCEDURE DIVISION.
    ENTER-DATACOM-DB.
    OPEN.
    FOR EACH STUDENT RECORD
      WHERE (STD1-STUDENT-SEX EQ 'M')
      COUNT IN TOTAL-MALES
        DISPLAY STUDENT-MAJOR
      WHEN END
        DISPLAY 'NUMBER OF MALE STUDENTS =>',
TOTAL-MALES.
    CLOSE.
    GOBACK.
```

5.3 DELETE

The DELETE statement deletes a record and and its corresponding index key values.

Format:

```
DELETE dataview-name
```

dataview-name
is a valid dataview defined in the Data Division.

Prerequisite Statements:

FOR (with the HOLD clause)
READ AND HOLD
READ AND HOLD PREVIOUS
READ NEXT AND HOLD WITHIN RANGE
READ NEXT DUPLICATE AND HOLD
READ PHYSICAL
READ SEQUENTIAL

Note: Before a record can be deleted, the record must be read with exclusive control, and the table from which the record is deleted must be defined in the User Requirements Table as a table which may be updated.

A DELETE may fail even when a record has been successfully held under exclusive control. This is particularly true when the DELETE is issued from a CICS program. If a DELETE fails, error processing may destroy the prerequisites on which a subsequent FOR, LOCATE NEXT, or READ NEXT statement depends. To prevent an error from destroying the prerequisites, save the request area before the DELETE and restore it afterwards.

Example:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
    . . .
DATAVIEW ENROLLMENT
    DICTIONARY NAME IS DD-ENR-1
    ACCESS KEY IS STDYC.
    . . .
PROCEDURE DIVISION.
    READ AND HOLD ENROLLMENT
       WHERE STDYC = '1214552222MATH101'
    DELETE ENROLLMENT
```

5.4 ENTER-DATACOM-DB

Use the ENTER-DATACOM-DB statement for batch processing only. In CICS, User Requirements Tables are opened by the CICS Service Facility.

Format:

```
ENTER-DATACOM-DB
```

Notes: When an application program is a subroutine of CA-DATACOM/DB at run time, the ENTER-DATACOM-DB statement is required. This statement provides the entry point ('DBMSCBL') that the CA-DATACOM/DB URT uses to transfer control to a the program. ENTRY BEGIN must also be specified in the link-edit step for programs that are subprograms to the URT.

If an application program is the main program and the CA-DATACOM/DB URT functions as a subprogram at run time, OPEN=USER must be specified in the URT. In addition, the URT(s) must be opened and closed by the program (see the OPEN and CLOSE statements.)

Example:

```
PROCEDURE DIVISION.
         ...
       ENTER-DATACOM-DB.
```

5.5 FOR Statements

The FOR Statement allows data selection based on multiple criteria and can order the set of records before returning them to the application program. After the set has been processed, processing may continue with the WHEN END, WHEN ERROR, and WHEN NONE clauses.

There are two formats:

- Format 1 is for programs using conventional COBOL.
- Format 2 is for programs using CA-MetaCOBOL+'s Structured Programming Facility.

Notes: FOR statements are only valid for set-at-a-time processing.

When FOR Statement processing terminates, the resources required to build the set of records matching the selection criteria are released, and exclusive control for the last set is released, if it is in effect.

The FOR Statement can be nested in a CA-MetaCOBOL+ Structured Programming Facility program. It cannot be nested in a conventional COBOL program.

If the FOR Statement is used with conventional COBOL and a GO TO statement is executed subordinate to a WHERE, WHEN END, WHEN ERROR, or WHEN NONE clause, the programmer must use the FREE SET statement to release resources.

5.5.1 FOR Statement Format 1 - Conventional COBOL

```
FOR {EACH  {                        } }
    {FIRST {record-count-identifier} }
    {ANY   {record-count-literal   } }
    {      {                        } }

          dataview-name {RECORD }
                        {RECORDS}

    [WHERE (condition)]

    [HOLD {RECORD } ]
    [     {RECORDS} ]
    [               ]

    [COUNT IN retrieval-count-identifier]

    [ORDER [UNIQUE] RECORDS ON                ]
    [ { [           ]                 }       ]
    [ { [ASCENDING ]                  }       ]
    [ { [DESCENDING] dataview-identifier } ... ]


        [imperative-statement-1] ...

    [WHEN END                      ]
    [    imperative-statement-2 ...]

    [WHEN ERROR                    ]
    [    imperative-statement-3 ...]

    [WHEN NONE                     ]
    [    imperative-statement-4 ...]    .
```

Note: A period is required to terminate the FOR Statement.

EACH

means that every record matching the selection criteria is to be retrieved for processing.

FIRST
means that the first *n* records from the ordered set matching the selection criteria are to be retrieved.

If the ORDER Clause is omitted, the FIRST or ANY clause selects the same set of records. If the ORDER BY clause is coded with FOR FIRST *n* RECORDS, all records matching the selection criteria are identified, sorted, and then up to *n* are retrieved.

ANY
means that any *n* records matching the selection criteria are to be retrieved.

If the ORDER Clause is omitted, the FIRST or ANY clause selects the same set of records. When FOR ANY *n* RECORDS is coded, *n* records matching the selection criteria are identified, sorted, and returned for processing.

record-count-identifier
For the FIRST and ANY clauses, the record counter can be a data item, which may not be defined within the dataview referenced by the FOR Statement. Do not include a count identifier when the EACH clause is specified.

record-count-literal
For the FIRST and ANY clauses, the record counter can be a numeric literal specified as a non-zero positive integer. Do not include a count literal when the EACH clause is specified.

dataview-name
specifies the CA-DATADICTIONARY dataview (from Format 1 of the DATAVIEW Statement).

WHERE
specifies the selection criteria a record must satisfy in order to be retrieved by the FOR Statement.

(condition)
is a COBOL complex condition enclosed in parentheses. CLASS, SIGN, and CONDITION-NAME conditions are excluded. AND and OR may be used to relate simple conditions (subject, relation, object); NOT can be used to negate a condition or relation. See the **Note** below for more information on the WHERE condition.

HOLD RECORD(S)
assigns exclusive control. Code this clause if records are to be updated or deleted.

COUNT IN
returns a count of the number of records satisfying the selection criteria.

retrieval-count-identifier
is a numeric data-item in which the record count is placed.

ORDER...RECORDS ON
specifies the sequence in which records are retrieved.

UNIQUE
means that only the first record of a group of duplicate records is to be retrieved.

ASCENDING
means the upward ordering of retrieved records. This ordering can be repeated for multiple fields and is the default.

DESCENDING
means the downward ordering of retrieved records. This ordering can be repeated for multiple fields.

dataview-identifier
is a data-item defined in the dataview workarea.

imperative-statement-1
is a statement executed for each record returned without error.

WHEN END
precedes the statements to be executed at the end condition.

imperative-statement-2
is executed when all records have been retrieved, after which the FOR Statement releases temporary indexes and exclusive control and terminates.

WHEN ERROR
precedes the procedural statements to be executed when there is an error condition. To determine an error condition, check the DB-STATUS-CODE fields (DB-CBS-ERROR-CODE) for a CBS return code. Refer to the **Note** below for more information on FOR Statement error processing.

imperative-statement-3
is executed when an error condition has been detected, regardless of whether any records have been retrieved. The FOR Statement then releases temporary indexes and exclusive control and terminates.

FOR Statement processing can resume if the error indication is cleared by setting the generated field *dataview-name***-**ERROR-CODE field equal to spaces. Processing then resumes with the next record, if one exists.

WHEN NONE
precedes the statements to be executed when no records satisfy the selection criteria.

imperative-statement-4
is executed when there are no records to retrieve, after which the FOR Statement releases temporary indexes and exclusive control and terminates.

. (period)
A period (**.**) terminates the FOR Statement. See the **Notes** below for more information on terminating the FOR Statement.

Note: With conventional COBOL, to nest FOR statements, use the PERFORM...THRU statement.

If the FOR Statement is used with conventional COBOL, and a GO TO statement is executed subordinate to a WHERE, WHEN END, WHEN ERROR, or WHEN NONE clause, release resources with the FREE SET statement.

WHERE *condition*:

A *subject* must be the data-name of a field in the dataview workarea; an *object* can be a data-name, literal, figurative constant, or an arithmetic expression. The object must be compatible with its subject.

When an object is a data item within the same dataview, the object must have the same PICTURE and USAGE characteristics as the subject.

When the object is not defined within the dataview referenced by the FOR Statement, compatibility is determined by conforming to the rules for COBOL data movement.

Relation (and Short Forms)

IS EQUAL TO	(IS = TO, EQ)
IS LESS THAN	(IS < THAN, LT)
IS GREATER THAN	(IS > THAN, GT)
IS NOT EQUAL TO	(IS NOT = TO, NE)
IS NOT LESS THAN	(IS NOT < THAN, NL)
IS GREATER THAN OR EQUAL TO	(> OR =, NL, GE)
IS EQUAL TO OR GREATER THAN	(= OR >, >=, =>)
IS NOT GREATER THAN	(IS NOT > THAN, NG)
IS LESS THAN OR EQUAL TO	(< OR =, NG, LE)
IS EQUAL TO OR LESS THAN	(= OR <, <=, =<)

The WHEN clauses are mutually exclusive and optional. The FOR Statement terminates if the terminating condition (END, ERROR, or NONE) occurs.

If WHEN NONE is omitted and there are no records to process, the WHEN END clause is executed.

FOR Statement Error Processing

Use the group data item DB-STATUS-CODE to report FOR Statement errors. DB-STATUS-CODE contains the error information for the most recent database call and is accurate for a given FOR Statement during WHEN END or WHEN ERROR processing only.

DB-STATUS-CODE is initialized as follows:

```
01  DB-STATUS-CODE.
    02 FILLER  PIC X(21)  VALUE 'CBS DATAVIEW STATUS:
'.
    02 DB-DATAVIEW-NAME PIC X(30)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-SL-NUMBER  PIC X(06)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-DL-FOR-STATUS PIC X(02)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-ERROR-CODE  PIC X(02)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-CBS-ERROR-CODE PIC X(03) VALUE SPACE.
```

The DB-STATUS-CODE fields are:

DB-DATAVIEW-NAME
identifies the dataview used by the FOR Statement.

DB-SL-NUMBER
indicates the CA-MetaCOBOL+ sequence number.

DB-DL-FOR-STATUS
contains an error code for the following FOR Statement error conditions. When a FOR Statement nesting error occurs, DB-DL-FOR-STATUS is initialized with NE to indicate a nesting error; this error results when concurrent access of the same dataview by more than one FOR Statement is attempted. When a FOR FIRST/ANY error occurs, DB-DL-FOR-STATUS is set to FA; this error occurs when the value of the *record-count-identifier* has become negative.

DB-ERROR-CODE
contains the CA-DATACOM/DB return code. This field contains 14 when an end-of-file condition exists or when no records were found. When a Compound Boolean Selection (CBS) error occurs, DB-ERROR-CODE is set to 91. DB-CBS-ERROR-CODE contains the CBS error code when DB-ERROR-CODE = 91. Refer to the table in Appendix E for a complete list of CA-DATACOM/DB return codes.

FOR Statement Termination

If you are coding nested FOR statements, certain fields under the group items *dataview-name*-STATUS-DATA and *dataview-name*-REQUEST-AREA--where *dataview-name* is designated by the programmer in the DATAVIEW Statement--are tested during error processing. As a general rule, the fields within these group items should not be modified by your program. Since they are used internally during FOR Statement processing, altering the STATUS-DATA and REQUEST-AREA fields during execution of the FOR Statement causes unpredictable results.

An exception to this rule is when it has been determined that processing may continue after a FOR Statement error occurs. Normally, the FOR Statement terminates when any error occurs. It is possible to continue processing after an error if the field *dataview-name*-ERROR-CODE is reset to spaces.

This approach is a useful response to CBS error 103. The description for this error is:

invalid data in record field; e.g., binary zeros in packed field

When the error is CBS error 103, an error procedure can evaluate or edit the record content and determine if the FOR should terminate or continue. If the FOR processing should continue, *dataview-name*-ERROR-CODE must be cleared to spaces by a statement in the WHEN ERROR clause.

When FOR Statement processing terminates, the resources required to build the set of records matching the selection criteria are released, and exclusive control for the last set retrieved is released, if it is in effect.

Example: Provide a total of all male students and display those with GPA's that are greater than 2.0:

```
77  TOTAL-MALES      PIC 9(05)
.   .   .
PROCEDURE DIVISION.
    ENTER-DATACOM-DB.
    OPEN.
    FOR EACH STUDENT RECORD
      WHERE (STUDENT-SEX EQ 'M')
      COUNT IN TOTAL-MALES
        IF STUDENT-GPA > 2.0
          DISPLAY STUDENT-NAME STUDENT-MAJOR
        ENDIF
      WHEN END
        DISPLAY 'NUMBER OF MALE STUDENTS =>',
TOTAL-MALES.
    CLOSE.
    GOBACK.
```

5.5.2 FOR Statement Format 2 - Structured COBOL

```
FOR {EACH  {                       }  }
    {FIRST {record-count-identifier}  }
    {ANY   {record-count-literal   }  }


     dataview-name {RECORD }
                   {RECORDS}

    [WHERE (condition)]

    [HOLD {RECORD } ]
    [     {RECORDS} ]


    [COUNT IN retrieval-count-identifier]

    [ORDER [UNIQUE] RECORDS ON                ]
    [ { [          ]                     }     ]
    [ { [ASCENDING ]                     }     ]
    [ { [DESCENDING] dataview-identifier } ... ]
    [ { [          ]                     }     ]

        [process-1] ...

    [WHEN END          ]
    [    process-2 ...]


    [WHEN ERROR        ]
    [    process-3 ...]


    [WHEN NONE         ]
    [    process-4 ...]

ENDFOR
```

EACH
means that every record matching the selection criteria is to be retrieved for processing.

FIRST means that the first *n* records from the ordered set matching the selection criteria are to be retrieved.

If the ORDER Clause is omitted, the FIRST or ANY clause selects the same set of records. If the ORDER BY clause is coded with FOR FIRST *n* RECORDS, records matching the selection criteria are identified, sorted, and then up to *n* are retrieved.

ANY
means that any *n* records matching the selection criteria are to be retrieved.

If the ORDER Clause is omitted, the FIRST or ANY clause selects the same set of records. When FOR ANY *n* RECORDS is coded, *n* records matching the selection criteria are identified, sorted, and returned for processing.

record-count-identifier
For the FIRST and ANY clauses, the record counter can be a data item, which may not be defined within the dataview referenced by the FOR Statement. Do not include a count identifier when the EACH clause is specified.

record-count-literal
For the FIRST and ANY clauses, the record counter can be a numeric literal specified as a non-zero positive integer. Do not include a count literal when the EACH clause is specified.

dataview-name
specifies the CA-DATADICTIONARY dataview (from Format 1 of the DATAVIEW Statement).

WHERE
specifies the selection criteria a record must satisfy in order to be retrieved by the FOR Statement.

(condition)
is a COBOL complex condition enclosed in parentheses. CLASS, SIGN, and CONDITION-NAME conditions are excluded. AND and OR may be used to relate simple conditions (subject, relation, object); NOT can be used to negate a condition or relation. See the **Note** below for more information on the WHERE condition.

HOLD RECORD(S)
assigns exclusive control. Code this clause if records are to be updated or deleted.

COUNT IN
returns a count of the number of records satisfying the selection criteria.

retrieval-count-identifier
is a numeric data-item in which the record count is placed.

ORDER...RECORDS ON
specifies the sequence in which records are retrieved.

UNIQUE
: means that only the first record of a group of duplicate records is to be retrieved.

ASCENDING
: means the upward ordering of retrieved records, which can be repeated for multiple fields. This is the default.

DESCENDING
: means the downward ordering of retrieved records, which can be repeated for multiple fields.

dataview-identifier
: is a data-item defined in the dataview workarea.

process-1
: is a statement executed for each record returned without error.

WHEN END
: precedes the statements to be executed at the end condition.

process-2
: is executed when all records have been retrieved, after which the FOR Statement releases temporary indexes and exclusive control and terminates.

WHEN ERROR
: precedes the procedural statements to be executed when there is an error condition. To determine an error condition, check the DB-STATUS-CODE fields (DB-CBS-ERROR-CODE) for a CBS return code. Refer to the **Note** below for more information on FOR Statement error processing.

process-3
: is executed when an error condition has been detected, regardless of whether any records have been retrieved. The FOR Statement then releases temporary indexes and exclusive control and terminates.

  FOR Statement processing can resume if the error indication is cleared by setting the generated field *dataview-name*-ERROR-CODE field equal to spaces. Processing then resumes with the next record, if one exists.

WHEN NONE
: precedes the statements to be executed when no records satisfy the selection criteria.

process-4
: is executed when there are no records to retrieve, after which the FOR Statement releases temporary indexes and exclusive control and terminates.

ENDFOR
: terminates the FOR Statement. See the Note below for more information on terminating the FOR Statement.

Note: WHERE *condition*:

A *subject* must be the data-name of a field in the dataview work area; an *object* can be a data-name, literal, figurative constant, or an arithmetic expression. The object must be compatible with its subject.

When an object is a data item within the same dataview, the object must have the same PICTURE and USAGE characteristics as the subject.

When the object is not defined within the dataview referenced by the FOR Statement, compatibility is determined by conforming to the rules for COBOL data movement:

Relation (and Short Forms)

IS EQUAL TO	(IS = TO, EQ)
IS LESS THAN	(IS < THAN, LT)
IS GREATER THAN	(IS > THAN, GT)
IS NOT EQUAL TO	(IS NOT = TO, NE)
IS NOT LESS THAN	(IS NOT < THAN, NL)
IS GREATER THAN OR EQUAL TO	(> OR =, NL, GE)
IS EQUAL TO OR GREATER THAN	(= OR >, >=, =>)
IS NOT GREATER THAN	(IS NOT > THAN, NG)
IS LESS THAN OR EQUAL TO	(< OR =, NG, LE)
IS EQUAL TO OR LESS THAN	(= OR <, <=, =<)

The WHEN clauses are mutually exclusive and optional. The FOR Statement terminates if the terminating condition (END, ERROR, or NONE) occurs.

If WHEN NONE is omitted and there are no records to process, the WHEN END clause is executed.

FOR Statement Error Processing:

Use the group data item DB-STATUS-CODE to report FOR Statement errors. DB-STATUS-CODE contains the error information for the most recent database call and is accurate for a given FOR Statement during WHEN END or WHEN ERROR processing only.

DB-STATUS-CODE is initialized as follows:

```
01  DB-STATUS-CODE.
    02 FILLER  PIC X(21)  VALUE 'CBS DATAVIEW STATUS:
'.
    02 DB-DATAVIEW-NAME PIC X(30)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-SL-NUMBER  PIC X(06)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-DL-FOR-STATUS PIC X(02)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-ERROR-CODE  PIC X(02)  VALUE SPACE.
    02 FILLER  PIC X(03)  VALUE ' - '.
    02 DB-CBS-ERROR-CODE PIC X(03) VALUE SPACE.
```

The **DB-STATUS-CODE** fields are:

DB-DATAVIEW-NAME
identifies the dataview used by the FOR Statement.

DB-SL-NUMBER
indicates the CA-MetaCOBOL+ sequence number.

DB-DL-FOR-STATUS
contains an error code for the following FOR Statement error conditions. When a FOR Statement nesting error occurs, DB-DL-FOR-STATUS is initialized with NE to indicate a nesting error; this error results when concurrent access of the same dataview by more than one FOR Statement is attempted. When a FOR FIRST/ANY error occurs, DB-DL-FOR-STATUS is set to FA; this error occurs when the value of the record-count-identifier has become negative.

DB-ERROR-CODE
contains the CA-DATACOM/DB return code. This field contains 14 when an end-of-file condition exists or when no records were found. When a Compound Boolean Selection (CBS) error occurs, DB-ERROR-CODE is set to 91. DB-CBS-ERROR-CODE contains the CBS error code when DB-ERROR-CODE = 91. Refer to the table in Appendix E for a complete list of CA-DATACOM/DB return codes.

FOR Statement Termination

If you are coding nested FOR statements, certain fields under the group items *dataview-name*-STATUS-DATA and *dataview-name*-REQUEST-AREA--where *dataview-name* is designated by the programmer in the DATAVIEW Statement--are tested during error processing. As a general rule, the fields within these group items should not be modified by your program. Since they are used internally during FOR Statement processing, altering the STATUS-DATA and REQUEST-AREA fields during execution of the FOR Statement causes unpredictable results.

An exception to this rule is when it has been determined that processing may continue after a FOR Statement error occurs. Normally, the FOR Statement terminates when any error occurs. It is possible to continue processing after an error if the field *dataview-name*-ERROR-CODE is reset to spaces.

This approach is a useful response to CBS error 103. The description for this error is:

invalid data in record field; e.g., binary zeros in packed field

When the error is CBS error 103, an error procedure can evaluate or edit the record content and determine if the FOR should terminate or continue. If the FOR processing should continue, *dataview-name*-ERROR-CODE must be cleared to spaces by a statement in the WHEN ERROR clause.

When FOR Statement processing terminates, the resources required to build the set of records matching the selection criteria are released, and exclusive control for the last set retrieved is released, if it is in effect.

Example: Provide a total of all male students and display those with GPA's that are greater than 2.0:

```
77  TOTAL-MALES      PIC9(05)
.   .   .
PROCEDURE DIVISION.
    ENTER-DATACOM-DB.
    OPEN.
    FOR EACH STUDENT RECORD
      WHERE (STUDENT-SEX EQ 'M')
      COUNT IN TOTAL-MALES
        IF STUDENT-GPA > 2.0
          DISPLAY STUDENT-NAME STUDENT-MAJOR
        ENDIF
      WHEN END
        DISPLAY 'NUMBER OF MALE STUDENTS =>',
TOTAL-MALES
    ENDFOR
    CLOSE.
    GOBACK.
```

5.6 FREE Statements

The FREE statement releases records held with exclusive control. There are three formats:

- FREE ALL
- FREE LAST
- FREE SET

5.6.1 FREE ALL

The FREE ALL statement releases exclusive control of all records held in a table for the specified dataview.

Format:

```
FREE ALL dataview-name
```

dataview-name
A valid dataview defined with a DATAVIEW Statement.

Prerequisite Statements:

READ AND HOLD
READ AND HOLD PREVIOUS
READ [NEXT] AND HOLD WITHIN RANGE
READ NEXT [DUPLICATE] AND HOLD

Note: This statement may not be used with the set-at-a-time dataview.

Example: The dataview ENROLLMENT is released from exclusive control:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
    . . .
DATAVIEW ENROLLMENT
DICTIONARY NAME IS DD-ENR-1
    ACCESS KEYS ARE STDYC STDYR.
    . . .
PROCEDURE DIVISION.
    . . .
    READ AND HOLD ENROLLMENT
    . . .
    FREE ALL ENROLLMENT
```

5.6.2 FREE LAST

The FREE LAST statement releases control of the last data record held for the specified dataview.

Format:

```
FREE LAST dataview-name
```

dataview-name
A valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

READ AND HOLD
READ AND HOLD PREVIOUS
READ [NEXT] AND HOLD WITHIN RANGE
READ NEXT [DUPLICATE] AND HOLD
READ PHYSICAL
READ SEQUENTIAL

Example: The dataview ENROLLMENT is released from exclusive control.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
    . . .
DATAVIEW ENROLLMENT
    DICTIONARY NAME IS DD-ENR-1
    ACCESS KEY IS STDYC.
    . . .

.
.
PROCEDURE DIVISION.
    . . .
    READ AND HOLD ENROLLMENT
    . . .
    FREE LAST ENROLLMENT
```

5.6.3 FREE SET

The FREE SET statement releases the resources acquired for a dataview with set-at-a-time processing. This includes set definition, temporary indices, and release of the last record retrieved, if exclusive control is in effect. It is used when transferring control out of the FOR Statement.

Format:

```
FREE SET dataview-name
```

dataview-name
A valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

FOR

Example: The resources acquired for the STUDENT-INFO dataview are released if one of the student records within the set have the last name of SMITH. Note the EXIT statement.

```
DATAVIEW STUDENT-INFO
    DATADICTIONARY IS DD-STD-1.
    . . .
PROCEDURE DIVISION.
    . . .
    FOR EACH STUDENT-INFO
       WHERE (STD-GPA < 2.0)
       HOLD RECORD
       IF STD-LNAME ='SMITH'
          FREE SET STUDENT-INFO
          EXIT
       ELSE
          DELETE STUDENT-INFO
       ENDIF
    ENDFOR
```

5.7 LOCATE Statements

The LOCATE statement searches a CA-DATACOM/DB index for a key entry that matches a specified key value. It establishes position in the index if a matching key is found. (Position in the index is established whenever a LOCATE or READ statement is successfully executed.) The LOCATE statement cannot be used to reference a set-at-a-time dataview.

The LOCATE statement does not retrieve a record. To retrieve records for the application program, use the READ statement.

The LOCATE statement has the following formats:

- LOCATE AT
- LOCATE NEXT
- LOCATE NEXT WITHIN RANGE
- LOCATE PHYSICAL
- LOCATE PREVIOUS
- LOCATE SEQUENTIAL
- LOCATE WHERE
- LOCATE WITHIN RANGE

5.7.1 Format 1 - LOCATE AT

LOCATE AT positions a dataview at the position of another dataview.

Format:

```
LOCATE dataview-name-1
    AT dataview-name-2
```

dataview-name-1
is a valid dataview defined with the DATAVIEW Statement.

dataview-name-2
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE WHERE
LOCATE [NEXT] WITHIN RANGE
READ [AND HOLD]
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]
READ [NEXT] [AND HOLD] WITHIN RANGE

Note: The request area field *dataview-name*-RA-STATUS-CODE is set to 98 when one of the key-names in the locating dataview is not the same as the located dataview. It is set to 99 when the table names specified in the dataviews are not identical.

Example: A generic index search is performed with the EMPLOYEE dataview to locate the record with a key value that equals the input file number. If the search is successful, the PAYROLL dataview is positioned at the EMPLOYEE record.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS IS GENERIC KEY IS EMPLOYEE-NUMBER.
DATAVIEW PAYROLL
    DATADICTIONARY NAME IS PAY-1
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    LOCATE EMPLOYEE
      WHERE EMPLOYEE-NUMBER EQUAL INPUT-FILE-NUMBER
    IF ENROLLMENT-RA-STATUS-CODE EQUAL SPACES
      LOCATE PAYROLL AT EMPLOYEE
    . . .
```

5.7.2 Format 2 - LOCATE NEXT

LOCATE NEXT locates the next index entry. It is a record-at-a-time statement.

Format:

```
LOCATE [NEXT] [DUPLICATE]
              [KEY      ] dataview-name
```

DUPLICATE
means the next sequential entry containing the same key value is located. DUP is a valid abbreviation.

KEY
means the LOCATE NEXT statement locates the next sequential key-value in the index.

If KEY is omitted, the next sequential entry in the index is returned, even if it is not the same key-value.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE WHERE
READ [AND HOLD]
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]

Note: Depending on the format of the DATAVIEW Statement and the setup of the user requirements table, generic or specific table searches can be implemented. Refer to the record-at-a-time DATAVIEW Statement for information on generic and specific searches.

Examples: The EMPLOYEE record with a key value that equals the input file number is located. The LOCATE NEXT statement then determines whether a record with a duplicate key value exists.

```
DATAVIEW EMPLOYEE
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    LOCATE EMPLOYEE
      WHERE EMPLOYEE-NUMBER EQUAL INPUT-FILE-NUMBER
    IF EMPLOYEE-RA-STATUS-CODE EQUAL SPACES
      LOCATE NEXT DUPLICATE EMPLOYEE
```

5.7.3 Format 3 - LOCATE NEXT WITHIN RANGE

LOCATE NEXT WITHIN RANGE locates the next record key value within the current range that matches or is greater than the current key value. It is a record-at-a-time statement and is valid for CA-DATACOM/DB 7.5 or later.

Format:

```
LOCATE NEXT dataview-name WITHIN RANGE
```

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statement:

LOCATE WITHIN RANGE

Example: The EMPLOYEE record with a key value nearest or matching LOW-NUMBER is located. The records falling within the range of key values are then displayed.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    LOCATE EMPLOYEE WITHIN RANGE
      WHERE EMPLOYEE-NUMBER IS LOW-NUMBER THRU
HIGH-NUMBER
    LOOP
    WHILE ENROLLMENT-RA-STATUS-CODE EQUAL SPACES
      DISPLAY EMPLOYEE
      LOCATE NEXT EMPLOYEE WITHIN RANGE
    ENDLOOP
```

5.7.4 Format 4 - LOCATE PHYSICAL

The LOCATE PHYSICAL statement establishes the table name and element list to set the start of a data area for physical block reads. By using LOCATE PHYSICAL with READ PHYSICAL, a data area can be read in its physical sequence from the first block through the end.

Format:

```
LOCATE PHYSICAL [AND HOLD] dataview-name
```

AND HOLD
is a clause that places exclusive control over the data area.

dataview-name
is a valid dataview defined with the DATAVIEW Statement. The dataview coded in the LOCATE PHYSICAL statement must be defined with Format 3 of the DATAVIEW Statement; the ACCESS IS PHYSICAL clause is required.

Notes: Physical sequential processing is the faster technique for retrieving a large volume of data.

The physical sequential statements reduce index processing by retrieving data via physical block reads. This technique retrieves large amounts of data efficiently (for example, for copying a CA-DATACOM/DB area to a VSAM file.)

Example: The EMPLOYEE dataview is located and read with physical sequential processing.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS IS PHYSICAL.
    . . .
    LOCATE PHYSICAL AND HOLD EMPLOYEE
    READ PHYSICAL EMPLOYEE
```

5.7.5 Format 5 - LOCATE PREVIOUS

LOCATE PREVIOUS establishes position at the record key value preceeding the current key value.

Format:

```
LOCATE PREVIOUS dataview-name
```

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE WHERE
READ [AND HOLD]
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]

Notes: A record key value must be successfully located before issuing LOCATE PREVIOUS. This may be accomplished with a LOCATE statement or a READ statement. When a record key matching or less than the requested value cannot be located, a return code of 14 is returned in the field *dataview-name*-RA-STATUS-CODE.

Example: An online transaction lists employee records, 20 records per screen, by the employee number. To browse through the records, the operator enters **f** or **b** lines to skip forward or backwards, respectively. The following example shows how LOCATE PREVIOUS can be used to scroll back one screen:

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
SCROLL-BACKWARD.
    LOOP 20 TIMES
      LOCATE PREVIOUS EMPLOYEE
      IF EMPLOYEE-RA-STATUS-CODE EQUAL '14'
        LOCATE EMPLOYEE                        Locate first record
          WHERE EMPLOYEE-NUMBER EQUAL '00000'  if there are less
    ENDLOOP                                    than 20 records to
       .                                       the front of the
       .                                       table
       .
DISPLAY-SCREEN.
    READ EMPLOYEE
    LOOP VARYING LINE-COUNT FROM 1 BY 1 THRU 20
      MOVE CORR TOTAL-EMPLOYEE-DATA TO SCREEN (LINE-COUNT)
      READ NEXT EMPLOYEE
    LEAVE WHEN EMPLOYEE-STATUS-CODE EQUAL '14'
    ENDLOOP
```

5.7.6 Format 6 - LOCATE SEQUENTIAL

LOCATE SEQUENTIAL positions a dataview on a key-value in the index greater than or equal to the key-value specified and establishes the key as the starting position for a subsequent READ SEQUENTIAL statement. This statement is valid for batch programs only.

Format:

```
LOCATE SEQUENTIAL dataview-name

          {'db-key-name-literal'}
    WHERE { dd-key-name         }>={'key-value-literal'  }
          { KEY-VALUE           }  { key-value-identifier}
```

SEQUENTIAL
fixes position on a key value in the index that is greater than or equal to the key value specified and establishes the key as the starting position for a succeeding OBTAIN SEQUENTIAL statement. SEQ is a valid abbreviation.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

WHERE
identifies the qualifier for the requested access.

db-key-name-literal
is a 2- to 5-character literal that specifies the CA-DATACOM/DB key name.

dd-key-name
is the valid CA-DATADICTIONARY key name of the dataview to be accessed.

KEY-VALUE
identifies the subject qualifier as the first key specified in the DATAVIEW Statement.

key-value-identifier
is a user-defined data item containing the key value to be located.

key-value-literal
is a user-defined data item containing the key value to be located.

Notes: One of the following may be specified as the relational operator in the condition of the WHERE Clause:

GREATER THAN OR EQUAL
GE
>=

Depending on the setup of the DATAVIEW Statement and the User Requirements Table, generic or specific table searches can be implemented. Refer to the Record-at-a-time DATAVIEW Statement for more information.

The advantage of sequential retrieval is that it is faster than random retrieval. However, native keys should be specified and the database should be reorganized periodically to reduce data segmentation.

Example: The LOCATE/READ SEQUENTIAL statements are used to increase the current rate for each PAYROLL record by ten percent. This example assumes that the parameters UPDATE=YES and AUTODXC=YES (default) are specified in the User Requirements Table.

```
DATAVIEW PAYROLL
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    LOCATE SEQUENTIAL PAYROLL
      WHERE EMPLOYEE-NUMBER >= '00000'  Locate first record
    LOOP                                        in table
      READ SEQUENTIAL PAYROLL
    UNTIL PAYROLL-RA-STATUS-CODE EQUAL '19' End of table for
       MULTIPLY CURRENT-RATE BY .1
                             READ SEQUENTIAL
               GIVING CURRENT-RATE
       UPDATE PAYROLL
    ENDLOOP
```

5.7.7 Format 7 - LOCATE WHERE

This format of the LOCATE statement establishes position at the specified key in the index or at the first key in the index that is greater than, equal to, or less than the specified key-value.

Format:

```
LOCATE dataview-name

           {'db-key-name-literal'}  {<=}  {                     }
     WHERE { dd-key-name          }  {>=}  {'key-value-literal'  }
           { KEY-VALUE            }  {= }  { key-value-identifier}
```

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

WHERE
identifies the qualifier for the requested access.

db-key-name-literal
a 2-5 character literal that specifies a valid CA-DATACOM/DB key name.

dd-key-name
is a valid CA-DATADICTIONARY key name of the dataview to be accessed.

KEY-VALUE
identifies the subject qualifier as the first key specified in the DATAVIEW Statement.

key-value-identifier
is a user-defined data item containing the key value to be located.

key-value-literal
is a user-defined data item containing the key value to be located.

Notes: Specify one of the following relational operators in the condition of the WHERE Clause:

EQUAL
=
GREATER THAN OR EQUAL
GE
>=
=>
LESS THAN OR EQUAL
LE
<=
=<

Depending on the setup of the DATAVIEW Statement and URT, generic or specific table searches can be implemented. Refer to the record-at-a-Time DATAVIEW Statement description for more information.

Example: The EMPLOYEE record with a key value nearest or matching LOW-NUMBER is retrieved. The records falling within the range of key values are then deleted.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    READ AND HOLD EMPLOYEE
      WHERE EMPLOYEE-NUMBER >= LOW-NUMBER
    LOOP
    WHILE ENROLLMENT-RA-STATUS-CODE EQUAL SPACES
      DELETE EMPLOYEE
      READ NEXT EMPLOYEE WITHIN RANGE
    ENDLOOP
```

5.7.8 Format 8 - LOCATE WITHIN RANGE

LOCATE WITHIN RANGE locates a record key value within the specified range that is nearest or matching the specified key value. It is a record-at-a-time statement and is valid for CA-DATACOM/DB 7.5 or later.

Format:

```
LOCATE dataview-name WITHIN RANGE

                {'db-key-name-literal'}
       WHERE    { dd-key-name          } IS { ident1} THRU { ident2}
                { KEY-VALUE            }    { 'lit1'}      { 'lit2'}
```

dataview-name
: is a valid dataview defined with the DATAVIEW Statement.

WITHIN RANGE
: specifies a range of key values. The WHERE Clause specifies the range of desired key values.

WHERE
: identifies the qualifier for the requested access.

db-key-name-literal
: is a 2-5 character literal that specifies a valid CA-DATACOM/DB key name.

dd-key-name
: is a valid CA-DATADICTIONARY key name of the dataview to be accessed.

KEY-VALUE
: identifies the subject qualifier as the first key specified in the DATAVIEW Statement.

key-value-identifier
: is a user-defined data item containing the key value to be located.

key-value-literal
: is a user-defined data item containing the key value to be located.

ident1
: is the beginning value of the range. It must be defined within the program and conform to the format of the key to be compared.

lit1
: is a literal that marks the beginning value of the range. It must be defined within the program and conform to the format of the key to be compared.

ident2
: is the ending value of the range. It must be defined within the program and conform to the format of the key to be compared.

lit2 is a literal that marks the ending value of the range. It must be defined within the program and conform to the format of the key to be compared.

Notes: Depending on the setup of the DATAVIEW Statement and URT, generic or specific table searches can be implemented. Refer to the record-at-a-time DATAVIEW Statement description for more information.

Example: The EMPLOYEE record with a key value nearest or matching LOW-NUMBER is located. The records falling within the range of key values are then displayed.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS KEY IS EMPLOYEE-NUMBER.
    . . .
    LOCATE EMPLOYEE WITHIN RANGE
      WHERE EMPLOYEE-NUMBER IS LOW-NUMBER THRU
HIGH-NUMBER
    LOOP
    WHILE ENROLLMENT-RA-STATUS-CODE EQUAL SPACES
      DISPLAY EMPLOYEE
      LOCATE NEXT EMPLOYEE WITHIN RANGE
    ENDLOOP
```

5.8 LOG Statements

LOG statements control restart and recovery functions that are available to the application program. There are four commands:

- BACKOUT LOG
- CHECKPOINT LOG
- READ LOG
- WRITE LOG

5.8.1 BACKOUT LOG

The BACKOUT LOG statement causes the reversal of all transactions for this task back to the last stable state.

Format:

```
BACKOUT LOG
```

Notes: The CA-DATACOM/DB logging option does not require or depend on any logging statements in your program. Therefore use BACKOUT LOG only when there is a special need to reverse changes made by the program to the database.

BACKOUT LOG may be used only when the logging option has been generated in the Master List and when the URT used by the task is defined for transaction backout. It is otherwise ignored.

Example:

```
PROCEDURE DIVISION.
     . . .
     BACKOUT LOG
```

5.8.2 CHECKPOINT LOG

The CHECKPOINT LOG statement causes a checkpoint record to be written to the log area. The checkpoint record implies the task has completed all work successfully.

Format:

```
CHECKPOINT LOG io-area-identifier
```

io-area-identifier
specifies a programmer-coded I/O area located in the Data Division. This area contains the checkpoint record, eight bytes of data, which is to be written to the log area.

Notes: For CICS programs, the CICS Service Facility checkpoints the task prior to the end of the CICS task. CHECKPOINT LOG is not necessary, but you can issue when there is a special need to implement checkpoint processing.

For batch programs that issue a large number of maintenance commands, a backout can take a long time. Issuing CHECKPOINT LOG can limit the size of the transaction backout.

CHECKPOINT LOG may only be used when the logging option has been generated in the Master List. It is ignored, otherwise.

Example: Below, CHECKPOINT LOG is issued for every 100 records that have been updated.

```
01  LOG-IOAREA.
    05  LOG-ID  PIC X(06)  VALUE 'TESTPGM'.
    05  LOG-NO  PIC 9(03)  VALUE 0.
    . . .
UPDATE-RECORDS.
    READ AND HOLD EMPLOYEE
      WHERE EMPLOYEE-NUMBER EQUAL INPUT-EMP-ID
    IF EMPLOYEE-RA-STATUS-CODE EQUAL SPACES
       ADD 1 TO UPDATE-COUNTER
       IF UPDATE-COUNTER > 100
          ADD 1 TO LOG-NO
          CHECKPOINT LOG LOG-IOAREA
          DISPLAY       '   PROCESSED   ' LOG-IOAREA
          MOVE ZEROS TO UPDATE-COUNTER
       ENDIF
    ENDIF
```

5.8.3 READ LOG

READ LOG retrieves records previously written to the CA-DATACOM/DB log area by the task with WRITE LOG.

Format:

```
READ LOG ioarea-identifier
```

ioarea-identifier

specifies a programmer-coded I/O area located in the Data Division. This area contains the length of the data read from the first four bytes, followed by the data itself.

Prerequisite Statement:

WRITE LOG

Notes: READ LOG may only be used when the logging option has been generated in the Master List. It is ignored if the logging option has not been generated.

The CA-DATACOM/DB logging option does not require or depend on any logging statements in your program. Therefore use READ LOG only when there is a special need to read data written to the log area with the WRITE LOG statement.

Example:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
    . . .
01 EMP-INFO.
    . . .
LOGGING.
   WRITE LOG EMP-INFO
   READ LOG EMP-INFO
   DISPLAY EMP-INFO
```

5.8.4 WRITE LOG

WRITE LOG writes records to the CA-DATACOM/DB log area.

Format:

```
WRITE LOG ioarea-identifier
```

ioarea-identifier
specifies a programmer-coded I/O area located in the Data Division. This area contains the length of the data read from the first four bytes, followed by the data itself.

Notes: WRITE LOG may only be used when the logging option has been generated in the Master List. It is ignored if the logging option has not been generated.

The CA-DATACOM/DB logging option does not require or depend on any logging statements in your application program. Therefore, use WRITE LOG only when there is a special need to write data to the log area.

Example:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
    . . .
01 EMP-INFO.
    . . .
    WRITE LOG EMP-INFO
```

5.9 OPEN

If processing with the extended options entry point or a URT specifying OPEN=USER, the OPEN statement is required. The OPEN statement opens the user requirements table prior to issuing other commands.

The OPEN statement is issued before any other. The CLOSE statement is issued when processing for the URT is complete.

Format:

```
OPEN
```

Example:

```
PROCEDURE DIVISION.
    ENTER-DATACOM-DB
    OPEN
    FOR EACH STUDENT RECORD
      WHERE (STD1-STUDENT-SEX EQ 'M')
      COUNT IN TOTAL-MALES
        DISPLAY STUDENT-MAJOR
      WHEN END
        DISPLAY 'NUMBER OF MALE STUDENTS =>',
TOTAL-MALES
    CLOSE
    GOBACK.
```

5.10 READ Statements

READ is a record-at-a-time statement that retrieves records for the application program. Depending on which READ construct you use, you can retrieve records that are currently located, retrieve a record whose key value has been specified explicitly, or scroll through the database.

Records can be read with exclusive control so that only one task may be deleted or updated. Since requests for exclusive control of a record already held for update must wait until the release of that record, use exclusive control for as short a time as possible. The maximum number of records which can be held under exclusive control at one time is specified in the Master List.

The READ statement has seven formats:

READ [**WHERE**]
: retrieves the currently located record or the record whose key value meets the WHERE condition.

READ NEXT
: retrieves the next record whose key value is greater than or equal to the current key value.

READ NEXT WITHIN RANGE
: retrieves the next record within the current range of key values.

READ PHYSICAL
: retrieves the next physical record.

READ PREVIOUS
: retrieves the next record whose key value is less than the current key value.

READ SEQUENTIAL
: retrieves the next sequential record (fast batch technique for retrieving records sequentially).

READ WITHIN RANGE
: retrieves the record within the specified range of key values.

5.10.1 Format 1 - READ [WHERE]

READ [**WHERE**] retrieves the currently located record or the record satisfying the WHERE condition.

Format:

```
READ [AND HOLD] dataview-name

  [       {'db-key-name-literal'}  {<=}  {'key-value-literal'  } ]
  [ WHERE { dd-key-name         }  {>=}  { key-value-identifier} ]
  [       { KEY-VALUE           }  {= }  {'record-id-literal'  } ]
  [       { ID                  }  {  }  { record-id-identifier} ]
  [       { BLOCK               }  {  }  {'block-id-literal'   } ]
  [       {                     }  {  }  { block-id-identifier } ]
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

AND HOLD
specifies exclusive control of a record.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

WHERE
specifies a condition for record retrieval. The condition identifies a subject that is less than or equal to, equal to, or greater than or equal to an object. For more information, see the Note that follows.

'db-key-name-literal'
is a 2- to 5-character CA-DATACOM/DB key name.

dd-key-name
is a valid key name that has been related to the desired dataview in CA-DATADICTIONARY.

KEY-VALUE
refers to the first key specified in the ACCESS clause of the DATAVIEW Statement.

ID
specifies the record identifier.

BLOCK
specifies the block.

key-value-identifier
is a programmer-defined data item containing the desired key value.

'key-value-literal'
is a programmer-defined data literal containing the desired key value.

record-id-identifier
is a programmer-defined data item containing the desired record-ID value.

'record-id-literal'
is a programmer-defined data literal containing the desired record-ID value.

block-id identifier
is a programmer-defined data item containing the desired block-ID value.

'block-id-literal'
is a programmer-defined data literal containing the desired block-ID value.

Prerequisite Statements:

1. For READ without the WHERE Clause:

   LOCATE NEXT [DUPLICATE] [KEY]
   LOCATE PREVIOUS
   LOCATE [WHERE]
   LOCATE WITHIN RANGE

2. For READ WHERE

   None

Notes: One of the following may be specified as the relational operator in the WHERE condition:

EQUAL
=
GREATER THAN OR EQUAL
GE
>=
=>
LESS THAN OR EQUAL
LE
<=
=<

The GREATER THAN OR EQUAL relational operator is valid for programs accessing CA-DATACOM/DB 7.5 or later.

Both specific and generic table searches can be implemented when the WHERE condition is coded in the READ statement. Refer to the Record-at-a-time DATAVIEW Statement for more information.

Example: In the following example, the STUDENT-GPA of the STUDENT-ENROLLMENT dataview is updated with NEW-STUDENT-GPA of the STUDENT-TRANS-FILE.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. STUDUPDT.
    . . .
ENVIRONMENT DIVISION.
    . . .
DATACOM SECTION.
    ID-AREA IS ID-AREA-X.
DATA DIVISION.
FILE SECTION.
FD  STUDENT-TRANS-FILE
    LABEL RECORDS ARE STANDARD.
01  STUDENT-TRANS-RECORD.
    05  STUDENT-TRANS-NUMBER       PIC X(9).
    05  STUDENT-TRANS-NAME         PIC X(15).
    05  STUDENT-TRANS-GPA          PIC 9V99.
    05  FILLER                     PIC X(53).
WORKING STORAGE SECTION.
    01  ID-AREA-X VALUE 'STUDUPDT' PIC X(32).
DATAVIEW STUDENT-ENROLLMENT
    DATADICTIONARY NAME IS STD-1
    ACCESS KEY IS STUDENT-ID.
PROCEDURE DIVISION.
MAIN-PROCESS.
    ENTER-DATACOM-DB
    START DATA
    77 FLAG MORE-TRANSACTIONS IS TRUE.
    END DATA
    LOOP
        READ STUDENT-TRANS-FILE
            AT END SET-FALSE MORE-TRANSACTIONS
            IF END
               SET-FALSE MORE-TRANSACTIONS
            ENDIF
    WHILE MORE-TRANSACTIONS
        LOCATE STUDENT-ENROLLMENT
            WHERE STUDENT-ID = STUDENT-TRANS-NUMBER
        IF STUDENT-ENROLLM-RA-STATUS-CODE EQUAL SPACES
            PERFORM UPDATE-STUDENT-ENROLLMENT
        ENDIF
    ENDLOOP
    GOBACK
UPDATE-STUDENT-ENROLLMENT.
    READ AND HOLD STUDENT-ENROLLMENT
    MOVE STUDENT-TRANS-GPA TO STUDENT-GPA
    UPDATE STUDENT-ENROLLMENT
```

5.10.2 Format 2 - READ NEXT

READ NEXT reads either the next data record in sequence or the next data record with the same key value (READ NEXT DUPLICATE). Position must be established in the database before this statement can be processed successfully.

Format:

```
READ NEXT [DUPLICATE] [AND HOLD] dataview-name
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

DUPLICATE
is the next data record that has the same key value. DUP is an abbreviation for DUPLICATE.

AND HOLD
specifies exclusive control of a record.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE WHERE
READ [AND HOLD] WHERE
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]
READ [AND HOLD] [NEXT] WITHIN RANGE

Example: The records matching the input employee number are deleted from the table containing records of each employee's dependents.

```
DATAVIEW DEPENDENT
    KEY IS EMPLOYEE-NUMBER.
    . . .
    READ AND HOLD DEPENDENT
      WHERE EMPLOYEE-NUMBER EQUAL INPUT-NUMBER
    LOOP
      DELETE DEPENDENT
    WHILE DEPENDENT-RA-STATUS-CODE EQUAL SPACES
      READ NEXT DUPLICATE AND HOLD DEPENDENT
    ENDLOOP
```

5.10.3 Format 3 - READ NEXT WITHIN RANGE

READ NEXT WITHIN RANGE retrieves the next record with a key value matching or greater than the current key value.

Format:

```
READ [AND HOLD] NEXT dataview-name WITHIN RANGE
```

READ
: is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

AND HOLD
: specifies exclusive control of a record.

dataview-name
: is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE [NEXT] WITHIN RANGE
LOCATE WHERE
READ [AND HOLD] WHERE
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]
READ [AND HOLD] [NEXT] WITHIN RANGE

Note: This statement is valid for CA-DATACOM/DB 7.5 or later.

Example: In the following example, records within a range of student ID's are deleted.

```
DATAVIEW STUDENT-ENROLLMENT
    DATADICTIONARY NAME IS STD-1
    ACCESS KEY IS STUDENT-ID.
         . . .
    LOCATE STUDENT-ENROLLMENT WITHIN RANGE
       WHERE STUDENT-ID IS '666666666' THRU
'777777777'
    LOOP
       READ AND HOLD NEXT STUDENT-ENROLLMENT WITHIN
RANGE
       IF STUDENT-ENROLLM-RA-STATUS-CODE EQUAL SPACES
          DELETE STUDENT-ENROLLMENT
       ELSE
          ESCAPE
       ENDIF
    ENDLOOP
```

5.10.4 Format 4 - READ PHYSICAL

The READ PHYSICAL statement reads data areas by physical block. The dataview to read must be specified with Format 3 of the DATAVIEW Statement and must have been set with the LOCATE PHYSICAL statement.

Format:

```
READ PHYSICAL dataview-name
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE PHYSICAL [AND HOLD]
READ PHYSICAL

Note: LOCATE and READ PHYSICAL represent a faster technique for retrieving a large volume of data. This is because they reduce index processing by retrieving data via physical block reads. Thus this technique retrieves large amounts of data efficiently (for example, for copying a CA-DATACOM/DB area to a VSAM file within a CA-MetaCOBOL+ CA-DATACOM/DB Facility program.)

Example: The EMPLOYEE dataview is located and read with physical sequential processing.

```
DATAVIEW EMPLOYEE
    DATADICTIONARY NAME IS EMP-1
    ACCESS IS PHYSICAL.
    . . .
    LOCATE PHYSICAL AND HOLD EMPLOYEE
    READ PHYSICAL EMPLOYEE
```

5.10.5 Format 5 - READ PREVIOUS

READ PREVIOUS (for CA-DATACOM/DB 7.5 or later) retrieves the record preceding the current key value. Position must be established in the database before this statement can be processed successfully.

Format:

```
READ [AND HOLD] PREVIOUS dataview-name
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

AND HOLD
specifies exclusive control of a record.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statements:

LOCATE AT
LOCATE NEXT [DUPLICATE/KEY]
LOCATE PREVIOUS
LOCATE [NEXT] WITHIN RANGE
LOCATE WHERE
READ [AND HOLD]
READ [AND HOLD] PREVIOUS
READ NEXT [DUPLICATE] [AND HOLD]
READ [AND HOLD] [NEXT] WITHIN RANGE

Notes: In comparison with forward processing statements, READ PREVIOUS uses additional CPU time. Additional I/O is also required during block transition.

READ PREVIOUS also releases control of the previous record read for update if specifications in the User Requirements Table (AUTODXC=YES and UPDATE=YES) require it to do so.

Example: An adjustment is added unilaterally to each employee PAYROLL record. READ PREVIOUS is used to process the PAYROLL records from the last to the first by descending employee number.

```
DATAVIEW PAYROLL
    KEY IS EMPLOYEE-NUMBER.
    . . .
    READ AND HOLD PAYROLL
      WHERE PAYROLL-NUMBER EQUAL '99999'
    LOOP
      ADD ADJUSTMENT TO EMP-YTD-PAY
      UPDATE PAYROLL
      DISPLAY EMP-NAME ' '
              EMP-YTD-PAY
      READ AND HOLD PREVIOUS PAYROLL
    UNTIL PAYROLL-RA-STATUS-CODE EQUAL '14'   Record
not found
    ENDLOOP
```

5.10.6 Format 6 - READ SEQUENTIAL

READ SEQUENTIAL retrieves records from a table sequentially, starting at a key established in a previous LOCATE SEQUENTIAL or READ SEQUENTIAL statement.

Format:

```
    READ SEQUENTIAL dataview-name
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

SEQUENTIAL
means that the next record in a table is to be retrieved. SEQ is an abbreviation for SEQUENTIAL.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

Prerequisite Statement:

LOCATE SEQUENTIAL
READ SEQUENTIAL

Notes: This statement is valid for batch programs only.

If the URT parameter UPDATE=YES, READ SEQUENTIAL obtains exclusive control of a record automatically. If the URT parameter AUTODXC=YES, exclusive control is released automatically when a subsequent UPDATE or DELETE statement is processed.

For fastest sequential retrieval, specify the native key in the ACCESS clause of the DATAVIEW Statement for the dataview accessed by READ SEQUENTIAL.

The first READ SEQUENTIAL for a table must be preceded by a LOCATE SEQUENTIAL.

Example: The LOCATE/READ SEQUENTIAL statements are used to increase the current rate for each PAYROLL record by ten percent. This example assumes that the parameters UPDATE=YES and AUTODXC=YES (default) are specified in the User Requirements Table.

```
DATAVIEW PAYROLL
 ACCESS KEY IS EMPLOYEE-NUMBER.
   . . .
LOCATE SEQUENTIAL PAYROLL
 WHERE EMPLOYEE-NUMBER >= '00000'           Locate first record
LOOP                                         in table
     READ SEQUENTIAL PAYROLL
UNTIL PAYROLL-RA-STATUS-CODE EQUAL '19'      End of table for
   MULTIPLY CURRENT-RATE BY .1               READ SEQUENTIAL
                      GIVING CURRENT-RATE
    UPDATE PAYROLL
   ENDLOOP
```

5.10.7 Format 7 - READ WITHIN RANGE

READ WITHIN RANGE retrieves a record with a key value within the specified range. It is a record-at-a-time statement and is valid for CA-DATACOM/DB 7.5 or later.

Format:

```
READ [AND HOLD] dataview-name WITHIN RANGE

          {'db-key-name-literal'}
    WHERE { dd-key-name          } IS {ident1} THRU {ident2}
          { KEY-VALUE            }    {'lit1'}      {'lit2'}
```

READ
is the keyword that begins the READ statement. OBTAIN is a synonym for READ.

AND HOLD
specifies exclusive control of a record.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

WITHIN RANGE
specifies a range of key values. The WHERE Clause specifies the range of desired key values.

WHERE
identifies the qualifier for the requested access.

db-key-name-literal
is a 2-5 character literal that specifies a valid CA-DATACOM/DB key name.

dd-key-name
is a valid CA-DATADICTIONARY key name of the dataview to be accessed.

KEY-VALUE
identifies the subject qualifier as the first key specified in the **DATAVIEW Statement**.

ident1
is the beginning value of the range. It must be defined within the program and conform to the format of the key to be compared.

lit1
is a literal that marks the beginning value of the range. It must be defined within the program and conform to the format of the key to be compared.

ident2
is the ending value of the range. It must be defined within the program and conform to the format of the key to be compared.

lit2

is a literal that marks the ending value of the range. It must be defined within the program and conform to the format of the key to be compared.

Notes: Depending on the setup of the DATAVIEW Statement and URT, generic or specific table searches can be implemented. Refer to the record-at-a-time DATAVIEW Statement description for more information.

Example: In the following example, all records in the range specified in the WHERE Clause are deleted.

```
DATAVIEW STUDENT-ENROLLMENT
    DATADICTIONARY NAME IS STD-1
    ACCESS KEY IS STUDENT-ID.
    . . .
    LOOP
       READ AND HOLD STUDENT-ENROLLMENT WITHIN RANGE
          WHERE STUDENT-ID IS '666666666' THRU '777777777'
       IF STUDENT-ENROLLM-RA-STATUS-CODE EQUAL SPACES
          DELETE STUDENT-ENROLLMENT
       ELSE
          ESCAPE
       ENDIF
    ENDLOOP
```

5.11 RECONSTRUCT Statement

The Dynamic RQA is built by coding a RECONSTRUCT Statement in the COBOL program, as shown below. This statement is coded in the PROCEDURE DIVISION logically preceding the associated FOR Statement.

The RECONSTRUCT Statement provides the ability to dynamically resolve the Request Qualification Area with information that becomes available only at program execution time.

Format:

```
RECONSTRUCT REQUEST QUALIFICATION AREA

        FOR DATAVIEW dataview-name

        USING QUALIFICATION EXPRESSION FOUND IN data-name
```

dataview-name
is the name of the DATAVIEW associated with the FOR Statement in your CA-MetaCOBOL+ or CA-MetaCOBOL+/PC program.

data-name
is the name of the data area in your COBOL program. The *data-name* contains a Qualification Statement Image (QSI) that specifies the criteria for ordering and selecting records. The QSI may have a WHERE Clause, an ORDER Clause, or both.

Note: A DATAVIEW named on a RECONSTRUCT Statement may have only one FOR Statement associated with it. There is no restriction on the number of RECONSTRUCT Statements naming the same DATAVIEW.

For a detailed explanation and example of how to use the RECONSTRUCT Statement, refer to Appendix F.

5.12 REWRITE

REWRITE modifies data elements in a specified table. REWRITE is a synonym for UPDATE.

Format:

```
REWRITE dataview-name
  FROM alternate-work-area-identifier
```

dataview-name
specifies a CA-DATACOM/DB dataview located in the Data Division.

FROM
specifies an alternate workarea.

alternate-work-area-identifier
is a user-defined name for an alternate CA-DATACOM/DB workarea or record description.

Prerequisite Statements:

FOR (with the HOLD clause)
READ AND HOLD
READ AND HOLD PREVIOUS
READ NEXT AND HOLD WITHIN RANGE
READ NEXT DUPLICATE AND HOLD
READ PHYSICAL
READ SEQUENTIAL

Notes: Before a record can be updated with REWRITE, it must be read with exclusive control.

The master key for a record may only be changed if the CA-DATACOM/DB Directory Master Key is defined to allow updating. If a table is defined to disallow changes in the Master Key value, an attempt to update the Master Key results in a CA-DATACOM/DB return code of 11.

A REWRITE may fail even when a record has been successfully held under exclusive control. This is particularly true when the REWRITE is issued from a CICS program. If a REWRITE fails, error processing may destroy the prerequisites on which a subsequent FOR, LOCATE NEXT, or READ NEXT statement depends. To prevent an error from destroying the prerequisites, save the request area before the REWRITE and restore it afterwards.

Example: The EMPLOYEE records are changed to reflect the new addresses provided from an input file.

```
DATAVIEW EMPLOYEE
    ACCESS KEY IS NUMBER.
    . . .
    READ AND HOLD EMPLOYEE
      WHERE NUMBER EQUAL INPUT-NUMBER
    IF EMPLOYEE-RA-STATUS-CODE EQUAL SPACES
      MOVE INPUT-NEW-ADDRESS TO EMPLOYEE-ADDRESS
      REWRITE ENROLLMENT
    ELSE
      DISPLAY "NO EMPLOYEE RECORD FOUND FOR "
INPUT-RECORD
    ENDIF
    . . .
```

5.13 SKIP

SKIP positions to the nth record within a record set. It is a set-at-a-time statement and is valid for CA-DATACOM 10.0 or later.

Format 1:

```
 SKIP dataview-name {FORWARD  }  <BY> dataname/literal <RECORDS>
                    {BACKWARD }
```

Format 2:

```
          {FIRST}  dataview-name <RECORD>
SKIP TO   {LAST }
          {SAME }
```

SKIP
is the keyword that begins the SKIP statement.

SKIP TO
are the keywords that begin the SKIP TO statement.

dataview-name
is a valid dataview defined with the DATAVIEW statement.

FORWARD - BACKWARD
identifies the direction to SKIP in the format 1.

BY
is optional.

dataname/literal
specifies the number of records to SKIP for format 1.

RECORD(s)
is optional.

FIRST LAST SAME
is the record to SKIP for format 2.

5.14 SET TEST OPTIONS

The SET TEST OPTIONS statement sets the test option switch for diagnostic execution.

Format:

```
SET TEST OPTIONS {=       }  {''hex-literal''     }
                 {EQUAL TO}  {figurative-constant}
```

hex-literal

is a string of two hexadecimal digits (0-F) bounded by pairs of apostrophes. The two digits represent one hexadecimal character. The two-digit representation is converted during translation to the proper one-character binary configuration.

"40"	means that the Master List area, which includes all buffers, should be dumped at the end of each request.
"02"	means that each request to read or write a data item, to expand the buffer, or to gain or release exclusive control should be dumped.
"42"	combines the above requests.

figurative-constant

is one of the COBOL fixed data-names:

ZERO
ZEROS
ZEROES
SPACE
SPACES
HIGH-VALUE
HIGH-VALUES
LOW-VALUE
LOW-VALUES

Example: The Master List area is dumped if a read for exclusive control fails for a reason other than the end-of-file condition.

```
     . . .
     READ AND HOLD EMPLOYEE
       WHERE NUMBER EQUAL INPUT-NUMBER
     IF EMPLOYEE-RA-STATUS-CODE NOT EQUAL SPACES OR
'14'
       SET TEST OPTIONS EQUAL TO ''42''
       READ AND HOLD EMPLOYEE
         WHERE NUMBER EQUAL INPUT-NUMBER
       SET TEST OPTIONS EQUAL TO ZERO
     ENDIF
```

5.15 WRITE

WRITE adds records to a table.

Format:

```
WRITE dataview-name
  FROM alternate-work-area-identifier
```

WRITE
is the keyword that begins the WRITE statement. INSERT is a synonym for WRITE.

dataview-name
is a valid dataview defined with the DATAVIEW Statement.

FROM
specifies an alternate workarea.

alternate-work-area-identifier
is a user-defined name for an alternate CA-DATACOM/DB workarea or record description.

Notes: If the table in which the record is to be added is defined to disallow records with duplicate Master Key values, an attempt to add a record with a Master Key value that already exists is rejected with a CA-DATACOM/DB error code of 10.

Not all elements in a record need be added. CA-DATACOM/DB will fill elements of the record not specified in the request with spaces, so be sure to initialize numeric fields. If specific key values are to be added to the index along with the record, include the key in the elements being added.

If a WRITE fails, error processing may destroy the prerequisites upon which a subsequent statement depends. To prevent an error from destroying the prerequisites, save the request area before the WRITE and restore it afterwards.

Example: A new record is added to the employee table:

```
DATAVIEW EMPLOYEE
    ACCESS KEY IS NUMBER.
    . . .
    WRITE EMPLOYEE
      FROM WS-NEW-EMPLOYEE-RECORD
```

Appendix A. CA-MetaCOBOL+ Statements/ CA-DATACOM/DB Commands;

The first table lists CA-MetaCOBOL+ statements and their associated CA-DATACOM/DB commands. The second table lists CA-DATACOM/DB commands and their associated CA-MetaCOBOL+ statements.

CA-DATACOM/DB Command listed by CA-MetaCOBOL+ statement.

CA-MetaCOBOL+ Statement	**CA-DATACOM/DB Command**
BACKOUT LOG	LOGTB
CHECKPOINT LOG	LOGCP
DELETE	DELET
ENTER-DATACOM-DB	ENTRY 'DBMSCBL'
FOR	SELFR, SELNR, SELPR
FREE ALL	RELFL
FREE ALL SETS	SELPA
FREE LAST	RELES
FREE SET	SELPR
INSERT	ADDIT
LOCATE	LOCKX, LOCKY, LOCKL
LOCATE NEXT	LOCNX
LOCATE NEXT DUPLICATE	LOCNE

CA-MetaCOBOL+ Statement	**CA-DATACOM/DB Command**
LOCATE NEXT KEY	LOCNK
LOCATE NEXT WITHIN RANGE	LOCNR
LOCATE PHYSICAL	GSETP
LOCATE PREVIOUS	LOCBR
LOCATE SEQUENTIAL	GSETL
LOCATE WITHIN RANGE	LOCKR
OBTAIN	REDKY REDKG REDID REDLE REDKL
OBTAIN AND HOLD	RDUKY RDUKG RDUID RDULE RDUKL
OBTAIN NEXT	REDNX
OBTAIN NEXT AND HOLD	RDUNX
OBTAIN NEXT DUPLICATE	REDNE
OBTAIN NEXT DUPLICATE AND HOLD	RDUNE
OBTAIN PHYSICAL	GETPS
OBTAIN SEQUENTIAL	GETIT
READ	RDUKY RDUKG RDUID RDULE RDUKL
READ AND HOLD	RDULE RDUKL
READ AND HOLD PREVIOUS	RDUBR
READ AND HOLD WITHIN RANGE	RDUKR
READ LOG	LOGLB
READ NEXT	REDNX
READ NEXT AND HOLD	RDUNR
READ NEXT AND HOLD	RDUNX
READ NEXT DUPLICATE	REDNE
READ NEXT DUPLICATE AND HOLD	RDUNE

CA-MetaCOBOL+ Statement	**CA-DATACOM/DB Command**
READ NEXT WITHIN RANGE	REDNR
READ PHYSICAL	GETPS
READ PREVIOUS	REDBR
READ SEQUENTIAL	GETIT
READ WITHIN RANGE	REDKR
REWRITE	UPDAT
SET TEST OPTIONS	TEST
UPDATE	UPDAT
WRITE	ADDIT
WRITE LOG	LOGIT

CA-MetaCOBOL+ statement listed by CA-DATACOM/DB command

DATACOM/DB Command	**MetaCOBOL+ Statement**
ADDIT	INSERT, WRITE
DELET	DELETE
ENTRY 'DBMSCBL'	ENTER-DATACOM-DB
GETIT	READ/OBTAIN SEQUENTIAL
GETPS	READ/OBTAIN PHYSICAL
GSETL	LOCATE SEQUENTIAL
GSETP	LOCATE PHYSICAL
LOCBR	LOCATE PREVIOUS
LOCKR	LOCATE WITHIN RANGE
LOCKL, LOCKX, LOCKY	LOCATE
LOCNE	LOCATE NEXT DUPLICATE
LOCNK	LOCATE NEXT KEY
LOCNR	LOCATE NEXT WITHIN RANGE
LOCNX	LOCATE NEXT
LOGCP	CHECKPOINT LOG
LOGIT	WRITE LOG
LOGLB	READ LOG
LOGTB	BACKOUT LOG
RDUBR	READ AND HOLD PREVIOUS
RDUKR	READ AND HOLD WITHIN RANGE
RDUKY RDUKG RDUID RDULE RDUKL	READ/OBTAIN AND HOLD
RDUNE	READ/OBTAIN NEXT DUPLICATE AND HOLD

DATACOM/DB Command	**MetaCOBOL+ Statement**
RDUNR	READ NEXT AND HOLD WITHIN RANGE
RDUNX	READ/OBTAIN NEXT AND HOLD
REDBR	READ PREVIOUS
REDKR	READ WITHIN RANGE
REDKY REDKG REDID REDLE REDKL	READ/OBTAIN
REDNE	READ/OBTAIN NEXT DUPLICATE
REDNR	READ NEXT WITHIN RANGE
REDNX	READ/OBTAIN NEXT
RELES	FREE LAST
RELFL	FREE ALL
SELFR, SELNR, SELPR	FOR
SELPA	FREE ALL SETS
SELPR	FREE SET
TEST	SET TEST OPTIONS
UPDAT	REWRITE/UPDATE

Appendix B. Generated Names

This appendix contains a description of all data-names generated by the CA-MetaCOBOL+ CA-DATACOM/DB Facility. The generated code for these statements is shown following a detailed description of a corresponding input statement.

B.1 Generated Dataview Areas

This section lists the data-names generated by DATAVIEW statements. These generated data areas are used internally by the CA-DATACOM/DB Facility and should not be changed by your program.

B.1.1 Set-at-a-time Dataview

The following are the generated data-names for the set-at-a-time dataview (Format 1).

```
01  dataview-name-GROUP.
    02 dataview-name-STATUS-DATA
      03 dataview-name-STATUS                        PIC X(02).
      03 dataview-name-ERROR-CODE                    PIC X(03).
      03 dataview-name-ERROR-NUMBER REDEFINES
         dataview-name-ERROR-CODE                    PIC 9(03).
      03 dataview-name-DECODE-ERROR                  PIC 9(04) COMP.
      03 dataview-name-ERROR-WA REDEFINES
         dataview-name-DECODE-ERROR.
        04 FILLER                                    PIC X(01).
        04 dataview-name-DECODE-WA                   PIC X(01).
    02 dataview-name-USER-ID                         PIC X(32).
    02 dataview-name-REQUEST-AREA.
      03 dataview-name-RA-FUNCTION              PIC X(05).
      03 dataview-name-RA-FILE                  PIC X(03).
      03 dataview-name-RA-KEY-NAME              PIC X(05).
      03 dataview-name-RA-STATUS-CODE           PIC X(02).
      03 dataview-name-RA-UPDATE                PIC X(01).
      03 dataview-name-RA-CBS-ERROR REDEFINES
         dataview-name-RA-UPDATE                PIC X(01).
      03 dataview-name-RA-DBID                  PIC X(02).
      03 dataview-name-RA-DBID-HW REDEFINES
         dataview-name-RA-DBID                  PIC 9(04) COMP.
      03 dataview-name-RA-RESERVED1             PIC X(22).
      03 dataview-name RA-COUNT                 PIC 9(08) COMP.
      03 dataview-name-RA-RESERVED2             PIC X(32).
      03 dataview-name-RA-KEY-VALUE             PIC X(180).
    02 dataview-name-WORKAREA.
      03 ELEMENT-element-1-name.
        04 element-1-name.
      03 ELEMENT-element-2-name.
        04 element-2-name.
    02 dataview-name-ELEMENT-LIST
      03 dataview-name-EL-element-1-name PIC X(05).
      03 dataview-name-EL-element-2-name PIC X(01).
01  DB-STATUS-CODE.
    02 FILLER               PIC X(21)  VALUE `CBS DATAVIEW
        STATUS:  '.
    02 DB-DATAVIEW-NAME     PIC X(30)  VALUE SPACE.
    02 FILLER               PIC X(03)  VALUE ' - '.
    02 DB-SL-NUMBER         PIC X(06)  VALUE SPACE.
    02 FILLER               PIC X(03)  VALUE ' - '.
    02 DB-DL-FOR-STATUS     PIC X(02)  VALUE SPACE.
    02 FILLER               PIC X(03)  VALUE ' - '.
    02 DB-ERROR-CODE        PIC X(02)  VALUE SPACE.
    O2 FILLER               PIC X(03)  VALUE ' - '.
    02 DB-CBS-ERROR-CODE    PIC X(03)  VALUE SPACE.
01  UIB-USER-INFORMATION-BLOCK.
    02 FILLER               PIC X(04)  VALUE SPACE.
    02 UIB-REGION           PIC X(01)  VALUE SPACE.
    02 UIB-SYSTEM-ID        PIC X(03)  VALUE '$DL'.
    02 FILLER               PIC X(01)  VALUE ' - '.
    02 UIB-PROGRAM-NAME     PIC X(08)  VALUE 'program-name'.
    02 FILLER               PIC X(01)  VALUE '('.
    02 UIB-VERSION          PIC X(03)  VALUE `001'.
    02 FILLER               PIC X(01)  VALUE ')'.
    02 UIB-STATEMENT-NUMBER  PIC X(06)  VALUE SPACE.
    02 UIB-DATE-TIME-STAMP   PIC S9(05) COMP-3 VALUE
'date-time-stamp'.
```

CA-DATADICTIONARY Information Specific to CA-DATACOM/DB Only

Note: The information about CA-DATADICTIONARY in the paragraphs below applies only to CA-DATACOM/DB; it does not apply to CA-DATACOM/PC. CA-DATACOM/PC uses its own data dictionary, not CA-DATADICTIONARY. Therefore, for CA-DATACOM/PC, the data dictionary name is always used.

When CA-DATADICTIONARY builds a COBOL copy book, (which happens automatically during the CA-MetaCOBOL+ processing of the CA-DATACOM/DB Facility), it has two sources of information for generating each COBOL name for elements and fields. First, the *COMPILER-NAME* attribute is checked. If the attribute is non-blank, the attribute value is used as the COBOL name. If the attribute is blank, the CA-DATADICTIONARY entity-occurrence name is used as the COBOL name.

Failure to assign elements and fields a *COMPILER-NAME* value in the CA-DATADICTIONARY can result in a duplicate COBOL data name conflict between the element name (04-level data item name) and a field name (05-level or higher). The conflict arises when the element's CA-DATADICTIONARY entity-occurrence name is the same as a field's CA-DATADICTIONARY entity-occurrence name.

To resolve the conflict, the Database Adminstrator must add a *COMPILER-NAME* value to the element. This allows the field name to remain unchanged.

DATAVIEWs are used by both CA-MetaCOBOL+/PC and CA-IDEAL. However, assigning a *COMPILER-NAME* value is strictly a CA-MetaCOBOL+ consideration. CA-IDEAL programs do not use the *COMPILER-NAME* values when processing a DATAVIEW.

B.1.2 Record-at-a-time Dataview

The following data names are generated for a record-at-a-time DATAVIEW Statement. If the DATAVIEW Statement is coded in the Linkage Section, these areas are generated without VALUE clauses and therefore are not initialized.

```
01  dataview-name-GROUP.
    02 dataview-name-REQUEST-AREA.
       03 dataview-name-RA-FUNCTION            PIC X(05).
       03 dataview-name-RA-FILE                PIC X(03).
       03 dataview-name-RA-KEY-NAME            PIC X(05).
       03 dataview-name-RA-STATUS-CODE         PIC X(02).
       03 dataview-name-RA-UPDATE              PIC X(01).
       03 dataview-name-RA-DBID                PIC X(02).
       03 dataview-name-RA-RECORD-ID           PIC X(07).
       03 dataview-name-RA-RESERVED            PIC X(51).
       03 dataview-name-RA-KEY-VALUE.
         04 dataview-name-RA-LONGEST-KEY    PIC X(length of longest
key).
         04 FILLER                          PIC X(180-length of longest
key).
       03 dataview-name-dbkeyname-RA-L-KEY REDEFINES
          dataview-name-RA-KEY-VALUE.
         04 dataview-name-dbkeyname-RA-KEY    PIC X(length of
dbkeyname)
           USAGE DISPLAY.
         04 FILLER                            PIC X(180-length of
longest key).
       03 dataview-name-RA-EOR-VALUE          PIC X(180).
       03 dataview-name-dbkeyname-RA-H-KEY REDEFINES
          dataview-name-RA-EOR-VALUE.
         04 dataview-name-RA-EOR              PIC X(length of
dbkeyname)
           USAGE DISPLAY.
         04 FILLER                            PIC X(180-length of
longest key).
    02 dataview-name-WORKAREA.
       03 ELEMENT-element-1-name.
         04  element-1-name.
                . . .
    02 dataview-name-ELEMENT-LIST.
       03  dataview-name-EL-element-name-n     PIC  X(05).
       03  dataview-name-EL-SEC-element-name-n PIC  X(01).
                . . .
       03  dataview-name-EL-END                PIC  X(05) VALUE
SPACE.
```

B.1.3 Physical Sequential Dataview

The request area for the physical sequential dataview (Format 3) is identical to that of the record-at-a-time dataview. No key-value area is generated for the physical sequential dataview.

B.2 Data-Name Descriptions

B.2.1 Dataview Data-Names

dataview-name-dbkeyname-**RA-L-KEY** PIC X(*length of dbkeyname*)

A group data item redefining the beginning value of a specified range of key values.

dataview-name-dbkeyname-**RA-H-KEY** PIC X(*length of dbkeyname*)

A group data item redefining the end value of a specified range of key values.

dataview-name-dbkeyname-**RA-KEY** PIC X(*length of dbkeyname*)

A data item containing the beginning value of a specified range of key values.

dataview-name-**DECODE-ERROR** PIC 9(04)

COMP
A data item containing the DB or CBS error code.

dataview-name-**DECODE-WA** PIC X(01)

A data item defining an area containing the DB or CBS error code.

dataview-name-**ERROR-CODE** PIC X(03)

A data item containing the DB or CBS error code. If the value is **91**, the numeric code for the CBS error is found in *dataview-name*-RA-CBS-ERROR. If the error is a DB error or "end of file" condition, see the *dataview-name*-RA-STATUS-CODE.

dataview-name-**ERROR-NUMBER** PIC 9(03)

A data item containing the sequence or line number of the error from the CA-DATACOM/DB Facility input listing.

dataview-name-**ERROR-WA** PIC X(01)

A data item defining an area containing the sequence or line number of the error from the CA-DATACOM/DB Facility input listing.

dataview-name-**EL-***element-name-n* PIC X(05)

The data-name of the element to be retrieved or processed by the database system.

dataview-name-**ELEMENT-***ddict-name*

A group item heading the TOTAL-*dataview-name*-DATA, which contains fields that make up the dataview.

dataview-name-**ELEMENT-LIST**

A data item containing the actual data elements to be retrieved from the database. Can be used for record-at-a-time processing.

dataview-name-**GROUP**

A group data item containing the DATAVIEW request area and element list. Can be used for record-at-a-time processing.

dataview-name-**RA-CBS-ERROR** PIC X(01)

A data item containing the CBS error code for the request area when the DB error code is 91.

dataview-name-**RA-COUNT** PIC 9(08)

COMP
A data item containing the number of records matching the selection criteria.

dataview-name-**RA-DBID** PIC X(02)

A data item containing the binary database literal specified in the **DATA-BASE-IDENTIFICATION** clause. Can be used for record-at-a-time processing.

dataview-name-**RA-DBID-HW** PIC 9(04)

COMP
A data item defining the area containing the DATA-BASE-IDENTIFICATION clause.

dataview-name-**RA-EOR** PIC X(*length of longest key)*

A data item defining the end of the specified range of key values.

dataview-name-dbkeyname-**RA-EOR-VALUE** PIC X(180)

A group data item containing the value of the end of the specified range of key values.

dataview-name-**RA-FILE** PIC X(03)

A data item containing the name of the table to be accessed. Can be used for record-at-a-time processing.

dataview-name-**RA-FUNCTION** PIC X(05)

A data item containing the function code parameter for **CALL**s to DATACOM/DB. Can be used for record-at-a-time processing.

dataview-name-**RA-KEY-NAME** PIC X(05)

The data-name of the key as defined in the database control file. Can be used for record-at-a-time processing.

dataview-name-**RA-KEY-VALUE** PIC X(180)

A data item containing the value of the CA-DATACOM/DB key to be searched.

dataview-name-dbkeyname-**RA-LONGEST KEY** PIC X(*length of longest key*)

A group data item containing the value of the longest CA-DATACOM/DB key to be searched.

*dataview-name***-RA-RECORD-ID**	PIC X(06)

A data item containing the physical record ID of the data record located. Can be used for record-at-a-time processing.

*dataview-name***-RA-RESERVED**	PIC X(51)

A data item containing CA-DATACOM/DB system information. It must not be modified. Can be used for record-at-a-time processing.

*dataview-name***-RA-RESERVED1**	PIC X(22)

A data item containing CA-DATACOM/DB system information. It must not be modified.

*dataview-name***-RA-RESERVED2**	PIC X(32)

A data item containing CA-DATACOM/DB system information. It must not be modified.

*dataview-name***-RA-STATUS-CODE**	PIC X(02)

A data item containing the status of a request when there is a DB error or "end of file" condition. Can be used for record-at-a-time processing.

*dataview-name***-RA-UPDATE**	PIC X(01)

A data item containing the exclusive control status. Can be used for record-at-a-time processing.

*dataview-name***-REQUEST-AREA**

A data-name defining an area used to interpret requests and to access the database. Can be used for record-at-a-time processing.

*dataview-name***-STATUS**	PIC X(02)

A data item defining a FOR nesting error (NE) or a FIRST/ANY specification of a non-positive integer (FA).

*dataview-name***-STATUS-DATA**

A group data item containing the dataview status area for decoding CBS error codes. (See Section B.1.1, "Set-at-a-time Dataview," for an illustration of the actual generated code.)

*dataview-name***-USER-ID** PIC X(32)

A data item containing the user identification area.

*dataview-name***-WORKAREA**

A group data item defining a user-supplied input/output area. Can be used for record-at-a-time processing.

B.2.2 Other Generated Data-Names

DATACOM/DB and CBS Error Codes

See Section B.1.1, "Set-at-a-time Dataview," for an illustration and explanation of the generated code.

DB-STATUS-CODE
A group data item containing CA-DATACOM/DB and/or CBS error codes. This area is generated once in the Working Storage Section.

DB-DATAVIEW-NAME PIC X(30)
A data item that contains the name of the dataview.

DB-SL-NUMBER PIC X(06)
A data item that contains the sequence or line number of the WHEN ERROR clause error from the CA-DATACOM/DB Facility input listing.

DB-DL-FOR-STATUS PIC X(02)
A data item that contains either spaces or the CA-DATACOM/DB Facility run-time error code NE when there is a FOR nesting error which can only be discovered at execution, or FA when there is a FIRST/ANY identifier error.

DB-ERROR-CODE PIC X(02)
A data item that contains the DB error code. When there is a CA-DATQCOM/DB nesting error, this data contains spaces.

DB-CBS-ERROR-CODE PIC X(03)
A data item that contains any DB error and the CBS error code when the DB error is 91. Otherwise, the CBS error code is zero. When there is a CA-DATACOM/DB nesting error, this data item contains spaces.

User Information Block

An initialized User Information Block (UIB) is required for a DATACOM/DB access utilizing the CBS interface. The first 5 bytes of the UIB are not used, including UIB-REGION, by the CA-DATACOM/DB Facility.

The UIB serves as a unique identifier at run time. It identifies the updating task and gives the line number of the invoking FOR Statement in the PXX report. If you provide an initialized UIB, the translate-time option **UPSI1=N** will disable the generation of the UIB and its initialization.

UIB-SYSTEM-ID PIC X(03)
A data item that identifies the system to CA-DATACOM/DB.

UIB-PROGRAM-NAME PIC X(8)
A data item that contains the Program-ID.

UIB-VERSION PIC X(03)
A data item that contains the version number of the program.

UIB-STATEMENT-NUMBER PIC X(6)
A data item that contains the statment number of the FOR Statement being executed.

UIB-DATE-TIME-STAMP PIC S9(5) COMP-3
This data item assures a unique UIB. It is intialized to the following at translation: The first byte contains the least significant character of the current day, and the remaining bytes contain the current time.

A data item containing the date and time of the access.

UIB-USER-ID PIC X(3)
A data item that can be used to activate set-at-a-time diagnostics.

UIB-CBS-DIAGNOSTICS REQUESTED
A condition-name indicating that the value '$$$' may be used to request set-at-a-time diagnostics.

FOR Statement Areas

FOR-*cn*-REQUEST-Q-AREA
A group data item containing the RQA for the FOR Statement.

FOR-*cn*-RQA-LENGTH PIC 9(04)
A data item containing the length of the FOR Statement request area.

FOR-*cn*-RQA-COUNT PIC 9(04)
A data item containing the count of request area sections.

Note:

cn	an integer that represents an occurence number of a FOR Statement during translation.
n	an integer that represents the occurrence of the selection criteria or ordered-by sequence keys.

ORDER SECTION

FOR-*cn*-RQA-OS-HEADER	PIC X(12)

A data item containing "ordered by" sequence from the ORDER BY clause.

un-subscripted ordered-by sequence key

FOR-*cn*-RQA-OS-ENTRY-*n*	PIC X(18)

subscripted ordered-by sequence key

FOR-*cn*-RQA-OS-EKH-*n*	PIC X(08)
FOR-*cn*-RQA-OS-EKO-*n*	PIC 9(04)
FOR-*cn*-RQA-OS-ERM-*n*	PIC X(06)

PARAMETER SECTION

FOR-*cn*-RQA-PS-HEADER	PIC X(12)

A data item containing the "parameter section" for the FIRST, ANY, COUNT, and UNIQUE clauses.

COUNT parameter entry

FOR-*cn*-RQA-PS-COUNT	PIC X(13)

FIRST parameter entry

FOR-*cn*-RQA-PS-FIRSTH	PIC X(03)
FOR-*cn*-RQA-PS-FIRSTN	PIC 9(09)
FOR-*cn*-RQA-PS-FIRSTM	PIC X(01)

ANY parameter entry

FOR-*cn*-RQA-PS-ANYH	PIC X(03)
FOR-*cn*-RQA-PS-ANYN	PIC 9(09)
FOR-*cn*-RQA-PS-ANYM	PIC X(01)

UNIQUE parameter entry

FOR-*cn*-RQA-PS-UNIQ	PIC X(01)

SELECTION SECTION

FOR-*cn*-RQA-SC-HEADER	PIC X(12)

A data item containing the "selection criteria" derived from the WHERE Clause.

-Comparison-

data-name versus data-name relation entries

un-subscripted data-names

FOR-*cn*-RQA-SC-E-n	PIC X(35)

subscripted data-names

FOR-*cn*-RQA-SC-ESH-*n*	PIC X(12)
FOR-*cn*-RQA-SC-ESO-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-ESL-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-EOH-*n*	PIC X(07)
FOR-*cn*-RQA-SC-EOO-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-EOL-*n*	PIC 9(04)

subscripted subject

FOR-*cn*-RQA-SC-ESH-*n*	PIC X(12)
FOR-*cn*-RQA-SC-ESO-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-ERM-*n*	PIC X(19)

subscripted object

FOR-*cn*-RQA-SC-EH-*n*	PIC X(27)
FOR-*cn*-RQA-SC-EOO-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-EOL-*n*	PIC 9(04)

-Comparison-

data-name vs literal, non-dataview data-name, or arithmetic expression

un-subscripted subject

FOR-*cn*-RQA-SC-E-n	PIC X(25)

subscripted subject

FOR-*cn*-RQA-SC-ESH-*n*	PIC X(12)
FOR-*cn*-RQA-SC-ESO-*n*	PIC 9(04)
FOR-*cn*-RQA-SC-ER-*n*	PIC X(09)

object value

FOR-*cn*-RQA-SC-E-V-*n*	PIC 'literal'

Program Service Statements

UTILITY-RA-DBID PIC X(01)
A data item containing the binary database literal specified in the DATA-BASE-IDENTIFICATION clause.

UTILITY-RA-FILE PIC X(03)
A data item containing the name of the table to be accessed.

UTILITY-RA-FUNCTION PIC X(5)
A data item containing the function code parameter for CALLs to CA-DATACOM/DB.

UTILITY-RA-KEY-NAME PIC X(05)
The data-name of the key as defined in the database control file.

UTILITY-RA-KEY-VALUE PIC X(360)
A data item required by CA-DATACOM/DB.

UTILITY-RA-RECORD-ID PIC X(07)
A data item containing the physical record ID of the data record located.

UTILITY-RA-RESERVED PIC X(51)
A data item containing CA-DATACOM/DB system information. It must not be modified.

UTILITY-RA-STATUS-CODE PIC X(02)
A data item containing the status of a request to CA-DATACOM/DB.

UTILITY-REQUEST-AREA
A data-name defining an area used to interpret requests and access the database. A UTILITY request is used on functions that do not reference a date field.

Miscellaneous Working Storage

ZZ-DLDB
The group item under which all CA-DATACOM/DB program storage data items are placed.

ZZ-DLDB-ABEND-CODE PIC X(05)
A data item containing the abend code issued with an ABEND command.

ZZ-DLDB-ABEND-DUMP PIC X(06)
A data item containing an indicator (DUMP or NODUMP) for CA-DATACOM to abend with or without providing a dump.

ZZ-DLDB-ABEND-REQUEST
The group item containing the required CA-DATACOM/DB parameter areas used when an ABEND command is specified.

ZZ-DLDB-DBID PIC X(01)
A data item containing the database identification when a DATA-BASE-IDENTIFICATION clause is used in a DATAVIEW being referenced in a database management statement. It is the low-order byte of the COBOL addressable half-word.

ZZ-DLDB-HW-DBID PIC X(04)
A data item containing the database identification when a DATA-BASE-IDENTIFICATION clause is used in a DATAVIEW being referenced in a database management statement. It is the COBOL addressable half-word.

Appendix C. Reserved Words

In addition to the words reserved by COBOL, the following are reserved words in the CA-DATACOM/DB Facility

ABEND
ACCESS
ACCESSED
ALL
AND
ANY
ANY-FILE
ASCENDING
AT
BACKOUT
BLOCK
CHECKPOINT
CICS
CLOSE
COUNT
DATA-BASE-ID
DATA-BASE-IDENTIFICATION
DATA-VIEW
DATAVIEW
DATACOM
DATACOM/DB
DATADICTIONARY
DBID
DD
DELETE
DESCENDING
DICTIONARY
DUMP
DUP
DUPLICATE
EACH
ELEMENT
ELEMENTS
END
ENDFOR
ENTER-DATACOM-DB
EQUAL
ERROR
FILE
FIRST
FOR
FREE
FROM
GEN
GENERIC
HOLD
ID
ID-AREA
INSERT
KEY
KEYS
KEY-VALUE
LAST
LATEST
LOCATE
LOG
LOW-VALUES
MONITOR
NAME
NEXT
NODUMP
NOGEN
NONE
OBTAIN
OPEN
OPTIONS
ORDER
ORGANIZATION

PHYSICAL
PREFIX
PREVIOUS
PRINT
RANGE
READ
RECORD(S)
REWRITE
SECTION
SEQ(UENTIAL)
SELECT
SAME
SET
SPACES
TEST
TEST-VERSION
UNIQUE
UPDATE
USAGE
WHEN
WHERE
WITHIN
WORKAREA
WRITE

Appendix D. Translate-Time Diagnostics

This appendix lists the diagnostics issued by the CA-DATACOM/DB Facility. The diagnostics assume the format:

DLBA*nnnc*

DLBA	is the prefix for diagnostics issued by the CA-DATACOM/DB Facilty.
nnn	is the number of the diagnostic.
c	is the severity code:

- **A** is an advisory message.
- **E** is an error message. The source program requires at least one modification.
- **F** is a fatal error message. Processing ends.
- **W** is a warning message. Some automatic change must be reviewed.

DLBA001E "*identifier-1*" **IS INVALID SYNTAX IN** "*identifier-2*"

Explanation: The displayed *identifier-1* is invalid syntax.

Action: Review the appropriate documentation, modify the syntax, and try again.

DLBA002E **DUPLICATE** "*identifier-1*" **ARE INVALID**

Explanation: The displayed *identifier-1* cannot appear more than once.

Action: Review the appropriate documentation, modify the syntax, and try again.

DLBA003E **EXPECTING** "*identifier-1*"**; FOUND** "*identifier-2*"

Explanation: The displayed *identifier-2* is an invalid element.

Action: Review the appropriate documentation, modify the syntax, and try again.

DLBA004E **INVALID SYNTAX IN EXEC CICS STATEMENT**

Explanation: The statement is invalid, either because no function is specified or the END-EXEC clause is missing.

Action: Review the appropriate IBM CICS documentation, modify the syntax, and try again.

DLBA006W "*identifier-1*" **FIRST CHARACTER MUST BE ALPHABETIC**

Explanation: The displayed *identifier-1* is invalid: the first character must be alphabetic.

Action: Modify the name of *identifier-1*, make sure the first character is alphabetic, and try again.

DLBA007W "*identifier-1*" **MUST CONSIST ONLY OF NUMERIC OR ALPHABETIC CHARACTERS**

Explanation: The displayed *identifier-1* is invalid because it contains at least one character that is neither numeric nor alphabetic.

Action: Modify the name of *identifier-1*, make sure the first character is alphabetic and that the rest are either numeric or alphabetic, then try again.

DLBA011E "*identifier-1*" **IS AN UNDEFINED DATAVIEW**

Explanation: The displayed *identifier-1* is an undefined dataview. A spelling error may have caused this message.

Action: Check the spelling of *identifier-1* against the spelling of the dataview name, and make sure it is correct. If needed, modify the spelling of *identifier-1* and try again.

DLBA015W **THE "*identifier-1*" OBJECT DECIMAL LOCATION DIFFERS FROM THE SUBJECT**

Explanation: The key-value subject in the dataview definition differs from the displayed comparand object, *identifier-1*.

Action: Review the specification in the dataview definition, modify the value of the key-value subject or the value of *identifier-1*, and try again.

DLBA016W **THE "*identifier-1*" PICTURE TYPE IS INCOMPATIBLE WITH THE SUBJECT TYPE**

Explanation: The key-value subject in the dataview definition differs from the displayed comparand object, *identifier-1*.

Action: Review the specification in the dataview definition, modify the value of the key-value subject or the value of *identifier-1*, and try again.

DLBA017W **THE OBJECT OF COMPARISON "*operand-1*" IS LARGER THAN THE SUBJECT**

Explanation: The key-value subject in the dataview definition differs from the displayed comparand object, *operand-1*.

Action: Review the specification in the dataview definition, modify the value of the key-value subject or the value of *operand-1*, and try again.

DLBA018E **THE "*CA-DATACOM/DB statement*" REQUIRES A KEY-NAME TO BE SPECIFIED IN THE DATAVIEW**

Explanation: The displayed *CA-DATACOM/DB statement* requires a key-name specification in the dataview definition.

Action: Check each reference to the dataview for compatibility. Make sure you have included a valid ACCESSED BY clause in the dataview and that the key-name is correctly spelled.

DLBA019E **THE "*CA-DATACOM/DB statement*" REQUIRES A FILE-ID TO BE SPECIFIED IN THE DATAVIEW**

Explanation: The displayed *CA-DATACOM/DB statement* requires a file-ID specification in the dataview definition.

Action: Check each reference to the dataview for compatibility. Make sure you have included a valid FILE clause in the dataview.

DLBA020E "*identifier-1*" **IS INVALID FILE ORGANIZATION FOR THE DATAVIEW**

Explanation: The displayed *identifier-1* is an invalid value.

Action: Review appropriate documentation. If necessary, re-specify the table organizations and try again.

DLBA021E **THE** "*CA-DATACOM/DB statement*" **REQUIRES A VALID** "*identifier-1*" **TO BE SPECIFIED IN THE "WHERE" QUALIFIER**

Explanation: The displayed *identifier-1* is an invalid value.

Action: Review appropriate documentation, modify the specification, and try again.

DLBA022E **THE COMMAND** "*identifier-1*" **IS INVALID FOR ONLINE EXECUTION UNDER** "*identifier-2*"

Explanation: You cannot sequentially access a CA-DATACOM/DB table when the MONITOR statement specifies *identifier-2*.

Action: Review appropriate documentation. Either determine an alternate method for procedure specification or delete the MONITOR statement, and try again.

DLBA023E "*identifier-1*" **IS INVALID COMMAND SYNTAX FOR ACCESS TO A** "*identifier-2*" **FILE**

Explanation: The displayed file type, *identifier-2*, is invalid.

Action: Review appropriate documentation, make sure you correctly specify the file type, and try again.

DLBA024E **DATAVIEW IS MISSING A REQUIRED** "*identifier-1*"

Explanation: The dataview definition is missing the displayed element, *identifier-1*.

Action: Review appropriate documentation, make sure you correctly specify *identifier-1* in the dataview definition, and try again.

DLBA025E **"ACCESSED BY" CLAUSE IS MISSING A REQUIRED** "*identifier-1*"

Explanation: The **ACCESSED BY** clause in the dataview definition is missing the displayed element, *identifier-1*.

Action: Review appropriate documentation, make sure you correctly specify *identifier-1* in the ACCESSED BY clause in the dataview definition, and try again.

DLBA027E **THE** "*identifier-1*" **KEY NAME IS UNDEFINED IN THE** "*identifier-2*" **DATAVIEW**

Explanation: The displayed key name, *identifier-1*, was not defined in the dataview definition. A spelling error may have caused this message.

Action: Check the spelling of *identifier-1* keys specified in the ACCESSED BY clause of the *identifier-2* dataview. Correct any spelling errors, and try again.

DLBA028E **THE** "*CA-DATACOM/DB statement*" **REQUIRES A WORKAREA IDENTIFIER TO BE SPECIFIED IN THE DATAVIEW**

Explanation: The displayed *CA-DATACOM/DB statement* was not specified with a WORKAREA clause.

Action: Review appropriate documentation, modify the specification, and try again.

DLBA029E **ELEMENT-SECURITY-CODE MISSING CLOSING RIGHT PARENTHESIS ")"**

Explanation: The ELEMENT-SECURITY-CODE lacks a closing right parenthesis.

Action: Modify the specification, and try again.

DLBA030E "*operand*" **IS AN INVALID RELATIONAL OPERATOR FOR THE** "*identifier-1*" **FUNCTION**

Explanation: You specified an invalid relational operator for the displayed *identifier-1* function.

Action: Review appropriate documentation. Specify a valid relational operator for the function, and try again.

DLBA031W "*identifier-1*" **IS OUTSIDE THE MINIMUM/MAXIMUM RANGE OF** "*identifier-2/identifier-3*" **FUNCTION**

Explanation: The value of *identifier-1* does not fall within the proper range.

Action: Review appropriate documentation, modify at least one of the specifications, and try again.

DLBA032A **USING** "*literal*" **AS THE QUALIFIER SUBJECT**

Explanation: The *literal* key will be used in the dataview when KEY-VALUE is the subject on the procedure qualifier.

Action: No action is required.

DLBA033E **THE** "*CA-DATACOM/DB statement*" **CAN NOT BE TRANSLATED BECAUSE THE PREVIOUS DATAVIEW STATEMENT DID NOT END PROPERLY WITH A PERIOD**

Explanation: the displayed *identifier-1* function cannot execute because the dataview definition does not end with **.** (a period).

Action: Put a **.** (period) on the end of the DATAVIEW or TERMINAL-VIEW statement preceding the *identifier-1* specification. Resubmit the job.

DLBA035E "*identifier-1*" **DATAVIEW HAS NO DEFINED ELEMENT LIST**

Explanation: The displayed *identifier-1* dataview definition is invalid because it does not contain an element list.

Action: Review documentation on the DATAVIEW format, modify the specification, and try again.

DLBA036E **INVALID RELATIONAL OPERATOR SEQUENCE**

Explanation: You specified an invalid relational operator.

Action: Review appropriate documentation. Specify a valid relational operator for the function, and try again.

DLBA038E "*identifier-1*" **IS AN INVALID COMMAND FOR ACCESSING A** "*identifier-2*" **FILE**

Explanation: You cannot access the displayed file type, *identifier-2*, with a *identifier-1* command.

Action: Review appropriate documentation, make sure you correctly specify the file type, and try again.

DLBA040E **MORE THAN** "*identifier-1*" "*identifier-2*" **SPECIFIED**

Explanation: The value of *identifier-2* cannot be greater than the value of *identifier-1*.

Action: Re-specify at least one of the values. Make sure that the value of *identifier-2* is less than or equal to the value of *identifier-1*.

DLBA041E **EXPECTING** "*identifier-1*" **IDENTIFIER; FOUND** "*identifier-2*"

Explanation: At least one specification is invalid.

Action: Refer to the appropriate documentation, review syntax, and re-specify at least one of the statements. Resubmit the job.

DLBA042E **EXPECTING** "*identifier-1*" **IDENTIFIER BUT THE DLDB STATEMENT PREMATURELY ENDED**

Explanation: At least one specification is invalid.

Action: Refer to the appropriate documentation, review syntax, and re-specify at least one of the statements. Resubmit the job.

DLBA044E **NUMERIC** "*identifier-1*" **IDENTIFIER**

Explanation: The displayed *identifier-1* is defined as having a non-numeric value.

Action: Modify the value, make sure the value is non-numeric, and resubmit the job.

DLBA045E **NON-NUMERIC** "*identifier-1*" **IDENTIFIER**

Explanation: The displayed *identifier-1* is defined as having a numeric value.

Action: Modify the value, make sure the value is numeric, and resubmit the job.

DLBA047A "*identifier-1*" **GREATER THAN MAXIMUM OF** "*identifier-2*" **CHARACTERS**

Explanation: The named *identifier-1* is longer than *identifier-2* characters, which may cause truncation or reference problems.

Action: Rename *identifier-1*. Make sure that the number of characters in *identifier-1* is less than or equal to the numeric value of *identifier-2*.

DLBA050A "*identifier-1*" **CLAUSE IS INVALID WITH A/AN** "*identifier-2*" **FUNCTION**

Explanation: The named *identifier-1* clause cannot be used with the *identifier-2* function, e.g., WHEN ERROR cannot be used with with the record-at-a-time DATAVIEW Statement.

Action: Refer to the appropriate documentation, and review syntax. Change or modify at least one of the specifications and resubmit the job.

DLBA051E **THE COMMAND** "*identifier-1*" **FUNCTION IS MISSING A REQUIRED** "*identifier-2*" **CLAUSE**

Explanation: The named *identifier-1* function is invalid because it is missing the *identifier-2* clause.

Action: Refer to the appropriate documentation, and review syntax. Add, change, or modify at least one of the specifications and resubmit the job.

DLBA057E **PROCEDURE DIVISION HEADER MISSING**

Explanation: The Procedure Division header is missing.

Action: Type in a header for the Procedure Division, and resubmit the job.

DLBA058E **ID-AREA MUST BE SPECIFIED IN THE DATACOM SECTION FOR USE WITH THE {OPEN|CLOSE} FUNCTION**

Explanation: The DATACOM SECTION requires an ID-AREA clause if the OPEN or CLOSE function is used.

Action: Review syntax. If you plan to use an OPEN or CLOSE function in the DATACOM SECTION, make sure you include an ID-AREA clause. Resubmit the job.

DLBA059A "*identifier-1*" **IDENTIFIER IS UNDEFINED IN THE PROGRAM**

Explanation: The named *identifier-1* identifier is undefined. CA-MetaCOBOL+ cannot perform source verification.

Action: Check the specification of *identifier-1*, and verify its location in the TCA-TWA.

DLBA062E "*identifier-1*" **IDENTIFIER MUST BE** "*identifier-2*" **CHARACTERS LONG**

Explanation: The number of characters in *identifier-1* is not equal to the value displayed in *identifier-2*.

Action: Rename *identifier-1*. Make sure that the number of characters in *identifier-1* is equal to the numeric value of *identifier-2*.

DLBA084E "*identifier-1*" **MAY NOT BE A LITERAL**

Explanation: The named *identifier-1* is invalid because it is a literal. A valid value is a literal.

Action: Modify the specification of *identifier-1*, and resubmit the job.

DLBA085E "*identifier-1*" **MAY NOT BE AN IDENTIFIER**

Explanation: The named *identifier-1* is invalid because it is an identifier. A valid value is a literal.

Action: Modify the specification of *identifier-1*, and resubmit the job.

DLBA091E **A DATAVIEW CAN BE CODED ONLY IN THE {FILE|COMMUNICATION|REPORT|NO SECTION} SECTION**

Explanation: At least one dataview section was entered into an invalid location in the program.

Action: Refer to the appropriate documentation, and review the sections into which dataview definitions can be coded. After you retype the specification into a valid location, delete the old specification, and resubmit the job.

DLBA092E **THE COMMAND "*command*" IS INVALID FOR USE WITH A DATAVIEW DEFINED FOR GENERIC ACCESS**

Explanation: The format of the command accessing the dataview cannot reference a dataview specified for generic access.

Action: Review appropriate documentation, choose a different command or modify the dataview definition, and try again.

DLBA093E **THE COMMAND "*command*" IS INVALID FOR USE WITH A DATAVIEW DEFINED FOR PHYSICAL ACCESS**

Explanation: The format of the command accessing the dataview cannot reference a dataview specified for physical access. For physical access, the dataview definition must contain the ACCESS IS PHYSICAL clause, and a user-defined dataview must contain the FILE IS *file-literal* clause.

Action: Review appropriate documentation, choose a different command or modify the dataview definition, and try again.

DLBA094E **THE COMMAND "*command*" STATEMENT IS INVALID FOR USE WITH A DATAVIEW DEFINED FOR PHYSICAL ACCESS**

Explanation: Only the following CA-DATACOM/DB statements are valid when used with a dataview in a physical search:

FOR
LOCATE PHYSICAL
OBTAIN PHYSICAL
READ PHYSICAL

Action: Review documentation on the DATAVIEW format, modify the specification, and try again.

DLBA101E **{FILE-ID|KEY NAME|PREFIX} IS INVALID FOR DATAVIEW** "*dataview-name*"

Explanation: Either the table-ID, key name, or the prefix is invalid.

Action: Review documentation on the DATAVIEW format, modify the specification, and try again.

DLBA104E **DATAVIEW** "*dataview-name*" **IS NOT RELATED TO PROGRAM** "*program-ID*"

Explanation: Either the *program-ID* is incorrect or the program entity-occurrence status and relationship are undefined. The *program-ID* must be defined as PROD or TEST in the CA-DATADICTIONARY and related to each dataview definition referenced in the program by the relationship name *pgm-dvw-use*.

Action: Enter a valid program-ID, and resubmit the job.

DLBA105E *definition* **FOR DICTIONARY DATAVIEW** "*dataview-name*" **NOT DEFINED.**
[CANNOT GENERATE {WORKAREA|ELEMENT-LIST}]

Explanation: One of the following is not defined:

DATAVIEW
ELEMENT(S)
FIELD(S)
RECORD(S)
FILE(S)
AREA(S)
DATABASE

Action: Review appropriate documentation, modify the CA-DATADICTIONARY definition or the reference, and try again.

DLBA111E **DATAVIEW** "*dataview-name*" **RELATED TO MORE THAN ONE RECORD**

Explanation: The displayed *dataview-name* is incorrectly related to more than one record.

Action: Review CA-DATADICTIONARY definitions, and change the status of all unnecessary files or records to HIST (history).

DLBA117E **COPYBOOK FOR DATAVIEW** "*dataview-name*" **UNDEFINED**

Explanation: The COBOL definition for a field is not defined. The dataview exists but the corresponding COBOL code for the fields in the dataview are not defined.

Action: Review appropriate documentation, either modify the CA-DATADICTIONARY COPYBOOK for the dataview workarea or the COBOL code, and try again.

DLBA118E **CORRECTED LEVEL NUMBER FOR DATAVIEW** "*dataview-name*" **WORKAREA EXCEEDS 49**

Explanation: The value of at least one level number was invalid: a valid value cannot be greater than 49.

Action: Review record descriptions and group items within affected records, make appropriate modifications, and try again.

DLBA119W **PREFIXED DATA-NAME FOR DATAVIEW** "*dataview-name*" **WORKAREA CAUSES TRUNCATION**

Explanation: Either the prefix is more than five characters long, or the prefix and the *dataview-name* are more than 30 characters long. In the first case, the prefix is truncated; in the second, the whole *dataview-name* is truncated.

Action: If the prefix, including the hyphen, is longer than five characters, use the DATAVIEW Statement's PREFIX IS clause to define a shorter prefix. If the *dataview-name* and its prefix are longer than 30 characters, re-specify the *dataview-name*.

DLBA131E **INVALID COMBINATION OF CLAUSES FROM BOTH DICTIONARY AND USER-DEFINED DATAVIEW**

Explanation: At least one clause in the dataview definition is invalid because it can only be specified for another type of dataview definition.

Action: Review the DATAVIEW Statement syntax for the CA-DATADICTIONARY or user-defined format. Modify the dataview definition, and resubmit the job.

DLBA134E **CA-DATADICTIONARY DATAVIEWS REQUIRE SPECIFICATION OF AN ID-AREA CLAUSE IN THE DATACOM SECTION**

Explanation: The ID-AREA clause is invalid.

Action: Refer to appropriate documentation on the CA-DATACOM SECTION statement for information on the ID-AREA clause. Modify the specification, and resubmit the job.

DLBA135E **DATACOM SECTION ID-AREA UNDEFINED**

Explanation: The ID-AREA clause does not exist. For CA-DATACOM/DB release 7.4 or higher, this clause is required in the DATACOM SECTION statement.

Action: Refer to appropriate documentation on the DATACOM SECTION statement for information on the ID-AREA clause. Add the specification, and resubmit the job.

DLBA136E **ACCESS CLAUSE NEEDED IN DATAVIEW** "*dataview-name*" **TO BE REFERENCED WITH A** "*CA-DATACOM/DB-statement*" **STATEMENT**

Explanation: An ACCESS clause is missing. This clause is required when a CA-DATADICTIONARY dataview is used with CA-DATACOM/DB Facility statements such as LOCATE, OBTAIN, or READ.

Action: Enter a valid ACCESS clause, and resubmit the job.

DLBA137E **DATAVIEW** "*dataview-name*" **INVALID FOR USE WITH** "*set-at-a-time-statement*"

Explanation: At least one clause is invalid.

Action: Refer to the description of the set-at-a-time dataview definition, change or modify the definition, and resubmit the job.

DLBA138W **DATAVIEW** "*dataview-name*" **REFERENCED BY A** ***"CA-DATACOM/DB-statement"*** **CANNOT BE UPDATED**.

Explanation: A CA-DATACOM/DB WRITE, REWRITE, INSERT, UPDATE, or DELETE statement is referencing a dataview whose CA-DATADICTIONARY attributes denote that updates are not allowed.

Action: Contact your CA-DATADICTIONARY system administrator to see if the update-intent attribute can be changed.

DLBA139E **"WITHIN RANGE" CLAUSE IS INVALID SYNTAX FOR THE** "*verb*" **STATEMENT**

Explanation: The WITHIN RANGE clause in invalid when used with the displayed *verb*. LOCATE [AND HOLD] or READ/OBTAIN [AND HOLD] are the only statements that can use this clause.

Action: Refer to the description of the *verb* change or modify the *verb* specification, and resubmit the job.

DLBA141E **INVALID DBID SPECIFICATION FOR USE WITH CA-DATADICTIONARY/USER-DEFINED DATAVIEW**

Explanation: The specification for the DATA-BASE-IDENTIFICATION clause in the dataview definition is invalid. You cannot specify a numeric value.

Action: Refer to appropriate documentation on the DATAVIEW Statement for information on the DATA-BASE-IDENTIFICATION clause. Modify the specification, and resubmit the job.

DLBA142E **INVALID TEST-VERSION OPERAND SPECIFIED**

Explanation: The version number for a dataview definition is invalid. A valid value is a number from 0 through 999.

Action: Modify the specification, and resubmit the job.

DLBA150A **CA-DATADICTIONARY SIGNON SUCCESSFUL FOR DBID** "*nnn*"

Explanation: The specified DATADICTIONARY was accessed.

Action: No action is required.

DLBA151A **PROGRAM** "*program-name*" **DEFINED TO DATADICTIONARY**
REQUESTED: STATUS "*status*" **VERSION** "*nnn*"
RETURNED: STATUS "*status*" **VERSION** "*nnn*"

Explanation: Attributes of the program-entity-occurrence have been updated in the DATADICTIONARY.

Action: No action is required.

DLBA153A **DATAVIEW "*data-view name*" HAS BEEN INHIBITED FROM EXECUTION SINCE DATADICTIONARY SHOWS** *reason*

The "reason" is any one of four reasons describing a missing relationship in the dictionary:

D NO RELATIONSHIP BEYOND RECORD ENTITY OCCURRENCE.
D NO RELATIONSHIP BEYOND TABLE ENTITY OCCURRENCE.
D NO RELATIONSHIP BEYOND AREA ENTITY OCCURRENCE.
D THE RELATED DATABASE ENTITY OCCURRENCE FAILS A STRUCTURE VERIFICATION TEST.

Explanation: Access to the specified DATAVIEW will not be executable.

Action: Consult your database administrator.

DLBA155A **RQA RESOLUTION STATISTICS:**

Explanation: This advisory diagnostic is followed by a series of lines showing the efficiency of the RQA resolution for numeric fields.

Action: None required.

DLBA166E **DATAVIEW "*dataview-name*" IS CODED USING A DISCONTINUED SYNTAX**

Explanation: At least one syntax statement is no longer supported.

Action: If you need to translate a user-defined DATAVIEW Statement that contains the discontinued syntax, include the UPSI3=U translate-time option, and resubmit the job.

DLBA175A **STATEMENT "*verb*" AS CODED IS VALID ONLY WITH CA-DATACOM/DB 7.5 AND LATER**

Explanation: Because you are not using CA-DATACOM/DB 7.5 or higher, the displayed *verb* is invalid.

Action: Refer to the appropriate documentation. You will probably have to redesign some of the program logic and code. Resubmit the job.

DLBA204F **MACRO SET SPP MUST BE LOADED BEFORE DLM**

Explanation: If you plan to use both the Structured Programming Facility (SPF) and the CA-DATACOM/DB Facility, you must load the SPF's SPP macro set before you load the CA-DATACOM/DB Facility's macro sets.

Action: Make sure the SPP macro set is loaded before the DLM macro set, then continue.

DLBA205F **MACRO SET SPP NOT LOADED; FOUND SPP** "*verb*"

Explanation: If you plan to use both the Structured Programming Facility (SPF) and the CA-DATACOM/DB Facility, you must load the SPF's SPP macro set before you load the CA-DATACOM/DB Facility's macro sets.

Action: Make sure the SPP macro set is loaded before the DLM macro set, then continue.

DLBA221F **PSTAT OPTION SPECIFIES AN INVALID STATUS**

Explanation: The **PSTAT=***value* option was specified with an invalid *value*. A valid value is a 1- through 4-character alphanumeric specification that defines the disposition of the CA-DATADICTIONARY entity-occurrence, for example, PROD or TEST.

Action: Refer to appropriate documentation, modify the PSTAT=*value* option, and resubmit the job.

DLBA222F **CA-DATADICTIONARY DBID** "*nnn*" **INVALID**

Explanation: The DBID=*nnn* translate-time option specification is invalid. A valid value for *nnn* is 0 through 999. The default is 000.

Action: Modify the DBID=*nnn* option, and resubmit the job.

DLBA223F **PROGRAM** "*program-name*" **OR VERSION** "*xxx*" **OF PROGRAM** "*program-name*" **NOT DEFINED IN DATADICTIONARY**

Explanation: CA-DATADICTIONARY requires a valid reference. This is a program entity-occurrence with the same name as the program-ID to be defined with TEST or PROD status.

Action: Refer to the appropriate documentation, and specify the entity-occurrence. For a program with TEST status, a version number is required. Resubmit the job.

DLBA241F **CA-DATADICTIONARY FATAL ERROR** "*error-code*"

Explanation: A fatal error occurred during a translation. The *error-code* is generated during an attempt to access CA-DATADICTIONARY.

Action: Refer to the CA-DATADICTIONARY *Service Facilities* manual for a description of the *error-code*. If you cannot correct the error, report this problem to CA Support.

DLBA242F **SIGNON TO CA-DATADICTIONARY VIA ADRXDDON HAS FAILED. REVIEW PRODUCT INSTALLATION/LINK-EDIT FOR ADRXDDON**

Explanation: A fatal error occurred during an attempt to sign on to CA-DATADICTIONARY.

Action: Review the product installation/link-edit for ADRXDDON. If the error recurs, report this problem to CA Support.

DLBA243F **SIGNON TO CA-DATADICTIONARY NOT ATTEMPTED. NECESSARY PROGRAM-ID PARAGRAPH NOT RECEIVED**

Explanation: A fatal error has occurred. The PROGRAM-ID paragraph is either missing or invalid.

Action: Review the appropriate documentation, add or modify the PROGRAM-ID paragraph, and resubmit the job. If the error recurs, report this problem to CA Support.

DLBA251F **MARKERS REQUIRED TO COMPLETE CA-DATACOM/DB TRANSLATION EXCEEDS MAXIMUM**

Explanation: The number of FOR statements and dataviews is greater than the value of the marker limit.

Action: Review the appropriate documentation, reduce the number of dataview definitions or FOR statements, and resubmit the job.

DLBA261F **INTERNAL MACRO PROCESSING ERROR**
submessage

Explanation: An internal error has occurred. One of the following *submessages* was generated:

INVALID INPUT EXIT CALL SEQUENCE
INVALID ERROR CODE "nn" FROM INPUT EXIT "subroutine-name"
INVALID OBJECT TYPE DESIGNATION
ELEMENT NAME CANNOT BE LOCATED

Either the input exit sequence is invalid or an invalid error number was received from the input exit routine.

Action: Review the syntax of the IXIT=*value* JCL parameter or translate-time option. Specify a valid *value*, and resubmit the job. If the error recurs, report this problem to CA Support.

DLBA262F **INVALID RETURN VALUES FOR CALL TO INPUT EXIT "subroutine-name"**

Explanation: The **IXIT=ADRXIXIT** option is absent or mis-specified.

Action: Review the syntax of the IXIT=*value* JCL parameter or translate-time option. Specify IXIT=ADRXIXIT, and resubmit the job. If the error recurs, report this problem to CA Support.

DLBA311E **FOUND UNEXPECTED** "*identifier-1*" **FOLLOWING** "*identifier-2*"

Explanation: The displayed *identifier-1* is part of a FOR Statement; it is invalid only because it is out of sequence. The displayed *identifier-2* is a word or phrase in a WHERE, COUNT, ORDER, or HOLD clause.

Action: Refer to appropriate documentation, and review syntax. Change or modify the specification, and try again.

DLBA312E **FOUND UNPAIRED {LEFT|RIGHT} PARENTHESIS**

Explanation: The FOR Statement's WHERE Clause is missing at least one parenthesis.

Action: If necessary, review the appropriate documentation. Modify the specification, and try again.

DLBA313E INVALID COMPOUND RELATIONAL OPERATOR

Explanation: You specified an invalid relational operator in the FOR Statement's WHERE Clause.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Specify a valid relational operator for the WHERE Clause, and try again.

DLBA314E FOUND INVALID OR UNDEFINED "*word***" IN WHERE CLAUSE**

Explanation: You specified an invalid *word* operator in the FOR Statement's WHERE Clause.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Modify the WHERE Clause, and try again.

DLBA316E FOUND "*identifier-1***" CLAUSE WHICH HAS NO CORRESPONDING "FOR" STATEMENT**

Explanation: A WHEN END or WHEN ERROR clause does not belong to a FOR Statement. An error in FOR processing may have occured.

Action: Refer to appropriate documentation, and review the syntax of the FOR Statement. Modify the FOR Statement or the clause, and try again.

DLBA342E "*identifier***" CANNOT BE USED IN ARITHMETIC EXPRESSION**

Explanation: Non-numeric fields are invalid in the FOR Statement's WHERE Clause arithmetic expression.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Specify a valid, numeric identifier for the WHERE Clause, and try again.

DLBA343E "*identifier***" IS INVALID "***description***" FOR DATAVIEW "***dataview-name***"**

Explanation: The dataview specified in the FOR Statement is not compatible with either the subject or the object specified in the WHERE Clause.

The displayed *identifier* is a unique occurrence of a data-name. The displayed *description* can be: the retrieval-count-identifier in a COUNT clause; a subject in a WHERE Clause condition; a field in an ORDER Clause.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Specify a valid *identifier*, and try again.

DLBA344E **OBJECT** "*identifier-1*" **HAS INCOMPATIBLE {USAGE|FIELD SIZE|DECIMAL LOCATION|SIGN} WITH SUBJECT** "*identifier-2*"

Explanation: The subject and object are not compatible.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Specify a valid *identifier*, and try again.

DLBA345E **{INVALID|EXTRANEOUS|INSUFFICIENT} SUBSCRIPTS/INDEXES FOUND FOR IDENTIFIER** "*identifier*"

Explanation: Incorrect subscripts or indexes are specified for the displayed *identifier*, which is in the FOR Statement's WHERE Clause.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Modify the specification, and try again.

DLBA346E **FOR STATEMENT NESTING LIMIT OF** "*limit*" **EXCEEDED**

Explanation: FOR Statement nesting levels now exceeds the displayed *limit.* By default, the *limit* is 9.

Action: If you require more than 9 nesting levels, call CA Support.

DLBA347E "*identifier*" **CANNOT BE USED AS A RECORD COUNT IDENTIFIER FOR DATAVIEW** "*dataview-name*"

Explanation: The record count identifier cannot be subordinate to the dataview referenced by the FOR Statement, and the identifier must have a value greater than zero.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Specify a valid record count identifier, and try again.

DLBA381E **{SUBJECT|OBJECT} TABLE CANNOT ACCOMMODATE ANOTHER ENTRY; LIMIT OF** "*limit*" **IS ABOUT TO BE EXCEEDED**

Explanation: The number of subjects and objects in the **FOR Statement**'s **WHERE Clause** exceeds the displayed *limit* on the number of conditions. By default, the *limit* is 20.

Action: If you require more than 20 conditions, call CA Support.

DLBA382E "*condition*" **EXCEEDS MAXIMUM LIMIT**

Explanation: The Request Qualification Area (RQA) generated by the FOR Statement's WHERE or ORDER Clauses exceeds either the maximum length of 363 bytes or the maximum limit of 160 keys.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. If the WHERE Clause generated the error, reduce the number of conditions in the WHERE Clause. If the ORDER Clause generated the error, reduce the number of keys in the ORDER Clause.

DLBA383E **ERROR ON CALL TO INPUT EXIT ADRXCOND IXIT LINE = "ADRXCOND** '*function*' **RETURNS ERROR** "*error-code*"

Explanation: An internal error has occurred.

Action: Write down the displayed *function* and *error-code*, then report this problem to CA Support.

DLBA391E **INVALID {CLASS|SIGN} CONDITION TEST**

Explanation: You specified an invalid condition test in the FOR Statement's WHERE Clause. The statement supports COBOL relation conditions - subject, relation, object - but CLASS and SIGN conditions are not supported.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Modify the specification, and try again.

DLBA392E **INVALID CONDITION-NAME TEST**

Explanation: You specified an invalid condition test in the FOR Statement's WHERE Clause. The statement supports COBOL relation conditions - subject, relation, object - but CONDITION-NAME tests are not supported.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Modify the specification, and try again.

DLBA393E **INVALID USE OF EDITED FIELDS**

Explanation: You specified an invalid field in the FOR Statement's WHERE Clause. The statement does not support edited picture fields.

Action: Refer to the appropriate documentation, and review the syntax of the FOR Statement. Modify the specification, and try again.

DLBA394E **CONDITIONAL STATEMENT "*cobol-statement*" CODED WITHIN A FOR CONSTRUCT SUPPORTED ONLY IN CONJUNCTION WITH STRUCTURED PROGRAMMING - REVIEW SP OPTION**

Explanation: Some constructs used with the FOR Statement require you to use the CA-MetaCOBOL+ Structured Programming Facility:

IF
SEARCH
ON SIZE ERROR
ON OVERFLOW
INVALID KEY
AT END
NO DATA KEY
AT END-OF-PAGE

Action: If you plan to use any of these constructs, make sure you use the Structured Programming Facility in conjunction with the CA-DATACOM/DB Facility.

DLBA410E **NUMBER OF "FOR" STATEMENTS IN THIS PROGRAM EXCEEDS EXPECTED MAXIMUM OF** *nn*

Explanation: More FOR Statements than expected have been coded in the program.

Action: Contact your CA-MetaCOBOL+ system administrator.

DLBA411E **DATAVIEW** *dataview-name* **HAS BEEN SPECIFIED IN MORE THAN ONE "FOR" STATEMENT. ITS USE IN A RECONSTRUCT STATEMENT IS INVALID**

Explanation: The named DATAVIEW cannot be used in a RECONSTRUCT Statement because it has been used in two or more FOR Statements. Only one FOR Statement can be used for a DATAVIEW referenced in a RECONSTRUCT Statement.

Action: If the correct DATAVIEW was specified in the RECONSTRUCT Statement, remove all but one FOR Statement for the DATAVIEW.

DLBA423E **MORE THAN MAXIMUM NUMBER OF** *nn* **RECONSTRUCT STATEMENTS FOUND**

Explanation: More RECONSTRUCT Statements than expected are coded in the program.

Action: Contact your CA-MetaCOBOL+ system administrator.

DLBA424E **DATAVIEW** *dataview-name* **IS UNDEFINED**

Explanation: The named DATAVIEW is not defined in your program.

Action: Check the spelling of the DATAVIEW in the RECONSTRUCT Statement. If it is spelled correctly, define it in the Data Division.

DLBA424E **DATAVIEW** *dataview-name* **IS INCAPABLE OF SET-AT-A-TIME ACCESS. CANNOT BE SPECIFIED ON A RECONSTRUCT STATEMENT.**

Explanation: Only DATAVIEWs without the ACCESS clause use a Request Qualification Area.

Action: Check the spelling of the DATAVIEW in the RECONSTRUCT Statement.

DLBA426E **FIELD** *field-name* **SPECIFIED AS QUALIFICATION EXPRESSION ON RECONSTRUCT STATEMENT IS UNDEFINED.**

Explanation: The name of the field specified as the qualification expression of the RECONSTRUCT Statement is undefined.

Action: Check the spelling of the field in the RECONSTRUCT Statement. If the spelling is correct, define the field in the Data Division.

DLBA492E **DYNAMIC RQA RESOLUTION INTERNAL ERROR. UNABLE TO RESOLVE** *dataview-name* **ASSOCIATED WITH FOR STATEMENT NUMBER** *nn*.

Explanation: A significant error was encountered while parsing the RECONSTRUCT Statement.

Action: Contact the CA-MetaCOBOL+ system administrator.

DLBA493E **DYNAMIC RQA RESOLUTION INTERNAL ERROR. UNEXPECTED NAME FOUND ON LEVEL-03 WORKAREA DATANAME** "*dataname*".

Explanation: A significant error was encountered while parsing the RECONSTRUCT Statement.

Action: Contact the CA-MetaCOBOL+ system administrator.

DLBA495E **DYNAMIC RQA RESOLUTION INTERNAL ERROR. UNABLE TO RESOLVE SECURITY CODE FOR ELEMENT** "*element-name*"

Explanation: A significant error was encountered while parsing the RECONSTRUCT Statement.

Action: Contact the CA-MetaCOBOL+ system administrator.

H78 **PROCESSING TERMINATED - SEE ABOVE MESSAGES**

Explanation: An error has occurred in the input exit to CA-DATADICTIONARY. Messages are displayed above this one.

Action: Check the preceding messages, make all necessary modifications, and resubmit the job.

Appendix E. Return Codes and Error Codes

This appendix contains two sections:

- CA-DATACOM/DB Return Codes
- Compound Boolean Selection (CBS) Error Codes

E.1 CA-DATACOM/DB Return Codes

After request processing is completed, CA-DATACOM/DB return codes are placed in the 2-digit return code field of either:

- *dataview-name*-RA-STATUS-CODE (the CA-DATACOM/DB request area), which is part of the expanded dataview.
- DB-ERROR-CODE (the generated field), which is in WORKING-STORAGE after dataview expansion is finished.

A one-byte hexadecimal error subcode is placed in relative byte 38 of the request area to aid in problem determination, especially for return code 13.

Code	**Reason**
" "	(blank space) Processing completed without error
01	Invalid request command
02	Invalid table name
03	Invalid key name
04	Invalid record ID
05	Table not open - check Multi-User Facility status
06	Table not open for update
07	Data area full
08	Index full
09	Request not preceded by prerequisite request

10	Duplicate Master Key not allowed
Code	**Reason**
11	Master Key has been modified - update was rejected
12	Special deleted record
13	Internal error - check the one-byte error subcode in relative byte 38 of the request area, and report this problem to CA Support.
14	No record found or no key found
16	Exclusive control interlock
17	Input/Output error
18	Exclusive Control Duplicate
19	End of Table for GETIT, End of area for GETPS
20	Control area key/element buffer too small
21	Error in compress/expand routine
22	Element name not found
23	Element security code violation
24	Exclusive control events exceeded
25	Invalid database ID
26	Insufficient control area buffer space
27	GETIT block size too small
28	Log area block size too small
29	EOF during LOGLB command
30	Table not open for this command
31	Key length inconsistency
36	User view not open
37	Invalid address
38	Previous logging area
39	Cannot process old request
40	Task save area extension too small
41	Not enough extra buffers in Master List
42	Not enough sequential extensions in Master List
43	No valid index
46	Table already open for update
47	Cannot open DB
51	Invalid mix of jobs including DB utilities
52	Recovery file OPEN/CLOSE failure
54	Insufficient open table buffer space
55	Bad user requirements table
56	Bad Master List
57	Bad RWTSA address - a missing OPEN/CLOSE clause can generate this return code.
58	Table not loaded
60	DB cannot open the log area
63	Bad device type
65	DD statement missing
66	Multi-volume open failure
67	CXX interlock
68	Multi-User Facility (MUF) is not up
69	Table has no current index
70	Block length too small
71	CMS open failure
72	Invalid data area control block
74	MVS open allocation area
76	Open error

Code	**Reason**
78	FBA block or extent error
79	CXX is wrong release
80	CA-DATACOM/D-NET error
81	CA-DATACOM/D-NET error (detected by CA-DATACOM/DB)
82	DB SVC program PSW-KEY error
83	SVC integrity error
84	Multitasking error
85	Insufficient tasks
86	The Multi-User Facility (MUF) abended
87	Security violation during open
88	Database has been disabled
89	VSE problem using 'CDLOAD'
91	Compound Boolean Selection (CBS) Facility error - refer to the following section on Compound Boolean Selection (CBS) Return Codes.
92	Set selection interrupt
93	Attempt to position past end/beg-of-set

E.2 Compound Boolean Selection (CBS) Error Codes

If CA-DATACOM/DB generates a return code of 91 in the DB-ERROR-CODE field, check DB-CBS-ERROR-CODE for one of the following values:

Code	**Reason**
91	Temporary CBS index not found
92	Invalid relational operator in the RQA
93	Invalid field to element relation
94	Sort order not A or D
95	Invalid ordered-by field CLASS
96	Invalid CLASS in *condition-operand-1*
97	Invalid CLASS in *condition-operand-2*
98	Invalid zoned field, QA length
99	Duplicate S section
100	Duplicate K section
101	Invalid section type
102	At least one CA-DATACOM/DB record contains errors
103	This CA-DATACOM/DB record contains data errors
104	Invalid logical operator in the RQA
106	Invalid zoned field in qualification area
107	Invalid Compound Boolean Selection command
108	Invalid packed literal in qualification area
109	Invalid operand lengths
110	Undefined element
111	RWTSA overflow
112	SELECT ended because of too many failures
113	SELECT ended because MAX records already in set
120	Invalid data type
121	Invalid sign field
123	Invalid string operator
124	Sign invalid with character data
125	Invalid contains scope
129	Undefined parameter name
130	Error in parameter specification
132	Records accepted interrupt interval reached
133	Records rejected interrupt interval reached
134	Start I/O interrupt interval reached
136	CBSBFR too small
137	ORDER-BY clause too long

Appendix F. Dynamic Request Qualification Area

In the FOR Statement, the programmer specifies the criteria for selecting and ordering records. These criteria are then resolved during translation into a Request Qualification Area (RQA). An RQA is a formatted data definition in a COBOL program indicating selection and ordering criteria to CA-DATACOM/DB or CA-DATACOM/PC. In other words, the selection and ordering criteria are converted into a "fixed" RQA specified at translate-time.

In contrast, The dynamic RQA enables the programmer to resolve the selection criteria at run-time based on information provided by the user of the program. This chapter describes how to set up CA-DATACOM/DB or CA-DATACOM/PC COBOL programs to take advantage of the Dynamic Request Qualification Area (RQA) Resolution Utility. This utility allows the programmer to select and sort records based on run-time user responses.

Features of Dynamic RQA

The two main features of Dynamic RQA are the MCTXDRRU Utility and the Qualification Statement Image (QSI).

MCTXDRRU Utility
The MCTXDRRU Utility is a run-time module which interprets a Qualification Statement Image and from it constructs the corresponding CA-DATACOM/DB or CA-DATACOM/PC Request Qualification Area. The object module MCTXDRRU is provided with CA-MetaCOBOL+ or CA-MetaCOBOL+/PC at installation time. MCTXDRRU must be included when you link-edit your program.

Qualification Statement Image
The Qualification Statement Image is a statement that specifies the criteria for selecting and ordering records. It may have an ORDER Clause, a WHERE Clause, or both. These clauses are similar to the ORDER and WHERE Clauses used in the CA-MetaCOBOL+ FOR Statement.

Each time the user requests information, the MCTXDRRU Utility is invoked and a new RQA is built. More information about the utility and the QSI is given later in this chapter.

F.1 Processing Overview

FOR Statement

The FOR Statement causes a Request Qualification Area to be built for the call to CA-DATACOM/DB or CA-DATACOM/PC. The qualification for records is specified in the WHERE and/or ORDER Clauses. The Request Qualification Area is resolved during the CA-MetaCOBOL+ translation.

The only variation of the FOR Statement that may be used with Dynamic RQA is FOR EACH. Using any other variation results in an error.

RECONSTRUCT Statement

The Dynamic RQA is built by coding a RECONSTRUCT Statement in the COBOL program, as shown below.

```
RECONSTRUCT REQUEST QUALIFICATION AREA

        FOR DATAVIEW dataview-name

        USING QUALIFICATION EXPRESSION FOUND IN data-name
```

This statement is coded in the PROCEDURE DIVISION logically preceding the associated FOR Statement. It will generate some MOVE and CALL statements to invoke the MCTXDRRU utility. The MCTXDRRU utility resolves the specified Qualification Statement Image into a Request Qualification Area used by the FOR Statement.

The RECONSTRUCT Statement provides the ability to dynamically resolve the Request Qualification Area with information that becomes available only at program execution time.

This statement will build some data areas in WORKING-STORAGE for communicating with the utility.

Note: A DATAVIEW named on a RECONSTRUCT Statement may have only one FOR Statement associated with it. There is no restriction on the number of RECONSTRUCT Statements naming the same DATAVIEW.

Request Qualification Area

The newly created Request Qualification Area is placed in the same location as the one created by the FOR Statement, and the FOR Statement continues to use this area in the call to CA-DATACOM/DB or CA-DATACOM/PC.

Warning for Runtime:

Since the newly created RQA can be much larger than the area generated by the original FOR Statement, the COBOL programmer should append an RQA expansion area to the DATAVIEW to accommodate a larger RQA. Failure to include this extra area compromises the integrity of the data areas following the DATAVIEW Statement once a call is made to the Dynamic RQA Resolution Utility. Example:

```
DATAVIEW dataview-name.
02 FILLER                              PIC X(200).
```

MCTXDRRU Utility

When called by the COBOL program at run-time, the MCTXDRRU Utility interprets a Qualification Statement Image and from it constructs the corresponding RQA for CA-DATACOM/DB or CA-DATACOM/PC. The MCTXDRRU utilty must be link-edited with the program that uses it.

An RQA of a DATAVIEW defined at translate-time by a FOR Statement will be destroyed if you use the Dynamic RQA Feature.

Activity and Return Codes

After the Dynamic RQA Utility is invoked, Activity and Return Codes are returned to the program. These codes indicate whether the Dynamic RQA was successfully created. See Section F.4 for more information.

F.2 Qualification Statement Image

The Qualification Statement Image (QSI) specifies the criteria for selecting and ordering records in the RECONSTRUCT Statement. The QSI may have either or both of two clauses:

- the QSI WHERE Clause indicates the record selection criteria
- the QSI ORDER Clause specifies the order in which selected records are to be returned

These QSI WHERE and ORDER Clauses are similar to the WHERE and ORDER Clauses of the CA-MetaCOBOL+ FOR Statement. Differences are discussed further in this section.

This image must be previously coded in the COBOL program or constructed by the COBOL program during execution.

Qualification Statement Image Format

The format of the QSI conforms to two concepts familiar to the CA-DATACOM/DB or CA-DATACOM/PC COBOL programmer. First, the statement is free-form in that the statement components may be separated by one, many, or no spaces. Second, the QSI statement's WHERE and ORDER Clauses are modeled after the CA-MetaCOBOL+ FOR Statement's WHERE and ORDER Clauses.

The QSI is contained in a COBOL data area which is inspected and interpreted by the Dynamic RQA Resolution Utility. It may have any number of leading and/or trailing blanks. All characters within the data area that are not part of a WHERE or ORDER Clause should be blank.

The major difference between a QSI WHERE or ORDER Clause and a CA-MetaCOBOL+ FOR Statement WHERE or ORDER Clause is that the FOR Statement clause is coded in the Procedure Division while the QSI clause is specified in a Data Division field. Other slight differences are noted where appropriate in this section.

F.2.1 QSI WHERE Clause

The QSI WHERE Clause indicates the selection criteria for records to be retrieved. It begins with the word "WHERE" and is followed by the selection condition(s) enclosed in parentheses. A WHERE Clause may indicate only one simple condition or several simple conditions joined by conjunctions. In the RECONSTRUCT Statement, the WHERE Clause image is contained in the data name referenced by the following clause:

`USING QUALIFICATION EXPRESSION FOUND IN` *data-name*

Simple Condition

A simple condition is composed of a subject, a relational operator, and an object:

Subject
Each subject must be the name of a field in the work area of the corresponding DATAVIEW.

Relational operators
Relational operators can be:

<	less than
<=	less than or equal
=	equal

^= not equal -- *the caret* (^) *represents the logical "not" character.* See the Note following.
>= greater than or equal
> greater than
~ string present
^~ string absent -- *the caret* (^) *represents the logical "not" character.* See the Note following.

Note: On mainframe terminals, the logical "not" character is located above the 6. If you are using a mainframe terminal, press the key for the logical not character instead of the caret. On the PC keyboard, press the key for the caret.

These operators are slightly different from those used in the FOR Statement. The FOR Statement allows the use of the words EQUAL, LESS THAN, etc.

Objects
There are three types of objects:

DATAVIEW object
the name of another field in the work area of the corresponding DATAVIEW.

literal object
a value that is enclosed in apostrophes and has the same length and format as the subject field.

The bounding apostrophes for a literal object have no implication on the value contained between them. Numeric and alphanumeric objects alike are enclosed this way. The method of comparison is determined by the attributes of the subject field. For example, if the Usage Clause of the subject is defined as COMP-3, then the object must be in packed-decimal format.

This syntax is slightly different from the syntax for the FOR Statement, which accommodates a literal value or a data name as the object. To achieve the same result as specifying a data name in a the WHERE Clause of a FOR Statement, the programmer must insert the *value* of the data name between the bounding apostrophes of the QSI.

string object
a value that is enclosed in apostrophes and is shorter than the length of the subject field. A string object may only follow a 'string present' or 'string absent' relational operator. If the first character is not the beginning apostrophe, it will be regarded as a mask character to be used in string matching, and the next character must be the beginning apostrophe.

Compound Condition

A compound condition is a series of simple conditions joined by conjunctions. Conjuctions can be:

Ampersand (&) - for AND

Vertical Bar (|) - for OR

These conjunctions differ slightly from the FOR Statement counterpart which uses the words AND and OR. The ampersand (&) and vertial bar (|) characters were chosen because neither can be used in a COBOL data name.

Since the vertical bar character does not appear on the PC keyboard, the vertical broken bar character (|) will also be accepted to indicate OR. In addition, the words AND and OR may be used in place of their respective ampersand and vertical bar characters, but if used, each must be preceded by a space.

The Qualification Statement Image is free-form; it does not require subjects and objects to be separated by spaces from relational operators or conjunctions. This free form provides both a way to keep the conjunction to one character and a way for the conjunction to be adjacent to a preceding DATAVIEW object and subsequent subject with no intervening spaces.

Also unlike the FOR Statement counterpart, a compound condition in a QSI WHERE Clause cannot indicate implied subjects. Moreover, the compound condition must not require normalization.

F.2.2 QSI Order Clause

The ORDER Clause indicates the order in which the selected records are to be retrieved. It begins with the word "ORDER" and is followed by the names of one or more DATAVIEW fields enclosed in parentheses. When more than one field is specified, commas or spaces may be used to separate the fields.

Each field may be accompanied by a preceding order indicator. This is specified as the letter A or D enclosed in parentheses to indicate ascending or descending order. The default is ascending.

(A) - ascending
(D) - descending

The order of the data is determined by the ASCII or EBCDIC collating sequence, whichever is in effect at execution. These order indicators are slightly different from those used in the ORDER Clause of the FOR Statement. In the FOR Statement, the words "ascending" and "descending" cannot be abbreviated; they must be spelled out.

In the RECONSTRUCT Statement, the ORDER Clause image is contained in the data name referenced by the following clause:

```
     USING QUALIFICATION EXPRESSION FOUND IN data-name
```

F.3 Building a Dynamic RQA

The FOR Statement causes an RQA to be built for the call to CA-DATACOM/DB or CA-DATACOM/PC. The qualification for records is specified in the WHERE and/or ORDER Clauses. The RQA is resolved during the CA-MetaCOBOL+ translation.

The RECONSTRUCT Statement provides the ability to dynamically resolve the RQA with information that becomes available only at program execution time.

To build the utility, perform the steps below.

Step 1

Define a data-name containing a QSI. Use the format:

```
01    dataname         PIC X(nn).
```

nn is the length of the Picture Clause.

Step 2

Move a valid value for a QSI into this *data-name*. The QSI can have a WHERE Clause, an ORDER Clause, or both. The order of the clauses is not important.

Blank spaces are essential only if they are needed to pad the object of a WHERE Clause to the exact length of its subject. Refer to Section 4.2.1 for complete instructions for coding a WHERE Clause. Refer to Section 4.2.2 for complete instructions for coding an ORDER Clause.

Step 3

Define the DATAVIEW in your program and provide an RQA expansion area. Use the format shown below for your code.

```
DATAVIEW dataview-name.
02  FILLER            PIC X(nnn).
```

Step 4

Code the RECONSTRUCT Statement using the format shown below. Then code a FOR EACH Statement to access the records from the CA-DATACOM/DB or CA-DATACOM/PC database. Immediately following the RECONSTRUCT Statement, be sure to include logic for checking the RETURN-CODE, as indicated below.

```
RECONSTRUCT REQUEST QUALIFICATION AREA
      FOR DATAVIEW dataview-name
      USING QUALIFICATION EXPRESSION FOUND IN dataname
.
IF RETURN-CODE IS GREATER THAN 0 THEN...
.
.
.
FOR EACH dataview-name...
```

A RETURN-CODE of zero indicates that the Dynamic RQA was successfully created. If the RETURN-CODE is not zero, refer to Section F.4 for diagnostic codes.

Refer to Section F.5 for a complete example of how to use the Dynamic RQA.

F.4 Verifying the Status of Dynamic RQA

At runtime, the Dynamic RQA Resolution Utility will scan the QSI and build the corresponding CA-DATACOM/DB or CA-DATACOM/PC Request Qualification Area. It also passes back to the caller the following information:

- a RETURN-CODE in the COBOL RETURN-CODE special register
- an activity status code
- the QSI parsing pointer value

The activity status code is found in a field defined by the RECONSTRUCT Statement as:

```
02 dataview-DYN-RQA-CODE PIC S9(04) COMP.
```

An additional value, the QSI parsing pointer value, is passed back to the program in a field defined by the RECONSTRUCT Statement as:

```
02 dataview-DYN-RQA-PNTR PIC S9(04) COMP.
```

RETURN-CODE

The RETURN-CODE is either 00 or 08.

00	indicates that the dynamic RQA has been succecssfully built. The Activity Status Code and Parsing Pointer Value will also have values of 00.
08	indicates that the Dynamic RQA Utility did not reconstruct the RQA. The first four bytes of the RQA are set to '0004' to guarantee its failure. (CA-DATACOM/DB fails the request with a CBS error code 91, subcode 98.) This safeguard is provided in case the activity status code is not checked and the FOR Statement is allowed to execute. Check the Activity Code and Parsing Pointer Value to help correct the error.

Activity Status Code

The activity status code will be 00 when the return code is 00. It otherwise will contain a value between 01 and 4095.

00	indicates that the return code is 00, and that a complete RQA has been built.

All Other Codes

The following activity codes found in dataview-DYN-RQA-CODE indicate problems resolving the RQA.

Parsing Problems

4012 - Item found is not a valid QSI Clause
4011 - No clauses found (QSI is all blank)
4009 - Where Duplicate WHERE Clause
4008 - Order Duplicate ORDER Clause
4005 - Where No WHERE Clause initiating left parenthesis
4004 - Order No ORDER Clause initiating left parenthesis

38nn - Where Specified subject exceeds 30 characters
37nn - Where Specified DATAVIEW object exceeds 30 characters
36nn - Where String object starting quote not found
34nn - Where Invalid relational operator
33nn - Where Invalid conjunction
32nn - Order Specified order-by field GT 30 characters
31nn - Order Invalid ascending/descending indicator

29nn - Where WHERE Clause length exceeds QSI at conjunction
28nn - Where WHERE Clause length exceeds QSI at object
27nn - Where WHERE Clause length exceeds QSI at relat operator
26nn - Where WHERE Clause length exceeds QSI at subject
25nn - Order ORDER Clause length exceeds QSI at nth field

Resolution Problems

19nn - Order order-by field not in DATAVIEW table
18nn - Where subject field not in DATAVIEW table
17nn - Where DATAVIEW object field not in DATAVIEW table
16nn - Where Literal object length not same as subject
14nn - Where String object length exceeds subject
13nn - Where String object specified has length 0
09nn - RQA size exceeded on this predicate

QSI Parsing Pointer Value

The QSI parsing pointer value will have a value of 00 when the return code and activity code are also 00, but otherwise it indicates the current character position within the QSI that was active at the time of a diagnostic. This can be helpful in debugging.

F.5 Sample Dynamic RQA

Objective 1

Code a program that uses a Dynamic RQA to search the employee database for employees who live in Houston, TX.

Step 1

Define a data-name containing a QSI. Call the data-name PERSONNEL-SELECTION-CRITERIA.

```
01    PERSONNEL-SELECTION-CRITERIA          PIC X(80).
```

which is to have a value of:

```
WHERE(EM-STATE-ADDRESS='TX'&EM-CITY-ADDRESS='HOUSTON~~bbbbbbbb~~')
```

Note that only essential blank spaces (~~b~~) are included. These blank spaces are essential because the defined length of EM-CITY-ADDRESS is 15 characters.

Step 2

Define a DATAVIEW named "PERSONNEL" in your program and provide an RQA expansion area. Use the format shown below for your code.

```
      DATAVIEW PERSONNEL.
      02  FILLER                          PIC X(200).
```

Step 3

Code the RECONSTRUCT Statement and the FOR EACH Statement below. The fields in the FOR EACH Statement are defined by a DATAVIEW in CA-DATACOM/DB or CA-DATACOM/PC. Immediately following the RECONSTRUCT Statement, be sure to include logic for checking the RETURN-CODE, as indicated below.

```
RECONSTRUCT REQUEST QUALIFICATION AREA
      FOR DATAVIEW PERSONNEL
      USING QUALIFICATION EXPRESSION
      FOUND IN PERSONNEL-SELECTION-CRITERIA
.
IF RETURN CODE IS GREATER THAN 0 THEN...
.
.
FOR EACH PERSONNEL
     DISPLAY EM-IDENTIFICATION-NUMBER
             EM-IDENTIFICATION-NAME
             EM-STREET-ADDRESS
             EM-CITY-ADDRESS
             EM-STATE-ADDRESS
             EM-ZIP-CODE-LOC
```

A RETURN-CODE of zero indicates that the Dynamic RQA was successfully created. If the RETURN-CODE is not zero, refer to Section F.4 for diagnostic codes.

Objective 2

Modify the code above to use a Dynamic RQA to search the employee database for employees who live in either Massachusetts or Georgia. Retrieve the records in descending alphabetical order according to employee last name.

Only one modification to the code is necessary. Redefine the value of *data-name* to be:

```
WHERE(EM-STATE-ADDRESS='MA'|EM-STATE-ADDRESS='GA')ORDER((D)EM-IDENTIFICATION-NAME)
```

Note that we added the OR indicator (|), the ORDER Clause, and the descending (D) code.

Index