CA IDMS - 19.0
Using CA IDMS DC Sort

Date: 15-Jan-2018
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time. This Documentation is proprietary information of CA and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2017 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
# Table of Contents

CA IDMS/DC Sort Overview ................................................................. 8

CA IDMS/DC Sort Examples ............................................................... 8
TPSEXPL1 .......................................................................................... 8
TPSEXPL2 .......................................................................................... 9
TPSEXPL3 .......................................................................................... 9
TPSEXPL4 .......................................................................................... 9
Selecting Sort Criteria on a User Screen ............................................. 40
Specifying Sequence and Sort Order .................................................. 41
Sample Sort Selection Screen .............................................................. 41
Field Error ......................................................................................... 42
Expanding Short Form Field Error Messages ....................................... 42
Processing Errors .............................................................................. 43
CA ADS Preprocessor ........................................................................ 43
CA ADS Preprocessor 1 ...................................................................... 44
  Step 1 -- Add CA IDMS/DC Sort Statements to Modules ................. 44
  Step 2 -- Execute the Preprocessor for Each Module ...................... 44
  Step 3 -- Execute the Dialog Generator ......................................... 45
  Step 4 -- Execute the Dialog ....................................................... 45
Preprocess Multiple Modules ............................................................. 45
System Flow ....................................................................................... 46
EDITOR ............................................................................................. 48
Key Settings ....................................................................................... 48
  ENTER Function ........................................................................... 48
  RESHOW Function ....................................................................... 48

CA ADS ............................................................................................... 49

CA IDMS/DC Sort parameters Overview ............................................. 52
CA IDMS/DC Sort Parameter Statements ............................................ 52
  Parameter Options ......................................................................... 52
  SETSORT Statement .................................................................... 52
Using CA IDMS DC Sort

This section provides examples of application programs that use CA IDMS/DC Sort. Each example is presented in four formats -- COBOL, Assembler, PLI, and CA ADS. The selection screens that appear when the USER parameter is selected in the SETSORT statement are also illustrated.

CA IDMS/DC Sort can be used in several ways. It can:

- Perform a single sort
- Perform multiple sorts
- Use predefined criteria
- Use criteria set by the user at runtime
- Operate within a single task
- Operate within multiple tasks (pseudo-conversational)

For more information, see the following topics:

- CA IDMS/DC Sort Overview (see page 8)
- CA IDMS/DC Sort Examples (see page 8)
- CA ADS (see page 49)
- CA IDMS/DC Sort parameters Overview (see page 52)
- CA IDMS/DC Sort System Flow (see page 53)
- COBOL/Assembler/PLI (see page 56)
- Customizing CA IDMS/DC Sort (see page 57)
- Notation Conventions and Syntax Rules (see page 59)
- Demonstration (see page 70)
- Parameter Statements Make CA IDMS/DC Sort Easy to Use (see page 73)
- Storage Requirements (see page 74)
- System Limits (see page 74)
CA IDMS/DC Sort Overview

This section describes operational procedures for CA IDMS/DC Sort. It begins with operational considerations, system flow, and system limits. Next are the steps (including model JCL) necessary to use CA IDMS/DC Sort with COBOL, Assembler, or PLI applications. Finally, this section discusses tuning CA IDMS/DC Sort for your environment. For information on using CA IDMS/DC Sort with CA ADS, see CA ADS Preprocessor (see page 43).

CA IDMS/DC Sort Examples

The examples in this section illustrate some of the ways that you can use CA IDMS/DC Sort. Each example is presented in the four languages supported by CA IDMS/DC Sort - COBOL Assembler, PLI, and CA ADS. Highlighted in the examples are the statements used to:

- Copy the control block
- Specify CA IDMS/DC Sort parameters: SETSORT, PUTSORT, GETSORT, ENDSORT, and SETLIMIT
- Check the CA IDMS/DC Sort return code (TPSRETN)
- Issue error messages when appropriate (TPSMG)
- TPSEXPL1 (see page 8)
- TPSEXPL2 (see page 9)
- TPSEXPL3 (see page 9)
- TPSEXPL4 (see page 9)
- Selecting Sort Criteria on a User Screen (see page 40)
- Specifying Sequence and Sort Order (see page 41)
- Sample Sort Selection Screen (see page 41)
- Field Error (see page 42)
- Expanding Short Form Field Error Messages (see page 42)
- Processing Errors (see page 43)
- CA ADS Preprocessor (see page 43)

TPSEXPL1

The first example uses CA IDMS/DC Sort to sort salespersons by sales volume and then to display the top five and bottom five salespersons.

Exhibit 4.2 COB is the COBOL version, Exhibit 4.3 ASM is the Assembler version, Exhibit 4.4 PLI is the PLI version, and Exhibit 4.5 ADS, the CA ADS version.
TPSEXPL2

The second example uses CA IDMS/DC Sort to sort accumulated employee sick leave in descending order and employee personal leave in descending order.

Exhibit 4.6 COB is the COBOL version, which also demonstrates the use of the IDMS option. Exhibit 4.7 ASM is the Assembler version, Exhibit 4.8 PLI is the PLI version, and Exhibit 4.9 CA ADS is the CA ADS version.

TPSEXPL3

The third example shows the use of CA IDMS/DC Sort in a pseudo-conversational mode. It sorts a given salesperson's sales by sales item and date.

Exhibit 4.10 COB is the COBOL version, Exhibit 4.11 ASM is the Assembler version, Exhibit 4.12 PLI is the PLI version, and Exhibit 4.13 ADS, the CA ADS version.

TPSEXPL4

The fourth example is similar to the third. In this example, the sort criteria are defined by the user at runtime. The user selection screens are illustrated with this example.

See Exhibit 4.1 for a table showing the features used in each example.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TPSEXPL1</td>
<td>salesperson</td>
<td>sales</td>
<td>top 5 and volume ascending</td>
<td>3.2 COB COBOL NEXT, PRIOR, 3.3 ASM Assembler FIRST, 3.4 PLI PLI LAST</td>
<td></td>
</tr>
<tr>
<td>TPSEXPL2</td>
<td>employees</td>
<td>accumulated sick leave descending</td>
<td>top 10</td>
<td>3.6 COB COBOL multiple sessions, 3.7 ASM Assembler FIRST, 3.8 PLI PLI NEXT</td>
<td></td>
</tr>
<tr>
<td>TPSEXPL3</td>
<td>sales data for a salesperson</td>
<td>item-name and date sold</td>
<td>20 items at a time</td>
<td>3.10 COB COBOL pseudo converse, 3.11 ASM Assembler, 3.12 PLI PLI, 3.13 ADS CA-ADS</td>
<td></td>
</tr>
<tr>
<td>TPSEXPL4</td>
<td>sales data for a salesperson</td>
<td>sales item data</td>
<td>20 items at a time</td>
<td>3.14 COB COBOL pseudo converse USER, 3.15 ASM Assembler, 3.16 PLI PLI, 3.17 ADS CA-ADS</td>
<td></td>
</tr>
</tbody>
</table>

*User selection screens are illustrated with this example.*

Exhibit 4.1: Table of Sort Examples
IDENTIFICATION DIVISION.
PROGRAM-ID.    TPSEXPL1.
REMARKS.    THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
DISPLAY THE TOP 5 AND BOTTOM 5 SALES PEOPLE IN A COMPANY USING A SINGLE
SORT WITHOUT READING THE SALES PEOPLE IN THE MIDDLE OF THE SORTED FILE.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
77    SALES-COUNT                 PIC S9(9) COMP.
77    END-OF-SALES               PIC X.
01    SALES-DATA.
   05    SALES-PERSON            PIC X(25).
   05    SALES-YTD               PIC S(9)V99 COMP-3.
COPY SALESREC.
COPY TPSCOMMC.
.
PROCEDURE DIVISION.
   PERFORM 0100-SORT-SALES.
   PERFORM 0200-DISPLAY-TOP-5-BOTTOM-5.

   ...return to CA IDMS/DC.
******************************************************************************
*                     SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE       *
*                     SALES. NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE    *
*                     SALES DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A *
*                     WORK RECORD FOR SORTING EFFICIENCY.                      *
******************************************************************************
0100-SORT-SALES    SECTION.
   SETSORT PROGRAM
   FOR SALES-DATA LENGTH 31
   FIELD SALES-YTD 6
   ASCENDING.
   IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
   MOVE 'N' END-OF-SALES.
   PERFORM 0150-PUT-SORT UNTIL END-OF-SALES = 'Y'.
   SECTION-EXIT.
   EXIT.
0150-PUT-SORT    SECTION.
   .
   ...read a sales record, set END-OF-SALES to 'Y' at end.
   IF END-OF-SALES = 'N'
   THEN
      MOVE SALESREC-SALES-PERSON TO SALES-PERSON
      MOVE SALESREC-SALES-YTD TO SALES-YTD
      PUTSORT;
      IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
   SECTION-EXIT.
   EXIT.
******************************************************************************
*                     GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY      *
*                     THEIR NAME AND YEAR TO DATE SALES.                       *
******************************************************************************
0200-DISPLAY-TOP-5-BOTTOM-5    SECTION.
   GETSORT LAST.
   IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
PERFORM 0220-GET-TOP-SALES VARYING SALES-COUNT
   FROM 1 BY 1 UNTIL SALES-COUNT > 5.

GETSORT FIRST.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
PERFORM 0240-GET-BOTTOM-SALES VARYING SALES-COUNT
   FROM 1 BY 1 UNTIL SALES-COUNT > 5.
ENDSORT.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

...display map

SECTION-EXIT.
EXIT.

0220-GET-TOP-SALES SECTION.

   MOVE SALES-PERSON TO ...map.
   MOVE SALES-YTD   TO ...map.

   GETSORT PRIOR.
   IF TPSRETN - '0000'
      THEN
         NEXT SENTENCE
   ELSE
      IF TPSRETN = '7020'
         THEN
            MOVE 5 TO SALES-COUNT
      ELSE
         PERFORM 9999-SORT-ERROR.
   ENDIF.

SECTION-EXIT.
EXIT.

0220-GET-BOTTOM-SALES SECTION.

   MOVE SALES-PERSON TO ...map.
   MOVE SALES-YTD   TO ...map>

   GETSORT NEXT.
   IF TPSRETN = '0000'
      THEN
         NEXT SENTENCE
   ELSE
      IF TPSRETN = '7020'
         THEN
            MOVE 5 TO SALES-COUNT
      ELSE
         PERFORM 9999-SORT-ERROR
   ENDIF.

SECTION-EXIT.
EXIT.

******************************************************************************
*             AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT    *
*             IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE  *
*             FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.           *
******************************************************************************

9999-SORT-ERROR SECTION.

   MOVE TPSMSG TO ...message line in map
   ...display map
   ...return to CA IDMS/DC

SECTION-EXIT.
EXIT.
Exhibit 4.2: COB FIRST, NEXT, LAST, PRIOR--COBOL

TITLE 'TPSEXPL1 -- 5 TOP AND BOTTOM SALES PERSONS'
***********************************************************************
* THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF TP/SORT TO DISPLAY   *
* THE TOP 5 AND BOTTOM 5 SALES PERSONS IN A COMPANY USING A SINGLE     *
* SORT WITHOUT READING THE SALES PERSONS IN THE MIDDLE OF THE        *
* SORTED FILE.                                                      *
***********************************************************************

name... DSECT
SLDATA DS 0XL31 EXTRACTED DATA FROM SALES RECORD
SLPERSON DS CL25 NAME OF SALES PERSON
SLYTD DS PL6 YTD SALES FOR SALES PERSON
END DS C END OF SALES INDICATOR

COPY SALESREC SALES RECORD
COPY TPSCOMMA TP/SORT COMMUNICATIONS BLOCK

R3 EQU 3 BAL - SUBROUTINE LINKAGE
R4 EQU 4 BCT - LOOP COUNTER

TPSEXPL1 CSECT

BAL R3,SORTSALE EXTRACT AND SORT SALES DATA
BAL R3,DISPLAY DISPLAY 5 TOP AND BOTTOM SALES PERSON

return to CICS or IDMS-DC
SPACE 2
***********************************************************************
* SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.         *
* NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA        *
* NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD FOR      *
* SORTING EFFICIENCY.                                                *
***********************************************************************

SORTSALE EQU *
SETSORT PROGRAM
FOR SLDATA LENGTH 31
FIELD SLYTD 6 ASCENDING.
CLC TPSRETN,=CL4'0000' SUCCESSFUL SETSORT ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVI END,C'N' INITIALIZE FOR LOOP

PUTLOOP EQU *

read a sales record, set END to 'Y' at end

CLI END,'Y' ANY MORE SALES PERSONS?
BER R3 NO, RETURN
MVC SLPerson,... SAVE SALES PERSON FOR SORT/DISPL
ZAP SLYTD,... SAVE YTD SALES FOR SORT/DISPLAY
PUTSORT.
CLC TPSRETN,=CL4'0000' SALES DATA ACCEPTED BY SORT ?
BE PUTLOOP YES, CONTINUE EXTRACTION
B BADSORT NO, REPORT ERROR AND ABORT

SPACE 2
***********************************************************************
* GET TOP 5 AND BOTTOM 5 SALES PERSONS AND DISPLAY THEIR NAME        *
* AND YEAR TO DATE SALES.                                           *
*--------------------------------------------------------------------*
DISPLAY EQU *

GETSORT LAST.
LA R4,5 NBR PERSONS TO GET FROM TP/SORT
B MAPTOP

TOPLOOP EQU *
GETSORT PRIOR.

MAPTOP EQU *

CLC TPSRETN,=CL4'7020' END OF SORTED DATA?
BE BOTTOM5 YES, GO GET BOTTOM 5
CLC TPSRETN,=CL4'0000' SORTED SALES DATA RETRIEVED ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVC ..MAP..,SLPERSON PUT SALES PERSON NAME IN MAP
UNPK ..MAP..,SLYTD PUT YTD SALES IN MAP
BCT R4,TOLOOP
SPACE
*-------------------- GET BOTTOM 5 SALES PERSONS ---------------------*
BOTTOM5 EQU *
GETSORT FIRST.
LA R4,5 NBR PERSONS TO GET FROM TP/SORT
B MAPBOT
BOTLOOP EQU *
GETSORT NEXT.
MAPBOT EQU *
CLC TPSRETN,=CL4'7020' END OF SORTED DATA ?
BE TERMSORT YES, GO END THE SORT SESSION
CLC TPSRETN,=CL4'0000' SORTED SALES DATA RETRIEVED ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVC ..MAP..,SLPERSON PUT SALES PERSON NAME IN MAP
MVC ..MAP..,SLYTD PUT YTD SALES IN MAP
BCT R4,BOTLOOP
SPACE
*-------------------- END SORT SESSION -------------------------------*
TERMSORT EQU *
ENDSORT.
CLC TPSRETN,=CL4'0000' SORT SESSION ENDED OK ?
BNE BADSORT NO, REPORT ERROR AND ABORT
SPACE
*-------------------- DISPLAY MAP -----------------------------------*
DSPLYMAP EQU *
.
.display map
BR R3
SPACE 2
***********************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY TP/SORT IN THE       *
* TPSRETN FIELD.  TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE       *
* FROM TP/SORT DESCRIBING THE BAD RETURN CODE.                       *
***********************************************************************
BADSORT EQU *
MVC ..MAP..,TPSMSG USE MESSAGE FROM TP/SORT
.
.display map
.
.return to CICS or IDMS-DC

Exhibit 4.3: ASM FIRST, NEXT, LAST--Assembler

TPSEXPL1: PROC OPTIONS(MAIN) REORDER;
/* REMARKS. THIS PLI EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
DISPLAY THE TOP 5 AND BOTTOM 5 SALES PEOPLE IN A COMPANY USING
A SINGLE SORT WITHOUT READING THE SALES PEOPLE IN THE MIDDLE
OF THE SORTED FILE.

CA IDMS/DC SORT REQUIRES COMPILIE OPTION "MARGINS(2,72)".
*/
/*REQURED FOR IDMS*/
DCL MODE (IDMS DC) DEBUG;
DCL IDMS ENTRY OPTIONS( INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);
/*END OF IDMS REQUIREMENT*/
DCL ADDR BUILTIN;
DCL SALES_COUNT FIXED BIN(31);
DCL END OF SALES CHAR(1);
DCL 1 SALES DATA,
2 SALES_PERSON CHAR(25),
2 SALES_YTD PIC'S99999V99';

%INCLUDE SALESREC;
%INCLUDE TPSCOMMPP;

CALL SORT_SALES_0100;
CALL DISPLAY_TOP_5_BOTTOM_5_0200;

...return to CA IDMS/DC;

****************************************************************************/
* SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.            *
* NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA            *
* NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD            *
* FOR SORTING EFFICIENCY.                                              *
****************************************************************************/

SORT_SALES_0100: PROC;
SETSORT PROGRAM
FOR SALES_DATALENGTH 31
FIELD SALES_YTD 7
ASCENDING;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END_OF_SALES = 'N';
DO UNTIL (END_OF_SALES = 'Y');
   CALL PUT_SORT_0150;
END;
END SORT_SALES_0100;

PUT_SORT_0150: PROC;
.
.
...read a sales record, set END_OF_SALE to 'Y' at end.
.
IF (END_OF_SALES = 'N')
THEN DO:
   SALES_PERSON = SALESREC_SALES_PERSON;
   SALES_YTD = SALESREC_SALES_YTD;
   PUTSORT;
   IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END;
END PUT_SORT_0150;

****************************************************************************/
* GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY THEIR              *
* NAME AND YEAR TO DATE SALES.                                         *
****************************************************************************/

DISPLAY_TOP_5_BOTTOM_5_0200: PROC;
GETSORT LAST;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
DO SALES_COUNT = 1 TO 5 BY 1;
   CALL GET_TOP_SALES_0220;
END;

GETSORT FIRST;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
DO SALES_COUNT = 1 TO 5 BY 1;
   CALL GET_BOTTOM_SALES_0240;
END;
ENDSORT;
IF(TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
.
.
get_top_sales_0220: proc;
    ...map = sales_person;
    ...map = sales_ytd;
    getsort prior;
    if (tpsretn = '0000')
        then
        else
            if (tpsretn = '7020')
                then
                    sales_count = 5; /* terminate do_loop */
                    else
                        call sort_error_9999;
        end if;
    end if;
end get_top_sales_0220;

get_bottom_sales_0240: proc;
    ...map = sales_person;
    ...map = sales_ytd;
    getsort next;
    if (tpsretn = '0000')
        then
        else
            if (tpsretn = '7020')
                then
                    sales_count = 5; /* terminate do_loop */
                    else
                        call sort_error_9999;
        end if;
    end if;
end get_bottom_sales_0240;

sort_error_9999: proc;
    ...message line in map = tpsmsg;
    ...display map
    ...return to ca idms/dc
end sort_error_9999;
end tpsexpl1;
DISPLAY.
!    RETURN TO TOP.
!
!*********************************************************************************************
!*    SORT SALES PEOPLE IN ASCENDING ORDER BY YEAR TO DATE SALES.  *
!*    NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA *
!*    NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO WORK RECORD *
!*    FOR SORTING EFFICIENCY.                                       *
!**********************************************************************************************

DEFINE SUBROUTINE SORT-SLS.
SETSORT PROGRAM FOR SALES-DATA
FIELD SALES-YTD 6 ASCENDING
IF TPSRETN NE ZERO
    CALL ERROR
MOVE 'N' TO END-OF-SALES.
WHILE END-OF-SALES NE 'Y'
    REPEAT.
    CALL PUT-SORT.
END
GOBACK.

DEFINE SUBROUTINE PUT-SORT.
...
....obtain a sales record, set END-OF-SALES to 'Y' at end.
...
IF END-OF-SALES = 'N'
DO.
    MOVE SALESREC-SALES-PERSON TO SALES-PERSON.
    MOVE SALESREC-SALES-YTD TO SALES-YTD.
    PUTSORT.
    IF TPSRETN NE ZERO
        DO.
            CALL ERROR.
    !*********************************************************************************************
   !*    DISPLAY-TOP-5-BOTTOM-5                                                                 *
   !*    GET THE TOP 5 AND BOTTOM 5 SALES PEOPLE AND DISPLAY                                 *
   !*    THEIR NAMES AND YEAR TO DATE SALES.                                                *
   !**********************************************************************************************

DEFINE SUBROUTINE DTOPBOT.
GETSORT LAST.
IF TPSRETN NE ZERO
    CALL ERROR.
MOVE 1 TO SALES-COUNT.
WHILE SALES-COUNT < 6
    REPEAT.
        CALL TOP-SLS.
        ADD 1 TO SALES-COUNT.
END.
GETSORT FIRST.
IF TPSRETN NE ZERO
    CALL ERROR.
MOVE 1 TO SALES-COUNT.
WHILE SALES-COUNT < 6
    REPEAT.
        CALL BOTSLS.
        ADD 1 TO SALES-COUNT.
END.
ENDSORT.
IF TPSRETN NE ZERO
    CALL ERROR.
GOBACK.
DEFINE SUBROUTINE TOPSLS.
MOVE SALES-PERSON TO ...map.
MOVE SALES-YTD TO ...map.
GETSORT PRIOR.
IF TPSRETN = '7020'
   MOVE 5 TO SALES-COUNT
ELSE
   IF TPSRETN NE ZERO
      CALL ERROR.
GOBACK.
DEFINE SUBROUTINE BOTSLS.
MOVE SALES-PERSON TO ...map.
MOVE SALES-YTD TO ...map.
GETSORT NEXT.
IF TPSRETN = '7020'
   MOVE 5 TO SALES-COUNT
ELSE
   IF TPSRETN NE ZERO
      CALL ERROR.
GOBACK.

************************************************************************
!*SORT-ERROR
!* * AN UNANTICIPATED RETURN CODE WAS RETURNED BY
!* * CA IDMS/DC SORT IN THE TPSRETN FIELD. TPSMSG CONTAINS
!* * A 79 CHARACTER MESSAGE FROM CA IDMS/DC SORT DESCRIBING
!* * THE BAD RETURN CODE.
************************************************************************

DEFINE SUBROUTINE ERROR.
DISPLAY MESSAGE TEXT TPSMSG.
! RETURN TO TOP.
GOBACK.

Exhibit 4.5: ADS FIRST, NEXT, LAST, PRIOR--ADS

IDENTIFICATION DIVISION.
PROGRAM-ID. TPSEXPL2.
REMARKS. THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT
TO EXECUTE TWO SORTS CONCURRENTLY. ONE ON ACCUMULATED SICK LEAVE
IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN DESCENDING
ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED.

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 END-OF-EMPLOYEES PIC X.
77 EMPLOYEE-COUNT PIC 9(2) COMP.3.
77 MAX-EMPLY-IN-MAP PIC 9(2) COMP.3 VALUE TO.
COPY IDMS EMPLYREC VER 22.
COPY TPSCOMMC.

PROCEDURE DIVISION.
PERFORM 0100-SORT-EMPLOYEES.
PERFORM 0200-DISPLAY-TOP-TEN.
...return to CA IDMS/DC.

************************************************************************
* EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON ACCUMULATED
* SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL TIME. NOTE: THE*
* RECORD BEING SORTED IS AN IDMS RECORD. THE ELEMENT ATTRIBUTES DO*
* NOT HAVE TO BE CODED IN THE SETSORT STATEMENT, THEY WILL BE      *
* EXTRACTED BY CA IDMS/DC SORT FROM THE DICTIONARY.                    *
**************************************************************************
0100-SORT-EMPLOYEES        SECTION.

SETSORT SESSION 1 PROGRAM IDMS
FOR EMPLYREC VER 22
IN DICT TEST
FIELD EMPLY-SICK-DAYS DESCENDING
EMPLY-NAME ASCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SETSORT SESSION 2 PROGRAM IDMS
FOR EMPLYREC VER 22
IN DICT TEST
FIELD EMPLY-PERSONAL-DAYS DESCENDING
EMPLY-NAME ASCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

MOVE 'N' END-OF-EMPLOYEES.
PERFORM 0150-PUT-SORT UNTIL END-OF-EMPLOYEES = 'Y'.
SECTION-EXIT.
EXIT.

0150-PUT-SORT        SECTION.

...read an employee record, set END-OF-EMPLOYEES to 'Y' at end.

IF END-OF-EMPLOYEES = 'N'
THEN
  PUTSORT SESSION 1.
  IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
  PUTSORT SESSION 2.
  IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
SECTION-EXIT.
EXIT.
**************************************************************************
* GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS.     *
* DISPLAY THE EMPLOYEE NAME AND TIME TAKEN.                        *
**************************************************************************
0200-DISPLAY-TOP-TEN        SECTION.

PERFORM 0220-GET-EMPLOYEE VARYING EMPLOYEE-COUNT
  FROM 1 BY 1 UNTIL EMPLOYEE-COUNT > MAX-EMPLY-IN-MAP.
ENDSORT SESSION 1.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
ENDSORT SESSION 2.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

...display map

SECTION-EXIT.
EXIT.

0200-GET-EMPLOYEE        SECTION.

GETSORT SESSION 1 NEXT.
IF TPSRETN = '7020'
THEN
  MOVE MAX-EMPLY-IN-MAY TO EMPLOYEE-COUNT
ELSE
  IF TPSRETN NOT = '0000'
    THEN
      PERFORM 9999-SORT-ERROR
    ELSE
      MOVE EMPLY-NAME TO ...map.
      MOVE EMPLY-PERSONAL-DAYS TO ...map.
  SECTION-EXIT.
EXIT.

**********************************************************************
*          AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN* *
*          THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE FROM* *
*          CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.             *
**********************************************************************

9999-SORT-ERROR SECTION.
MOVE TPSMSG TO ...message line in map
  ...display map
  ...return to CA IDMS/DC
SECTION-EXIT.
EXIT.

Exhibit 4.6: COB Multiple Sessions--COBOL

TITLE 'TPSEXPL2  2 CONCURRENT SORTS'
**********************************************************************
*          THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT * *
*          TO EXECUTE TWO SORTS CONCURRENTLY. ONE ON ACCUMULATED SICK   * *
*          LEAVE IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN    * *
*          DESCENDING ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED.* *
**********************************************************************
name... DSECT
  @COPY IDMS,RECORD=EMPLYREC,VERSION=22

  END DS C END OF SALES INDICATOR
  .
  COPY TPSCOMMA CA IDMS/DC SORT COMMUNICATIONS BLOCK
  .
  R3 EQU 3 BAL - SUBROUTINE LINKAGE
  R4 EQU 4 BCT - LOOP COUNTER

TPSEXPL1 CSECT
  .
  BAL R3.
SORTEMPL EXTRACT AND SORT SICK & PERSONAL TIME
  BAL R3,DISPLAY DISPLAY TOP TEN IN EACH CATEGORY
  .
  return to CA IDMS/DC
SPACE 2
**********************************************************************
*          EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON ACCUMULATED * *
*          SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL TIME. NOTE: THE* *
*          RECORD BEING SORTED IS AN IDMS RECORD. THE ELEMENT ATTRIBUTES DO* *
*          NOT HAVE TO BE CODED IN THE SETSORT STATEMENT, THEY WILL BE    * *
*          EXTRACTED BY CA IDMS/DC SORT FROM THE DICTIONARY.              *
**********************************************************************
SORTEMPL EQU
  SETSORT SESSION 1 PROGRAM IDMS
  FOR EMLYREC VER 22
  IN DICT TEST
  FIEL EMLPSICK DESCENDING
  EMPNAME ASCENDING.
  CLC TPSRETN,=CL4'0000' SUCCESSFUL SETSORT ?
  BNE BADSORT NO, REPORT ERROR AND ABORT
SETSORT SESSION 2 PROGRAM IDMS
FOR EMPLYREC VER 22
IN DICT TEST
FIELD EMPPRSNL DESCENDING
EMPNAME ASCENDING.

CLC TPSRETN,=CL4'0000' SUCCESSFUL SETSORT ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVI END,C'N' INITIALIZE FOR LOOP
SPACE
PUTLOOP EQU *

* read an employee record, set END to 'Y' at end

CLI END,'Y' ANY MORE EMPLOYEES ?
BER R3 NO, RETURN
PUTSORT SESSION 1.

CLC TPSRETN,=CL4'0000' SICK LEAVE ACCEPTED BY SORT ?
BNE BADSORT NO, REPORT ERROR AND ABORT
PUTSORT SESSION 2.

CLC TPSRETN,=CL4'0000' PERSONAL TIME ACCEPTED BY SORT?
BE PUTLOOP YES, CONTINUE EXTRACTION
B BADSORT NO, REPORT ERROR AND ABORT
SPACE

******************************************************************************
* GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS. *
* DISPLAY THE EMPLOYEE NAME AND TIME TAKEN. *
******************************************************************************

DISPLAY EQU *

*---------------------GET TOP TEN EMPLOYEES IN EACH CATEGORY-----------------*
LA R4,10 NBR EMPLOYEES FOR DISPLAY

SPACE
LOOP EQU *

GETSORT SESSION 1 NEXT.

CLC TPSRETN,=CL4'7020' END OF SORTED DATA ?
BE TERMSORT YES, GO END THE SORT SESSION

CLC TPSRETN,=CL4'0000' SORTED SALES DATA RETRIEVED ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVC ..MAP..,EMPNAME PUT EMPLOYEE NAME IN MAP
UNPK ..MAP..,EMPSICK PUT SICK TIME IN MAP
SPACE

GETSORT SESSION 2 NEXT.

CLC TPSRETN,=CL4'0000' SORTED SALES DATA RETRIEVED ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVC ..MAP..,EMPNAME PUT EMPLOYEE NAME IN MAP
UNPK ..MAP..,EMPPRSNL PUT PERSONAL TIME IN MAP
BCT R4,LOOP
SPACE

*----------------------------- END SORT SESSION *

TERMSORT EQU *

ENDSORT SESSION 1.

CLC TPSRETN,=CL4'0000' SORT SESSION ENDED OK ?
BNE BADSORT NO, REPORT ERROR AND ABORT

ENDSORT SESSION 2.

CLC TPSRETN,=CL4'0000' SORT SESSION ENDED OK ?
BNE BADSORT NO, REPORT ERROR AND ABORT

SPACE

*----------------------------- DISPLAY MAP *

DSPLYMAP EQU *

* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN *
* THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE* *

******************************************************************************

BADSORT EQU *
MVC..MAP..,TPMSG

USE MESSAGE FROM CA IDMS

/DC SORT

display map

return to CA IDMS/DC

Exhibit 4.7: ASM Multiple Sessions--Assembler

TPEXPL2: PROC OPTIONS(MAIN) REORDER;

/* REMARKS. THIS PL1 EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO EXECUTE TWO SORTS CONCURRENTLY. ONE ON ACCUMULATED SICK LEAVE IN DESCENDING ORDER AND ONE ON PERSONAL TIME USED IN DESCENDING ORDER. THE TOP TEN IN EACH CATEGORY ARE DISPLAYED. TPSORT REQUIRED COMPILe OPTION "MARGINS(2,72)". */

/*REQUIRED FOR IDMS*/
DCL (subschema_name SUBSCHEMA,schema_name SCHEMA) MODE (IDMS DC) DEBUG;
DCL IDMS ENTRY OPTIONS(INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);

/*END OF IDMS REQUIREMENT*/
DCL ADDR BUILTIN;
DCL END_OF_EMPLOYEES CHAR(1);
DCL EMPLOYEE_COUNT PIC'S99';
DCL MAX_EMPLY_IN_MAP PIC'S99' INIT(10);

%INCLUDE TPSCOMMP;

CALL SORT_EMPLOYEES_0100;
CALL DISPLAY_TOP_TEN_0200;

...return to CA IDMS/DC.

*******************************************************************************
* EXECUTE TWO SORTS, BOTH IN DESCENDING ORDER--ONE ON ACCUMULATED SICK LEAVE AND THE OTHER ON ACCUMULATED PERSONAL TIME. NOTE: THE RECORD BEING SORTED IS AN IDMS RECORD. THE ELEMENT ATTRIBUTES DO NOT HAVE TO BE CODED IN THE SETSORT STATEMENT, THEY WILL BE EXTRACTED BY CA IDMS/DC SORT FROM THE DICTIONARY. THE RECORD DEFINITION "EMPLYREC" WILL BE INSERTED BY THE PLI_IDMS PREPROCESSOR THROUGH THE IDMS DCL. *
*******************************************************************************

SORT_EMPLOYEES_0100: PROC;

SETSORT SESSION 1 PROGRAM IDMS FOR EMPLYREC VER 22 IN DICT TEST
FIELD EMPLY_SICK_DAYS DESCENDING EMPLY_NAME ASCENDING;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

SETSORT SESSION 2 PROGRAM IDMS FOR EMPLYREC VER 22 IN DICT TEST
FIELD EMPLY_PERSONAL_DAYS DESCENDING EMPLY_NAME ASCENDING.
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

END_OF_EMPLOYEES = 'N';
DO UNTIL (END_OF_EMPLOYEES = 'Y');
CALL PUT_SORT_0150;
END;
END SORT_EMPLOYEES_0100;
PUT_SORT_0150: PROC;
  IF (END_OF_EMPLOYEES = 'N') THEN DO;
    PUTSORT SESSION 1;
    IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
    PUTSORT SESSION 2;
    IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
  END;
END PUT_SORT_0150;

*****************************************************************************
* GET THE TOP TEN FROM THE SICK LEAVE AND PERSONAL TIME SORTS. *
* DISPLAY THE EMPLOYEE NAME AND TIME TAKEN. *
*****************************************************************************
DISPLAY_TOP_TEN_0200: PROC;
DO EMPLOYEE_COUNT = 1 TO MAX_EMPLY_IN_MAP BY 1;
  CALL GET_EMPLOYEE_0220
END;
ENDSORT SESSION 1;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
ENDSORT SESSION 2;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;

GET_EMPLOYEE_0220: PROC;
GETSORT SESSION 1 NEXT;
  IF (TPSRETN = '7020') THEN
    EMPLOYEE_COUNT = MAX_EMPLY_IN_MAP; /*CLOSE DO_LOOP*/
  ELSE
    IF (TPSRETN = '0000') THEN
      CALL SORT_ERROR_9999;
    ELSE
      DO;
        ...map =EMPLY_NAME;
        ...map =EMPLY_SICK_DAYS;
        GETSORT SESSION 2 NEXT;
        IF (TPSRETN = '0000') THEN
          CALL SORT_ERROR_9999;
        ELSE
          DO;
            ...map =EMPLY_NAME;
            ...map =EMPLY_PERSONAL_DAYS;
          END;
      END;
  END;
END GET_EMPLOYEE_0220;

*****************************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT *
* IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE *
*****************************************************************************
SORT_ERROR_9999: PROC;
Exhibit 4.8: PLI Multiple Sessions--PLI

SETSORT SESSION 1 PROGRAM IDMS FOR EMPLOYEE-SICK PERSONAL
FIELDS EMPLOYEE-SICK-LEAVE DESCENDING
    EMPLOYEE-NAME ASCENDING.
WHILE (NOT DB-END-OF-SET) AND (TPSRETN EQUAL ZERO) REPEAT.

    OBTAIN NEXT EMPLOYEE WITHIN EMPLOYEE-MASTER.
    PUTSORT SESSION 1.
    IF TPSRETN NE ZERO DO.
        DISPLAY MESSAGE TEXT TPSMSG.
        RETURN TO TOP.
    END.

    PUTSORT SESSION 2.
    IF TPSRETN NE ZERO DO.
        DISPLAY MESSAGE TEXT TPSMSG.
        RETURN TO TOP.
    END.

GETSORT SESSION 1 FIRST.
    IF TPSRETN NE ZERO DO.
        DISPLAY MESSAGE TEXT TPSMSG.
        RETURN TO TOP.
    END.

MOVE ZERO TO MAP-FIELD-SUBSCRIPT.
WHILE (END-OF-SICK-LEAVE EQUAL 'N') AND (MAP-FIELD-SUBSCRIPT LE 10) REPEAT.

    MOVE SICK-LEAVE-MSG TO MAP-SICK-LEAVE-MSG (MAP-FIELD-SUBSCRIPT).
    MOVE EMPLOYEE-NAME TO MAP-EMPLOYEE-NAME (MAP-FIELD-SUBSCRIPT).
    MOVE EMPLOYEE-NUM TO MAP-EMPLOYEE-NUM (MAP-FIELD-SUBSCRIPT).
    MOVE EMPLOYEE-SICK-LEAVE TO MAP-EMPLOYEE-SICK-MSG (MAP-FIELD-SUBSCRIPT).
    ADD 1 TO MAP-FIELD-SUBSCRIPT.

GETSORT SESSION 1 NEXT.
    IF TPSRETN EQUAL '7020' DO.
        MOVE 'Y' TO END-OF-SICK-LEAVE.
        END.
    ELSE IF TPSRETN NE ZERO DO.
        DISPLAY MESSAGE TEXT TPSMSG.
        RETURN TO TOP.
    END.

    IF TPSRETN = ZERO CONTINUE IN ITERATION.
END.

GETSORT SESSION 2 FIRST.
IF TPSRETN NE ZERO
  DO.
    DISPLAY MESSAGE TEXT TPSMSG.
    !               RETURN TO TOP.
  END.
MOVE ZERO TO MAP-FIELD-SUBSCRIPT.
WHILE (END-OF-PERSONAL-LEAVE EQUAL 'N') AND
  (MAP-FIELD-SUBSCRIPT LE 10)
REPEAT.
  MOVE PERSONAL-LEAVE-MSG TO
    MAP-PERSONAL-LEAVE-MSG (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-NAME TO MAP-EMPLOYEE-NAME (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-NUM TO MAP-EMPLOYEE-NUM (MAP-FIELD-SUBSCRIPT).
  MOVE EMPLOYEE-PERSONAL-LEAVE TO
    MAP-EMPLOYEE-PERSONAL-LEAVE (MAP-FIELD-SUBSCRIPT).
  ADD 1 TO MAP-FIELD-SUBSCRIPT.
GETSORT SESSION 2 NEXT.
IF TPSRETN EQUAL '7020'
  DO.
    MOVE 'Y' TO END-OF-PERSONAL-LEAVE.
  END.
ELSE
  IF TPSRETN NE ZERO
    DO.
      DISPLAY MESSAGE TEXT TPSMSG.
      !               RETURN TO TOP.
    END.
    ! IF TPSRETN = ZERO CONTINUE IN ITERATION.
  END.
ENDSORT SESSION 1.
IF TPSRETN NE ZERO
  DO.
    DISPLAY MESSAGE TEXT TPSMSG.
    !               RETURN TO TOP.
  END.
ENDSORT SESSION 2.
IF TPSRETN NE ZERO
  DO.
    DISPLAY MESSAGE TEXT TPSMSG.
    !               RETURN TO TOP.
  END.
DISPLAY.

Exhibit 4.9: ADS Multiple Sessions--ADS

IDENTIFICATION DIVISION.
PROGRAM-ID.   TPSEXPL3.
REMARKS.     THIS COBOL EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO
             DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON. THIS PROGRAM IS
             PSEUDO CONVERSATIONAL.

ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
  77    DISPLAY-COUNT             PIC S9(9) COMP.
  77    END-OF-DISPLAY           PIC X.
  77    END-OF-SALES             PIC X.
  01   SALES-DATA.
  05   SALES-ITEM-NAME         PIC X(25).
  05   SALES-AMOUNT             PIC S9(9)V99 COMP-3.
  05   SALES-AMOUNT             PIC S9(9) COMP-3.
  05   SALES-DATE               PIC X(08).
COPY SALESREC.
COPY TPSCOMM.

PROCEDURE DIVISION.

IF ...first time
  ...set first time off
  PERFORM 0100-GET-SORTED-SALES-DATA.

MOVE 'N' TO END-OF-DISPLAY.
PERFORM DISPLAY-SALES-DATA.

IF END-OF-DISPLAY = 'Y'
  THEN
    PERFORM 9000-END-SORT
    ...return to CA IDMS/DC
  ELSE
    ...return to CA IDMS/DC with next task code
    for this program.

******************************************************************************
*                      SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE  *
*                      SOLD. NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES *
*                      DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK  *
*                      RECORD FOR SORTING EFFICIENCY.                             *
******************************************************************************

0100-GET-SORTED-SALES-DATA SECTION.

SETSORT PROGRAM
FOR SALES-DATA LENGTH 44
  FIELD SALES-ITEM-NAME 25 ASCENDING
  SALES-DATE 8 DESCENDING.
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
MOVE 'N' END-OF-SALES.
PERFORM 0150-PUT-SORT UNTIL END-OF-SALES = 'Y'.

SECTION-EXIT.
EXIT.

0150-PUT-SORT SECTION.
.
...read a sales record for the sales person,
  when all records have been read for sales person
  move 'Y' to END-OF-SALES
.

IF END-OF-SALES = 'N'
  THEN
    MOVE SALESREC-SALES-ITEM TO SALES-ITEM
    MOVE SALESREC-SALES-AMOUNT TO SALES-AMOUNT
    MOVE SALESREC-SALES-QTY TO SALES-QTY
    MOVE SALESREC-SALES-DATE TO SALES-DATE
    PUTSORT.
    IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.

SECTION-EXIT.
EXIT.

******************************************************************************
*                      DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY SALES PERSON.  *
******************************************************************************

0200-DISPLAY-SALES-DATA SECTION.

PERFORM 0220-GET-SALES-ITEM
  VARYING DISPLAY-COUNT
  FROM 1 BY 1 UNTIL (DISPLAY-COUNT > 20)
  OR (END-OF-DISPLAY = 'Y')
IF END-OF-DISPLAY = 'Y'
THEN
    MOVE 'NO MORE ITEMS FOR SALES PERSON'
    TO ...message in map.
ELSE
    MOVE 'MORE ITEMS FOLLOW FOR SALES PERSON'
    TO ...message in map.
.
...display map
.
SECTION-EXIT.
EXIT.
0220-GET-SALES-ITEM SECTION.

** CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE BETWEEN
** PSEUDO CONVERSES. THEREFORE, NO REPOSITIONING IS REQUIRED.

GETSORT NEXT.
IF TPSRETN = '0000'
THEN
    MOVE SALES-ITEM-NAME TO ...map
    MOVE SALES-AMOUNT TO ...map
    MOVE SALES-QtY TO ...map
    MOVE SALES-DATE TO ...map
ELSE
    IF TPSRETN = '7020'
    THEN
        MOVE 'Y' TO END-OF-DISPLAY
    ELSE
        PERFORM 9999-SORT-ERROR.
    END-ELSE.
END-ELSE.

SECTION-EXIT.
EXIT.
**************************************************************************
* END CURRENT SESSION OF CA IDMS/DC SORT.                              *
**************************************************************************

9000-END-SORT SECTION.

ENDSORT.
IF TPSRETN NOT = '0000' THEN PERFORM 9999-SORT-ERROR.

SECTION-EXIT.
EXIT.
**************************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT          *
* IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE          *
* FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.                  *
**************************************************************************

9999-SORT-ERROR SECTION.

MOVE TPSMSG TO ...message line in map
...display map
...return to CA IDMS/DC

SECTION-EXIT.
EXIT.

Exhibit 4.10: COB Pseudo Conversational--COBOL

TITLE 'TPSEXPL3 PSEUDO CONVERSATIONAL'
**************************************************************************
* THIS ASSEMBLER EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO      *
* DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON.  THIS PROGRAM IS*
/*  PSEUDO CONVERSATIONAL. */

* ************************************************* *
name... DSECT
SLDATA DS OXL44 EXTRACTED DATA FROM SALES REC
  D
SLITEM DS CL25 NAME OF ITEM SOLD
SLAMT DS PL6 AMOUNT ITEM SOLD FOR
SLAMT DS PL5 NUMBER OF ITEMS SOLD
SLDATE DS CL8 DATE ITEM WAS SOLD
ENDSALE DS C END OF SALES INDICATOR
ENDDSPLY DS C END OF DISPLAY INDICATOR

* COPY SALESREC SALES RECORD
* COPY TPSCOMMA CA IDMS/DC SORT COMMUNICATIONS BLOCK

R3 EQU 3 BAL - SUBROUTINE LINKAGE
R4 EQU 4 BCT - LOOP COUNTER

TPSEXPL3 CSECT

CLI ...first time IS THIS FIRST TIME ?
BNE MAIN0100 NO, SKIP EXTRACT AND SORT THEN
MVI ... first time TURN FIRST TIME INDICATOR
OFF
BAL R3,SORTSALE EXTRACT AND SORT SALES DATA
SPACE
MAIN0100 EQU
MVI ENDDSPLY,C'N' INITIALIZE FOR LOOP
BAL R3,DISPLAY DISPLAY SALES DATA
SPACE
CLI ENDDSPLY,C'Y' HAVE ALL SALES BEEN DISPLAYED ?
BE NOMORE YES, END SORT PSEUDO CONVERSE
...return to CA IDMS/DC with next code for this program
SPACE
NOMORE EQU *
BAL R3,TERMSORT RELEASE SORT
...return to CA IDMS/DC
SPACE 2

**************************************************************************
* SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE SOLD. *
* NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA *
* NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD FOR *
* SORTING EFFICIENCY. *
**************************************************************************

SORTSALE EQU *

SETSORT PROGRAM
FOR SLDATA LENGTH 44
FIELD SLITEM 25 ASCENDING
SLDATE 8 DESCENDING.

CLC TPSRETN,=CL4'0000' SUCCESSFUL SETSORT ?
BNE BADSORT NO, REPORT ERROR AND ABORT
MVI ENDDSLEY,C'N' INITIALIZE FOR LOOP

SPACE
EQU *

...read a sales record, set ENDSALEY to 'Y' at end

CLI ENDSALEY,'Y' ANY MORE SALES PERSONS ?
BER R3 NO, RETURN
MVC SLITEM,... SAVE NAME OF ITEM SOLD
ZAP SLAMT,... SAVE AMOUNT OF SALE
ZAP SLQTY,... SAVE QUANTITY SOLD
MVC SLDATE,... SAVE DATE ITEM SOLD

CLC TPSRETN,=CL4'0000' SALES DATA ACCEPT
ED BY SORT ?
BE PUTLOOP YES, CONTINUE EXTRACT
DISPLAY EQU *
LA    R4,20 NUMBER OF ITEMS PER SCREEN
SPACE
GETLOOP EQU *
GETSORT NEXT.

CLC    TPSRETN,=CL4'7020' END OF SORTED DATA ?
BE     LAST YES, INDICATE NO MORE ITEMS
CLC    TPSRETN,=CL4'0000' SORTED SALES DATA RETRIEVED ?
BNE    BADSORT NO, REPORT ERROR AND ABORT
MVC    ..map..,SLITEM NAME OF ITEM SOLD
UNPK   ..map..,SLAMT AMOUNT ITEM SOLD FOR
UNPK   ..map..,SLQTY NUMBER OF ITEMS SOLD
MVC    ..map..,SLDATE DATE ITEM SOLD
BCT     R4,GETLOOP
MVC    ..message in map..,MSGMORE
SPACE
DSPLYMAP EQU *
  ...display map
  .
BR     R3
SPACE
LAST EQU *
MVI     ENDDSPLY,C'Y' INDICATE LAST SCREEN OF DISPLAY
MVC    ..message in map..,MSGLAST
B      DSPLYMAP

SPACE 2
******************************************************************************
*                     END CURRENT SESSION OF CA IDMS/DC SORT.                   *
******************************************************************************
TERMSORT EQU *
ENDED.
CLC    TPSRETN,=CL4'0000' SESSION ENDED OK ?
BER    R3 YES
B      BADSORT NO
SPACE 2
******************************************************************************
*                  AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN *
*                THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER MESSAGE*  
******************************************************************************

BADSORT EQU *
MVC    ..map..,TPSMSG USE MESSAGE FROM CA IDMS/DC SORT
  ...display map
  ...return to CA IDMS/DC
SPACE 2

MSGMORE DC  CL79'MORE ITEMS FOLLOW FOR SALESPERSON'
MSGLAST  DC  CL79'NO MORE ITEMS FOR SALE PERSON'

Exhibit 4.11: ASM Pseudo Conversational--Assembler
CA IDMS/DC SORT REQUIRES COMPILe OPTION "MARGINS(2,72)".

/*
 * REQUIRED FOR IDMS*/
DCL MODE (IDMS_DC) DEBUG;
DCL IDMS ENTRY OPTIONS(INTER,ASM);
INCLUDE IDMS(SUBSCHEMA_CTRL);
/*END OF IDMS REQUIREMENT*/

DCL ADDR BUILTIN;
DCL DISPLAY_COUNT FIXED BIN(31);
DCL END_OF_DISPLAY CHAR(1);
DCL END_OF_SALES CHAR(1);
DCL 1 SALES_DATA,
  2 SALES_ITEM_NAME CHAR(25),
  2 SALES_AMOUNT PIC 'S999999999V99',
  2 SALES_QTY PIC 'S999999999',
  2 SALES_DATE CHAR(8);

%INCLUDE SALESREC;
%INCLUDE TPSCOMMP;
.
.
IF (...first time)
  THEN DO;
    ...set first time off;
    CALL GET_SORTED_SALES_DATA_0100;
  END;
END_OF_DISPLAY = 'N';
CALL DISPLAY_SALES_DATA_0200;
IF (END-OF-DISPLAY = 'Y')
  THEN DO;
    END_SORT_9000;
    ...return to CA IDMS/DC;
  END;
ELSE
  ...return to CA IDMS/DC with next task code for this program;
/
***************************************************************************
* SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE SOLD.*
* SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD FOR SORTING EFFICIENCY.*
***************************************************************************

GET_SORTED_SALES_DATA_0100: PROC;

SET SORT PROGRAM
  FOR SALES DATA LENGTH 53
  FIELD SALES ITEM NAME 25 ASCENDING
  SALES_DATE 8 DESCENDING.

IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END_OF_SALES = 'N';
DO UNTIL (END OF SALES = 'Y');
  CALL PUT_SORT_0150;
END;
END GET_SORTED_SALES_DATA_0100;

PUT_SORT_0150: PROC;
.
...read sales record for the sales person;
  when all records have been read for sales person
  END_OF_SALES = 'Y';
IF (END_OF_SALES = 'N') THEN DO;
    SALES_ITEM = SALESREC_SALES_ITEM;
    SALES_AMOUNT = SALESREC_SALES_AMOUNT;
    SALES_QTY = SALESREC_SALES_QTY;
    SALES_DATE = SALESREC_SALES_DATE;
    Putsort;
    IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END;
END PUT_SORT_0150;;

/*****************************************************************************/
/* DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON. */
/*****************************************************************************/

DISPLAY SALES DATA 0200: PROC;
DO DISPLAY_COUNT = 1 TO 20 BY 1
    UNTIL (END_OF_DISPLAY = 'Y');
    CALL GET_SALES_ITEM_0220;
END;

IF (END-OF-DISPLAY = 'Y') THEN
    ...message in map = 'NO MORE ITEMS FOR SALES PERSON';
ELSE
    ...message in map = 'MORE ITEMS FOLLOW FOR SALES PERSON';
    ...display map
END DISPLAY_SALES_DATA_0200;

GET_SALES_ITEM_0220: PROC;
/*
** CA IDMS/DC SORT keeps its currency within the sorted file between
** pseudo converses, therefore, no repositioning is required.
*/
GETSORT NEXT INTO SALES_DATA;
IF (TPSRETN = '0000') THEN DO;
    ...map = SALES_ITEM_NAME;
    ...map = SALES_AMOUNT;
    ...map = SALES_QTY;
    ...map = SALES_DATE;
END;
ELSE
    IF (TPSRETN = '7020') THEN
        END_OF_DISPLAY = 'Y';
    ELSE
        CALL SORT_ERROR_9999;
END GET_SALES_ITEM_0220;

/*****************************************************************************
* END CURRENT SESSION OF CA IDMS/DC SORT.                                  *
*****************************************************************************
END_SORT_9000: PROC;
ENDSORT;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END END_SORT_9000;

*****************************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT              *
* IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE             *
* FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.                     *
*****************************************************************************
SORT_ERROR_9999: PROC;
...message line in map = TPSMSG;
...display map
...return to CA IDMS/DC
END SORT_ERROR_9999;
END TPSEXPL3;

Exhibit 4.12: PLI Pseudo Conversational--PLI

!  ************************************************************************
!  *  THIS ADS EXAMPLE ILLUSTRATES THE USE OF CA IDMS/DC SORT TO        *
!  *  DISPLAY THE SALES DATA FOR A GIVEN SALES PERSON. THE ENTIRE       *
!  *  SORTED DETAILS CANNOT BE DISPLAYED ON A SINGLE SCREEN. HENCE,*    *
!  *  THE DETAILS ARE KEPT IN SORTED ORDER ACROSS MAP DISPLAYS.        *
!  ************************************************************************
IF ...first time
  ...set first time off
  CALL GET-SORTED-SALES-DATA.
END.

MOVE 'N' TO END-OF-DISPLAY.
WHILE (DISPLAY-COUNT < 21) AND (END-OF_DISPLAY NE 'Y')
  REPEAT.
    ************************************************************************
    *  CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE          *
    *  BETWEEN PSEUDO CONVERSES; THEREFORE, NO REPOSITIONING IS          *
    *  REQUIRED.                                                            *
    ************************************************************************
    GETSORT NEXT.
    IF TPSRETN EQUAL ZERO
      DO.
        MOVE SALES-ITEM-NAME TO MAP-SALES-ITEM NAME (DISPLAY-COUNT).
        MOVE SALES-AMOUNT TO MAP-SALES-AMOUNT (DISPLAY-COUNT).
        MOVE SALES-QTY TO MAP-SALES-QTY (DISPLAY-COUNT).
        MOVE SALES-DATE TO MAP-SALES-DATE (DISPLAY-COUNT).
        ADD 1 TO DISPLAY-COUNT
      END.
    ELSE
      IF TPSRETN EQUAL '7020'
        MOVE 'Y' TO END-OF-DISPLAY.
      ELSE
        CALL SORT-ERROR.
      END.
    END.

    IF END-OF-DISPLAY = 'Y'
      DO.
        ENDSORT.
        IF TPSRETN NOT EQUAL ZERO
          DO.
            CALL SORT-ERROR.
          END.
        DISPLAY MESSAGE TEXT 'NO MORE ITEMS FOR SALES PERSON'.
      END.
    ELSE
      DISPLAY CONTINUE
      MESSAGE TEXT 'MORE ITEMS FOLLOW FOR SALES PERSON'.

    ************************************************************************
    *GET-SORTED-SALES-DATA                                                 *
    *  *  SORT SALES DATA FOR A GIVEN SALES PERSON BY ITEM AND DATE SOLD. *
    *  *  NOTE: SINCE THE SALES RECORD IS VERY LARGE, THE SALES DATA       *
    *  *  NEEDED FOR THE SORT AND DISPLAY ARE MOVED TO A WORK RECORD        *
    ************************************************************************


DEFINE SUBROUTINE GET-SORTED-SALES-DATA.

SETSORT PROGRAM IDMS FOR SALES-DATA
   FIELD SALES-ITEM-NAME ASCENDING
   SALES-DATE DESCENDING.
IF TPSRETN NOT EQUAL ZERO
   CALL SORT-ERROR.
MOVE 'N' TO END-OF-SALES
WHILE END-OF-SALES NOT EQUAL 'Y'
   REPEAT.
   ...
   ...obtain a sales record for the sales person, when all records
   have been processed for this sales person, move 'Y' to
   END-OF-SALES.
   ...
   IF END-OF SALES = 'N'
   DO.
      MOVE SALESREC-SALES-ITEM TO SALES-ITEM.
      MOVE SALESREC-SALES-AMOUNT TO SALES-AMOUNT.
      MOVE SALESREC-SALES-QTY TO SALES-QTY.
      MOVE SALESREC-SALES-DATE TO SALES-DATE.
      Putsort.
      IF TPSRETN NOT EQUAL ZERO
      CALL SORT-ERROR.
   END.
END.
GOBACK.

DEFINE SUBROUTINE SORT-ERROR.
DISPLAY MESSAGE TEXT TPMSG.
RETURN TO TOP.
GOBACK.

Exhibit 4.13: ADS Pseudo Conversational--ADS

IDENTIFICATION DIVISION.
PROGRAM-ID. TPSEXPL4
REMARKS. THIS COBOL EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A "USER"
SORT HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT. THE PROGRAM
IS PSEUDO CONVERSATIONAL AND CAN SORT ANY OR ALL OF THE SALES DATA
FIELDS IN EITHER ASCENDING OR DESCENDING ORDER AT THE USERS
DISCRETION AT EXECUTION TIME.

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
  77 DISPLAY-COUNT PIC S9(9) COMP.
  77 END-OF-DISPLAY PIC X.
  77 END-OF-SALES PIC X.
  01 SALES-DATA.
  05 SALES-ITEM-NAME PIC X(25).
  05 SALES-AMOUNT PIC S9(9)V99 COMP-3.
  05 SALES-QTY PIC S9(9) COMP-3.
  05 SALES-DATE PIC X(08).
COPY SALESREC
.
COPY TPSCOMMC
.
.
PROCEDURE DIVISION.
IF ...first time
THEN
   ...set first time off
   PERFORM 0100-GET-SORTED-SALES-DATA.
MOVE 'N' TO END-OF-DISPLAY.
PERFORM DISPLAY-SALES-DATA.
IF END-OF-DISPLAY = 'Y'
THEN
   PERFORM 9000-END-SORT
   ...return to CA IDMS/DC
ELSE
   ...return to CA IDMS/DC with next task code
   for this program.
**************************************************************************
*            SORT ORDER WILL BE CONTROLLED BY THE USER. THE USER CAN      *
*            SELECT ANY OR ALL OF THE FIELDS IN THE SALES-DATA-WORK      *
*            RECORD AS A SORT KEY. EACH SELECTED SORT KEY CAN BE ORDERED  *
*            EITHER IN ASCENDING OR DESCENDING SEQUENCE.                  *
**************************************************************************
0100-GET-SORTED-SALES-DATA SECTION.
SETSORT USER
   FOR SALES-DATA LENGTH 44
   FIELD SALES-ITEM-NAME 25
   SALES-AMOUNT 6
   SALES-QTY 5
   SALES-DATE 8
IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
MOVE 'N' END-OF-SALES.
PERFORM 0150-PUT-SORT UNTIL END-OS-SALES = 'Y'.
SECTION-EXIT.
EXIT.
0150-PUT-SORT SECTION.
   ...read a sales record for the sales person,
   when all records have been read for sales person
   move 'y' to END-OF-SALES
   IF END-OF-SALES = 'N'
   THEN
      MOVE SALESREC-SALES-ITEM TO SALES-ITEM
      MOVE SALESREC-SALES-AMOUNT TO SALES-AMOUNT
      MOVE SALES-REC-SALES-QTY TO SALES-QTY
      MOVE SALESREC-SALES-DATE TO SALES-DATE
      PUTSORT.
      IF TPSRETN NOT = '0000' PERFORM 9999-SORT-ERROR.
   SECTION-EXIT.
   EXIT.
**************************************************************************
*            DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY SALES PERSON      *
**************************************************************************
0200-DISPLAY-SALES-DATA SECTION.
PERFORM 0220-GET-SALES-ITEM
   VARYING DISPLAY-COUNT FROM 1 BY 1 UNTIL (DISPLAY-COUNT > 20)
OR  (END-OF-DISPLAY = 'Y')
THEN
  MOVE 'NO MORE ITEMS FOR SALES PERSON' TO ...message in map.
ELSE
  MOVE 'MORE ITEMS FOLLOW FOR SALES PERSON' TO ...message in map.

...display map
SECTION-EXIT.
EXIT.

0220-GET-SALES-ITEM SECTION.

** CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE BETWEEN
** PSEUDO CONVERSES. THEREFORE, NO REPOSITIONING IS REQUIRED.

GETSORT NEXT.
IF TPSRETN = '0000'
THEN
  MOVE SALES-ITEM-NAME TO ...map
  MOVE SALES-AMOUNT TO ...map
  MOVE SALES-QTY TO ...map
  MOVE SALES-DATE TO ...map
ELSE
  IF TPSRETN = '7020'
    THEN
      MOVE 'Y' TO END-OF-DISPLAY
    ELSE
      PERFORM 9999-SORT-ERROR.
  SECTION-EXIT.
EXIT.

*****************************************************************************
* END CURRENT SESSION OF CA IDMS/DC SORT.                                *
*****************************************************************************

9000-END-SORT SECTION.
ENDSORT.
IF TPSRETN NOT = '0000' THEN PERFORM 9999-SORT-ERROR.
SECTION-EXIT.
EXIT.

*****************************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT IN          *
* THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE FROM          *
* CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.                        *
*****************************************************************************

9999-SORT-ERROR SECTION.
MOVE TPSMSG TO ...message line in map
...display map
...return to CA IDMS/DC
SECTION-EXIT.
EXIT.

Exhibit 4.14: COB Pseudo Conversational, USER Option--COBOL
* USER'S DISCRETION AT EXECUTION TIME. *

User specifies the fields and sort order.

** SALE DATA RECORD **

- **SLDATA** DS OXL44: Extracted data from sales record.
- **SLITEM** DS CL25: Name of item sold.
- **SLAMT** DS PL6: Amount item sold for.
- **SLQTY** DS PL5: Number of items sold.
- **SLDATE** DS CL8: Date item was sold.
- **ENDSALE** DS C: End of sales indicator.
- **ENDDSPLY** DS C: End of display indicator.

** COMMUNICATIONS BLOCK **

- **R3** EQU 3: Bal - subroutine linkage.
- **R4** EQU 4: BCT - loop counter.

** TPSEXPL4 CSECT **

- **CLI** ...first time.
- **BNE** MAIN0100 NO, SKIP EXTRACT AND SORT.
- **MVI** ...first time.
- **BAL** R3, SORTSALE: Extract and sort sales data.

** MAIN0100 EQU **

- **EQU**.
- **MVI** ENDDSPLY, C'N': Initialize for loop.

** PUTLOOP EQU **

- **EQU**.
- **CLI** ENDSALE, 'Y': Any more sales persons?
- **BER** R3 NO, RETURN.
- **MVC** SLITEM,...: Save name of item sold.
- **ZAP** SLAMT,...: Save amount of sale.
- **ZAP** SLQTY,...: Save quantity sold.
- **MVC** SLDATE,...: Save date item sold.
- **PUTSORT.**
- **CLC** TPSRETN, = CL4'0000': Sales data accepted by sort?
CA IDMS - 19.0

BE PUTLOOP YES, CONTINUE EXTRACT
B BADSORT NO, REPORT ERROR AND ABORT
SPACE 2

******************************************************************************
* DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON. *
* NOTE: CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE *
* BETWEEN PSEUDO CONVERSES. THEREFORE, NO REPOSITIONING IS *
* REQUIRED. *
******************************************************************************

DISPLAY EQU *
LA R4,20 NUMBER OF ITEMS PER SCREEN
SPACE
GETLOOP EQU *
GETSORT NEXT.
CLC TPSRETN,=CL4'7020'
BE LAST YES, INDICATE NO MORE ITEMS
CLC TPSRETN,=CL4'0000'
BNE BADSORT NO, REPORT ERROR AND ABORT
MVC ..map...SLITEM NAME OF ITEM SOLD
UNPK ..map...SLAMT AMOUNT ITEM SOLD FOR
UNPK ..map...SLQTY NUMBER OF ITEMS SOLD
MVC ..map...SLDATE DATE ITEM SOLD
BCT R4,GETLOOP
MVC ..message in map...,MSGMORE
SPACE
DSPLYMAP EQU *
..display map
BR R3
SPACE
LAST EQU *
MVI ENDDSPLY,C'Y' INDICATE LAST SCREEN OF DISPLAY
MVC ..message in map...,MSGLAST
B DSPLYMAP
SPACE 2

******************************************************************************
* END CURRENT SESSION OF CA IDMS/DC SORT. *
******************************************************************************

TERMSORT EQU *
ENDSORT.
CLC TPSRETN,=CL4'0000'
BER R3 YES
B BADSORT NO
SPACE 2

******************************************************************************
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT *
* IN THE TPSRETN FIELD. TPSMSG FIELD CONTAINS A 79 CHARACTER *
* MESSAGE FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE. *
******************************************************************************

BADSORT EQU *
MVC ..map...,TPSMSG USE MESSAGE FROM CA IDMS
/DC SORT
..display map
..return to CA IDMS/DC
SPACE 2

MSGMORE DC CL79'MORE ITEMS FOLLOW FOR SALES PERSON'
MSGLAST DC CL79'NO MORE ITEMS FOR SALES PERSON'

Exhibit 4.15: ASM Pseudo Conversational, USER Option--Assembler

TPSEXPL4: PROC OPTIONS(MAIN) REORDER;
/* REMARKS. THIS PLI EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A *
"USER" SORT HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT. *
THE PROGRAM IS PSEUDO CONVERSATIONAL AND CAN SORT ANY OR
ALL OF THE SALES DATA FIELDS IN EITHER ASCENDING OR
DESCENDING ORDER AT THE USERS DISCRETION AT EXECUTION TIME.

CA IDMS/DC SORT REQUIRES COMPILATION OPTION "MARGINS(2,72)".

/*
* REQUIRED FOR IDMS */
DCL MODE (IDMS DC) DEBUG;
DCL IDMS ENTRY OPTIONS (INTER, ASM);
INCLUDE IDMS (SUBSCHEMA CTRL);
/* END OF IDMS REQUIREMENT */

DCL ADDR BUILTIN;
DCL DISPLAY_COUNT FIXED BIN(31);
DCL END_OF_DISPLAY CHAR(1);
DCL END_OF_SALES CHAR(1);
DCL 1 SALES_DATA,
  2 SALES_ITEM_NAME CHAR(25),
  2 SALES_AMOUNT PIC'S999999999V99',
  2 SALES_QTY PIC'S999999999',
  2 SALES_DATE CHAR(8);

%INCLUDE SALESREC;
%INCLUDE TPSCOMMP;
.
.
IF (..first time)
  THEN DO:
    ...set first time off;
    CALL GET_SORTED_SALES_DATA_0100
  END;

END_OF_DISPLAY = 'N';
CALL DISPLAY_SALES_DATA_0200;
IF (END_OF_DISPLAY = 'Y')
  THEN DO;
    ...return to CA IDMS/DC;
  END;
ELSE
  ...return to CA IDMS/DC with next task code
  for this program;
*****************************************************************************
* SORT ORDER WILL BE CONTROLLED BY THE USER. THE USER CAN SELECT* *
* ANY OR ALL OF THE FIELDS IN THE SALES-DATA WORK RECORD AS A        *
* SORT KEY. EACH SELECTED SORT KEY CAN BE ORDERED EITHER IN          *
* ASCENDING OR DESCENDING SEQUENCE.                                  *
*****************************************************************************
GET_SORTED_SALES_DATA_0100: PROC;

SETSORT USER
  FOR SALES DATA LENGTH 53
  FIELDS SALES_ITEM_NAME 25
        SALES_AMT 11
        SALES_QTY 9
        SALES_DATE 8;
IF (TPSRETN = '0000) CALL SORT_ERROR_9999;

END_OF_SALES = 'N';
DO UNTIL (END_OF_SALES = 'Y');
  CALL PUT_SORT_0150;
END;
END GET_SORTED_SALES_0100;
PUT_SORT_0150: PROC;

...read a sales record for the sales person;
when all records have been read for sales person
END_OF_SALES = 'Y';

IF (END_OF_SALES = 'N')
THEN DO;
    SALES_ITEM = SALESREC_SALES_ITEM;
    SALES_AMOUNT = SALESREC_SALES_AMOUNT;
    SALES_QTY = SALESREC_SALES_QTY;
    SALES_DATE = SALESREC_SALES_DATE;
    PUTSORT;
    IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END;
END_PUT_SORT_0150;

*****************************************************************************/
*             DISPLAY UP TO 20 OF THE NEXT ITEMS SOLD BY THE SALES PERSON.          *
*****************************************************************************/
DISPLAY_SALES_DATA_0200: PROC;

DO DISPLAY_COUNT = 1 TO 20 BY 1
UNTIL (END_OF_DISPLAY = 'Y');
    CALL GET_SALES_ITEM_0220;
END;

IF (END_OF_DISPLAY = 'Y')
THEN
    ...message in map = 'NO MORE ITEMS FOR SALES PERSON';
ELSE
    ...message in map = 'MORE ITEMS FOLLOW FOR SALES PERSON';

...display map

END DISPLAY_SALES_DATA_0200;

GET_SALES_ITEM_0220: PROC;
/*
**     CA IDMS/DC SORT keeps its currency within the sorted file between
**     pseudo converses, therefore, no repositioning is required.
*/
GETSORT NEXT INTO SALES_DATA;
IF (TPSRETN = '0000')
THEN DO;
    ...map = SALES_ITEM_NAME;
    ...map = SALES_AMOUNT;
    ...map = SALES_QTY;
    ...map = SALES_DATE;
END;
ELSE IF (TPSRETN = '7020')
THEN
    END_OF_DISPLAY = 'Y';
ELSE
    CALL SORT_ERROR_9999;
END GET_SALES_ITEM_0220;

*****************************************************************************/
*             END CURRENT SESSION OF CA IDMS/DC SORT.                           *
*****************************************************************************/
END_SORT_9000: PROC;
ENDSORT;
IF (TPSRETN = '0000') THEN CALL SORT_ERROR_9999;
END END_SORT_9000;

/*****************************************************************************/
* AN UNANTICIPATED RETURN CODE WAS RETURNED BY CA IDMS/DC SORT      *
* IN THE TPSRETN FIELD. TPSMSG CONTAINS A 79 CHARACTER MESSAGE      *
* FROM CA IDMS/DC SORT DESCRIBING THE BAD RETURN CODE.               *
****************************************************************************/

SORT_ERROR_9999: PROC;
...message line in map = TPSMSG;
...display map;
...return to CA IDMS/DC;
END SORT_ERROR_9999;
END TPSEXPL4;

Exhibit 4.16: PLI Pseudo Conversational, USER Option--PLI

! **************************************************************************
! * THIS ADS EXAMPLE IS THE SAME AS EXAMPLE 3, EXCEPT A "USER" SORT      *
! * HAS BEEN SPECIFIED INSTEAD OF A "PROGRAM" SORT. THE DIALOGUE IS*      *
! * PSEUDO CONVERSATIONAL AND CAN SORT ANY OR ALL OF THE SALES DATA *      *
! * DISCRETION AT EXECUTION TIME.                                       *
! **************************************************************************
IF ...first time
  DO.
    ...set first time off
    CALL GET-SORTED-SALES-DATA.
  END.
MOVE 'N' TO END-OF-DISPLAY.
WHILE (DISPLAY-COUNT < 21) AND
  (END-OF-DISPLAY NE 'Y')
REPEAT.
! **************************************************************************
* CA IDMS/DC SORT KEEPS ITS CURRENCY WITHIN THE SORTED FILE              *
* BETWEEN PSEUDO CONVERSES; THEREFORE, NO REPOSITIONING IS              *
* REQUIRED.                                                           *
**************************************************************************
GETSORT NEXT.
IF TPSRETN EQUAL ZERO
  DO.
    MOVE SALES-ITEM-NAME TO MAP-SALES-ITEM-NAME  (DISPLAY-COUNT).
    MOVE SALES-AMOUNT TO MAP-SALES-AMOUNT  (DISPLAY-COUNT).
    MOVE SALES-QTY TO MAP-SALES-QTY  (DISPLAY-COUNT).
    MOVE SALES-DATE TO MAP-SALES-DATE  (DISPLAY-COUNT).
    ADD 1 TO DISPLAY-COUNT.
  END.
ELSE IF TPSRETN EQUAL '7020'
  MOVE 'Y' TO END-OF-DISPLAY.
ELSE
  CALL SORT-ERROR.
END.

IF END-OF-DISPLAY = 'Y'
  DO.
    ENDSORT.
    IF TPSRETN NE ZERO
      DO.
        CALL ERROR.
      END.
    DISPLAY MESSAGE TEXT 'NO MORE ITEMS FOR SALES PERSON'.
Define subroutine GET-SORTED-SALES-DATA.

Setsort user IDMS for sales-data
If TPSRETN ne zero
  Call error.

Move 'N' to END-OF-SALES
While END-OF-SALES ne 'Y'
  Repeat.
    ...
    ...obtain a sales record for the sales person,
    when all records have been processed for this sales
    person, move 'y' to END-OF-SALES.
    ...
  If END-OF-SALES = 'N'
    Do.
      Move SALESREC-SALES-ITEM TO SALES-ITEM.
      Move SALESREC-SALES-AMOUNT TO SALES-AMOUNT.
      Move SALESREC-SALES-QTY TO SALES-QTY.
      Move SALESREC-SALES-DATE TO SALES-DATE.
      Putsort.
      If TPSRETN ne zero
        Call error.
    End.

End.
Goback.

Define subroutine error.

Display message text TPSMSG.
  !  Return to top.
Goback.

Exhibit 4.17: ADS Pseudo Conversational, USER Option--ADS

Selecting Sort Criteria on a User Screen

When you specify USER in the SETSORT statement, at processing time CA IDMS/DC Sort displays a
screen for selecting sort criteria. The screen shows the fields in the record specified in the SETSORT
statement. You can make entries for 1 to 16 fields in the columns headed Sequence and Sort Order.
To cancel the sort at any time, press PA2.
A sort selection screen is illustrated in Exhibit 4.17 SCR1.

**Specifying Sequence and Sort Order**

You can easily specify sequence and sort order by following these steps:

1. On the line containing the element that is to be the first sort key, in the sequence column enter 1.

2. On the same line, in the Sort Order column, you must enter either A for ascending or D for descending.

3. On the line of the element that is to be the second sort key, in the Sequence column enter 2 and in the Sort Order column enter either A or D.

4. Continue entering sequence and sort order for up to 16 elements. Do not skip any sequence numbers. Any element for which you assign a sequence number must also be assigned a sort order.
   If a record has more elements that can fit on one page, you can page backwards or forwards by using the PF keys indicated at the bottom of the screen.

5. For elements that are not sort keys, leave the Sequence and Sort Order columns blank.

6. When you are finished specifying sequence and sort order, press PF3 to execute the sort.

7. If you correct an error, press ENTER to validate the corrections before you execute the sort again.

**Sample Sort Selection Screen**

A sample Sort Selection screen is shown in Exhibit 3.18 SCR1. Here are descriptions of the fields on the screen:

- **SORT KEY DESCRIPTION** -- Names of the fields or elements in the record.

- **SEQUENCE** -- Column in which you can enter a number from 01 to 16 to indicate the sequence in which the fields are to be sorted.

- **SORT ORDER** -- Column in which you enter A for ascending or D for descending for each element given a sequence number.

- **PF Keys** -- List of PF key assignments.
Field Error

When an invalid value is entered in one of the columns on the Sort Selection Screen, a "field error" occurs. Such an error is shown in Exhibit 4.19 SCR2, below. In the detail line for SORT-LANGUAGE, the user entered a sequence value outside the range 1-16.

- **Generalized Error Message** -- A generalized error message appears in the message area, line 2 of the screen, indicating that one or more field errors have been detected.

- **Specific Error Message, Short Form** -- At the same time, short forms of more specific error messages appear next to the items in error. When necessary, these messages can be expanded to provide further information. The next example illustrates how to expand a short form error message.

```
CA IDMS/DC SORT  Rnn.nn   USER SORT KEY SPECIFICATION  hh:mm:ss mm/dd/yy
TPU7066E-ONE OR MORE DETAIL FIELDS ARE IN ERROR

Sort Key Description          Sequence  Sort Order
SORT-NAME                      01        A
SORT-LANGUAGE                  31 7057-BAD SEQ #  A

HELP: PF1-(Expand Error Message)
CONTROL: ENTER-(Validate Screen) PA2-(Cancel Sort) PF3-(Execute Sort)
PAGING: PF6-(Page First) PF7-(Page Prior) PF8-(Page Next)
```

Exhibit 4.19: SCR2 Sort Selection Screen--Field Related Error

Expanding Short Form Field Error Messages

To expand a short form field error message and obtain more information, follow these steps:

1. **Position the cursor** on the detail line item that precedes the short form message to be expanded.

2. **Press the PF1 key.**

3. **Look at the Message Area** (line 2 of the screen). It now contains the long form of the message in the detail line, including the message code (first eight characters).

If the long form of the message still does not provide enough information, use the message code to look up a detailed explanation in Messages (https://docops.ca.com/display/IDMSM/CA+IDMS+Extractor+Messages).
Processing Errors

Processing errors can occur when the USER option is specified in the SETSORT statement. A screen illustrating a processing error message is shown in Exhibit 4.21 SRC4.

- **Message code and text.** For a detailed explanation of the message, see Messages (https://docops.ca.com/display/IDMSM/CA+IDMS+Extractor+Messages).

Exhibit 4.21: SRC4 Sort Selection Screen--Processing Error

CA ADS Preprocessor

This section describes the steps necessary to use CA IDMS/DC Sort with CA ADS. It also describes how to use the EDITOR to correct syntax errors encountered by the CA ADS Preprocessor in CA IDMS/DC Sort.

- CA ADS Preprocessor 1 (see page 44)
- Preprocess Multiple Modules (see page 45)
- System Flow (see page 46)
- EDITOR (see page 48)
- Key Settings (see page 48)
CA ADS Preprocessor 1

To use the CA IDMS/DC Sort preprocessor for CA ADS, follow these four steps:

Step 1 -- Add CA IDMS/DC Sort Statements to Modules

Add the necessary SETSORT, PUTSORT, GETSORT, ENDSORT and SETLIMIT statements to the appropriate dictionary modules. See Chapter 3, Examples for sample dialogs.

Add a statement to check the return code (TPSRETN).

Add statements to issue error messages when appropriate.

Step 2 -- Execute the Preprocessor for Each Module

For each dictionary module containing CA IDMS/DC Sort syntax, execute the CA ADS preprocessor of CA IDMS/DC Sort (TPSG), as shown in Exhibit 6.1.

If an error is detected in the user-supplied parameters, CA IDMS/DC Sort automatically invokes an online EDITOR (see page 48) which allows you to correct syntax errors.

If you have installed the CA IDMS DME, the module source can be preprocessed at any time during a CA IDMS DME session by entering TPSG on the command line.

Note: For more information, see the CA IDMS Dictionary Module Editor User Guide.

If you are using CA IDMS Dictionary Migrator to move module source to another dictionary, the CA ADS Batch preprocessor, TPSBADSI, can be used to preprocess the DDDLUPD file.

To execute the preprocessor, enter the task code for CA IDMS/DC Sort (TPSG) on the system prompt screen. The system will display a screen on which you can enter parameters identifying the module.

The screen is illustrated in Exhibit 6.1. If you use the PA2 key to cancel the preprocessor, only the current TPSG command is affected.
Exhibit 6.1: CA ADS Module Preprocessor Screen

Another method of executing the preprocessor is to enter all of the preprocessor parameters when invoking the preprocessor:

\texttt{TPSG \textit{module-name} \textit{module-version} \textit{alternate-dictionary} \textit{node}}

where:

- \textit{module-name} indicates the name of the module.
- \textit{module-version} indicates the version number of the module.
- \textit{alternate-dictionary} indicates the dictionary in which the module resides.
- \textit{node} indicates the DDS node in which the alternate dictionary is found.

Each of these parameters is positional, with each default represented by a comma (,). A single space delimiter between parameters is required.

Example

\texttt{V10 ENTER NEXT TASK CODE: TPSG TEST-MOD, DICTB}

Step 3 -- Execute the Dialog Generator

Execute the dialog compiler, ADSC, for the dialog containing the altered modules. Specify TPSCOMM as a work record.

Step 4 -- Execute the Dialog

Execute the dialog.

Preprocess Multiple Modules

CA IDMS users can create a CLIST consisting of a number of CA IDMS/DC Sort preprocessor task statements and the CA ADS compiler task statement. Once the CLIST is set up, you need execute only one instruction instead of several.

The following example illustrates the creation and execution of a CLIST to preprocess multiple CA ADS modules.

Example

Create a CLIST to preprocess multiple ADS modules and invoke the ADSC compiler.

\texttt{ADD MODULE PRESORT VER 1 LANG DCMODULE SOURCE FOLLOWSTPSG MODULE-1, DICTBTPSG MODULE-4 2 DICTBTPSG MODULE-6 99 DICTBADSCMSEN D.}

Execute the CLIST.
V10 ENTER NEXT TASK CODE: CLIST PRESORT

In this way, each of the modules containing CA IDMS/DC Sort syntax will be preprocessed prior to executing the CA ADS Compiler, ADSC.

System Flow

The preprocessor uses SETSORT, PUTSORT, GETSORT, and ENDSORT statements to generate required programming logic for CA IDMS/DC Sort.

A diagram of system flow is shown in Exhibit 6.2. The dialogs are placed in the dictionary (IDD), and from there read into the CA IDMS/DC Sort precompiler. If there are errors, the CA EDITOR is automatically invoked and you can make syntax corrections, replace the module in the dictionary, and re-execute the preprocessor. Then the dialogs are generated. After they have been preprocessed, CA IDMS/DC Sort statements appear as comments.

At execution time, if the PROGRAM option was selected in the SETSORT statement, the dialogs issue calls to CA IDMS/DC Sort. Then the sorts are done by CA IDMS/DC Sort, using main and auxiliary sort-work areas as necessary. After the sorts are completed, the results are displayed as directed by the application.

If the USER option was selected in the SETSORT statement, before the sorting is done CA IDMS/DC Sort presents the user with a sequence selection screen, where the user can designate up to 16 sort keys and the sort order (ascending or descending) for each key.
IDMSDB--System Flow

Exhibit 6.2: CA ADS System Flow
EDITOR

If the CA IDMS/DC Sort CA ADS Preprocessor detects sort control or formatting errors then the CA
EDITOR is invoked during the preprocess step.

The EDITOR is conveniently invoked directly from the preprocessor. When a syntax error is detected,
the EDITOR displays the module source with highlighted error lines describing the problem. The
programmer can correct the syntax and RETRY the preprocessor. If the syntax is correct, it is replaced
in the dictionary and processing continues.

If syntax errors remain, the EDITOR is reinvoked until the syntax is correct or the user cancels the
preprocessor session.

Key Settings

PF keys are set for most Primary Commands. This allows you to enter a Primary Command from any
position on the screen (not from just the Command Line). PF keys also reduce keystrokes. You do not
need to press the ENTER key after pressing a PF key. Exhibit 6.11 summarizes the functions of the key
settings.

ENTER Function

The PF keys set for the ENTER function work in the same way as the ENTER key: they execute Primary
or Line Commands.

RESHOW Function

The RESHOW function on the PA1, PA2 and CLEAR keys cannot be performed with a Primary
Command. Use RESHOW to view the last screen display.

<table>
<thead>
<tr>
<th>Key</th>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1/13</td>
<td>HELP</td>
<td>(currently unavailable).</td>
</tr>
<tr>
<td>PF2/14 PF3/15</td>
<td>END/RETRY</td>
<td>Saves the module and reinvokes the preprocessor.</td>
</tr>
<tr>
<td>PF4/16</td>
<td>CANCEL</td>
<td>Saves the module and exits the Edit Screen.</td>
</tr>
<tr>
<td>PF5/17</td>
<td>RFIND</td>
<td>Repeats the last FIND command.</td>
</tr>
<tr>
<td>PF6/18</td>
<td>RCHANGE</td>
<td>Repeats the last CHANGE command.</td>
</tr>
<tr>
<td>PF7/19</td>
<td>UP</td>
<td>Scrolls to display the lines above your current view.</td>
</tr>
<tr>
<td>PF8/20</td>
<td>DOWN</td>
<td>Scrolls to display the lines below your current view.</td>
</tr>
<tr>
<td>PF9/21</td>
<td>RESET</td>
<td>Removes Line Commands, column markers and extraneous messages.</td>
</tr>
<tr>
<td>PF10/22</td>
<td>LEFT</td>
<td>Scrolls to display the lines to the left of your current view.</td>
</tr>
<tr>
<td>PF11/23</td>
<td>RIGHT</td>
<td>Scrolls to display the lines to the right of your current view.</td>
</tr>
<tr>
<td>PF12/24</td>
<td>ENTER</td>
<td>Functions as the ENTER key.</td>
</tr>
<tr>
<td>PA1/PA2 CLEAR</td>
<td>RESHOW</td>
<td>Redisplays previous screen.</td>
</tr>
</tbody>
</table>
Exhibit 6.10: Summary of Key Settings

CA ADS

To use the preprocessor with CA ADS, see CA IDMS/DC Sort Parameter Statements (see page 52).

Exhibit 5.2: Modelz/OS JCL for Execution of CA IDMS/DC Sort

Exhibit 5.3: Sample Z/OS Link Edit Control Statements Assembler or COBOL

Exhibit 5.4: Sample Z/OS Link Edit Control Statements--PLI
Exhibit 5.4: Sample Z/OS Link Edit Control Statements--PLI

// OPTION PARTDUMP
// UPSI 00000001
* **** PRIVATE CORE IMAGE LIBRARY WHERE TP/SORT INSTALLED
// DLBL DBMS,'your.loadlib'
// EXTENT .volser
* FOR DOS/SP USE THE FOLLOWING:
// LIBDEF PHASE,SEARCH=(DBMS.sublibrary,IDMS.sublibrary)
* FOR Z/VSE USE THE FOLLOWING:
// LIBDEF CL,SEARCH=(DBMS,IDMS)
* 
* *************** REPORT FILE *********************
* 
// ASSGN SYS013,SYSLST
* 
* ******** SOURCE CODE INPUT TO PREPROCESSOR ***********************
* 
// ASSGN SYS014,SYSRDR SYNTAX FILE
* 
* ******** PREPROCESSED SOURCE CODE FROM PREPROCESSOR **************
* 
// DLBL OUTPUT,'work.file.output',0,SD
// EXTENT SYS015,volser,,,00250,003
// ASSGN SYS015,DISK,VOL=volser,SHR
* 
// EXEC TPSBXXXX,SIZE=(TPSBXXXX,400K)
PLACE LANGUAGE SOURCE CODE HERE
/*
*/
SS EOJ

Exhibit 5.5: Model Z/VSE JCL for Execution of CA IDMS/DC Sort

*****************************************************************************
* Sample Z/VSE link-edit control cards for inclusion of CA IDMS/DC SORT with Assembler or COBOL programs. *
*****************************************************************************
PHASE your-program,*
INCLUDE CICS-module
INCLUDE your-program
ENTRY DFHEI1

Exhibit 5.6: Sample Z/VSE Link Edit Control Statements Assembler or COBOL

*****************************************************************************
* Sample Z/VSE link-edit control cards for inclusion of CA IDMS/DC SORT with PL/I programs. *
*****************************************************************************
PHASE your-program,*
INCLUDE your-program
ENTRY PLISTART

Exhibit 5.7: Sample Z/VSE Link Edit Control Statements--PLI

/* TPSBEXEC */
/* *
/* KEY TO VARIABLES */
/* ca.loadlib The filename of the load library into which you downloaded CA IDMS Tools. */
/* idms.loadlib The filename of the load library containing your IDMS SUBSCHEMA and DMCL modules. */
/* tsprogn Change to: TPSBASMI for IDMS/DC-Assembler */
/* TPSBCOBI for IDMS/DC-COBOL */
/* TPSBPLII for IDMS/DC-PLI */

TRACE OFF; SIGNAL ON ERROR

CA_LOADLIB_FN = 'ca.loadlib'
IDMS_LOADLIB_FN = 'idms.loadlib'
INPUT_FN = 'input-source-fn'
INPUT_FT = 'input-source-ft'
INPUT_FM = '*'
OUTPUT_FN = 'output-source-fn'
OUTPUT_FT = 'output-source-ft'
OUTPUT_FM = '*'

/* LINK AND ACCESS MINIDISKS CONTAINING REQUIRED LIBRARIES */

'CP SPOOL PRINTER NOCONT CLOSE'
'CP SPOOL PRINTER TO * NOHOLD CONT FORM OFF DIST OFF'
'GLOBAL LOADLIB ' CA_LOADLIB_FN IDMS_LOADLIB_FN
'FILEDEF AUDIT PRINTER'
'FILEDEF INPUT DISK ' INPUT_FN INPUT_FT INPUT_FM
'FILEDEF OUTPUT DISK ' OUTPUT_FN OUTPUT_FT OUTPUT_FM
SIGNAL OFF ERROR
SAY 'STARTING CA IDMS/DC SORT PREPROCESSOR'
'EXECOS OSRUN tsprogn'
TPSBXEC_RC = RC
'CP SPOOL PRINTER NOCONT'
'CP CLOSE PRINTER NAME TPSBEXEC LISTING'
'CP SPOOL PRINTER OFF'
SAY 'TPSBEXEC FINISHED WITH A RETURN CODE OF' TPSBXEC_RC
'GLOBAL LOADLIB'
'FILEDEF * CLEAR'
EXIT TPSBEXEC_RC

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
ERROR:
/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/
ERROR_RC = RC
TRACE OFF; SIGNAL OFF ERROR
SAY 'NON-ZERO RETURN CODE ENCOUNTERED IN EXEC AT LINE' SIGL
'CP SPOOL PRINTER NOCONT'
'CP CLOSE PRINTER NAME TPSBEXEC LISTING'
'CP SPOOL PRINTER OFF'
'GLOBAL LOADLIB'
'FILEDEF * CLEAR'
EXIT ERROR_RC

/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

Exhibit 5.8: Z/VM EXEC for CA IDMS/DC Sort Execution ── Asm, COBOL, PLI
CA IDMS/DC Sort parameters Overview

This section provides information on the CA IDMS/DC Sort parameters. It begins with an overview of the five parameter statements that can be entered in your application program. The overview includes a complete parameter summary chart, notation conventions, and syntax rules. The overview is followed by a description of each parameter, showing its appropriate syntax, rules for use, and defaults. The parameters are presented in the order shown in the parameter summary chart.

- CA IDMS/DC Sort Parameter Statements (see page 52)
- record-name Parameter: CA IDMS/DC Sort (see page 53)

CA IDMS/DC Sort Parameter Statements

Five parameter statements are entered in the application programs: SETSORT, PUTSORT, GETSORT, ENDSORT and SETLIMIT. Sample application programs including these statements are shown in Examples (https://docops.ca.com/display/IDMS19/Examples).

CA IDMS/DC Sort has a work-saving preprocessor which generates programming statements appropriate for your sorting requirements. This section explains the parameter statements that you must include in your program.

Parameter Options

Within the parameter statements, CA IDMS/DC Sort provides a variety of options for tailoring your sort session to meet your needs. These options allow you to select a record and define sort criteria.

Your choice of options entered in the parameter statements depends on the programming language, the operating environment, and the type of sort you want generated.

In a CA IDMS environment, you can direct CA IDMS/DC Sort to extract some of the control information from the dictionary.

SETSORT Statement

The SETSORT statement initiates a CA IDMS/DC Sort session. From this statement, values are initialized in the CA IDMS/DC Sort control blocks. These values are in effect until an ENDSORT statement is issued for the session.

In the SETSORT statement, you can specify the record name and sort criteria, or you can indicate that the user is to define criteria at execution time. The SETSORT statement can also indicate that some of the values are to be extracted from a dictionary.
PUTSORT Statement

The PUTSORT statement takes a record defined in the SETSORT statement for the session and transfers it to CA IDMS/DC Sort processing.

GETSORT Statement

The GETSORT statement retrieves a record defined in the SETSORT statement for the session. You can specify one of four different sequences for retrieval: NEXT, PRIOR, FIRST, or LAST.

ENDSORT Statement

The ENDSORT parameter statement is used to terminate a CA IDMS/DC Sort session and to release the resources used by CA IDMS/DC Sort.

SETLIMIT Statement

The SETLIMIT statement is used to override runtime storage limits and functional page organization. The use of SETLIMIT is enabled or disabled through the TPSPARM tuning macro described in Operations (https://docops.ca.com/display/IDMS19/Operations).

record-name Parameter: CA IDMS/DC Sort

record-name is the name of the record into which CA IDMS/DC Sort is to place each sorted record when it is retrieved.

If record-name is not specified, the record named in the SETSORT statement will be used. The 'INTO<record-name>' clause must be specified whenever the GETSORT command is separated from the SETSORT command by a pseudo-converse.

Note that in CA ADS, a pseudo-converse occurs whenever a DISPLAY command that has no continue option is encountered.

CA IDMS/DC Sort System Flow

CA IDMS/DC Sort provides preprocessors for use with COBOL, Assembler, PLI, and CA ADS. The preprocessors use SETSORT, PUTSORT, GETSORT, ENDSORT, and SETLIMIT statements to generate required programming logic for CA IDMS/DC Sort.
A diagram of system flow is shown in Exhibit 5.1. The application program, including CA IDMS/DC Sort parameter statements, is fed into the CA IDMS/DC Sort precompiler. Then the program is compiled and linked.

At execution time, if the PROGRAM option was selected in the SETSORT statement, the program issues calls to CA IDMS/DC Sort. Then the sorts are done by CA IDMS/DC Sort, using main and auxiliary sort-work areas as necessary. After the sorts are completed, the results are displayed as directed by the application.

If the USER option was selected in the SETSORT statement, before the sorting is done CA IDMS/DC Sort presents the user with a sequence selection screen, where the user can designate up to 16 sort keys and the sort order (ascending or descending) for each key.
Exhibit 5.1: CA IDMS/DC Sort System Flow
COBOL/Assembler/PLI

The steps required to use CA IDMS/DC Sort with COBOL, Assembler, or PLI programs are listed below. Sample application programs are shown in Examples (https://docops.ca.com/display-IDMS19/Examples).

1. Add to your program the statements necessary to accomplish these tasks:

   a. Copy the appropriate CA IDMS/DC Sort control block.
      COBOL: TPSCOMMC
      Assembler: TPSCOMMA
      PLI: TPSCOMMP

   b. Initiate the sort and establish criteria (SETSORT).
      Optionally, issue a SETLIMIT to alter the runtime environment.
      - Pass records to CA IDMS/DC Sort (PUTSORT).
      - Retrieve sorted records (GETSORT).
      - Terminate a session (ENDSORT).
      The details of using these four parameter statements are described in CA IDMS/DC Sort Parameter Statements (see page 52).

   c. Check the return code (TPSRETN). Do not ignore a non-zero return code. After a GETSORT, the content of the sorted record is unpredictable if the return code is non-zero.

   d. Issue error messages when appropriate (TPSMSG).

2. Execute the preprocessor as shown in Exhibit 5.2 (z/OS), Exhibit 5.5 (Z/VSE), or Exhibit 5.8 (Z/VM).

   ! Note: CA IDMS 16.0 supports z/OS V2R10 as well as z/OS 1.1 and above. However, we refer to z/OS in this document.

3. Compile your program.

4. Link your program as shown in Exhibit 5.3 (z/OS, COBOL or Assembler), Exhibit 5.4 (z/OS, PLI), Exhibit 5.6 (Z/VSE, COBOL or Assembler), or Exhibit 5.7 (Z/VSE, PLI).

5. Execute your program.

6. If USER was specified in the SETSORT statement, respond to the user screens.
Customizing CA IDMS/DC Sort

CA IDMS/DC Sort provides a customization macro that gives you the ability to:

- Specify the amount of main storage and auxiliary storage to be made available to CA IDMS/DC Sort.
- Indicate how space is to be allocated to buffers at runtime. The allocation of buffers also depends on the record length in a particular sort.
- Indicate whether or not developers are allowed to alter MAIN, AUX, or MINRBUF at runtime. It does this by enabling or disabling the SETLIMIT parameter.
- Indicate the CA IDMS/DC Sort CA ADS Preprocessor termination key.

These runtime options can be changed at anytime after initial product installation, either before or after SMP/E ACCEPT processing. See the CA IDMS installation guides for detailed instructions on processing customization macro changes under SMP/E. Additional customization considerations and examples for CA IDMS/DC Sort are shown below.

Customization Considerations

At installation time, the MAIN and AUX parameters are each assigned a value of 10000 bytes, unless you changed the default values during the SMP/E installation process. During each sort session in an application, CA IDMS/DC Sort acquires the main and auxiliary storage as necessary, up to the value assigned. (A session is defined by the session number in a SETSORT statement.) If you want to run the most efficient sorts possible, you may want to consider the following points:

The most efficient sort is one in which

There are many small records in a buffer

- All of the buffers reside in main storage

To increase efficiency in a given sort session, use a work record that contains only the fields necessary for sorting. With only those fields, the work record is as small as possible to meet the requirements.

In an ideal situation,

Main storage is slightly larger than the space needed for an average sort

- Auxiliary storage adds the extra space needed for large sorts

Increasing the proportion of auxiliary storage to main storage may affect response time.
Sample CA IDMS/DC Sort Customization

At execution time, CA IDMS/DC Sort allocates sort buffers in multiples of 2000 bytes. To determine the size of a sort buffer:

1. Multiply the MINRBUF value times the record size.
2. Round the result up to the next multiple of 2000 bytes.
3. Add 12 bytes for CA IDMS/DC Sort overhead.

Maximum: Sort buffer size can be no greater than 32K.

\[ \text{Sort buffer size} = (\text{MINRBUF} \times \text{record length}) + 12 \]

\[ \text{Round up to next multiple of 2000} \]

\[ \text{Maximum: Sort buffer size} \leq 32K \]

**Note:** CA IDMS/DC Sort will not split a buffer between main and auxiliary storage. Therefore it is necessary to make efficient use of main and auxiliary storage.

The product of the MINRBUF value and the record length cannot exceed either the MAIN value or the AUX value, whichever is larger, because there would not be enough space to store one sort buffer.

In the following four examples, the MAIN and AUX parameters are not changed. The default for each is 10000 bytes.

**Example 1**

MINRBUF=20
record-length=100

The sort buffer used by CA IDMS/DC Sort will be 2012 bytes:

\[ 20 \times 100 = 2000 \]
\[ 2000 \text{ is a multiple of } 2000 \]
\[ 2000 + 12 = 2012 \]

CA IDMS/DC Sort can store four sort buffers (80 records) in main storage and four sort buffers (80 records) in auxiliary storage.

**Example 2**

MINRBUF=20
record-length=150

The sort buffer used by CA IDMS/DC Sort will be 4012 bytes:

\[ 20 \times 150 = 3000 \]
\[ 3000 \text{ is the next multiple of } 2000 \]
\[ 4000 + 12 = 4012 \]

CA IDMS/DC Sort can store two sort buffers (40 records) in main storage and two sort buffers (40 records) in auxiliary storage.

**Example 3**
MINRBUF=100 (default)
record-length=31

The sort buffer used by CA IDMS/DC Sort will be 4012 bytes:

\[ 31 \times 100 = 3100 \]
\[ \text{next multiple of 2000 is } 4000 \]
\[ \text{sort buffer is } 4012 \]

CA IDMS/DC Sort can store two sort buffers (200 records) in main storage and two sort buffers (200 records) in auxiliary storage.

**Example 4**

MINRBUF=100 (default)
record-length=51

The sort buffer used by CA IDMS/DC Sort will be 6012 bytes:

\[ 51 \times 100 = 5100 \]
\[ \text{next multiple of 2000 is } 6000 \]
\[ \text{sort buffer is } 6012 \]

CA IDMS/DC Sort can store one sort buffer (100 records) in main storage and one sort buffer (100 records) in auxiliary storage.

### Notation Conventions and Syntax Rules

The rest of this section explains in detail how to use the parameter statements. Be sure to review the following Exhibits.

- Exhibit 3.1 -- CA IDMS/DC Sort Parameter Summary
- Exhibit 3.2 -- CA IDMS/DC Sort Parameter Summary with the IDMS Extension
- Exhibit 3.3 -- Notation Conventions
- Exhibit 3.4 -- Parameter Syntax Rules
IDMSDB--Notation Conventions and Syntax Rules

Exhibit 3.1: CA IDMS/DC Sort Parameter Summary
Exhibit 3.2: CA IDMS/DC Sort Parameter Summary with IDMS Extension

<table>
<thead>
<tr>
<th>Example</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETSORT</td>
<td>Keywords appear in UPPERCASE.</td>
</tr>
</tbody>
</table>

IDMSDB--Notation Conventions and Syntax Rules (2)
Example | Function
--- | ---
SESSION | The minimum required portion of each keyword is UNDERSCORED. You can omit the portion of a keyword that is not underscored without altering the meaning.
record-name | Variables appear in lowercase. You must substitute an appropriate value for a variable.
[ field-length ] | Brackets indicate optional clauses.
/PRIOR\ FIRST/ | Braces enclose two or more options. You must select one of them.
|| field-name-1 |||| field-name-16 || |
| | A pair of double bars encloses two or more options. You must select one or more of the options.

Exhibit 3.3: Notation Conventions

<table>
<thead>
<tr>
<th>Item</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of Delimiters</td>
<td>Use one or more blanks as a delimiter between keywords. Use a period or semicolon to end each parameter statement.</td>
</tr>
<tr>
<td>Coding Conventions</td>
<td>When inserting the CA IDMS/DC Sort parameters into your application program, follow the coding conventions of the application program language: PLI, Assembler, COBOL or CA ADS.</td>
</tr>
<tr>
<td>Parameter Statement Limits</td>
<td>Parameter statements can be continued on more than one line. However, you cannot exceed 50 lines of syntax for a single statement.</td>
</tr>
</tbody>
</table>

Exhibit 3.4: Parameter Syntax Rules

**ENDSORT Statement 1**

The ENDSORT statement terminates a SETSORT sequence. Although optional, its use is recommended in order to free up resources.

A sort session can be terminated at any time with an ENDSORT statement. It need not follow a PUTSORT and GETSORT statement.

\/.ENDSORT [ SESSION session-number ] <;}
SESSION Parameter 3

SESSION session-number

is an optional parameter which identifies a sorting process for a single terminal user. The session-number can be an integer from 0 to 9.

Default: default is 0.

The SESSION parameter in the ENDSORT statement must have the same value as the SESSION parameter in a corresponding SETSORT statement.

GETSORT Statement 1

The GETSORT statement lets you define a retrieval location for the sorted record. Once you issue a GETSORT statement, you cannot issue another PUTSORT statement for the same session until that session is ended with an ENDSORT.

NEXT/PRIOR/FIRST/LAST Parameters

One of these parameters is required and indicates to CA IDMS/DC Sort how assorted record should be retrieved.

NEXT

returns next sequential record in the sort queue.

PRIOR

returns previous sequential record in the queue.

FIRST

returns first sequential record in the sort queue.
returns the first record in the sort queue.

\textit{LAST}

returns the last record in the sort queue.

\textbf{PUTSORT Statement 1}

The PUTSORT statement transfers a record to CA IDMS/DC Sort.

\texttt{PUTSORT [ SESSION session-number] }<;}>

The \texttt{SESSION} Parameter

\texttt{SESSION session-number} is an optional parameter that identifies a sorting process for a single terminal user. The session-number can be a number from 0 to 9.

Default: The default value is 0.

The session-number in the PUTSORT statement must have the same value as the session-number in a corresponding SETSORT statement.

\textbf{SETSORT Statement 1}

A single SETSORT statement is required for each CA IDMS/DC Sort session. The SETSORT statement must be the first of the four statements which are coded into the application program.

The SETSORT statement identifies the particular session and indicates to CA IDMS/DC Sort the requirements of this session.
IDMSDB--SETSORT Statement

Exhibit 3.5: SETSORT Syntax

IDMSDB--SETSORT Statement (2)

Exhibit 3.6: SETSORT Syntax with IDMS Extension
SESSION Parameter

`SESSION session-number`

SESSION is an optional parameter that identifies a sorting process for a single terminal user. The session number can be a value from 0 to 9. This number allows you to differentiate concurrent sorting of various lists or of one list using different sort keys.

Default: The default value is 0.

Once the SESSION parameter has been set in the SETSORT statement, all sort criteria remain intact until the session is terminated with an ENDSORT statement. Within a particular task, a second SETSORT statement for the same session number cannot be issued until an ENDSORT statement has been issued. The same session number may then be reused, with a different set of sort criteria.

If a new task is begun and no ENDSORT statement for a given session number was issued in the old task, you can use a SETSORT statement with the same session number in the new task. In that case, CA IDMS/DC Sort automatically issues an ENDSORT for the session in the old task.

USER/PROGRAM Parameter

`/USER\ \PROGRAM/`

is a required parameter which indicates to CA IDMS/DC Sort whether sort criteria will be defined dynamically by the user at processing time, or within the program.

⚠️ **Note:** Blank spaces are not valid; only options USER and PROGRAM are valid.

**USER**

indicates that CA IDMS/DC Sort will prompt the terminal user for sort criteria on a screen at processing time. See Chapter 4, Examples for a description of selection screen.

**PROGRAM**

indicates that the sort criteria are defined in the program and the user cannot change them at processing time. When PROGRAM is specified, all sort criteria must be included in the SETSORT statement within the application program.

IDMS Parameter

`IDMS`
is an optional parameter which indicates to CA IDMS/DC Sort that it must access a CA IDMS dictionary for information about the specified record. If the application program is written in CA ADS, IDMS is assumed automatically.

DICTIONARY and NODE, which are explained on a later page, may be specified to further identify a CA IDMS dictionary.

record-name Parameter

record-name is a required parameter and specifies the name of the record that CA IDMS/DC Sort will use for put and get requests. There is a limitation in the manner DCSORT navigates the Integrated Data Dictionary to obtain the record element list when the IDMS option is used. This limitation means that only the last NAMESYN-083 within the SET SDR-NAMESYN will be recognized by DCSORT as a valid field name.

The IDD is navigated as follows to determine the element name:

- OBTAIN CALC SR-036 to obtain the correct record name and version.
- Repeat the DML command for all elements.
- OBTAIN NEXT SDR-042 WITHIN SET SR-SDT to obtain the SDR-042 junction record between record and element.
- OBTAIN LAST NAMESYN-083 WITHIN SET SDR-NAMESYN to obtain the element name.

For any SETSORT statement where IDMS is specified, or if your application is written in CA ADS, the record name must be the 01-level name of the record as it resides in the dictionary. For all other applications, it may be any symbolic item.

LENGTH Parameter

LENGTH record-length

is required if IDMS is not specified. It specifies the length of the record to be sorted. It may be specified as a numeric integer or as a symbolic data name that will satisfy an assignment to a halfword field.

If IDMS is specified, CA IDMS/DC Sort extracts the record length from the dictionary. If IDMS is specified and a length is specified, CA IDMS/DC Sort returns an error message.

VERSION Parameter

VERSION version-number

is an optional parameter used to further qualify a CA IDMS record if the IDMS keyword was used previously. If you specify VERSION without IDMS, CA IDMS/DC Sort returns an error message.
Version-number must be an unsigned integer from 0 to 9999.
Default: The default value is 1.

**DICTIONARY Parameter**

```
DICTIONARY dictionary-name
```

is an optional parameter which indicates to CA IDMS/DC Sort an alternate dictionary in which the CA
IDMS record resides. If you use the DICTIONARY parameter without the IDMS parameter, CA IDMS
/DC Sort returns an error message.

Dictionary-name must be a 1- to 8-character alphanumeric name.

Default: The primary dictionary.

**NODE Parameter**

```
NODE node-name
```

is an optional parameter which represents a DDS node in which a CA IDMS record resides. If you use
the NODE parameter without the IDMS parameter, CA IDMS/DC Sort returns an error message.

Node-name must be a 1- to 8-character alphanumeric name.

Defaults: For CA IDMS DME and TPSG interfaces, the default is the dictionary/node combination in
which the current MODULE-SOURCE is stored.

For batch preprocessors, the default may be specified in a DDDL format "signon" statement. For CA
ADS batch processing, the signon statement is left in place. For other languages, the signon
statement is removed.

**FIELDS Parameter**

The FIELDS parameter is required if you specified any of the following parameters:

- USER without IDMS
- PROGRAM

```
FIELDS field-name-1 field-length-1 [sort-order] .
   .
   .
   field-name-16 field-length-16 [sort-order]
```

Format 1 (USER without IDMS)

```
field-name field-length
```

You can define 1 to 16 sets of field-name and field-length.
Format 2 (PROGRAM without IDMS)

field-name field-length sort-order

You can define 1 to 16 sets of field-name, field-length, and sort-order.

Format 3 (PROGRAM with IDMS)

field-name sort-order

You can define 1 to 16 sets of field-name and sort-order.

- **field-name**
  - indicates the symbolic name of a data item that is subordinate to the record-name specified in the SETSORT statement. This field cannot be subscripted.

- **field-length**
  - indicates the length, in bytes, of the data item indicated by the field name.

- **sort-order**
  - is either **ASCENDING** or **DESCENDING**.

## SETLIMIT Statement 1

The SETLIMIT statement, if enabled, allows a developer to override system installed defaults for the amount of main and/or auxiliary storage per sort session, and for the minimum number of records to be placed in each sort buffer. These values are fully described under “Tuning Considerations” in Operations (https://docops.ca.com/display/IDMS19/Operations).

The SETLIMIT statement that applies to a particular session must appear after the SETSORT statement that identifies the session and before any PUTSORT statements for the session.

```
SETLIMIT [SESSION session-number] [MAIN n] [AUX n] [MINRBUF n] SESSION
```

- **session-number**
  - is an optional parameter which identifies a sorting process for a single terminal user. The session number can be an integer from 0 to 9.

- **MAINn**
  - is an optional parameter which allows you to specify the amount of main storage to be made available to CA IDMS/DC Sort where n is an integer from 0 to 9999999.

- **AUXn**
  - is an optional parameter which allows you to specify the amount of auxiliary storage to be made available to CA IDMS/DC Sort where n is an integer from 0 to 9999999.

- **MINRBUFn**
  - is an optional parameter which indicates how space is to be allocated to buffers at runtime where n is an integer from 0 to 9999999. The allocation of buffers also depends on the record length in a particular sort.
Demonstration

The CA IDMS tape contains a member that has information on how to run demonstrations. The member was downloaded to your source library during installation.

CA IDMS Environment

Read the instructions in TPSDEMO to find out how to run these CA IDMS demonstrations:

1. Test of CA IDMS/DC Sort runtime facilities. If the task TPS1 is specified, 50 random records are sorted on a key selected on the USER screen.

2. CA ADS prototype. An intentional syntax error is embedded in a SETSORT statement, in order to invoke the EDITOR.

CA IDMS/DC and CA IDMS UCF

The CA IDMS/DC system is central to the CA IDMS multiuser operating environment. CA IDMS/DC (or CA IDMS UCF) controls:

- Task management
- Terminal communications
- Scratch and queue management
- Storage and program management

Defining the System

You define the CA IDMS/DC or CA IDMS UCF system in the system dictionary through a process called system generation using the system generation compiler. The system definition includes:

- Definitions for system resources, programs, tasks, logs, and statistical reporting.
- Teleprocessing component definitions

⚠️ **Note:** For more information on CA IDMS/DC and CA IDMS UCF, see the CA IDMS Administering section.

CA IDMS/DB Components

Contents
CA IDMS - 19.0

- Database Management System (see page 71)
- Dictionaries (see page 71)
- Physical Database Definition (see page 72)
- Logical Database Definition (see page 72)

CA IDMS/DB components include the following:

- Database management system
- Dictionaries
- Physical database definition
- Logical database definition

Database Management System

The database management system makes it possible to access the data in your database. It ensures that the data is consistent and coordinates access to data through the use of locks. The DBMS provides data integrity through automatic recovery services and has a number of tuning options such as clustering, linked lists, and data compression.

Dictionaries

What is a Dictionary

To support the runtime environment, certain information is needed to define and control that environment. This information is stored in dictionaries.

A dictionary is a special CA IDMS defined database used to hold definitions of:

- Other databases
- CA IDMS/DC or CA IDMS UCF systems
- User-written applications

There are two kinds of dictionaries used in the CA IDMS environment: system dictionaries and application dictionaries.

System Dictionary

The system dictionary contains DC/UCF system definitions and physical database definitions.

There can be only one system dictionary in a runtime environment.

Application Dictionary

An application dictionary contains application definitions and logical database definitions. This includes records, relationships, areas, schemas, subschemas, maps, and dialogs.

There can be zero, one, or more application dictionaries in a runtime environment.
Physical Database Definition

In addition to defining the logical components of the database, you define the physical characteristics of the data and the environment in which it will be accessed. This is called the physical database definition.

The physical database definition includes the following:

- Segments, areas, and files that will hold the data
- Buffers used in retrieving and storing data
- Journal files used for recovery

The physical database definition is stored in the system dictionary, since it represents all data accessible through the runtime environment.

Logical Database Definition

The logical database definition identifies the user’s view of the data.

The logical database definition includes the following:

- Definition of records, tables, and views
- Definitions of relationships between these entities
- Specification of integrity rules
- Specification of indexes and other access keys

Logical database definitions reside in the application dictionary.

Tools for Database Definition and Maintenance

You define and maintain your database using a number of facilities.

**Command Facility**

The command facility is a tool used to enter:

- Physical database definition and maintenance statements
- SQL logical database definition and maintenance statements
Utility statements

It can be run in either online or batch mode.

Note: For more information on the command facility, see the Using section.

Schema, Subschema, and DDDL Compilers

The batch and online schema, subschema, and data dictionary definition language (DDDL) compilers are used to define and maintain the logical definition of non-SQL databases:

- Schema compiler -- Used to create a complete logical non-SQL database definition
- Subschema compiler -- Used to create a subset view of the logical database definition for use with application programs.
- DDDL compiler -- Used to create record and element definitions in the dictionary.

Utilities

You use utilities to perform maintenance operations on the database. Most utilities are executed as statements through the command facility; however, some are standalone programs.

Note: For more information on utilities, see the CA IDMS Administrating section.

CICS Environment

Read the instructions in TPSDEMOC to find out how to run the CICS demonstration:

Test of CA IDMS/DC Sort run-time facilities, including the USER screen. After you select a sort field and sort order, a display of 50 random records should appear in the proper sequence.

Parameter Statements Make CA IDMS/DC Sort Easy to Use

CA IDMS/DC Sort is easily controlled by five parameter statements:

- The SETSORT statement initiates the CA IDMS/DC Sort process. In addition, the SETSORT statement either explicitly states the criteria for a sort to be performed or invokes the user option, so that the user can specify the sort criteria.
The PUTSORT statement transfers a record to CA IDMS/DC Sort for processing.

The GETSORT statement retrieves a record after CA IDMS/DC Sort processing, (FIRST, LAST, NEXT or PRIOR).

The ENDSORT statement terminates the CA IDMS/DC Sort process.

The SETLIMIT statement overrides internal system storage limits.

By simply entering these five statements in the online program, you can have CA IDMS/DC Sort efficiently perform the sorts in whatever order you specify.

Then all you have to do is enter the code that directs the system to display data or use it in some other way.

Storage Requirements

For each sort session CA IDMS/DC Sort requires main and auxiliary storage.

- The MAIN storage default at installation time is 10000 bytes, maximum.
- The AUX (auxiliary) storage default at installation is 10000 bytes, maximum.

Online Program Storage

For COBOL and Assembler, runtime program storage requires a maximum of 38K.

- For CA ADS, the preprocessor requires a maximum of 147K (including the EDITOR).

Runtime program storage for CA ADS requires a maximum of 43K.

System Limits

The following limits are in effect for CA IDMS/DC Sort:

- Record size can be no greater than 32000 bytes for CA IDMS and 16000 bytes for CICS.
- Element size can be no greater than 256 bytes.
- Sort buffer size can be no greater than 32000 bytes for CA IDMS and 16000 bytes for CICS (See MINRBUF parameter in this section.)
- The maximum number of sort keys is 16.
- Up to ten sessions may operate concurrently at one terminal.
- All sorting is performed on a binary basis, in EBCDIC collating sequence.