

WRDC-TR-90-8007
Volume V
Part 9
Section 5 of 5



AD-A252 527



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume V - Common Data Model Subsystem
Part 9 - Neutral Data Manipulation Language (NDML) Precompiler
Development Specification
Section 5 of 5

J. Althoff, M. Apicella

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

DTIC
ELECTE
JUN 19 1992
S A D

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

92-16144



NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations


DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:


BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE			FORM APPROVED OMB NO. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE September 1990	3. REPORT TYPE AND DATES COVERED Final Technical Report 1Apr87 - 31Dec90	
4. TITLE AND SUBTITLE INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) Volume V - Common Data Model Subsystem Part 9 - Neutral Data Manipulation Language (NDML) Precompiler Development Specification Section 5 of 5			5. FUNDING NUMBERS Contract No.: F33600-87-C-0464 PE: 78011F Proj. No.: 595600 Task No.: P95600 WU: 20950607	
6. AUTHOR(S) J. Althoff, M. Apicella				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Controld Data Corporation Integration Technology Services 2970 Presidential Drive Fairborn, OH 45324-6209			8. PERFORMING ORGANIZATION REPORT NUMBER DS 6203, 1200	
9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES) Manufacturing Technology Directorate (WRDC/MTI) Wright-Patterson AFB, OH 45433-6533			10. SPONSORING/MONITORING AGENCY REP NUMBER WRDC-TR-90-8007, Vol. V, Part 9 Section 5 of 5	
11. SUPPLEMENTARY NOTES WRDC/MTI Project Priority 6203				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution is Unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT This development Specification (DS) describes the functions, performance, environment, interfaces, and design requirements for the Neutral Data Manipulation Language (NDML) Precompiler. The NDML Precompiler is a component of the Common Data Model Processor (CDMP) and it is used to generate various programs (e.g., request processor or RP, RP drivers, CS-ES transformers, and local subroutine callers) tailored to satisfy the NDML requests in a specific application program. This report is divided into five (5) sections.				
14. SUBJECT TERMS			15. NUMBER OF PAGES 885	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT SAR	18. SECURITY CLASS OF THIS PAGE SAR	19. SECURITY CLASS OF ABSTRACT SAR	20. LIMITATION ABSTRACT SAR	

SECTION 27

FUNCTION PRE10 - Build Calls and Messages

This function will control the processing logic for the generation of all code into the users application program to satisfy a NDML conceptual schema transaction.

This function:

1. Generates code into the procedure parcel of the application program which assembles information needed by the Distributed Request Supervisor into a message to satisfy an NDML request and sends that message to the Distributed Request Supervisor.
2. Generates code in the procedure parcel for receipt of results from the CS/ES Transformer and for presentation of the results to the user.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

27.1 Inputs

1. External Schema representation of the data
ES-ACTION-LIST
ES-QUALIFY-LIST
2. Conceptual schema representation of the data
CS-ACTION-LIST
CS-QUALIFY-LIST
3. Internal Schema representation of the data
IS-ACTION-LIST
IS-QUALIFY-LIST
4. Join Query Graph for the NDML request
JQG
JQG-ATTRIBUTE-PAIR-LIST

Accession For	
NTIS CRA&I	☒
DNC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

5. Result Field Table
RFT
6. Subtransaction table for the NDML request
SUBTRANS-PROCESS-ID-TABLE
7. Code generation table
CODE-GENERATOR-TABLE
8. Application Program parcel names
IDFILE-NAME
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
9. Application Program error file name
ERROR-FILE
10. Conceptual/External Schema transform program name
CS-ES-MOD-NAME
11. Source Language of the Application Program
SOURCE-LANGUAGE
12. User View Abbreviation List
UV-ABBR-LIST
13. Input-Output Section Indicator
IO-SECTION-INDICATOR
14. Block Stack
BLOCK-STACK
15. Logical Unit Work Name
LUW
16. First Inner Select Flag
FIRST-INNER-SELECT
17. Fortran Variable Association Table
FORTRAN-VARIABLE-TABLE
18. Target Host Name
TARGET-HOST

27.2 CDM Requirements

None

27.3 Internal Requirements

A temporary conceptual schema action list to be used during processing of inner selects of a query combination command. RET-STATUS and QCS-CDMP-CHECK-STATUS

27.4 Processing

1. Initialize the program variables and files.

- 1.1 Initialize return status of function to good status.
- 1.2 Open the four parcels of the users application program.
- 1.3 Determine the source language of the program and set local variable.

```
If SOURCE-LANGUAGE = "COBOL"
  set LANG-IND to 1
else
  set LANG-IND to -1
```

2. Determine if this is the start of a transaction and process the insert values, if it is also an Insert transaction.

- 2.1 Generate working storage required for each new NDML statement that is not an End Curly, Exit, Break, Next or Continue. Call "CDMACR" utility with the following:

```
Library Name - COBOL
Macro Name - UAPESWS
Parameters
  EE = ES-NDML-NO
```

```
Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - UAPESWS
Parameters
```

```
P1 = CDM-CS-RESULTS-FILE-ee
P2 = CDM-INPUT-NAME-ee
P3 = CDM-CS-RESULTS-ee
P4 = FCB-CDM-RESULTS-ee
P5 = FCB-INPUT-ee
P6 = CDM-INPUT-RETURN-LENGTH-ee
P7 = CDM-INPUT-RECORD-LENGTH-ee
```

where ee = ES-NDML-NO

30 September 1990

- 2.2 If it is not the case that ES-ACTION = "I" and ES-NDML-NO does not equal the previous ES-NDML-NO continue at step 2.7 otherwise continue at step 2.3.
- 2.3 Generate code for the start of the loop for an Insert command.
- Determine if the Insert is from a user file, user structure or a list of user values.
- 2.3.1 If ES-FILE-NAME = SPACE and ES-STRUCTURE = SPACE continue processing at step 2.4.
- 2.3.2 If ES-FILE-NAME NOT = SPACE continue processing at step 2.5.
- 2.3.3 If ES-STRUCTURE NOT = SPACE continue processing at step 2.6.
- 2.4 Generate code for an insert from a list of user values.
- 2.4.1 Determine how many rows of values the program will insert by counting the used ES-LOCAL-VARIABLE or ES-VALUE variables in the ES-ACTION-LIST and store the results in local variable ES-VALUE-USED.
- 2.4.2 Generate the temporary table to hold the insert values. Call "CDMACR" utility with the following:
- Library Name - COBOL
Macro Name - UAPWSI
Parameters
P1 = ES-VALUE-USED
EE = ES-NDML-NO
- Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - UAPWSI
Parameters
P1 = ES-VALUE-USED
P2 = CDM-INPUT-INDEX-ee
P3 = CDM-INPUT-USED-ee
- where ee = ES-NDML-NO
- 2.4.3 Generate data definitions for insert values.
Call function "CDP10E" with the following parameters.
- LANG-NO
IDFILE-NAME
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
ES-ACTION-LIST

ES-VALUE-USED
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

2.4.4 Generate code for the start of the insert loop.

Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - INSVAL1
Parameters
EE = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - INSVAL1
Parameters
P1 = CDM-INPUT-INDEX-ee
P2 = CDM-INPUT-USED-ee
EE = ES-NDML-NO

where ee = ES-NDML-NO

2.5 Generate the code for an insert from a user file.

2.5.1 Generate variable name into WS parcel if language is COBOL. Generate:

01 CDM-INPUT-ee.

where ee = ES-NDML-NO

2.5.2 Generate data definitions for insert values. Call function "CDP10E" with the following parameters:

LANG-NO
IDFILE-NAME
FDFILE -NAME
WORKFILE-NAME
PROCFIL-NAME
ES-ACTION-LIST
ES-VALUE-USED
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

2.5.3 Generate code for beginning of loop of insert from a file. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro name - INSFIL1
Parameters
EE = ES-NDML-NO
F1 = file name specified by the user.

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - INSFIL1
Parameters
P1 = CDM-INPUT-NAME-ee

P2 = FCB-INPUT-ee

P3 = CDM-INPUT-RECORD-LENGTH-ee
P5 = CDM-INPUT-ee
P6 = CDM-INPUT-RETURN-LENGTH-ee

2.5.4 Move 1 to START-POS.

2.5.5 Generate code that will move the insert values from the input record to the insert variables generated in the program. For each non-deleted entry in the ES-ACTION-LIST, perform steps 2.5.5.1 through 2.5.5.5.

2.5.5.1 If ES-TYPE = "C" or "I"
calculate END-POS = START-POS +
 ES-SIZE -1
If ES-TYPE = "F"
calculate END-POS = START-POS +
 ES-SIZE

2.5.5.2 If ES-TYPE = "C", generate:
ES-VAR-INS-ee-ii =
CDM-INPUT-ee(sp:ep)
where ee = ES-NDML-NO
 ii = ES-INDEX
 sp = START-POS
 ep = END-POS

2.5.5.3 If ES-TYPE = "I", generate:
CHAR-ES-VAR-INS-ee-ii =
CDM-INPUT-ee(sp:ep)
CALL
 CHRINT(CHAR-ES-VAR-INS-ee-ii,
 ES-VAR-INS-ee-ii, NDMLST)

where ee = ES-NDML-NO
 ii = ES-INDEX
 sp = START-POS
 ep = END-POS

2.5.5.4 If ES-TYPE = "F", generate:
CHAR-ES-VAR-INS-ee-ii =
CDM-INPUT-ee(sp:ep)
CALL
 CHREAL(CHAR-ES-VAR-INS-ee-ii,
 ES-VAR-INS-ee-ii, NDMLST)

 where ee = ES-NDML-NO
 ii = ES-INDEX
 sp = START-POS
 ep = END-POS

2.5.5.5 Calculate START-POS = END-POS +
 1.

2.6 Generate code for an insert from a user structure.

2.6.1 Generate an internal table to correspond to user's structure.

2.6.1.1 Generate the 01 level of the table if language is COBOL:

01 CDM-INPUT-eee.

where eee = ES-NDML-NO

2.6.1.2 Generate the data definitions for the table containing the insert values. Call function "CDP10E" with the following parameters:

LANG-NO
IDFILE-NAME
FDFILE-NAME
WORKFILE-NAME
PROCFE-NAME
ES-ACTION-LIST
ES-VALUE-USED
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

2.7 Set LAST-ES-NDML-NO to ES-NDML-NO.

3. Determine the type of Conceptual Schema transaction and update the parcels containing the users application source code.

3.1 If CS-ACTION = "S" (Select) or CS-ACTION = "Q" (Combination Query) go to step 4.

3.2 If CS-ACTION = "M" (Modify) go to step 5.

3.3 If CS-ACTION = "D" (Delete) go to step 6.

3.4 If CS-ACTION = "I" (Insert) go to step 7.

3.5 If CS-ACTION = "1" (Type 1 Referential Integrity) go to step 8.

3.6 If CS-ACTION = "2" (Type 2 Referential Integrity) go to step 9.

3.7 If CS-ACTION = "K" (Key Uniqueness) go to step 10.

3.8 If CS-ACTION = "B" (Begin) go to step 11.

3.9 If CS-ACTION = "C" (Commit) go to step 12.

3.10 If CS-ACTION = "R" (Rollback) go to step 13.

3.11 If CS-ACTION = "N" or (Next or Continue) go to step 14.

30 September 1990

- 3.12 If CS-ACTION = "E" (End Curley) go to step 15.
- 3.13 If CS-ACTION = "X" (Exit or Break) go to step 16.

4. Process a Select Conceptual Schema transaction.

- 4.0 If ES-SEMI-CURLY-IND not equal spaces, add an entry to the BLOCK-STACK.

4.0.1 Add 1 to BLOCK-INDEX.

4.0.2 Set MOD-NAME-STACK (BLOCK-INDEX) to CS-ES-MOD-NAME.

4.0.3 Set CS-NDML-NO-STACK (BLOCK-INDEX) to CS-NDML-NO.

- 4.1 Determine the type of SELECT command:

1. Select retrieved values into a user file.
2. Select retrieved values into a user structure.
3. Select retrieved values into user variables.
4. Inner Select of a Query combination command.

If ES-FILE-NAME NOT = SPACE continue processing at step 4.2.

If ES-STRUCTURE NOT = SPACE or ES-LOCAL-VARIABLE NOT = SPACE continue processing at step 4.3.

If ES-SELECT-COMB continue processing at step 4.4.

- 4.2 Process a Select where the results are to be stored in a user specified file.

4.2.1 Generate variable containing file name in WS parcel if language is COBOL:

O1 CDM-RESULTS-REC-eee

where eee = ES-NDML-NO

4.2.2 Generate variables to hold results

4.2.2.1 Call function "CDP1OF" with the following parameters:

LANG-NO
CS-ACTION-LIST
ES-ACTION-LIST
FDFILE-NAME

WORKFILE-NAME
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

4.2.2.2 If language is COBOL generate:

01 CDM-RESULTS-NAME-ee PIC
X(80).

else generate:

CHARACTER*80
CDM-RESULTS-NAME-ee

where ee = ES-NDML-NO

4.2.3 Generates code to transform runtime qualification values from external to conceptual schema format.

Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

4.2.4 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function "CDP10B" with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-LIST
TARGET-HOST

QCS-CDMP-CHECK-STATUS.

4.2.5 Generate code to call the DRS and receive status back.

4.2.5.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or
IBMFORTRAN
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 =
CDM-CS-RESULTS-FILE-ee

where ee = ES-NDML-NO
cc = CS-NDML-NO

4.2.5.2 If the CS-ACTION is not BEGIN, COMMIT, or ROLLBACK, call "CDMACR" utility with the following:

Library Name - COBOL or
VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

4.2.6 Generate code to initialize NDML-COUNT for the retrieval loop. Generate if COBOL:

MOVE ZERO TO NDML-COUNT.

Otherwise, generate:

NDMLCT = 0

4.2.7 Generate code to call the C/E Transform Program for the first time:

4.2.7.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 1
EE = ES-NDML-NO
CC = CS-NDML-NO if
ES-SEMI-CURLY-IND
equal spaces, otherwise use
CS-NDML-NO-STACK (BLOCK-INDEX)
MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND
equal spaces, otherwise use
MOD-NAME-STACK (BLOCK-INDEX)

Library Name - VAXFORTRAN or
IBMFORTRAN
Macro Name - CECALL
Parameters
P1 = '1'
MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND equal
spaces, otherwise use
MOD-NAME-STACK(BLOCK-INDEX)
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-cc
P4 = CDM-RESULTS-ee

where ee = ES-NDML-NO
cc = CS-NDML-NO if
ES-SEMI-CURLY-IND equal
spaces, otherwise use
MOD-NAME-STACK(BLOCK-INDEX)

4.2.7.2 If language is COBOL generate:

IF NOT CDM-CD-EOF
ADD 1 TO NDML-COUNT.

else generate:
IF (EOFFLA.NE.'1') NDMLCT =
NDMLCT + 1

4.2.7.3 Call "CDMACR" utility with the
following:

Library Name - COBOL or
VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHKCE
Parameters
EE = ES-NDML-NO

4.2.8 Generate code to move the values in the
result record to the named file.

30 September 1990

- 4.2.8.1 Call function CDP10C to generate the moves with the following parameters:

LANG-NO
PROCFIL-NAME
ES-ACTION-LIST
CS-NDML-NO
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

- 4.2.9 Generate code to save results to a user's file.

- 4.2.9.1 Generate code to begin saving results into user's file.

Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - FILSAV1
Parameters
EE = ES-NDML-NO
F1 = ES-FILE-NAME
(Variable or Constant)

Library Name - VAXFORTRAN or
IBMFORTRAN
Macro Name - FILSAV1
Parameters
EE = ES-NDML-NO
F1 = ES-FILE-NAME
(Variable or Constant)
P1 = CDM-RESULTS-NAME-ee
P2 = FCB-CDM-RESULTS-ee
P3 = CDM-RECORD-LENGTH-ee

where ee = ES-NDML-NO

- 4.2.10 Generate paragraph name for program loop to save results to a file.
Generate if COBOL:

LOOP-eee.

- 4.2.11 Generate code to save the null flag values for the retrieved data.

- 4.2.11.1 If COBOL, for each projected data item in the ES-ACTION-LIST, generate:

MOVE FLAG-X(ii) TO ES-NULL-cc-nn.

where ii = current index into the null flag array

cc = ES-NDML-NO
nn = ES-INDEX

4.2.11.2.1 Calculate the number
of non-deleted
ES-ACTION-LIST
entries, and set
REAL-ES-USED.

4.2.11.2.2 Generate:

CDM-RESULTS-REC-ee(
1:rr) =
FLAGAR(1:rr)

where ee =
ES-NDML-NO
rr =
REAL-ES-USED

4.2.12 Generate code to write the results to
the user specified file.

4.2.12.1 If COBOL, generate:

MOVE CDM-RESULTS-eee TO
CDM-RESULTS-RECORD-eee.

where eee = ES-NDML-NO

4.2.12.2 If FORTRAN, perform steps
4.2.12.2.1 through 4.7.12.2.5
for each projected data item in
the ES-ACTION-LIST. Initialize
START-POS to 1 and START-POSF to
REAL-ES-USED plus 1.

4.2.12.2.1 If ES-FCTN-NAME is "COUNT",
perform steps 4.2.12.2.1.1
through 4.2.12.2.1.3.

4.2.12.2.1.1 Set END-POS
equal START-POS
+8.
Set END-POSF
equal START-POSF
+8.

4.2.12.2.1.2 Generate:
CALL CONDIG
(CDM-RESULTS-ee
(ep:ep), SIGN,
DIGIT, NDMLST)
CDM-RESULTS-REC-
ee
(sf:ef) =
CDM-RESULTS-ee
(sp:ep)
where ee =
ES-NDML-NO
ep = END-POS

DS 620341200
30 September 1990
sp = START-POS
ef = END-POSF
sf = START-POSF

- 4.2.12.2.1.3 Set START-POS
equal END-POS
plus 1.
Set START-POSF
equal END-POSF
plus 1.
Continue at step
4.2.12.2.
- 4.2.12.2.2 If ES-FCTN-NAME is equal to
"MEAN", or "AVG," or "SUM,"
perform steps 4.2.12.2.2.1
through 4.2.12.2.2.3.
- 4.2.12.2.2.1 Set START-POS
equal START-POS
plus 8. Set
END-POSF equal
START-POSF plus
18.
- 4.2.12.2.2.2 Generate:
DECIML = 9
CALL RELFTN
(DECIML,
ES-RES-cc-ii,
LONG-ES-RES-cc-i
i,CDM-RESULTS-RE
C-ee
(sf:ef))
where cc =
CS-NDML-NO
ii = ES-INDEX
sf = STAT-POSF
ef = END-POSF
ee = ES-NDML-NO
- 4.2.12.2.2.3 Set START-POSF
equal END-POSF
plus 1.
Continue at step
4.2.12.2.
- 4.2.12.2.3 If ES-TYPE equals "I", perform
steps 4.2.12.2.3.1 through
4.2.12.2.3.3.
- 4.2.12.2.3.1 Set END-POSF
equal START-POSF
plus 9.
- 4.2.12.2.3.2 Generate:
DIGIT =
ES-RES-cc-ii
Call INTFIN

DS 620341200
30 September 1990
(DIGIT,
CDM-RESULTS-REC-
ee
(sf:ef))
where cc =
CS-NDML-NO
ii = ES-INDEX
ee = ES-NDML-NO
sf = START-POSF
ef = END-POSF

4.2.12.2.3.3 Set START-POSF
equal END-POSF
plus 1.
If ES-SIZE is
greater than 4
Set START-POS
equal
START-POS plus
4.
Else
Set START-POS
equal
START-POS plus
2.
Continue at step
4.2.12.1.

4.2.12.2.4 If ES-TYPE equals "F", same
processing as step 4.2.12.2.2.

4.2.12.2.5 If ES-TYPE equals "C", perform
steps 4.2.12.2.5.1 through
4.2.12.2.5.3.

4.2.12.2.5.1 Set END-POSF
equal START-POSF
plus ES-SIZE
minus 1.
Set END-POS
equal START-POS
plus ES-SIZE
minus 1.

4.2.12.2.5.2 Generate:
CDM-RESULTS-REC-
ee (sf:ef) =
CDM-RESULTS-ee(s
p:ep)
where ee =
ES-NDML-NO
sf = START-POSF
ef = END-POSF
sp = START-POS
ep = END-POS

4.2.12.2.5.3 Set START-POS
equal END-POS
plus 1.

DS 620341200
30 September 1990
Set START-POSF
equal END-POSF
plus 1.
Continue at step
4.2.12.2.

4.2.12.3 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - UAPWR
Parameters
EE = ES-NDML-NO

Library Name - VAXFORTRAN or
IBMFORTTRAN
Macro Name - UAPWR
Parameters
P1 = FCB-CDM-RESULTS-ee
P2 = CDM-RESULTS-REC-ee
P3 = CDM-RECORD-LENGTH-ee
where ee = ES-NDML-NO

4.2.13 Generate code to call the C/E Transform Program for the 2-N time.

4.2.13.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 2
EE = ES-NDML-NO
CC = CS-NDML-NO if ES-SEMI-CURLY-IND
equal spaces, otherwise use
CS-NDML-NO-STACK (BLOCK-INDEX)
MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND
equal spaces, otherwise use
MOD-NAME-STACK (BLOCK-INDEX)

Library Name - VAXFORTRAN or IBMFORTTRAN
Macro Name - CECALL
Parameters
P1 = '2'
MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND equal spaces,
otherwise use MOD-NAME-STACK
(BLOCK-INDEX)
P2 = CDM-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-cc
P4 = CDM-RESULTS-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO if ES-SEMI-CURLY-IND
equal spaces, otherwise use
MOD-NAME-STACK (BLOCK-INDEX)

4.2.13.2 If language is COBOL, generate:

IF NOT CDM-CE-EOF
ADD 1 TO NDML-COUNT.

else generate:

IF (EOFFLA.NE.'1') NDMLCT =
NDMLCT +1

4.2.13.3 Call "CDMACR" utility with the
following:

Library Name = COBOL or
VAXFORTRAN or
IBMFORTRAN
Macro Name = ERRCHK
Parameters
EE = ES-NDML-NO

4.2.14 Generate code to move the values in the
result record to the named variables,
structure or file.

4.2.14.1 Call function CDP10C to generate
the moves with the following
parameters:

LANG-NO
PROCFIle-NAME
ES-ACTION-LIST
CS-NDML-NO
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

4.2.15 Generate code for completion of the loop
for saving results into a file. Call
"CDMACR" utility with the following:

Library Name - COBOL
Macro Name - FILSAV2
Parameters
EE = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - FILSAV2
Parameters
EE = ES-NDML-NO
P1 = FCB-CDM-RESULTS-ee
where ee = ES-NDML-NO

4.2.16 Continue processing at step 17.

4.3 Process a Select where the results are to be
stored in a user specified structure or user
variables.

30 September 1990

- 4.3.1 Generate working storage definition for the 01 level of the results. Generate if COBOL:

01 CDM-RESULTS-eee.

where eee = ES-NDML-NO

- 4.3.2 Generate variables to hold results.

Call function CDP10F with the following parameters:

LANG-NO
CS-ACTION-LIST
ES-ACTION-LIST
FDFILE-NAME
WORKFILE-NAME
FORTRAN-VARIABLE-TABLE
QCS-CDMP-CHECK-STATUS

- 4.3.3 Generate code to transform runtime qualification values from External to Conceptual Schema format.

Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 4.3.4 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor.

Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO

JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

4.3.5 Generate code to call the DRS and receive status back.

4.3.5.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or
IBMFORTRAN
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee

where ee = ES-NDML-NO
cc = CS-NDML-NO

4.3.5.2 Call "CDMACR" utility with the following:

Library Name - COBOL or
VAXFORTRAN or IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

4.3.6 Generate code to bypass the call to the C/E transform program if no results were retrieved. Generate if COBOL:

IF NDML-COUNT = 0
GO TO END-NDML-eee.

else generate:

IF (CHARCT.EQ. '00000') GO TO 93eee

where eee = NDML-NO

- 4.3.7 Generate code to initialize NDML-COUNT for the retrieval loop. Generate if COBOL:

MOVE ZERO TO NDML-COUNT.

else generate:

NDMLCT = 0

- 4.3.8 Generate code to call the C/E Transform Program for the first time.

- 4.3.8.1 Call "CDMACR" utility with the following:

Library Name - COBOL

Macro Name - CECALL

Parameters

P1 = 1

EE = ES-NDML-NO

CC = CS-NDML-NO if
ES-SEMI-CURLY-IND equal
spaces, otherwise use

CS-NDML-NO-STACK
(BLOCK-INDEX)

MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND
equal spaces, otherwise
use MOD-NAME-STACK
(BLOCK-INDEX)

Library Name - VAXFORTRAN or
IBMFORTRAN

Macro Name - CECALL

Parameters

P1 = '1'

MMMMM = CS-ES-MOD-NAME if
ES-SEMI-CURLY-IND
equal spaces, otherwise
use MOD-NAME-STACK
(BLOCK-INDEX)

P2 = CDM-CS-RESULTS-FILE-ee

P3 = CDM-CSQ-TABLE-cc

P4 = CDM-RESULTS-ee

where ee = ES-NDML-NO
cc = CS-NDML-NO if
ES-SEMI-CURLY-IND
equal spaces, otherwise
use CS-NDML-NO-STACK
(BLOCK-INDEX)

- 4.3.8.2 If language is COBOL, generate:

IF NOT CDM-CD-EOF
ADD 1 TO NDML-COUNT.

else generate:

IF (EOFFLA.NE.'1') NDMLCT =
NDMLCT +1

4.3.8.3 Call 'CDMACR' utility with the
following:

Library Name - COBOL or
VAXFORTRAN or

IBMFORTRAN

Macro Name - ERRCHKCE

Parameters

EE = ES-NDML-NO

4.3.9 Generate code to move the values in the result
record to the named variables, structure or file.

4.3.9.1 Generate paragraph name for program loop
of saving results. Generate if COBOL:

LOOP-eee.

else generate:

94eee

where eee = ES-NDML-NO

4.3.9.2 Call function "CDP10C" to generate the
moves with the following parameters:

LANG-NO

PROCFIL-NAME

ES-ACTION-LIST

CS-NDML-NO

FORTRAN-VARIABLE-TABLE

QCS-CDMP-CHECK-STATUS

4.3.10 Generate code to close and delete the results file
and terminate the loop structure for SELECT into
variables or structure that did not have an NDML
loop structure.

If ES-SEMI-CURLY-IND equal spaces generate code to
call the C/E Transform Program to close and delete
the results file.

4.3.10.1 Call "CDMACR" utility with the following:

Library Name - COBOL

Macro Name - CECALL

Parameters

P1 = 3

EE = ES-NDML-NO

CC = CS-NDML-NO if ES-SEMI-CURLY-IND equal
spaces, otherwise use
CS-NDML-NO-STACK(BLOCK-INDEX)

30 September 1990

MMMMM = CS-ES-MOD-NAME if
 ES-SEMI-CURLY-IND
 equal spaces otherwise use
 MOD-NAME-STACK(BLOCK-INDEX)

Library Name - VAXFORTRAN or IBMFORTRAN

Macro Name - CECALL

Parameters

P1 = '3'

MMMMM = CS-ES-MOD-NAME if
 ES-SEMI-CURLY-IND
 equal spaces, otherwise use
 MOD-NAME-STACK (BLOCK-INDEX)

P2 = CDM-CS-RESULTS-FILE-ee

P3 = CDM-CSQ-TABLE-cc

P4 = CDM-RESULTS-ee

where ee = ES-NDML-NO

cc = CS-NDML-NO if ES-SEMI-CURLY-IND
 equal spaces, otherwise use
 CS-NDML-NO-STACK (BLOCK-INDEX)

4.3.10.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
 IBMFORTRAN

Macro Name - ERRCHK

Parameters

EE = ES-NDML-NO

4.3.10.3 If ES-SEMI-CURLY-IND equal space generate closing loop structure. Generate if COBOL:

END-NDML-eee.

else generate:

93eee

where eee = ES-NDML-NO

4.3.11 Continue processing at step 17.

4.4 Process an inner select of a Query combination command.

4.4.1 Determine if each inner Select has the same number of attributes to be retrieved and if each attribute matches in data type.

4.4.1.1 Determine if this is first inner Select for Query combination command, and if it is, then populate SAVE-CS-ACTION-LIST.

If FIRST-INNER-SELECT flag is not set, calculate REAL-CS-USED by counting only CS-ENTRIES that are not generated or have not been deleted. Transfer REAL-CS-USED to SAVE-CS-USED and transfer each used

CS-TYPE to SAVE-CS-TYPE. Set the FIRST-INNER-SELECT flag to indicate we have processed the first inner Select of the Query combination command.

If FIRST-INNER-SELECT flag has been set, just calculate REAL-CS-USED by the method described in the above paragraph.

- 4.4.1.2 Check to see that the number of attributes match.

If REAL-CS-USED NOT = SAVE-CS-USED issue an error message, set function status to bad status and exit processing of CDP10.

- 4.4.1.3 Check to see that the data type of each attribute matches.

For each used entry in the CS-ACTION-LIST if CS-TYPE NOT = SAVE-CS-TYPE issue an error message, set function status to bad status and exit processing of CDP10.

- 4.4.2 Generate working storage variables to hold the names of result files from DRS and CS selector and CS count. Generate if COBOL:

```
01 CDM-CS-RESULTS-eee      PIC X(80).  
01 CDM-CS-COUNT-eee       PIC 9(6).
```

else generate:

```
CHARACTER*80 CDM-CS-RESULTS-eee  
CHARACTER*6  CDM-CS-COUNT-eee
```

where eee = CS-NDML-NO

- 4.4.3 Generate code to transform runtime qualification values from External to Conceptual Schema format. Call function CDP10A with the following parameters:

```
LANG-NO  
FDFILE-NAME  
WORKFILE-NAME  
PROCFILE-NAME  
CS-ACTION-LIST  
CS-QUALIFY-LIST  
ES-ACTION-LIST  
ES-QUALIFY-LIST  
IS-ACTION-LIST  
IS-QUALIFY-LIST  
UV-ABBR-LIST  
CODE-GENERATOR-TABLE  
SUBTRANS-PROCESS-ID-TABLE  
NEXT-PARAMETER-NO  
ERROR-FILE  
LUW
```

FORTTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 4.4.4 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor.

Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTTRAN-VARIABLE-TABLE
QCS-CDM-CHECK-STATUS

- 4.4.5 Generate code to call the DRS and receive status back.

- 4.4.5.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTTRAN
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

- 4.4.5.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

- 4.4.6 Generate code to call the CS selector program to obtain the final file of Conceptual Schema results. Call the "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CCCALL
Parameters
P2 = ES-NDML-NO
P3 = CS-NDML-NO
P1 = CS-ES-MOD-NAME

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CCCALL
Parameters
P1 = CS-ES-MOD-NAME
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-cc
P4 = CDM-CS-RESULTS-cc
P5 = CDM-CS-COUNT-cc
where ee = ES-NDML-NO
cc = CS-NDML-NO

- 4.4.7 Generate code to perform error checking for the C/C transformer call. Call the "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

- 4.4.8 Continue processing at step 17.

5. Process a Modify Conceptual Schema transaction.

- 5.1 Generates code to transform runtime qualification/update values from External to Conceptual Schema format. Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 5.2 Generate code to test for assertion/constraints.
Call function CDCONS with the following parameters:

LANG-NO
CS-ACTION-LIST
CS-QUALIFY-LIST
WORKFILE-NAME
PROCFILE-NAME
NEXT-PARAMETER-NO
ES-NDML-NO
ERROR-FILE
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 5.3 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 5.4 Generate code to call the DRS and receive status back.

- 5.4.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

IBMFORTRAN Library Name - VAXFORTRAN or
Macro Name - DRSCALL
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

5.4.2 Call "CDMACR" utility with the following:

Library Name - COBOL or
VAXFORTRAN or IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

5.5 Generate code to define the bypass point for the
command. Generate if
COBOL:

END-NDML-eee.

else generate:

93eee

where eee = ES-NDML-NO

5.6 Continue processing at step 17.

6. Process a Delete Conceptual Schema transaction.

6.1 Generate code to transform runtime qualification
values from External to Conceptual Schema format.
Call function CDP10A with the following
parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

6.2 Generate code to test for assertion/constraints.
Call function CDCONS with the following
parameters:

LANG-NO
CS-ACTION-LIST
CS-QUALIFY-LIST
WORKFILE-NAME
PROCFILE-NAME

NEXT-PARAMETER-NO
ES-NDML-NO
ERROR-FILE
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 6.3 Generate code to transfer the precompiler tables required at runtime for the distributed request supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 6.4 Generate code to call the DRS and receive status back.

- 6.4.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO
Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - DRSCALL
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

- 6.4.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

- 6.5 Generate code to define the bypass point for the command. Generate if COBOL:

END-NDML-eee.

else generate:

93eee

where eee = ES-NDML-NO

6.6 Continue processing at step 17.

7. Process an Insert Conceptual Schema transaction.

7.1 Generate code to transform runtime insert values from External to Conceptual Schema format. Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

7.2 Generate code to test for assertion/constraints. Call function CDCONS with the following parameters:

LANG-NO
CS-ACTION-LIST
CS-QUALIFY-LIST
WORKFILE-NAME
PROCFILE-NAME
NEXT-PARAMETER-NO
ES-NDML-NO
ERROR-FILE
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

7.3 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

7.4 Generate code to call the DRS and receive status back.

7.4.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - DRSCALL

Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

7.4.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

7.5 Generate code for the termination of the loop for insert values

7.5.1 If ES-STRUCTURE NOT = SPACE generate if COBOL:

END-NDML-eee.
BREAK-NDML-eee.

else generate:

93eee
95eee

where eee = ES-NDML-NO

Continue processing at step 7.6.

7.5.2 If ES-FILE-NAME NOT = SPACE
MACRO-NAME = INSFIL2
else
MACRO-NAME = INSVAL2

Call "CDMACR" utility with the
following:

Library Name - COBOL
Macro Name - from above
Parameters
EE = ES-NDML-NO

IBMFORTRAN
Library Name - VAXFORTRAN or
Macro Name - from above
Parameters
EE = ES-NDML-NO
P1 = FCB-INPUT-ee
(if MACRO NAME is INSFIL2)
where ee = ES-NDML-NO

7.6 Continue processing at step 17.

8. Process a Referential Integrity Type 1 Conceptual Schema transaction.

8.1 Generate code to transform runtime qualification value from External to Conceptual Schema format. Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 8.2 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 8.3 Generate code to call the DRS and receive status back.

8.3.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTAN
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

8.3.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

- 8.4 Generate code to call the CS selector program to obtain the final count of results from the Referential Integrity Type 1 transaction. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CS2CALL
Parameters
P1 = CS-ES-MOD-NAME
P2 = ES-NDML-NO
P3 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CS2CALL
Parameters
P1 = CS-ES-MOD-NAME
P2 = ES-NDML-NO
P3 = CDM-CS-RESULTS-FILE-ee
P4 = CDM-CSQ-TABLE-cc
where ee = ES-NDML-NO
cc = CS-NDML-NO

- 8.5 Generate code for the test of the results of a Referential Integrity Type 1 test. Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or IBMFORTRAN
Macro Name - RITCHK
Parameters
OP = SPACE
SIGN = "=" if COBOL
".EQ." if FORTRAN
EE = ES-NDML-NO
ECODE = 49901

- 8.6 Continue processing at step 17.

9. Process a Referential Integrity Type 2 Conceptual Schema transaction.

- 9.1 Generate code to transfer runtime qualification values from External to Conceptual Schema format. Call function CDP10A with the following parameters:

LANG-NO
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
IS-ACTION-LIST
IS-QUALIFY-LIST
UV-ABBR-LIST
CODE-GENERATOR-TABLE
SUBTRANS-PROCESS-ID-TABLE
NEXT-PARAMETER-NO
ERROR-FILE
LUW
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 9.2 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFIELD-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 9.3 Generate code to call the DRS and receive status back.

- 9.3.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters

P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - DRSCALL

Parameters

P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee

where ee = ES-NDML-NO
cc = CS-NDML-NO

- 9.3.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN

Macro Name - ERRCHK

Parameters

EE = ES-NDML-NO

- 9.4 Generate code to call the CS selector program to obtain the final count of results from the Referential Integrity Type 2 transaction. Call "CDMACR" utility with the following:

30 September 1990

Library Name - COBOL

Macro Name - CS2CALL

Parameters

P1 = CS-ES-MOD-NAME

P2 = ES-NDML-NO

P3 = CS-NDML-NO

Library Name -VAXFORTRAN or
IBMFORTRAN

Macro Name - CS2CALL

Parameters

P1 = CS-ES-MOD-NAME

P2 = ES-NDML-NO

P3 = CDM-CS-RESULTS-FILE-ee

P4 = CDM-CSQ-TABLE-cc

where ee = ES-NDML-NO

cc = CS-NDML-NO

- 9.5 Generate code for the test of the results of a Referential Integrity Type 2 test. Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN

Macro Name - RITCHK

Parameters

OP = "NOT" if COBOL
spaces if FORTRANSIGN = "=" if COBOL
".NE." if FORTRAN

EE = ES-NDML-NO

ECODE = 49902

- 9.6 Continue processing at step 17.

10. Process a Key Uniqueness Conceptual Schema transaction.

- 10.1 Generate code to transform runtime qualification values from external to conceptual schema format. Call function CDP10A with the following parameters:

LANG-NO
 FDFILE-NAME
 WORKFILE-NAME
 PROCFILE-NAME
 CS-ACTION-LIST
 CS-QUALIFY-LIST
 ES-ACTION-LIST
 ES-QUALIFY-LIST
 ES-ACTION-LIST
 IS-QUALIFY-LIST
 UV-ABBR-LIST
 CODE-GENERATOR-TABLE
 SUBTRANS-PROCESS-ID-TABLE
 NEXT-PARAMETER-NO
 ERROR-FILE
 LUW

FORTTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 10.2 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 10.3 Generate code to call the DRS and receive status back.

- 10.3.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO

VAXFORTRAN or IBMFORTTRAN	Library Name -
DRSCALL	Macro Name -
	Parameters
	P1 = SUB-USED
	P2 = CS-ACTION
CDM-POOL-ee-cc	P3 =
CDM-CSAL-ee-cc	P4 =
CDM-JQG-ee-cc	P5 =
CDM-APL-ee-cc	P6 =
CDM-RFT-ee-cc	P7 =
CDM-CS-RESULTS-FILE-ee	P8 =
= ES-NDML-NO	where ee
CS-NDML-NO	cc =

30 September 1990

10.3.2 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

10.4 Generate code to call the CS selector program to obtain the final count of results from the Key Uniqueness Referential Integrity transaction. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CS2CALL
Parameters
P1 = CS-ES-MOD-NAME
P2 = ES-NDML-NO
P3 = CS-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CS2CALL
Parameters
P1 = CS-ES-MOD-NAME
P2 = ES-NDML-NO
P3 = CDM-CS-RESULTS-FILE-ee
P4 = CDM-CSQ-TABLE-cc
where ee = ES-NDML-NO
cc = CS-NDML-NO

10.5 Generate code for the test of the results of a Key Uniqueness test. Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTTRAN
Macro Name - RITCHK
Parameters
OP = "NOT" if COBOL
spaces if FORTRAN
SIGN = "=" if COBOL
".NE." if FORTRAN
EE = ES-NDML-NO
ECODE = 49903

10.6 Continue processing at step 17.

11. Process a Begin Conceptual Schema transaction.

11.1 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10F with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFIL-NAME
ES-NDML-NO

JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 11.2 Generate code to call the DRS and receive status back.
Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO
Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - DRSCALL
Parameters
P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

- 11.3 Continue processing at step 17.

12. Process a Commit transaction.

- 12.1 Generate code to transfer the precompiler tables
required at runtime for the Distributed Request
Supervisor. Call function CDP10B with the
following parameters:

LANG-NO
WORKFILE-NAME
PROCFILE-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

- 12.2 Generate code to call the DRS and receive status
back. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters

DS 620341200
30 September 1990

P1 = SUB-USED
P2 = CS-ACTION
P3 = ES-NDML-NO
P4 = CS-NDML-NO
Library Name - VAXFORTRAN or
 IBMFORTTRAN

Macro Name - DRSCALL

Parameters

P1 = SUB-USED
P2 = CS-ACTION
P3 = CDM-POOL-ee-cc
P4 = CDM-CSAL-ee-cc
P5 = CDM-JQG-ee-cc
P6 = CDM-APL-ee-cc
P7 = CDM-RFT-ee-cc
P8 =

CDM-CS-RESULTS-FILE-ee

where ee = ES-NDML-NO
 cc = CS-NDML-NO

12.3 Continue processing at step 17.

13. Process a Rollback transaction.

13.1 Generate code to transfer the precompiler tables required at runtime for the Distributed Request Supervisor. Call function CDP10B with the following parameters:

LANG-NO
WORKFILE-NAME
PROCFIL-NAME
ES-NDML-NO
JQG
JQG-ATTRIBUTE-PAIR-LIST
RFT
CS-ACTION-LIST
FORTRAN-VARIABLE-TABLE
TARGET-HOST
QCS-CDMP-CHECK-STATUS

13.2 Generate code to call the DRS and receive status back. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - DRSCALL
Parameters
 P1 = SUB-USED
 P2 = CS-ACTION
 P3 = ES-NDML-NO
 P4 = CS-NDML-NO
Library Name - VAXFORTRAN or IBMFORTTRAN
Macro Name - DRSCALL
Parameters
 P1 = SUB-USED
 P2 = CS-ACTION
 P3 = CDM-POOL-ee-cc
 P4 = CDM-CSAL-ee-cc
 P5 = CDM-JQG-ee-cc
 P6 = CDM-APL-ee-cc

P7 = CDM-RFT-ee-cc
P8 = CDM-CS-RESULTS-FILE-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO

13.3 Continue processing at step 17.

14. Process a NEXT/CONTINUE transaction.

Generate code to leave the current retrieval loop.
Generate if COBOL:

GO TO CE-LOOP-eee

else generate:

GO TO 92eee

where eee = ES-NDML-NO

Continue processing at step 17.

15. Process an End Curly (E) Conceptual Schema transaction.

15.1 If BLOCK-INDEX = 0
Go to 15.8.

15.2 Generate label for start of retrieval loop.
Generate if COBOL:

CE-LOOP-eee.

else generate:

92eee

where eee = ES-NDML-NO

15.3 Generate code to call the C/E transformer for the
next record. Call the "CDMACR" utility with the
following:

Library Name - COBOL

Macro Name - CECALL

Parameters

P1 = 2

EE = ES-NDML-NO

CC = CS-NDML-NO-STACK(BLOCK-INDEX)

MMMMM = MOD-NAME-STACK(BLOCK-INDEX)

Library Name - VAXFORTRAN or IBMFORTAN

Macro Name - CECALL

Parameters

P1 = '2'

MMMMM = MOD-NAME-STACK(BLOCK-INDEX)

P2 = CDM-CS-RESULTS-FILE-ee

P3 = CDM-CSQ-TABLE-cc

P4 = CDM-RESULTS-ee

where ee = ES-NDML-NO

cc = CS-NDML-NO-STACK(BLOCK-INDEX)

15.4 If language is COBOL generate:

IF NOT CDM-CE-EOF
ADD 1 TO NDML-COUNT.

else generate:

If (EDFFLA.NE. '1') NDMLCT = NDMLCT + 1

15.5 Generate code to perform error checking for the
C/E transformer call. Call the "CDMACR" utility
with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

15.6 Generate code to terminate the retrieval loop,
the break point and the bypass point. Call
"CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ENDLOOP
Parameters
EE = ES-NDML-NO

15.7 Subtract 1 from BLOCK-INDEX

15.8 Continue processing at step 17.

16. Process an EXIT or BREAK transaction.

16.1 If BLOCK-INDEX = 0
Go to 17.

16.2 Generate code to call the C/E transformer to close all
files. Call the "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 3
EE = ES-NDML-NO
CC = CS-NDML-NO-STACK(BLOCK-INDEX)
MMMMM = MOD-NAME-STACK(BLOCK-INDEX)
Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CECALL
Parameters
P1 = '3'
MMMMM = MOD-NAME-STACK(BLOCK-INDEX)
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-cc
P4 = CDM-RESULTS-ee
where ee = ES-NDML-NO
cc = CS-NDML-NO-STACK(BLOCK-INDEX)

- 16.3 Generate code to exit the retrieval loop.
Generate if COBOL:

GO TO BREAK-eee

else generate:

GO TO 9leee

where eee = ES-NDML-NO

17. Terminate processing of function PRE10.

17.1 Close the four parcels of the user's application
program.

17.2 Set the function status variable and exit processing.

27.5 Outputs

1. Error status of the function

RETURN-STATUS

2. Code generated into the parcels of the Application
Program.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - CECALL

PARAMETERS - P1
 EE
 MMMM

*
*
*

CALL CS-ES-TRANSFORM

MOVE P1 TO CDM-CE-FLAG
CALL "MMMM" USING
 CDM-CE-FLAG
 CDM-CS-RESULTS-FILE-EE
 CDM-CSQ-TABLE-CC
 CDM-FLAG-ARRAY
 CDM-RESULTS-EE
 CDM-CE-EOF-FLAG
 NDML-STATUS

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - DRSCALL

PARAMETERS - P1
 P2
 P3
 P4

*

* CALL THE DRS:

*

MOVE P1 TO CDM-NO-SUBTRANS
MOVE "P2" TO CDM-DRS-ACTION
CALL "CDS01" USING
 CDM-NO-SUBTRANS
 CDM-DRS-ACTION
 CDM-POOL-P3-P4
 CDM-CSAL-P3-P4
 CDM-JQG-P3-P4
 CDM-APL-P3-P4
 CDM-RFT-P3-P4

*

CDM-CS-RESULTS-FILE-P3
NDML-COUNT
NDML-STATUS

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME- ENDLOOP

PARAMETERS - EE

```
*  
*   TERMINATION OF RETRIEVAL LOOP  
*  
    IF NOT CDM-CE-EOF  
      GO TO LOOP-EE.  
BREAK-EE.  
END-NDML-EE.
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - ERRCHK

PARAMETERS - EE

IF NOT OK
GO TO END-NDML-EE.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - ERRCHKCE

PARAMETERS - EE

IF NOT OK OR CDM-CE-EOF
GO TO END-NDML-EE.

LIRARY NAME - COBOL

MACRO NAME - FILSAV1

PARAMETERS - EE

```
*  
* BEGIN SAVING RESULTS INTO USERS FILE  
*  
  IF CDM-CE-EOF  
      GO TO END-NDML-EE.  
  MOVE F1 TO CDM-RESULTS-NAME-EE.  
  MOVE "W" TO DISPOSITION.  
  CALL "OPNFIL" USING FCB-CDM-RESULTS-EE,  
      RET-STATUS,  
      CDM-RESULTS-NAME-EE,  
      DISPOSITION,  
      CDM-RECORD-LENGTH-EE,  
      NUMBER-OF-RECORDS.  
  IF RET-STATUS NOT = KES-FILE-OK  
      MOVE "ERROR OPENING FILE CDM-RESULTS-NAME-EE"  
          TO MESG-DESC  
      PERFORM PROCESS-ERROR  
      GO TO EXIT-PROGRAM.
```

LIBRARY NAME - COBOL

MACRO NAME - FILSAV2

PARAMETERS - EE

*

* COMPLETION OF LOOP SAVING RESULTS INFO A FILE

*

IF NOT CDM-CE-EOF

GO TO LOOP-EE.

MOVE "K" TO DISPOSITION.

CALL "CLSFIL" USING FCB-CDM-RESULTS-EE,
RET-STATUS,
DISPOSITION.

IF RET-STATUS NOT = KES-FILE-OK

MOVE "ERROR CLOSING FILE CDM-RESULTS-NAME-EE"
TO MMSG-DESC

PERFORM PROCESS-ERROR

GO TO EXIT-PROGRAM.

END-NDML-EE.

LIBRARY NAME - COBOL

MACRO NAME - INSFIL1

PARAMETERS - EE
 F1

```
*
*   BEGINNING OF LOOP OF INSERT FROM A FILE
*
MOVE F1 TO CDM-INPUT-NAME-EE.
MOVE "R" TO DISPOSITION.
CALL "OPNFIL" USING FCB-INPUT-EE,
                   RET-STATUS,
                   CDM-INPUT-NAME-EE,
                   DISPOSITION,
                   CDM-INPUT-RECORD-LENGTH-EE,
                   NUMBER-OF-RECORDS.
IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR OPENING FILE CDM-INPUT-NAME-EE"
      TO MSG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
LOOP-EE.
CALL "INPFIL" USING FCB-INPUT-EE,
                   RET-STATUS,
                   CDM-INPUT-EE,
                   CDM-INPUT-RECORD-LENGTH-EE,
                   CDM-INPUT-RETURN-LENGTH-EE.
IF RET-STATUS = KES-END-OF-FILE-INPUT
    GO TO BREAK-EE.
IF RET-STATUS NOT = KES-FILE-OK
    MOVE "ERROR READING FILE CDM-INPUT-NAME-EE"
      TO MSG-DESC
    PERFORM PROCESS-ERROR
    GO TO EXIT-PROGRAM.
```

LIBRARY NAME - COBOL

MACRO NAME - INSFIL2

PARAMETERS - EE

```
*  
*   END OF INSERT LOOP (FROM FILE)  
*  
END-NDML-EE.  
  GO TO LOOP-EE.  
BREAK-EE.  
  MOVE "K" TO DISPOSITION.  
  CALL "CLSFIL" USING FCB-INPUT-EE,  
    RET-STATUS,  
    DISPOSITION.  
  IF RET-STATUS NOT = KES-FILE-OK  
    MOVE "ERROR CLOSING FILE CDM-INPUT-EE"  
      TO MMSG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO EXIT-PROGRAM.
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - INSVAL1

PARAMETERS - EE

```
*  
*   BEGINNING OF LOOP TO INSERT COMMAND VALUES  
*  
    MOVE 0 TO CDM-INPUT-INDEX-EE.  
LOOP-EE.  
    ADD 1 TO CDM-INPUT-INDEX-EE.  
    IF CDM-INPUT-INDEX-EE > CDM-INPUT-USED-EE  
        GO TO BREAK-EE.
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - INSVAL2

PARAMETERS - EE

*
* END OF INSERT LOOP, FROM COMMAND VALUES
*
END-NDML-EE.
 GO TO LOOP-EE.
BREAK-EE.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - RITCHK

PARAMETERS - OP
 SIGN
 EE
 ECODE

*
*
*

CHECK RESULTS OF REFERENTIAL INTEGRITY TEST:

IF NDML-COUNT OP SIGN 0
 MOVE "ECODE" TO NDML-STATUS
GO TO END-NDML-EE.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPESWS

PARAMETERS - EE

01	CDM-CS-RESULTS-FILE-EE	PIC X(80).
01	CDM-INPUT-NAME-EE	PIC X(80).
01	CDM-CS-RESULTS-EE	PIC X(80).
01	FCB-CDM-RESULTS-EE	PIC S9(9) COMP.
01	FCB-INPUT-EE	PIC S9(9) COMP.
01	CDM-INPUT-RETURN-LENGTH-EE	PIC S9(9) COMP.
01	CDM-INPUT-RECORD-LENGTH-EE	PIC S9(9) COMP.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWS

PARAMETERS - none

*

* ITEMS FOR EACH NDML REQUEST

*

```
01 CDM-NO-SUBTRANS          PIC 999.
01 CDM-DRS-ACTION           PIC X.
01 CDM-PTR                  PIC 9(5).
01 NDML-COUNT                PIC 9(6).
01 NDML-STATUS              PIC X(5).
    88 OK                    VALUE "00000".
01 CDM-CE-FLAG              PIC 9.
01 CDM-CE-EOF-FLAG          PIC 9.
    88 CDM-CE-EOF            VALUE 1.
01 CDM-FLAG-ARRAY.
    03 FLAG-X OCCURS 25 TIMES PIC 9.
01 NDML-CS-COUNT            PIC 9(6).
01 NDML-RFT-COUNT           PIC 9(6).
COPY ERRFS OF IISSCLIB.
01 DISPOSITION              PIC X.
01 NUMBER-OF-RECORDS        PIC S9(9) COMP VALUE 2000.
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWSI

PARAMETERS - P1
EE

```
*  
*      TABLE TO STORE INSERT VALUES  
*  
*      FOUND IN NDML COMMAND  
01      CDM-INPUT-INDEX-EE          PIC 999 VALUE 0.  
01      CDM-INPUT-USED-EE          PIC 999 VALUE P1.  
01      CDM-INPUT-EE.  
        02 CDM-INPUT-ENTRY OCCURS P1 TIMES.
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - CCCALL

PARAMETERS - P1
P2
P3

```
*  
* CALL CS SELECTOR  
*  
CALL "P1" USING  
    CDM-CS-RESULTS-FILE-P2  
    NDML-COUNT  
    CDM-CSQ-TABLE-P3  
*  
    CDM-CS-RESULTS-P3  
    CDM-CS-COUNT-P3  
    NDML-STATUS
```

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - CS2CALL

PARAMETERS P1
 P2
 P3

```
IF NDML-COUNT > 0
  CALL "P1" USING
    CDM-CS-RESULTS-FILE-P2
    CDM-CSQ-TABLE-P3
    NDML-COUNT
    NDML-STATUS
IF NOT OK
  GO TO END-NDML-P2.
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - CCCALL

PARAMETERS - P1
 P2
 P3
 P4
 P5

```
      CALL P1 ( %REF(P2), %REF(NDMLCT), %REF(P3)
*           , %REF(P4), %REF(P5), %REF(NDMLST) )
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MARCO NAME - CECALL

PARAMETERS - P1
 MMMMM
 P2
 P3
 P4

```
CEFLAG = P1
CALL MMMMM( %REF(CEFLAG), %REF(P2), %REF(P3)
*           , %REF(FLAGAR), %REF(P4), %REF(EOFFLA)
*           , %REF(NDMLST))
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - CS2CALL

PARAMETERS - P1
 P2
 P3
 P4

```
IF (CHARCT .NE. '000000') THEN
  CALL P1( %REF(P3), %REF(P4), %REF(CHARCT)
*         , %REF(NDMLST))
  IF (NDMLST .NE. '00000' ) GO TO 93P2
ENDIF
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - DRSCALL

PARAMETERS - P1
 P2
 P3
 P4
 P5
 P6
 P7
 P8

```
NOSSUB  =  P1
DRSACT  =  'P2'
CALL CDS01( %REF(NOSSUB), %REF(DRSACT), %REF(P3)
*          , %REF(P4), %REF(P5)
*          , %REF(P6)
*          , %REF(P7), %REF(P8), %REF(CHARCT)
*          , %REF(NDMLST))
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - ENDLOOP

PARAMETERS - EE

IF (EDFFLA .NE. '1') GO TO 94EE
91EE CONTINUE
93EE CONTINUE

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - ERRCHK

PARAMETERS - EE

IF (NDMLST .NE. '00000') GO TO 93EE

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - ERRCHKCE

PARAMETERS - EE

```
      IF (NDMLST .NE. '00000' .OR.  
*      EOFFLA .EQ. '1') GO TO 93EE
```

DS 620341200
30 September 1990

LIRARY NAME - FORTRAN

MACRO NAME - FILSAV1

PARAMETERS - EE

F1
P1
P2
P3

```
      IF (EOFFLA .EQ. '1') GO TO 93EE
      FILEST = 'W'
      P1 = F1
      CALL OPNFIL ( %REF(P2), %REF(NDMLST), %REF(P1)
*           , %REF(FILEST), %REF(P3), %REF(NUMREC))
      IF (NDMLST .NE. '00000') GO TO 93EE
94EE CONTINUE
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - FILSAV2

PARAMETERS - EE
 P1

```
          IF (EOFFLA .NE. '1') GO TO 94EE
          FILEST = 'K'
          CALL CLSFIL ( %REF(P1), %REF(NDMLST), %REF(FILEST))
          IF (NDMLST .NE. '00000') GO TO 93EE
93EE      CONTINUE
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - INSFIL1

PARAMETERS - EE

F1
P1
P2
P3
P5
P6

```
FILEST = 'R'
P1      = F1
CALL OPNFIL ( %REF(P2), %REF(NDMLST), %REF(P1),
*             %REF(FILEST), %REF(P3),
*             %REF(NUMREC))
IF (NDMLST .NE. '00000') GO TO 91EE
94EE CONTINUE
CALL INPFIL ( %REF(P2), %REF(NDMLST), %REF(P5),
*             %REF(P3), %REF(P6))
IF (NDMLST .NE. '00000') GO TO 91EE
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - INSFIL2

PARAMETERS - EE
 P1

93EE CONTINUE
 GO TO 94EE

91EE CONTINUE
 FILEST = 'K'
 CALL CLSFIL(%REF(P1), %REF(NDMLST), %REF(FILEST))

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - INSVAL1

PARAMETERS - EE

P1
P2

```
94EE      P1 = 0
          CONTINUE
          P1 = P1 + 1
          IF (P1 .GT. P2) GO TO 91EE
```

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - INSVAL2

PARAMETERS - EE

93EE CONTINUE
GO TO 94EE
91EE CONTINUE

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - RITCHK

PARAMETERS - OP
SIGN
EE
ECODE

IF (CHARCT OP SIGN '000000') THEN
NDMLST = 'ECODE'
GO TO 93EE
END IF

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - UAPESWS

PARAMETERS - P1

P2
P3
P4
P5
P6
P7

CHARACTER*80	P1
CHARACTER*80	P2
CHARACTER*80	P3
INTEGER	P4
INTEGER	P5
INTEGER	P6
INTEGER	P7

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - UAPWS

PARAMETERS - NONE

CHARACTER*3	NOSSUB
CHARACTER*1	DRSACT
CHARACTER*5	PTRCDM
INTEGER	NDMLCT
CHARACTER*6	CHARCT
CHARACTER*5	NDMLST
CHARACTER*1	CEFLAG
CHARACTER*1	EOFFLA
CHARACTER*25	FLAGAR
INTEGER	RFTCTI
CHARACTER*3	RFTCTC
CHARACTER*2	RFTCTJ
INTEGER	CSCNTI
CHARACTER*3	CSCNTC
CHARACTER*1	FILEST
INTEGER	NUMREC
DATA	NUMREC /2000/
REAL	CSUSED
INTEGER	DECIML
INTEGER	SIGN
INTEGER	DIGIT
INTEGER	ENPOSR
INTEGER	ENPOSC
INTEGER	SPOSR
INTEGER	SPOSC

DS 620341200
30 September 1990

LIBRARY NAME - FORTRAN

MACRO NAME - UAPWSI

PARAMETERS - P1
P2

P3

INTEGER	P2	
INTEGER	P3	
DATA	P3	/P1/

SECTION 28

FUNCTION CDP10A - Generate code to transform external schema values to conceptual schema values.

This function will:

1. Generate the transformation of ES values and variables for runtime search parameters and update values to Conceptual Schema format.
2. Generate code to build the pool of values and information for the DRS.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

28.1 Inputs

1. Source Language Indicator of the Application Program
LANG-NC
2. Application Program parcel names
FD-FILE-NAME
WORK-FILE-NAME
PROC-FILE-NAME
3. Conceptual Schema representation of the data
CS-ACTION-LIST
CS-QUALIFY-LIST
4. External Schema representation of the data
ES-ACTION-LIST
ES-QUALIFY-LIST

5. Internal Schema representation of the data
IS-ACTION-LIST
IS-QUALIFY-LIST
6. User View Abbreviation List
UV-ABBR-LIST
7. Code generation table
CODE-GENERATOR-TABLE
8. Subtransaction Identification table for the NDML
request
SUBTRANS-PROCESS-ID-TABLE
9. Next Parameter Number for complex mapping calls
NEXT-PARAMETER-NO
10. Application Program error file name
ERROR-FILE
11. Logical Unit Work Name
LUW
12. Fortran Variable Association Table
FORTRAN-VARIABLE-TABLE
13. Target Host Name
TARGET-HOST

28.2 CDM Requirements

None

28.3 Internal Requirements

None

28.4 Processing

1. Generate External/Conceptual Schema data definitions for runtime update/search values. Call function "CDECWS" with the following parameters:

LANG-NO
WORK-FILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
FORTRAN-VARIABLE-TABLE
RET-STATUS

2. Generate "MOVE" statements from user defined variables or constants to External Schema variables for update/search values. Call function "CDUEMV" with the following parameters if it is not the case that CS-ACTION = "I" or "K" or CS-ACTION = "1" and ES-ACTION = "I":

LANG-NO
PROC-FILE-NAME
CS-ACTION-LIST
CS-QUALIFY-LIST
ES-ACTION-LIST
ES-QUALIFY-LIST
FORTRAN-VARIABLE-TABLE
RET-STATUS

3. Perform steps 3.1 through 3.4 for each subtransaction. This will perform logic to calculate the length of the pool.

3.1 Initialize local variables

Set SUB-POOL-LEN to ZERO
Set WS-DBID to STR-DBID(SUB-INDEX).
Set TEMP-SUB-INDEX to SUB-INDEX

- 3.2 Perform steps 3.2.1 through 3.2.2 for each entry in the IS-QUALIFY-LIST if CS-ACTION not equal "I".

- 3.2.1 If ISQ-CSQ-PTR equals zero and ISQ-TYPE equals 2, continue at step 3.2.

- 3.2.2 If ISQ-TYPE equals 2, ISQ-DBIDL equals WS-DBID, ISQ-SUBTRANS-IDL equals TEMP-SUB-INDEX, and ISQ-OP is not equal "NN" or "NL" then:

Set CSQ-INDEX to ISQ-CSQ-PTR.
Set SUB-POOL-LEN equal to
SUB-POOL-LEN plus CSQ-L-SIZE.

- 3.3 Perform step 3.3.1 for each entry in the IS-ACTION-LIST if IS-ACTION equals "I" or "M".

30 September 1990

- 3.3.1 If IS-DBID equals WS-DBID, IS-SUTRANS-ID equals TEMP-SUB-INDEX, IS-MAPPED-TO, IS-USER, IS-CS-PTR is greater than zero, and IS-SOURCE is not equal "G", then:

Set CS-INDEX to IS-CS-PTR.
Set SUB-POOL-LEN equal to SUB-POOL-LEN
plus CS-SIZE.

- 3.4 Set SUB-POOL-LEN equal to SUB-POOL-LEN
plus FIXED-LEN.
Set TOTAL-POOL-LEN equal to TOTAL-POOL-LEN
plus SUB-POOL-LEN.

4. Generate code to define the pool.

If the language is COBOL, call "CDMACR" utility with
the following:

Library Name - COBOL
Macro Name - WSPPOOL
Parameters
EE = ES-NDML-NO
P1 = CS-NDML-NO
P5 = TOTAL-POOL-LEN

Else, if FORTRAN, generate:

CHARACTER*tt CDM-POOL-ee-cc
where tt = TOTAL-POOL-LEN
ee = ES-NDML-NO
cl = CS-NDML-NO

5. Perform steps 5.1 through 5.10 for each subtransaction.

- 5.1 Initialize local variables.

Set SUB-POOL-LEN to ZERO.
Set WS-DBID to STR-DBID(SUB-INDEX).
Set TEMP-SUB-INDEX to SUB-INDEX.

- 5.2 Perform steps 5.2.1 through 5.2.2 for each entry
in the IS-QUALIFY-LIST if CS-ACTION NOT = "I".

- 5.2.1 If ISQ-CSQ-PTR equals zero and ISQ-TYPE
equals 2, continue at step 5.2.

- 5.2.2 If ISQ-TYPE equals 2, ISQ-DBIDL equals
WS-DBID, ISQ-SUBTRANS-IDL equals
TEMP-SUB-INDEX, and ISQ-OP is not equal
"NN" or "NL" then perform steps 5.2.2.1
through 5.2.2.10.

- 5.2.2.1 Set CSQ-INDEX and CLIST-INDEX to
ISQ-CSQ-PTR

- 5.2.2.2 If SOURCE-IS-VIEW, set
SUB-POOL-LEN equal to
SUB-POOL-LEN plus CSQ-L-SIZE.

DS 620341200
30 September 1990
Continue at step 5.2.

- 5.2.2.3 Set ELIST-INDEX to CSQ-ES-PTR
- 5.2.2.4 If CS-ACTION equals "K" or "1", scan the CS-ACTION-LIST searching for CSQ-AUCL equal to CS-AUC. When found, perform step 5.2.2.4.1.
 - 5.2.2.4.1 Set DI-NO to ES-DI-NO
Set SEC-NO to ES-UV-NO
- 5.2.2.5 If CS-ACTION is not equal "K" or "1", perform step 5.2.2.5.1.
 - 5.2.2.5.1 Set ESQ-INDEX to CSQ-ES-PTR
Set DI-NO to ESQ-L-DI-NO
Set SEC-NO to ESQ-L-UV-NO
Set AUC-NO to CSQ-AUCL
- 5.2.2.6 Scan the UV-ABBR-LIST, searching for UV-NO equal SEC-NO. When found, set ENTRY-STATUS to 1 and SEC-ID to UV-NAME. If not found, perform proper error handling.
- 5.2.2.7 If ES-FILE-NAME not equal spaces or ES-STRUCTURE not equal spaces, set FILE-STRUC-VAR-FLAG equal "F" else set FILE-STRUC-VAR-FLAG equal "V".
- 5.2.2.8 Generate code required for the transformation of retrieved/qualified datafields from external schema format to conceptual schema format. Call function CDEC with the following parameters:

LANG-NO
WORK-FILE-NAME
PROC-FILE-NAME
NEXT-PARAMETER-NO
CLIST-INDEX
ELIST-INDEX
ACTION-TYPE
ES-ACTION
ES-NDML-NO
CS-NDML-NO
DI-NO
AUC-NO
FILE-STRUC-VAR-FLAG

DS 620341200
30 September 1990
FORTRAN-VARIABLE-TABLE
TARGET-HOST
RET-STATUS

5.2.2.9 Add CSQ-L-SIZE to SUB-POOL-LEN

5.2.2.10 Continue at step 5.2.

5.3 Perform steps 5.3.1 through 5.3.7 for each entry in the IS-ACTION-LIST if CS-ACTION equals "I" or "M".

5.3.1 If CS-ACTION equals "I"
Set ACTION-TYPE equal "I"
else set ACTION-TYPE equal "U".

5.3.2 If IS-DBID equals WS-DBID, the entry is not generated, the entry is mapped to, IS-SUBTRANS-ID equals TEMP-SUB-INDEX, the entry is mapped to and IS-CS-PTR is greater than zero, perform steps 5.3.2.1 through 5.3.2.6.

5.3.2.1 Set CLIST-INDEX equal IS-CS-PTR.
Set CS-INDEX equal IS-CS-PTR.
Set ES-INDEX equal CS-ES-PTR.
Set ELIST-INDEX equal CS-ES-PTR.
Set DI-NO equal ES-DI-NO.
Set SEC-NO equal ES-UV-NO.
Set AUC-NO equal CS-AUC.

5.3.2.2 Scan the UV-ABBR-LIST, searching for UV-NO equal SEC-NO. When found, set ENTRY-STATUS to 1 and SEC-ID to UV-NAME. If not found, perform proper error handling.

5.3.2.3 If ES-FILE-NAME not equal spaces or ES-STRUCTURE not equal spaces,
set FILE-STRUC-VAR-FLAG equal "F"
else set FILE-STRUC-VAR-FLAG equal "V".

5.3.2.4 Generate code required for the transformation of retrieved/qualified datafields from external schema format to conceptual schema format. Call function CDEC with the following parameters:

LANG-NO
WORK-FILE-NAME
PROC-FILE-NAME
NEXT-PARAMETER-NO
CLIST-INDEX

ELIST-INDEX
ACTION-TYPE
ES-ACTION
ES-NDML-NO
CS-NDML-NO
DI-NO
AUC-NO
FILE-STRUC-VAR-FLAG
FORTRAN-VARIABLE-TABLE
TARGET-HOST
RET-STATUS

5.3.2.5 Add CS-SIZE to SUB-POOL-LEN.

5.3.2.6 Continue at step 5.3.

5.4 Add FIXED-LEN to SUB-POOL-LEN.

5.5 Scan the CODE-GENERATOR-TABLE searching for
CGT-DBID equal WS-DBID and
the CGT-MOD-TYPE is RP-MAIN. Perform
steps 5.5.1 through 5.5.2.

5.5.1 If the entry is found
Set RP-DRIVER-NAME to CGT-MOD-NAME.
Set REMOTE-LOCAL to CGT-LOCALITY.
Set ENTRY-STATUS to 1.

5.5.2 If the entry is not found, perform steps
5.5.2.1 through 5.5.2.4.

5.5.2.1 Call function CDF1RP with
parameters:

LUW
WS-DBID
TEMP-MOD-NAME
REMOTE-LOCAL
FOUND-FLAG
CODE-GENERATOR-TABLE
RET-STATUS

5.5.2.2 If entry not found, perform
proper error handling.

5.5.2.3 If entry found and is local, set
RP-MAIN-DIR and RP-MAIN-END to
spaces. Set RP-MAIN-NAME to
TEMP-MOD-NAME.

5.5.2.4 If entry found and is remote,
Set RP-DRIVER-NAME to
TEMP-MOD-NAME.

5.6 If language is COBOL, perform steps 5.6.1 through
5.6.2.

30 September 1990

- 5.6.1 Scan the CODE-GENERATOR-TABLE searching for CGT-CASE-NO equal CS-NDML-NO, CGT-SUBTRANS-ID equal TEMP-SUB-INDEX, and CGT-MOD-TYPE is RP-SUB. When found, set RP-SUB-NAME to CGT-MOD-NAME.

- 5.6.2 Generate:

STRING "aabbccdd"

where aa = SUB-POOL-LEN
 bb = RP-SUB-NAME
 cc = RP-DRIVER-NAME
 dd = REMOTE-LOCAL

- 5.7 If language is FORTRAN, perform steps 5.7.1 through 5.7.10.

- 5.7.1 Calculate $\text{END-POS} = \text{START-POS} + 3 - 1$

- 5.7.2 Generate:

CDM-POOL-eee-ccc(ss:pp) = 'll'

where eee = ES-NDML-NO
 ccc = CS-NDML-NO
 ss = START-POS
 pp = END-POS
 ll = SUB-POOL-LEN

- 5.7.3 Calculate $\text{START-POS} = \text{END-POS} + 1$.

Calculate $\text{END-POS} = \text{START-POS} + 6 - 1$.

- 5.7.4 Scan the CODE-GENERATOR-TABLE searching for CGT-CASE-NO equal CS-NDML-NO, CGT-SUBTRANS-ID equal TEMP-SUB-INDEX, and CGT-MOD-TYPE is RP-SUB. When found, set RP-SUB-NAME to CGT-MOD-NAME.

- 5.7.5 Generate:

CDM-POOL-eee-ccc(ss:pp) = 'rr'

where eee = ES-NDML-NO
 ccc = CS-NDML-NO
 ss = START-POS
 pp = END-POS
 rr = RP-SUB-NAME

- 5.7.6 Calculate $\text{START-POS} = \text{END-POS} + 1$.
 Calculate $\text{END-POS} = \text{START-POS} + 10 - 1$.

- 5.7.7 Generate:

CDM-POOL-eee-ccc(ss:pp) = 'dd'

where eee = ES-NDML-NO
ccc = CS-NDML-NO
ss = START-POS
pp = END-POS
dd = RP-DRIVER-NAME

5.7.8 Calculate START-POS equal END-POS + 1.
Calculate END-POS equal START-POS + 1 - 1.

5.7.9 Generate:

CDM-POOL-eee-ccc(ss:pp) = 'mm'

where eee = ES-NDML-NO
ccc = CS-NDML-NO
ss = START-POS
pp = END-POS
mm = REMOTE-LOCAL

5.7.10 Calculate START-POS equal END-POS + 1.

5.8 Perform steps 5.8.1 through 5.8.2 for each entry in the IS-QUALIFY-LIST if CS-ACTION not equal "I".

5.8.1 If ISQ-CSQ-PTR equals zero and ISQ-TYPE equals 2:

Continue at step 5.8.

5.8.2 If ISQ-TYPE equals 2, ISQ-DBIDL equals WS-DBID, ISQ-SUBTRANS-IDL equals TEMP-SUB-INDEX, and ISQ-OP is not equal "NN" or "NL", then perform steps 5.8.2.1 through 5.8.2.4.

5.8.2.1 Set CSQ-INDEX and CLIST-INDEX to ISQ-CSQ-PTR.

5.8.2.2 If language is COBOL, generate:

CSQ-VAR-ccc-ii

where ccc = CS-NDML-NO
ii = CLIST-INDEX

5.8.2.3 If language is FORTRAN, perform steps 5.8.2.3.1 through 5.8.2.3.3.

5.8.2.3.1 Calculate END-POS equal
START-POS + CSQ-L-SIZE - 1

5.8.2.3.2 If CSQ-L-TYPE not equal "C",
generate:

DS 620341200
30 September 1990

```
DECIML = nn  
CALL RELFTN(DECIML,  
CSQ-VAR-ccc-ii,  
XHQccci,  
CSQ-LONG-VAR-ccci  
CDM-POOL-eee-ccc(ss:pp) =  
XHQccci
```

else generate:

```
CDM-POOL-eee-ccc(ss:pp) =  
CSQ-VAR-ccc-ii
```

```
where eee = ES-NDML-NO  
ccc = CS-NDML-NO  
nn = CSQ-L-ND  
ii = CLIST-INDEX  
ss = START-POS  
ee = END-POS
```

5.8.2.3.3 Calculate START-POS equal
END-POS + 1.

5.8.2.3.4 Sum the CSQ-L-SIZE's of all
CSQ-ENTRYS before the one
currently pointed to by
CSQ-INDEX, where CSQ-R-SIZE
equals zero, and put the total
in START-POSC.

5.8.2.3.5 Add 1 to START-POSC.
Set END-POSC equal to
START-POSC plus CSQ-L-SIZE
minus 1.

5.8.2.3.6 If CSQ-L-TYPE equals "C",
generate:
CDM-POOL-eee-ccc(sp:pp) =
CSQ-VAR-ccc-ii
Else, generate:
CDM-POOL-eee-ccc(ss:pp) =
XHQccci
where eee = ES-NDML-NO
ccc = CS-NDML-NO
ss = START-POSC
pp = END-POSC
ii = CLIST-INDEX

5.8.2.4 Continue at step 4.8.

5.9 Perform steps 5.9.1 through 5.9.1.4. for each
entry in the IS-ACTION-LIST if CS-ACTION equals
"I" or "M".

5.9.1 If IS-DBID equals WS-DBID, the entry is
not generated, the entry is mapped to,
IS-SUBTRANS-ID equals TEMP-SUB-INDEX,
and IS-CS-PTR is greater than zero,
perform steps 5.9.1.1 through 5.9.1.4.

5.9.1.1 Set CS-INDEX and CLIST-INDEX to
IS-CS-PTR.

5.9.1.2 If language is COBOL, generate:
CS-VAR-ccc-ii

where ccc = CS-NDML-NO
ii = CLIST-INDEX

5.9.1.3 If language is FORTRAN, perform
steps 5.9.1.3.1 through
5.9.1.3.3.

5.9.1.3.1 Calculate END-POS =
START-POS + CS-SIZE - 1

5.9.1.3.2 If CS-TYPE is not equal to "C",
generate:

DECIML = nn
CALL RELFTN(DECIML,
CS-VAR-cccii, XHScccii,
CS-LONG-VAR-ccc-ii)
CDM-POOL-eee-ccc(ss:pp) =
XHScccii

else generate:

CDM-POOL-eee-ccc(ss:pp) =
CS-VAR-ccc-ii

where eee = ES-NDML-NO
ccc = CS-NDML-NO
nn = CS-ND
ii = CLIST-INDEX
ss = START-POS
ee = END-POS

5.9.1.3.3 Calculate START-POS =
END-POS + 1

5.9.1.4 Continue at step 5.9.

5.10 If language is COBOL, generate:

DELIMITED BY SIZE INTO
CDM-POOL-eee-ccc WITH POINTER CDM-PTR.

where eee = ES-NDML-NO
ccc = CS-NDML-NO

6. Terminate processing of function PRE10A.

6.1 Close the two parcels of the user's application
program.

6.2 Set the function status variable and exit processing.

28.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - WSPool

PARAMETERS - P1
 P5
 EE

*
* POOL OF INPUT PARAMETERS FOR DRS TO PASS ON TO RP:
*
01 CDM-POOL-EE-P1 PIC X(P5).

SECTION 29

FUNCTION CDP10B Generate precompiler tables into the users
 Application Program.

This function will:

1. Generate the transfer of precompiler tables into the users Application Program. These tables include:

JQG - Join Query Graph
APL - Attribute Pair List
RFT - Result Field Table
CSAL - Conceptual Schema Action List

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

29.1 Inputs

1. Source Language Indicator of the Application Program
 LANG-NO
2. Application Program parcel names
 WORKFILE-NAME
 PROCFIELD-NAME
3. External Schema representation of the data
 ES-NDML-NO
4. Join Query Graph for the NDML request
 JQG
 JQG-ATTRIBUTE-PAIR-LIST
5. Result Field Table
 RFT

6. Conceptual Schema representation of the data

CS-ACTION-LIST

7. Fortran Variable Association Table

FORRAN-VARIABLE-TABLE

8. Target Host Name

TARGET-HOST

29.2 CDM Requirements

None

29.3 Internal Requirements

None

29.4 Processing

1. If CS-ACTION is BEGIN, CDMMIT, or ROLLBACK, generate a variable into the WS parcel to hold the pool.

- 1.1 If language is COBOL, generate:

01 CDM-POOL-ee-cc PIC X.

where ee = ES-NDML-NO
cc = CS-NDML-NO

- 1.2 If language is FORTRAN, generate:

CHARACTER*1 CDM-POOL-eee-cc

where ee = ES-NDML-NO
cc = CS-NDML-NO

2. Generate working storage variables to hold the precompiler tables required for runtime execution of the NDML request. The tables required are:

JQG
APL
RFT
CSAL

If language is COBOL, perform steps 2.1 and 2.2.
If language is FORTRAN, perform step 2.3.

- 2.1 For each non-empty table (USED > 0), generate a "COPY" statement to include the copy member using the "CDMACR" utility.

TABLE

JQG
APL
RFT
CSAL

MACRO

UAPWS2
UAPWS4
UAPWS3
UAPWS1

- 2.2 For each empty table, generate the following data definition.

TABLE

JQG	01	CDM-JQG-ee-cc	PIC 9999	VALUE 0.
APL	01	CDM-APL-ee-cc	PIC 9999	VALUE 0.
RFT	01	CDM-RFT-ee-cc	PIC 9999	VALUE 0.
CSAL	01	CDM-CSAL-ee-cc	PIC 9999	VALUE 0.

where

ee = ES-NDML-NO

cc = CS-NDML-NO

- 2.3 For each non-empty table (USED > 0), generate a variable to hold the copy member.

- 2.3.1 If CS-USED is greater than zero
calculate TEMP-NO = CS-ENTRY * CS-USED +
CS-HEAD

Generate:

CHARACTER*nn CDM-CSAL-ee-cc
DATA CDM-CSAL-ee-cc /'uu'/

If CS-USED is equal zero, generate

CHARACTER*4 CDM-CSAL-ee-cc
DATA CDM-CSAL-ee-cc
/'0000'/

where ee = ES-NDML-NO

cc = CS-NDML-NO

nn = TEMP-NO

uu = CS-USED

- 2.3.2 If JQG-USED is greater than zero
calculate TEMP-NO = JQG-ENTRY * JQG-USED
+ JQG-HEAD

Generate:

CHARACTER*nn CDM-JQG-ee-cc
DATA CDM-JQG-ee-cc /'uu'/

If JQG-USED is equal zero, generate

CHARACTER*4 CDM-JQG-ee-cc
DATA CDM-JQG-ee-cc /'0000'/

where ee = ES-NDML-NO

cc = CS-NDML-NO

nn = TEMP-NO

uu = JQG-USED

- 2.3.3 If APL-USED is greater than zero
calculate $TEMP-NO = APL-ENTRY * APL-USED + APL-HEAD$

Generate:

CHARACTER*nn CDM-APL-ee-cc
DATA CDM-APL-ee-cc
/'uu0022'/

If APL-USED is zero, generate:

CHARACTER*4 CDM-APL-ee-cc
DATA CDM-APL-ee-cc /'0000'/

where ee = ES-NDML-NO
cc = CS-NDML-NO
nn = TEMP-NO
uu = APL-USED

- 2.3.4 If RFT-USED is greater than zero
calculate $TEMP-NO = RFT-NO * RFT-USED + RFT-HEAD$

Generate:

CHARACTER*nn CDM-RFT-ee-cc
DATA CDM-RFT-ee-cc
/'uu000024'/

If RFT-USED is zero, generate

CHARACTER*6 CDM-RFT-ee-cc
DATA CDM-RFT-ee-cc
/'000000'/

where ee = ES-NDML-NO
cc = CS-NDML-NO
nn = TEMP-NO
uu = RFT-USED

3. Generate code to populate the tables with precompiled results required for runtime execution of the NDML request.

- 3.1 For each non-empty table (USED > 0) generate MOVE statements to move the non-repeating data of each table and each used entry of the table to the newly defined variables.

29.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWS1

PARAMETERS - EE
 CC
 P1

*

COPY CSAL OF IISSCLIB REPLACING
==CS-ACTION-LIST== BY
==CDM-CSAL-EE-CC==
==50== BY ==P1==.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWS2

PARAMETERS - EE
 CC
 P2

COPY JQGTBL OF IISSCLIB REPLACING
==01 JQG== BY
==01 CDM-JQG-EE-CC==
==30== BY ==P2==.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWS3

PARAMETERS - EE
 CC
 P4

COPY RFTABLE OF IISSCLIB REPLACING
==01 RFT== BY
==01 CDM-RFT-EE-CC==
==200== BY ==P4==.

DS 620341200
30 September 1990

LIBRARY NAME - COBOL

MACRO NAME - UAPWS4

PARAMETERS - EE
 CC
 P3

COPY APL OF IISSCLIB REPLACING
==JQG-ATTRIBUTE-PAIR-LIST== BY
==CDM-APL-EE-CC==
==60== BY ==P3==.

SECTION 30

FUNCTION CDP10C - Generate External Schema Results into User Variables or Structures

This function will:

1. Generate code to transfer the External Schema results
record into user specified variables or user
specified structure.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

30.1 Inputs

1. Source Language Indicator of the Application Program
LANG-NO
2. Application Program parcel names
PROC-FILE-NAME
3. External Schema representation of the data
ES-ACTION-LIST
4. Conceptual Schema representation of the data
CS-NDML-NO
5. Fortran Variable Association Table
FORTRAN-VARIABLE-TABLE

30.2 CDM Requirements

None

30.3 Internal Requirements

None

30.4 Processing

1. Generate code to transfer the results of the NDML query into user specified variables or a user specified structure.

- 1.1 If results are to be placed in a user structure (ES-STRUCTURE NOT = SPACE) continue at step 1.2. If results are not to be placed into a file (ES-FILE-NAME = SPACES) continue.

For each ES field projected generate a move of the results to the user specified variable. Set VARIABLE-NAME to ES-LOCAL-VARIABLE (ES-INDEX, 1). If language is FORTRAN, perform step 1.1.2, else perform step 1.1.1.

- 1.1.1 Generate:

MOVE ES-RES-ccc-nn TO vv

where

ccc = CS-NDML-NO
nn = ES-INDEX
vv = VARIABLE-NAME

Exit processing.

- 1.1.2 If the function chosen for the entry is COUNT, MEAN, AVG, or SUM or ES-TYPE is not equal to "C", continue at step 1.1.2.1, else continue at step 1.1.2.5.

- 1.1.2.1 If the function chosen for the entry is COUNT, perform steps 1.1.2.1.1 through 1.1.2.1.

1.1.2.1.1 Calculate
END-POS equals START-POS plus 8.

- 1.1.2.1.2 Generate:

Call
CONDIG(CDM-RESULTS-ee(ss:pp),
SIGN, DIGIT, NDMLST)
Call
CHRINT(CDM-RESULTS-ee(ss:pp)
* vv, NDMLST)
vv =
* vv*SIGN
where ee =

ES-NDML-NO

ss =

START-POS

pp =

END-POS

vv =

VARIABLE-NAME

1.1.2.1.3 Calculate
START-POS equals END-POS plus 1.

1.1.2.2 If the function chosen for the
entry is MEAN, SUM, or AVG, or
ES-TYPE is F, perform steps
1.1.2.2.1 through 1.1.2.2.2.

1.1.2.2.1 Generate:

vv =

ES-RES-cc-nn

where vv =

VARIABLE-NAME

cc =

CS-NDML-NO

nn =

ES-INDEX

1.1.2.2.2 Calculate
START-POS equals START-POS plus
8.

1.1.2.3 If ES-TYPE is "I", perform steps
1.1.2.3.1 through 1.1.2.3.2.

1.1.2.3.1 Generate:

vv = ES-RES-cc-nn

where vv = VARIABLE-NAME

cc = CS-NDML-NO

nn = ES-INDEX

1.1.2.3.2 If ES-SIZE is
greater than 4, add 4 to
START-POS.

Else add 2 to
START-POS.

1.1.2.4 Continue at step 1.1.2.7.

1.1.2.5 Calculate END-POS = START-POS +
ES-SIZE - 1.

1.1.2.6 Generate:

vv = CDM-RESULTS-ee(ss:pp)

where ee = ES-NDML-NO

ss = START-POS

pp = END-POS

vv = VARIABLE-NAME

30 September 1990

1.1.2.7 Calculate $\text{START-POS} = \text{END-POS} +$
1

1.2 Generate a move of the results to the user
specified

structure. Generate if COBOL:

MOVE CDM-RESULTS-ee TO (es-structure)

else generate:

(es-structure) = CDM-RESULTS-ee

where

ee = ES-NDML-NO

30.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application
Program

SECTION 31

FUNCTION - CDP10E Process External Schema Insert Value

This function will:

1. Generate External Schema Data
2. Generate definitions for runtime insert values and procedure division code to move the values to temporary defined table.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

31.1 Inputs

1. External Schema representation of the data
ES-ACTION-LIST
ES-VALUE-USED
2. Application Program parcel names
ID-FILE-NAME
FD-FILE-NAME
WORK-FILE-NAME
PROC-FILE-NAME
3. Source Language Indicator of the Application Program
LANG-NO
4. Fortran Variable Association Table
FORTRAN-VARIABLE-TABLE

31.2 CDM Requirements

None

31.3 Internal Requirements

None

31.4 Processing

1. Perform steps 1.1 through 1.3 if language is COBOL.

1.1 Generate External Schema Data Definitions for insert values.

For each entry in the ES-ACTION list generate data definitions into WORKFILE.

03 ES-VAR-INS-eee-nn PIC clause

where .

eee = ES-NDML-NO

nn = ES-INDEX

clause = picture clause built by routine
"CDPIC" using ES-META-DATA

1.2 Generate procedure division code to populate the temporary table to hold insert values. If ES-VALUE-USED is greater than zero, generate:

MOVE 0 TO CDM-INPUT-INDEX-ee

Generate the following procedure division MOVE statements for each entry in ES-ACTION list, ES-VALUE-USED times.

ADD 1 TO CDM-INPUT-INDEX-ee.

MOVE {var, value, constant} TO
ES-VAR-INS-ee-nn(CDM-INPUT-INDEX-ee)

where

ee = ES-NDML-NO

nn = ES-INDEX

1.3 If ES-STRUCTURE not equal spaces, generate:

MOVE ss TO CDM-INPUT-ee.

where

ee = ES-NDML-NO

ss = ES-STRUCTURE

2. Perform steps 2.1 through 2.5.2 if language is FORTRAN.

2.1 Calculate BUFFER-SIZE which is the total of ES-SIZE of all non-deleted ES-ACTION-LIST entries plus one for each non-deleted entry with ES-TYPE equal "F".

2.2 Generate:

CHARACTER*bb CDM-INPUT-ee

where bb = BUFFER-SIZE
ee = ES-NDML-NO

2.3 Perform steps 2.3.1 through 2.3.3 for each entry in the ES-ACTION-LIST.

2.3.1 If ES-TYPE is equal "C", generate:

CHARACTER*ss ES-VAR-INS-ee-nn(uu)

where ss = ES-SIZE
ee = ES-NDML-NO
nn = ES-INDEX

uu = ES-VALUE-USED, if greater than zero.

Continue at step 2.3.5.

2.3.2 If ES-TYPE is equal "F", generate:

DOUBLE PRECISION
ES-VAR-INS-ee-nn(uu)

CHARACTER*SS
CHAR-ES-VAR-INS-ee-nn

where ee =
ES-NDML-NO
nn =
ES-INDEX
ss =
ES-SIZE +1
uu =
ES-VALUE-USED, if greater than zero

2.3.3 If ES-TYPE is equal "I", generate

INTEGER ES-VAR-INS-ee-nn(uu)
CHARACTER*ss CHAR-ES-VAR-INS-ee-nn
where ee = ES-NDML-NO
nn = ES-INDEX
ss = ES-SIZE +1
uu = ES-VALUE-USED, if
greater than zero

2.4 If ES-VALUE-USED is greater than zero, generate:

CDM-INPUT-INDEX-ee = 0

- 2.5 Generate code in the procedure division to populate the temp table to hold insert values from NDML select statement. Perform steps 2.5.1 through 2.5.2 ES-VALUE-USED times.

2.5.1 Generate:

CDM-INPUT-INDEX-ee = CDM-INPUT-INDEX-ee
+ 1

where ee = ES-NDML-NO

- 2.5.2 For each entry in the ES-ACTION-LIST, generate:

ES-VAR-INS-ee-nn(CDM-INPUT-INDEX-ee) =
{var, value, constant}

where ee = ES-NDML-NO
nn = ES-INDEX
Continue at step 2.5.

- 2.6 If ES-STRUCTURE is not equal spaces, perform steps 2.6.1 through 2.6.3 for each entry in the ES-ACTION-LIST.

- 2.6.1 Calculate END-POS = START-POS + ES-SIZE
- 1

2.6.2 Generate:

ES-VAR-INS-ee-nn = tt(ss:pp)

where ee = ES-NDML-NO
nn = ES-INDEX
tt = ES-STRUCTURE
ss = START-POS
pp = END-POS

- 2.6.3 Calculate START-POS = END-POS + 1
Continue at step 2.6.

31.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program

SECTION 32

FUNCTION CDP10F - Generate Data Definitions for Retrieved Results

This function will:

1. Generate program variables to hold the External Schema results of the NDML query.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

32.1 Inputs

1. External Schema representation of the data
ES-ACTION-LIST
2. Conceptual Schema representation of the data
CS-ACTION-LIST
3. Application Program parcel names
FD-FILE-NAME
WORK-FILE-NAME
4. Source Language Indicator of the Application Program
LANG-NO
5. Fortran Variable Association Table
FORTRAN-VARIABLE-TABLE

32.2 CDM Requirements

None

32.3 Internal Requirements

None

32.4 Processing

1. For each projected entry in the ES action list, generate a program variable to hold the results of the NDML query.

- 1.1 Generate variables to hold the null flags if results are to be written to a user file. If ES-FILE-NAME not = space, for each projected field generate if COBOL:

03 ES-NULL-ccc-nn PIC 9.

where

ccc = CS-NDML-NO
nn = ES-INDEX

Add 1 to TOTAL-SIZE to compute actual number of non-deleted ES entries.

- 1.2 Generate a group item for retrieved data fields to be stored in a file. If ES-FILE-NAME not = space generate if COBOL:

03 CDM-RESULTS-RECORD-eee

where

eee = ES-NDML-NO

- 1.3 Generate 05 entries for each projected field in the ES-ACTION list. Generate if ES-FILE-NAME not = space into WORK parcel if COBOL:

05 ES-RES-RECORD-ccc-nn PIC clause.

where

ccc = CS-NDML-NO
nn = ES-INDEX
clause = picture clause from table below

Add appropriate total size from table below to
TOTAL-SIZE.

- 1.4 If language is FORTRAN and ES-FILE-NAME not =
spaces, generate:

```
CHARACTER*tt  CDM-RESULTS-REC-ee
INTEGER       CDM-RECORD-LENGTH-ee
DATA          CDM-RECORD-LENGTH  /tt/
```

where tt = TOTAL-SIZE
ee = ES-NDML-NO

- 1.5 If language is COBOL and ES-FILE-NAME not =
spaces, generate:

01 CDM-RECORD-LENGTH-ee PIC S9(9) COMP VALUE tt.

01 CDM-RESULTS-ee.

where ee = ES-NDML-NO
tt = TOTAL-SIZE

- 1.6 Generate 05 entries for each projected field in
the ES-ACTION-LIST. If COBOL, generate:

05 ES-RES-ccc-nn PIC clause.

where cc = CS-NDML-NO
nn = ES-INDEX
clause = picture clause from table below

Else, if FORTRAN, and function is COUNT, or
ES-TYPE equals "C", don't generate anything.

If function is MEAN, AVG, or SUM, generate:

```
DOUBLE PRECISION  ES-RES-ccc-nn
CHARACTER*18      LONG-ES-RES-ccc-nn
```

Else, if ES-TYPE equals "I", generate:

```
INTEGER*ii  ES-RES-ccc-nn
Else, if ES-TYPE equals "F", generate
DOUBLE PRECISION  ES-RES-ccc-nn
CHARACTER*18      LONG-ES-RES-ccc-nn
```

where ccc = CS-NDML-NO
nn = ES-INDEX
ss = ES-SIZE
ii = 4 if ES-SIZE >4
2 if ES-SIZE <=4

Add appropriate size to TOTAL-SIZE.

1.7 If language is FORTRAN, generate:

CHARACTER*tt CDM-RESULTS-ee

where tt = TOTAL-SIZE
ee = ES-NDML-NO

1.8 If language is FORTRAN, perform steps 1.8.1 through 1.8.4 for each ES-ACTION entry that is projected, has ES-TYPE not equal "C", and ES-FCTN-NAME not equal "COUNT."

1.8.1 Calculate START-POS by summing up sizes of the ES entries up to the one currently pointed to by ES-INDEX. Use the following criteria:

If ES-FCTN-NAME equals "COUNT", add 9 to START-POS.
Else, if ES-FCTN-NAME equals "SUM", "AVG", or "MEAN", add 8 to START-POS.
Else, if ES-TYPE equals "I" and ES-SIZE is greater than 4, add 4 to START-POS.
Else, if ES-TYPE equals "I" and ES-SIZE is less than 5, add 2 to START-POS.
Else, if ES-TYPE equals "F", add 8 to START-POS.
Else, if ES-TYPE equals "C", add ES-SIZE to START-POS.

1.8.2 Add 1 to START-POS.

1.8.3 Calculate END-POS using the following criteria:

If ES-FCTN-NAME equals "MEAN", "SUM", or "AVG", set END-POS equal to START-POS plus 7.
Else, if ES-TYPE equals "I" and ES-SIZE is greater than 4, set END-POS equal to START-POS plus 3.
Else, if ES-TYPE equals "I" and ES-SIZE is less than 5, set END-POS equal to START-POS plus 1.
Else, if ES-TYPE equals "F", set END-POS equal to START-POS plus 7.

1.8.4 Generate:

EQUIVALENCE (CDM-RESULTS-ee(sp:ep),
ES-RES-cc-nn)
where ee = ES-NDML-NO
sp = START-POS
ep = END-POS
cc = CS-NDML-NO

30 September 1990

NOTE: Generate the following PIC clause for
functions and calculate TOTAL-SIZE:

ES-FCTN-NAME	PIC clause	TOTAL-SIZE
" "	CDPIC generated	ES-SIZE
COUNT	S9(9)	9
MEAN	S9(9)V9(9)	18
AVG	S9(9)V9(9)	18
SUM	S9(9)V9(9)	18
MIN	CDPIC generated	ES-SIZE
MAX	CDPIC generated	ES-SIZE

32.5 Outputs

1. Error status of the function

RETURN-STATUS

2. Generated code in the WORK parcels.

SECTION 33

FUNCTION CDEC - Generate External/Conceptual Transformation

This routine will generate code required for the transformation of search or update data items to their corresponding conceptual attributes.

CDEC generates either ANSI X3.23-1974 COBOL or ANSI X3.9-1978 Fortran source code.

For COBOL programs, CDEC generates working storage and procedure division code. For Fortran programs, CDEC generates type statements and executable statements.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

33.1 Inputs

1. LANG-NO included in LANG-NO copy member

If this parameter contains the value 1, COBOL source code is to be generated.

If this parameter contains the value -1, Fortran source code is to be generated.

2. WORK-FILE-NAME PIC X(30)

This parameter contains the name of the file where COBOL working storage or Fortran type statements are generated.

3. PROC-FILE-NAME PIC X(30)

This parameter contains the name of the file where COBOL or Fortran executable statements will be generated.

4. NEXT-PARAMETER-NUMBER PIC 9(3)

 This parameter is both an input and output parameter. It contains the number of the last parameter which was generated in the request processor. It is incremented prior to the generation of a new parameter.
5. CLIST-INDEX PIC 99

 This parameter contains the index value of either the CS-ACTION-LIST or CS-QUALIFY-LIST entry which describes the transformed external field.
6. ELIST-INDEX PIC 99

 This parameter contains the index value of either the ES-ACTION-LIST entry or ES-QUALIFY-LIST entry which describes the external field to be transformed.
7. ACTION-TYPE PIC X

 This parameter contains the CS action to be performed.
8. ES-ACTION-TYPE PIC X

 This parameter contains the ES action to be performed.
9. ES-NDML-NO PIC 999

 This number uniquely identifies the user's NDML request.
10. CS-NDML-NO PIC 9(6)

 This number uniquely defines the case in a logical unit of work.
11. DI-NO PIC 9(6)

 This parameter contains the internal identifier of the external data item.
12. AUC-NO PIC 9(6)

 This parameter contains the internal identifier of the conceptual attribute use class.
13. VAR-OR-FILE-IND PIC X

 This parameter is used to determine whether key uniqueness and type 1 referential integrity tests for inserts as well as the insert itself have their values coming from user variables and constant values (VAR-OR-FILE-IND equals V) or from a user structure or file (VAR-OR-FILE-IND equals F).

14. FORTRAN-VARIABLE-TABLE

This table contains the FORTRAN variables that have been defined. Associated with each of these variables in the table is the corresponding COBOL variable.

15. TARGET-HOST PIC XXX.

This variable is used if the source language is FORTRAN in the calls to CDMACR. It must know whether to use IBMFORTRAN or VAXFORTRAN as the library name.

33.2 CDM Requirements

ENTITY CLASS

COMPLEX-MAPPING-PARM
MODULE-PARAMETER
USER-DEF-DATA-TYPE

33.3 Internal Requirements

None

33.4 Processing

1. Open for EXTEND WORK-FILE-NAME.
2. Open for EXTEND PROC-FILE-NAME.
3. Determine whether a complex mapping algorithm exists for the data item/AUC combination.
 - 3.1 If the following SQL statement returns at least 1 row, a complex mapping exists for the data item/AUC combination.

```
SELECT MOD_ID, PARM_ID, CONSTANT_VALUE, UNION_DISC
FROM COMPLEX_MAPPING_PARM
WHERE ALG_USE_CODE = U AND
MOD_ID IN
(SELECT MOD_ID FROM COMPLEX_MAPPING_PARM
WHERE TAG_NO = :AUC-NO-WS) AND
MOD_INST IN
(SELECT MOD_INST FROM COMPLEX_MAPPING_PARM
WHERE TAG_NO = :AUC-NO-WS) AND
MOD_ID IN
```

```
(SELECT MOD_ID FROM COMPLEX_MAPPING_PARM
WHERE DI_NO = :DI-NO-WS) AND
MOD_INST IN
(SELECT MOD_INST FROM COMPLEX_MAPPING_PARM
WHERE DI_NO = :DI-NO-WS)
ORDER BY PARM_ID
```

4. If the previous SQL statement returned no data, a complex mapping algorithm does not exist for the data item/AUC combination. Single moves from the external variable to the conceptual variable must be generated on PROC-FILE-NAME. No code is generated on WORK-FILE-NAME.

- 4.1 If COBOL is to be generated (LANG-NO equals 1), perform the following steps.

- 4.1.1 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following code:

```
MOVE ES-VAR-INS-esndml-elist
(CDM-INPUT-INDEX-esndml)
TO CSQ-VAR-csndml-clist
```

where esndml is the value contained in input parameter ES-NDML-NO, elist is the value contained in input parameter ELIST-INDEX, csndml is the value contained in input parameter CS-NDML-NO and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.2 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following code:

```
MOVE ES-VAR-INS-esndml-elist
TO CSQ-VAR-csndml-clist
```

where esndml is the value contained in input parameter ES-NDML-NO, elist is the value contained in input parameter

30 September 1990

ELIST-INDEX, csndml is the value contained in input parameter CS-NDML-NO and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.3 If processing a type 1 referential integrity test (ACTION-TYPE equals 1) for a modify (ES-ACTION-TYPE equals M), generate the following code:

```
MOVE ES-VAR-csndml-elist  
TO CSQ-VAR-csndml-clist
```

where csndml is the value contained in input parameter CS-NDML-NO, elist is the value contained in input parameter ELIST-INDEX and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.4 If processing the qualification for a type 2 referential integrity test or a select or a delete or a modify (ACTION-TYPE equals 2 or S or D or M), generate the following code:

```
MOVE ESQ-VAR-csndml-elist  
TO CSQ-VAR-csndml-clist
```

where csndml is the value contained in input parameter CS-NDML-NO, elist is the value contained in input parameter ELIST-INDEX and clist is the value contained in input parameter ELIST-INDEX.

- 4.1.5 If processing the new column values for a modify (ACTION-TYPE equals U), generate the following code:

```
MOVE ES-VAR-csndml-elist  
TO CS-VAR-csndml-clist
```

where csndml is the value contained in input parameter CS-NDML-NO, elist is the value contained in input parameter ELIST-INDEX and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.6 If processing an insert (ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following code:

```
MOVE ES-VAR-INS-esndml-elist  
      (CDM-INPUT-INDEX-esdml)  
TO CS-VAR-csndml-clist
```

where esndml is the value contained in input parameter ES-NDML-NO, elist is the value contained in input parameter ELIST-INDEX, csndml is the value contained in input parameter CS-NDML-NO and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.7 If processing an insert (ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following code:

```
MOVE ES-VAR-INS-esndml-elist  
TO CS-VAR-esndml-clist
```

where esndml is the value contained in input parameter es-ndml-no, elist is the value contained in input parameter LIST-INDEX, csndml is the value contained in input parameter CS-NDML-NO and clist is the value contained in input parameter CLIST-INDEX.

- 4.1.8 Continue processing at step 6.

- 4.2 If Fortran is to be generated (LANG-NO equals -1), perform the following steps.

- 4.2.1 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following code:

```
CSQ-VAR-csndml-clist =  
ES-VAR-INS-esndml-elist(CDM-INUT-INDEX-e  
sndml)
```

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CLIST-INDEX, esndml is the value contained in input parameter ES-NDML-NO and elist is the value contained in input parameter ELIST-INDEX.

- 4.2.2 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following code:

30 September 1990

CSQ-VAR-csndml-clist =
ES-VAR-INS-esndml-elist

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CLIST-INDEX, esndml is the value contained in input parameter ES-NDML-NO and elist is the value contained in input parameter ELIST-INDEX.

- 4.2.3 If processing a type 1 referential integrity test (ACTION-TYPE equals 1) for a modify (ES-ACTION-TYPE equals M), generate the following code:

CSQ-VAR-csndml-clist =
ES-VAR-csndml-elist

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CLIST-INDEX and elist is the value contained in input parameter ELIST-INDEX.

- 4.2.4 If processing the qualification for a type 2 referential integrity test or a select or a delete or a modify (ACTION-TYPE equals 2 or S or D or M), generate the following code:

CSQ-VAR-csndml-clist =
ESQ-VAR-csndml-elist

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CLIST-INDEX and elist is the value contained in input parameter ELIST-INDEX.

- 4.2.5 If processing the new column values for a modify (ACTION-TYPE equals U), generate the following code:

CS-VAR-csndml-clist =
ES-VAR-csndml-elist

where csndml is the value contained in input parameter CS-NDML-NO, elist is the value contained in input parameter CLIST-INDEX and elist is the value contained in input parameter ELIST-INDEX.

30 September 1990

- 4.2.6 If processing an insert (ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following code:

```
CS-VAR-csndml-clist =
ES-VAR-INS-esndml-elist(CDM-INPUT-INDEX-
esndml)
```

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CUST-INDEX, esndml is the value contained in input parameter ES-NDML-NO and elist is the value contained in input parameter ELIST-INDEX.

- 4.2.7 If processing an insert (ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following code:

```
CS-VAR-csndml-clist =
ES-VAR-INS-esndml-elist
```

where csndml is the value contained in input parameter CS-NDML-NO, clist is the value contained in input parameter CLIST-INDEX, esndml is the value contained in input parameter ES-NDML-NO and elist is the value contained in input parameter ELIST-INDEX.

4.2.8 Continue processing at step 6.

5. If data was returned from SQL statement 1, a complex mapping algorithm exists for the data item/AUC combination.

- 5.1 Initialize the following structure.

```
01 PARAMETER-TABLE
   03 PARM-USED          PIC 99.
   03 PARM-MAX          PIC 99 VALUE 25.
   03 PARM-ENTRY OCCURS 25 TIMES INDEXED BY
   PARM-INDEX.
   05 PARM-NAME          PIC X(30).
```

Set PARM-USED to zero and all PARM-NAME entries to spaces.

- 5.2 If COBOL is to be generated (LANG-NO equals 1), perform the following steps:

- 5.2.1 If the current UNION-DISC from SQL statement 1 execution equals "5" indicating a data item, perform the following steps.

5.2.1.1 Generate a working storage entry for the data item.

Perform the following SQL statement to extract type, size, and number of decimal digits of the parameter from the CDM:

```
SELECT TYPE_ID, MAX_SIZE,
       NO_OF_DECIMALS
FROM USER_DEF_DATA_TYPE
A,
       MODULE_PA
PARAMETER B
WHERE MOD_ID =
:SQL1-MOD-ID AND
       PARM_ID =
:SQL1-PARM-ID AND
       B.DATA_TY
PE_NAME = A.DATA_TYPE_NAME
```

where SQL1-MOD-ID is the MOD-ID retrieved from SQL statement 1 and SQL1-PARM-ID is the PARM-ID from SQL statement 1.

Increment the input parameter NEXT-PARAMETER-NUMBER by 1.

Generate the following working storage entry in WORK-FILE-NAME.

```
01 PARM-npn
PIC clause
```

where npn is the current value of input parameter NEXT-PARAMETER-NUMBER and PIC clause is the picture clause generated by CDPIC. Send CDPIC the current TYPE-ID, MAX-SIZE and NO-OF-DECIMALS from SQL statement 2.

5.2.1.2 Add the parameter name to the PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter name into PARM-NAME (PARM-USED):

PARM-npn

where npn is the
value of
NEXT-PARAMETER-NUMBER.

- 5.2.1.3 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following MOVE statement into PROC-FILE-NAME:

```
MOVE ES-VAR-INS-esndml-elist  
(CDM-INPUT-INDEX-esndml)  
TO PARM-npn
```

where esndml is the value contained in input parameter ES-NDML-NO, elist is the value contained in input parameter ELIST-INDEX and npn is the current value of NEXT-PARAMETER-NUMBER.

- 5.2.1.4 If processing either a key uniqueness test or a type 1 referential integrity test (ACTION-TYPE equals K or 1) for an insert (ES-ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following MOVE statement into PROC-FILE-NAME:

```
MOVE ES-VAR-INS-esndml-elist TO  
PARM-npn
```

where esndml is the value contained in input parameter ES-NDML-NO, elist is the value contained in input parameter ELIST-INDEX and npn is the current value of NEXT-PARAMETER-NUMBER.

- 5.2.1.5 If processing a type 1 referential integrity test (ACTION-TYPE equals 1) for a modify (ES-ACTION-TYPE equals M), generate the following MOVE statement into PROC-FILE-NAME:

30 September 1990

MOVE ES-VAR-csndml-elists TO
PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, elists is the value
contained in input parameter
ELIST-INDEX and npn is the value
of NEXT-PARAMETER-NUMBER.

- 5.2.1.6 If processing the qualification
for a type 2 referential
integrity test or a select or a
delete or a modify (ACTION-TYPE
equals 2 or S or D or M),
generate the following MOVE
statement or PROC-FILE-NAME: .

MOVE ESQ-VAR-csndml-elists TO
PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, elists is the value
contained in input parameter
ELIST-INDEX and npn is the value
contained in
NEXT-PARAMETER-NUMBER.

- 5.2.1.7 If processing the new column
values for a modify (ACTION-TYPE
equals U), generate the
following MOVE statement on
PROC-FILE-NAME:

MOVE ES-VAR-csndml-elists TO
PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, elists is the value
contained in input parameter
NEXT-PARAMETER-NUMBER.

- 5.2.1.8 If processing an insert
(ACTION-TYPE equals I) and the
data will be residing in user
variables or constants
(VAR-OR-FILE-IND equals V),
generate the following MOVE
statement into PROC-FILE-NAME:

MOVE ES-VAR-INS-esndml-elists
(CDM-INPUT-INDEX-esndml)
TO PARM-npn

where esndml is the value
contained in input parameter
ES-NDML-NO, elists is the value

DS 620341200
30 September 1990
contained in input parameter
ELIST-INDEX and npn is the value
contained in
NEXT-PARAMETER-NUMBER.

- 5.2.1.9 If processing an insert
(ACTION-TYPE equals I) and the
data will be residing in a file
or structure (VAR-OR-FILE-IND
equals F), generate the
following MOVE statement into
PROC-FILE-NAME:

MOVE ES-VAR-INS-esndml-elist TO
PARM-npn

where esndml is the value
contained in input parameter
ES-NDML-NO, enlist is the value
contained in input parameter
ELIST-INDEX and npn is the value
contained in
NEXT-PARAMETER-NUMBER.

- 5.2.1.10 Fetch the next row from SQL
statement 1. If another row is
successfully fetched, continue
processing at step 5.2.1. If no
MOVE rows are successfully
fetched, continue processing at
5.2.4.

- 5.2.2 If the current UNION-DISC from SQL
statement 1 execution equals "1"
indicating a tag, perform the following
steps:

- 5.2.2.1 Generate a working storage entry
for the tag.

Extract type, size and number of
decimals for the parameter by
executing SQL statement 2 as
described in step 5.2.1.1.

Increment the input parameter
NEXT-PARAMETER-NUMBER by 1.
Remember this value for later
use in step 5.2.5.

Generate the following working
storage entry in WORK-FILE-NAME:

01 PARM-npn PIC clause

where npn is the current value
of input parameter
NEXT-PARAMETER-NUMBER and PIC
clause is the picture clause

DS 620341200
30 September 1990
generated by CDPIC. Send CDPIC
the current TYPE-ID, MAX-SIZE
and NO-OF-DECIMALS from SQL
statement 2.

- 5.2.2.2 Add the parameter name to the
PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter
name into PARM-NAME(PARM-USED):

PARM-npn

where npn is the value of
NEXT-PARAMETER-NUMBER.

- 5.2.2.3 Fetch the next row from SQL
statement 1. If another row is
successfully fetched, continue
processing at 5.2.1. If no MOVE
rows are successfully fetched,
continue processing at step
5.2.4.

- 5.2.3 If the current UNION-DISC from SQL
statement 1 equals "2" indicating a
constant, perform the following steps:

- 5.2.3.1 Generate a working storage entry
for the constant.

Extract type, size and number of
decimals for the parameter by
executing SQL statement 2 as
described in step 5.2.1.1.

Increment the input parameter
NEXT-PARAMETER-NUMBER by 1.

Generate the following working
storage entry in WORK-FILE-NAME:

01 PARM-npn PIC clause

where npn is the current value
of input parameter
NEXT-PARAMETER-NUMBER and PIC
clause is the picture clause
generated by CDPIC. Send CDPIC
the current TYPE-ID, MAX-SIZE
and NO-OF-DECIMALS from SQL
statement 2.

- 5.2.3.2 Add the constant's name to the
PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter name into PARM-NAME (PARM-USED):

PARM-npn

where npn is the value of NEXT-PARAMETER-NUMBER.

- 5.2.3.3 Generate the move of the constant value into the constant parameter name.

If the TYPE-ID of the constant equals C, generate the following code into PROC-FILE-NAME:

MOVE "constval" TO PARM-npn

where constval is the current CONSTANT-VALUE from SQL statement 1 and npn is the current value of NEXT-PARAMETER-NUMBER.

Place the quote marks around the CMA-CONST-VAL to generate a character literal.

If the TYPE-ID of the constant does not equal C, generate the following code into PROC-FILE-NAME:

MOVE constval TO PARM-npn

where constval is the current CONSTANT-VALUE from SQL statement 1 and npn is the current value of NEXT-PARAMETER-NUMBER.

- 5.2.3.4 Fetch the next row from SQL statement 1. If another row is successfully fetched, continue processing at 5.2.1. If no MOVE rows are successfully fetched, continue processing at step 5.2.4.

- 5.2.4 Generate the call to the user module.

- 5.2.4.1 Generate the following code on PROC-FILE-NAME:

CALL "mod" USING

where mod is the current MOD-ID from SQL statement 1.

- 5.2.4.2 For each used PARM-NAME,
generate a parameter call list
entry on PROC-FILE-NAME.

parml
.
.
.
parmn

where parml through parmn are
the values contained in all used
PARM-NAME entries.

- 5.2.4.3 Generate the status parameter
and terminating period on
PROC-FILE-NAME.

RET-STATUS.

- 5.2.4.4 Generate the status checking
logic on PROC-FILE-NAME.

MOVE RET-STATUS TO NDML-STATUS
IF RET-STATUS NOT = KES-SUCCESSFUL
STRING "mod"
" TRANSFORM PROGRAM FAILED"
DELIMITED BY SIZE INTO MSG-DESC
PERFORM PROCESS-ERROR
GO TO END-NDML-esndml.

where mod is the current MOD-ID from SQL
statement 1 and esndml is the value of
input parameter ES-NDML-NO.

- 5.2.5 Generate the MOVE from the module output
parameter to the conceptual variable.

- 5.2.5.1 If processing a type 1
referential integrity test or a
key uniqueness test (ACTION-TYPE
equals 1 or K) for an insert
(ES-ACTION-TYPE equals I),
generate the following MOVE
statement into PROC-FILE-NAME

MOVE PARM-npn TO CSQ-VAR-csndml-clist

where npn is the value
remembered from step 5.2.2.1,
csndml is the value contained in
input parameter CS-NDML-NO and
clist is the value contained in
input parameter CLIST-INDEX.

- 5.2.5.2 If processing a type 1
referential integrity test
(ACTION-TYPE equals 1) for a

DS 620341200
30 September 1990
modify (ES-ACTION-TYPE equals
M), generate the following MOVE
statement into PROC-FILE-NAME

MOVE PARM-npn TO CSQ-VAR-csndml-clist

where npn is the value
remembered from step 5.2.2.1,
csndml is the value contained in
input parameter CS-NDML-NO and
clist is the value contained in
input parameter CLIST-INDEX.

- 5.2.5.3 If processing the qualification
for a type 2 referential
integrity test or a select or a
delete or a modify (ACTION-TYPE
equals 2 or S or D or M),
generate the following MOVE
statement into PROC-FILE-NAME:

MOVE PARM-npn TO CSQ-VAR-csndml-clist

where npn is the value
remembered from step 5.2.2.1,
csndml is the value contained in
input parameter CS-NDML-NO and
clist is the value contained in
input parameter CLIST-INDEX.

- 5.2.5.4 If processing the new column
values for a modify (ACTION-TYPE
equals U), generate the
following MOVE statement into
PROC-FILE-NAME:

MOVE PARM-npn TO CS-VAR-csndml-clist

where npn is the value
remembered from step 5.2.2.1,
csndml is the value contained in
input parameter CS-NDML-NO and
clist is the value contained in
input parameter CLIST-INDEX.

- 5.2.5.5 If processing an insert
(ACTION-TYPE equals I), generate
the following MOVE statement
into PROC-FILE-NAME:

MOVE PARM-npn TO CS-VAR-csndml-clist

where npn is the value
remembered from step 5.2.2.1,
csndml is the value contained in
input parameter CS-NDML-NO and
clist is the value contained in
input parameter CLIST-INDEX.

5.2.6 Continue processing at step 6.

5.3 If Fortran is to be generated (LANG-NO = -1),
perform the following steps.

5.3.1 If the current UNION-DISC from SQL
statement 1 execution equals "5"
indicating a data item, perform the
following steps.

5.3.1.1 Generate a type declaration for
the data item.

Extract type, size and number of
decimals from SQL statement 2 as
described in step 5.2.1.1.

Increment the input parameter
NEXT-PARAMETER-NUMBER by 1.

If the TYPE-ID returned from SQL
statement 2 equals C, generate
the following statement in
WORK-FILE-NAME:

CHARACTER*maxsize PARM-npn

where maxsize is the value of
MAX-SIZE from SQL statement 2
and npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE-ID returned from SQL
statement 2 equals I, generate
the following statement in
WORK-FILE-NAME:

INTEGER PARM-npn
CHARACTER*6 CHAR-PARM-npn

where npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE-ID returned from SQL
statement 2 equals F, generate
the following in WORK-FILE-NAME:

REAL*sizeno PARM-npn
CHARACTER*maxsize
CHAR-PARM-npn

where npn is the value contained
in NEXT-PARAMETER-NUMBER,
maxsize is the value contained
in MAX-SIZE, and sizeno is 16 if
MAX-SIZE > 15, or 8 if MAX-SIZE
> 7, or 4 if MAX-SIZE < 7.

30 September 1990

- 5.3.1.2 Add the parameter name to the
PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter
name into PARM-NAME (PARM-USED):

```
CHAR-PARM-npn    if TYPE-ID = "I"
PARM-npn         otherwise
```

where npn is the value of
NEXT-PARAMETER-NUMBER.

- 5.3.1.3 If processing either a key
uniqueness test or a type 1
referential integrity test
(ACTION-TYPE equals K or I) for
an insert (ES-ACTION-TYPE equals
I) and the data will be residing
in user variables or constants
(VAR-OR-FILE-IND equals V),
generate the following statement
into PROC-FILE-NAME:

```
PARM-npn =
ES-VAR-INS-esndml-elists(CDM-INPU
T-INDEX-esndml)
```

where npn is the value of
NEXT-PARAMETER-NUMBER, esndml is
the value contained in input
parameter ES-NDML-NO and elists
is the value contained in input
parameter ELIST-INDEX.

- 5.3.1.4 If processing either a key
uniqueness test or a type 1
referential integrity test
(ACTION-TYPE equals K or I) for
an insert (ES-ACTION-TYPE equals
I) and the data will be residing
in a file or structure
(VAR-OR-FILE-IND equals F),
generate the following statement
into PROC-FILE-NAME:

```
PARM-npn =
ES-VAR-INS-esndml-elists
```

where npn is the value of
NEXT-PARAMETER-NUMBER, esndml is
the value contained in input
parameter ES-NDML-NO and elists
is the value contained in input
parameter ELIST-INDEX.

30 September 1990

- 5.3.1.5 If processing a type 1 referential integrity test (ACTION-TYPE equals 1) for a modify (ES-ACTION-TYPE equals M), generate the following statement into PROC-FILE-NAME:

PARM-npn = ES-VAR-csndml-elists

where npn is the value of NEXT-PARAMETER-NUMBER, csndml is the value contained in input parameter CS-NDML-NO and elists is the value contained in input parameter ELIST-INDEX.

- 5.3.1.6 If processing the qualification for a type 2 referential integrity test or a select or a delete or modify (ACTION-TYPE equals 2 or S or D or M), generate the following statement into PROC-FILE-NAME:

PARM-npn = ESQ-VAR-csndml-elists

where npn is the value of NEXT-PARAMETER-NUMBER, csndml is the value contained in input parameter CS-NDML-NO and elists is the value contained in input parameter ELIST-INDEX.

- 5.3.1.7 If processing an insert (ACTION-TYPE equals I) and the data will be residing in user variables or constants (VAR-OR-FILE-IND equals V), generate the following statement into PROC-FILE-NAME:

PARM-npn =
ES-VAR-INS-esndml-elists(CDM-INPUT-INDEX-esndml)

where npn is the value of NEXT-PARAMETER-NUMBER, esndml is the value contained in input parameter ES-NDML-NO and elists is the value contained in input parameter ELIST-INDEX.

- 5.3.1.8 If processing an insert (ACTION-TYPE equals I) and the data will be residing in a file or structure (VAR-OR-FILE-IND equals F), generate the following statement into PROC-FILE-NAME:

PARM-npn =
ES-VAR-INS-esndml-elists

where npn is the value of
NEXT-PARAMETER-NUMBER, esndml is
the value contained in input
parameter ES-NDML-NO and elists
is the value contained in input
parameter ELIST-INDEX.

5.3.1.9 Fetch the next row from SQL
statement 1. If another row is
successfully fetched, continue
processing at step 5.3.1. If no
more rows are successfully
fetched, continue processing at
step 5.3.4.

5.3.2 If the current UNION-DISC from SQL
statement 1 execution equals "1"
indicating a tag, perform the following
steps:

5.3.2.1 Generate a type declaration for
the tag.

Extract type, size and number of
decimal digits by executing SQL
statement 2 as described in step
5.2.1.1.

Increment the input parameter
NEXT-PARAMETER-NUMBER by 1.
Remember this value for later
use in step 5.7.5.

If the TYPE-ID returned from SQL
statement 2 equals C, generate
the following statement in
WORK-FILE-NAME:

CHARACTER*maxsize PARM-npn

where maxsize is the value of
MAX-SIZE from SQL statement 2
and npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE ID returned from SQL
statement 2 equals I, generate
the following statement in
WORK-FILE-NAME:

INTEGER PARM-npn
CHARACTER*6 CHAR-PARM-npn

where npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE-ID returned from SQL statement 2 equals F, generate the following statement in WORK-FILE-NAME:

```
REAL*sizeo          PARM-npn
CHARACTER*maxsize   CHAR-PARM-npn
```

where npn is the value contained in NEXT-PARAMETER-NUMBER, maxsize is the value contained in MAX-SIZE, and sizeo is 16 if MAX-SIZE > 15, or 8 if MAX-SIZE > 7, or 4 if MAX-SIZE < 7.

- 5.3.2.2 Add the parameter name to the PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter name into PARM-NAME (PARM-USED)

```
CHAR-PARM-npn    if TYPE-ID = "I"
PARM-npn         otherwise
```

where npn is the value of NEXT-PARAMETER-NUMBER.

- 5.3.2.3 Fetch the next row from SQL statement 1. If another row is successfully fetched, continue processing at step 5.3.1. If no more rows are successfully fetched, continue processing at step 5.3.4.

- 5.3.3. If the current UNION-DISC from SQL statement 1 equals "2" indicating a constant, perform the following steps:

- 5.3.3.1 Generate a type declaration for the constant.

Extract type, size and number of decimal digits by executing SQL statement 2 as described in step 5.2.1.1.

Increment the input parameter NEXT-PARAMETER-NUMBER by 1.

If the TYPE-ID returned from SQL statement 2 equals C, generate the following statement in WORK-FILE-NAME:

```
CHARACTER*maxsize   PARM-npn
```

where maxsize is the value of
MAX-SIZE from SQL statement 2
and npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE-ID returned from SQL
statement 2 equals I, generate
the following statement in
WORK-FILE-NAME:

```
INTEGER          PARM-npn  
CHARACTER*6      CHAR-PARM-npn
```

where npn is the value contained
in NEXT-PARAMETER-NUMBER.

If the TYPE-ID returned from SQL
statement 2 equals F, generate
the following statement in
WORK-FILE-NAME:

```
REAL*szeno       PARM-npn  
CHARACTER*maxsize CHAR-PARM-npn
```

where npn is the value contained
in NEXT-PARAMETER-NUMBER,
maxsize is the value contained
in MAX-SIZE, and szeno is 16 if
MAX-SIZE > 15, or 8 if MAX-SIZE
> 7, or 4 if MAX-SIZE < 7.

- 5.3.3.2 Add the constant's name to the
PARAMETER-TABLE.

Increment PARM-USED by 1.

Generate the following parameter
name into PARM-NAME (PARM-USED):

```
CHAR-PARM-npn    if TYPE-ID =  
"I"  
PARM-npn         otherwise
```

where npn is the value of
NEXT-PARAMETER-NUMBER.

- 5.3.3.3 Generate the assignment of the
constant value into the constant
parameter name.

If the TYPE-ID of the constant
equals C, generate the following
code into PROC-FILE-NAME:

```
PARM-npn = 'constval'
```

30 September 1990

where npn is the current value of NEXT-PARAMETER-NUMBER and constval is the current CONSTANT-VALUE from SQL statement 1.

If the TYPE-ID of the constant does not equal C, generate the following code into PROC-FILE-NAME:

PARM-npn = constval

where npn is the current value of NEXT-PARAMETER-NUMBER and constval is the current CONSTANT-VALUE from SQL statement 1.

- 5.3.3.4 Generate a call to a routine to convert the data from numeric to character data, if it already isn't character.

If TYPE-ID equals "I", generate:

Call INTFTN(PARM-npn,
CHAR-PARM-npn)

where npn is the current value of NEXT-PARAMETER-NO

If TYPE-ID equals "F", generate:

DECIML = 00

Call RECFIN(DECIML, PARM-npn,
CHAR-PARM-npn)

where npn is the current value of NEXT-PARAMETER-NO

- 5.3.3.5 Fetch the next row from SQL statement 1. If another row is successfully fetched, continue processing at step 5.3.1. If no more rows are successfully fetched, continue processing at step 5.3.4.

- 5.3.4 Generate the call to the user module.

- 5.3.4.1 Generate the following code on PROC-FILE-NAME:

CALL mod (

where mod is the current MOD-ID from SQL statement 1.

5.3.4.2 For each used PARM-NAME,
generate a continuation marker
(* in column 6) followed by a
parameter call list entry on
PROC-FILE-NAME followed by a
comma.

If TARGET-HOST equals
VAX:

```
* parml,  
.  
.  
.  
* parmnn,
```

If TARGET-HOST equals
IBM:

```
*%REF(parml),  
.  
.  
.  
*%REF(parmnn),
```

where parml through parmnn are
the values contained in all used
PARM-NAME entries.

5.3.4.3 Generate a continuation marker,
the status variable and the
terminating parenthesis on
PROC-FILE-NAME
If TARGET-HOST equals VAX:
* NDMLST)

If TARGET-HOST equals
IBM:
*%REF(NDMLST))

5.3.4.4 Generate the status checking
logic on PROC-FILE-NAME.

```
IF (NDMLST .NE. '00000') THEN  
GO TO 93esndml  
ENDIF
```

where mod is the current MOD-ID
from SQL statement 1 and esndml
is the value of input parameter
ES-NDML-NO.

5.3.5 Generate the assignment of the module
output parameter to the conceptual
variable.

5.3.5.1 If processing a type 1
referential integrity test or a
key uniqueness test (ACTION-TYPE
equals 1 or K) for an insert

DS 620341200
30 September 1990
(ES-ACTION-TYPE equals I),
generate the following statement
on PROC-FILE-NAME:

CSQ-VAR-csndml-clist = PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, clist is the value
contained in input parameter
CLIST-INDEX and npn is the value
remembered from step 5.3.2.1.

- 5.3.5.2 If processing a type 1
referential integrity test
(ACTION-TYPE equals 1) for a
modify (ES-ACTION-TYPE equals
M), generate the following
statement on PROC-FILE-NAME:

CSQ-VAR-csndml-clist = PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, clist is the value
contained in input parameter
CLIST-INDEX and npn is the value
remembered from step 5.3.2.1.

- 5.3.5.3 If processing the qualification
for a type 2 referential
integrity test or a select or a
delete or a modify (ACTION-TYPE
equals 2 or S or D or M),
generate the following statement
on PROC-FILE-NAME:

CSQ-VAR-csndml-clist = PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, clist is the value
contained in input parameter
CLIST-INDEX and npn is the value
remembered in step 5.3.2.1.

- 5.3.5.4 If processing the new column
values for a modify (ACTION-TYPE
equals U), generate the
following statement on
PROC-FILE-NAME:

CS-VAR-csndml-clist = PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, clist is the value

DS 620341200
30 September 1990
contained in input parameter
CLIST-INDEX and npn is the value
remembered in step 5.3.2.1.

- 5.3.5.5 If processing an insert
(ACTION-TYPE equals I), generate
the following statement on
PROC-FILE-NAME:

CS-VAR-csndml-clist = PARM-npn

where csndml is the value
contained in input parameter
CS-NDML-NO, clist is the value
contained in input parameter
CLIST-INDEX and npn is the value
remembered in step 5.3.2.1.

6. Close WORK-FILE-NAME and PROC-FILE-NAME.
7. Terminate processing.

33.5 Outputs

1. RET-STATUS PIC X(5)

Error status. A value equal to KES-SUCCESSFUL as
defined in the ERRCDM copy marker indicates success.

SECTION 34

FUNCTION CDECWS - Generate Data Definitions for runtime
 update/search values.

This function will:

1. Generate External/Conceptual Schema Data Defintions for runtime insert, modify and qualify data values.
2. Generate code into the Working Storage section of the modified AP. Depending on CS-ACTION-TYPE the following code will be generated:

01	CS-VAR-ccc-nn	PIC type(size)[V9(nd)]
01	CSQ-VAR-ccc-nn	PIC type(size)[V9(nd)]
01	ES-VAR-ccc-nn	PIC type(size)[V9(nd)]
01	ESQ-VAR-ccc-nn	PIC type(size)[V9(nd)]

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

34.1 Inputs

1. Source Language Indicator of the Application Program
LANG-NO
2. Application Program parcel names
WORK-FILE-NAME
3. Conceptual Schema representation of the data
CS-ACTION-LIST
CS-QUALIFY-LIST
4. External Schema representation of the data
ES-ACTION-LIST
ES-QUALIFY-LIST

5. Fortran Variable Association Table

FORTTRAN-VARIABLE-TABLE

34.2 CDM Requirements

None

34.3 Internal Requirements

None

34.4 Processing

1. Generate Conceptual schema definitions for CS-ACTION of "1" or "K".

Scan CS-QUALIFY list for

CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-ES-PTR NOT = 0

- 1.1 Generate 01 level for table containing qualify values if entries were found. If language is COBOL, generate:

01 CDM-CSQ-TABLE-ccc

Else, generate:

CHARACTER*tt CDM-CSQ-TABLE-ccc

where

ccc = CS-NDML-NO
tt = total size of all entries found in the
above scan.

- 1.2 Generate the following using the same process as in scan above. If language is COBOL, generate:

03 CSQ-VAR-ccc-ii PIC clause.

where

ccc = CS-NDML-NO
ii = CSQ-INDEX
PIC clause = meta data from current CSQ entry (L)

else perform steps 2.2.1 through 2.2.2.

- 1.2.1 If CSQ-L-TYPE is equal "C", generate:

CHARACTER*11 CSQ-VAR-ccc-nn

Else, generate:

30 September 1990

DOUBLE PRECISION CSQ-VAR-ccc-nn
 CHARACTER*11 XHQcccnn
 CHARACTER*ss CSQ-LONG-VAR-ccc-nn

where 11 = CSQ-L-SIZE
 ccc = CS-NDML-NO
 nn = CSQ-INDEX
 ss = CSQ-L-SIZE + 1

1.2.2 If ES-ACTION not Insert, generate additional code.

1.2.2.1 If ES-TYPE is equal "C", generate:

CHARACTER*ee ES-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION ES-VAR-ccc-nn
 CHARACTER*ee XHSCccnn

where ee = ES-SIZE
 ccc = ES-NDML-NO
 nn = CSQ-ES-PTR

1.3 If nothing was generated in step 2.1 generate if COBOL:

01 CDM-CSQ-TABLE-ccc PIC X.

else if FORTRAN, generate:

CHARACTER*1 CDM-CSQ-TABLE-ccc

where

ccc = CS-NDML-NO

2. Generate External Schema definitions for CS-ACTION of "1" or "K" if language is COBOL.

2.1 Scan CS-QUALIFY list for

CSQ-ECNOR = 0
 CSQ-AUCR = 0
 CSQ-ES-PTR NOT = 0

For each entry found generate:

01 ES-VAR-ccc-nn PIC type(size)(nd).

where

ccc = CS-NDML-NO
 nn = CSQ-ES-PTR

Call CDPIC to generate picture clause using meta data in the ES-ACTION-LIST.

ES-INDEX = CSQ-ES-PTR

3. Generate Conceptual Schema definitions for CS-ACTION of "S", "D", "2", or "M".

Scan CS-QUALIFY list for

CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-ES-PTR NOT = 0
CSQ-SOURCE = "U"
or
CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-SOURCE = "V"

- 3.1 Generate 01 level for table containing qualify values if entries were found. If language is COBOL, generate:

01 CDM-CSQ-TABLE-ccc.

Else, generate:

CHARACTER*tt CDM-CSQ-TABLE-ccc

where

ccc = CS-NDML-NO
ttt = total size of CSQ entries found

above

- 3.2 Generate the following using the same procedure as in above scan. If language is COBOL, generate:

03 CSQ-VAR-ccc-ii PIC clause

where

ccc = CS-NDML-NO
ii = CSQ-INDEX
clause = meta data from current CSQ entry

(L)

else perform steps 3.2.1 through 3.2.2.

- 3.2.1 If CSQ-L-TYPE is equal "C", generate:

CHARACTER*11 CSQ-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION CSQ-VAR-ccc-nn
CHARACTER*11 XHQcccn
CHARACTER*ss CSQ-LONG-VAR-ccc-nn

where ll = CSQ-L-SIZE
ccc = CS-NDML-NO
nn = CSQ-INDEX
ss = CSQ-L-SIZE + 1

3.2.2 Generate additional code:

If ESQ-L-TYPE is equal "C", generate:

CHARACTER*ee ESQ-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION ESQ-VAR-ccc-nn
CHARACTER*ee XQHcccnn

where ee = ESQ-L-SIZE
ccc = CS-NDML-NO
nn = CSQ-ES-PTR

3.3 If nothing was generated in step 3.1, generate if COBOL:

01 CDM-CSQ-TABLE-ccc PIC X.

Else, generate:

CHARACTER*1 CDM-CSQ-TABLE-ccc

where

ccc = CS-NDML-NO

4. Generate External Schema definitions for CS-ACTION of "S", "D", "2", or "M".

Scan CS-QUALIFY list for

CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-ES-PTR NOT = 0
CSQ-SOURCE NOT = "V"

For each entry found if language is COBOL generate:

01 ESQ-VAR-ccc-nn PIC clause

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR
clause = meta data in ES-QUALIFY list using CSQ-ES-PTR
as ESQ-INDEX

5. Generate External/Conceptual definitions for CS-ACTION of "I" or "M".

Scan CS-ACTION-LIST for
CS-ES-PTR NOT = 0
CS-SOURCE NOT = "G"

For each entry found, if COBOL, generate:

01 ES-VAR-ccc-nn PIC clause.

where

ccc = CS-NDML-NO
nn = CS-ES-PTR
clause = meta data from ES-ACTION list using
CS-ES-PTR as ES-INDEX

01 CS-VAR-ccc-ii PIC clause.

where

ccc = CS-NDML-NO
ii = CS-INDEX
clause = meta data from CS-ACTION-LIST

else if FORTRAN, perform steps 5.1 through 5.2.

5.1 If CS-TYPE is equal "C", generate:

CHARACTER*11 CS-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION CS-VAR-ccc-nn
CHARACTER*11 XHScccnn
CHARACTER*ss CS-LONG-VAR-ccc-nn

where 11 = CS-SIZE
ccc = CS-NDML-NO
nn = CS-INDEX
ss = CS-SIZE + 1

5.2 If CS-ACTION not Insert, generate additional code.

5.2.1 If ES-TYPE is equal "C", generate:

CHARACTER*ee ES-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION ES-VAR-ccc-nn

where ee = ES-SIZE
ccc = CS-NDML-NO
nn = CSQ-ES-PTR

34.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program.

ccc = CS-NDML-NO
 nn = CS-ES-PTR
 clause = meta data from ES-ACTION list using
 CS-ES-PTR as ES-INDEX

01 CS-VAR-ccc-ii PIC clause.

where

ccc = CS-NDML-NO
 ii = CS-INDEX
 clause = meta data from CS-ACTION-LIST

else if FORTRAN, perform steps 5.1 through 5.2.

5.1 If CS-TYPE is equal "C", generate:

CHARACTER*11 CS-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION CS-VAR-ccc-nn
 CHARACTER*11 XHSCccnn
 CHARACTER*ss CS-LONG-VAR-ccc-nn

where 11 = CS-SIZE
 ccc = CS-NDML-NO
 nn = CS-INDEX
 ss = CS-SIZE + 1

5.2 If CS-ACTION not Insert, generate additional code.

5.2.1 If ES-TYPE is equal "C", generate:

CHARACTER*ee ES-VAR-ccc-nn

Else, generate:

DOUBLE PRECISION ES-VAR-ccc-nn

where ee = ES-SIZE
 ccc = CS-NDML-NO
 nn = CSQ-ES-PTR

34.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program.

SECTION 35

FUNCTION CDUEMV Generate "MOVE" statements for runtime update/
search values.

This function will:

1. Generate MOVE statements for insert, modify and qualify values from user defined variables or constants to External Schema format data definitions.
2. Code will be generated in the Procedure Division of the modified AP to move the user defined variable value or constant to External Schema format data definitions. Depending on CS-ACTION-TYPE the following code will be generated:

```
MOVE {var  
value TO {ES-VAR-cc-nn  
'constant'} ESQ-VAR-ccc-nn}
```

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

35.1 Inputs

1. External Schema representation of the data
ES-ACTION-LIST
ES-QUALIFY-LIST
2. Conceptual Schema representation of the data
CS-ACTION-LIST
CS-QUALIFY-LIST
3. Application Program parcel names
PROC-FILE-NAME
4. Source Language Indicator of the Application Program

LANG-NO

5. Fortran Variable Association Table

FORTTRAN-VARIABLE-TABLE

35.2 CDM Requirements

None

35.3 Internal Requirements

None

35.4 Processing

1. Generate "MOVE" statements for CS-ACTION of "1" or "K".

Scan CS-QUALIFY-LIST for

CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-ES-PTR NOT = 0

Generate the following code for each entry:

- 1.1 If COBOL, generate the "Move" statement:

```
      {var  
MOVE  value          TO      ES-VAR-ccc-nn.  
      'constant'}
```

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT or ES-VALUE
in ES-ACTION-LIST using CSQ-ES-PTR as ES-INDEX

If FORTRAN, generate the assign statement:

```
      ES-VAR-ccc-nn = {var,  
                      value,  
                      'constant'}
```

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT or ES-VALUE
in ES-ACTION-LIST

2. Generate "MOVE" statements for select, delete or modify values (CS-ACTION = "S", "D", "M" or "2").

2.1 Scan CS-QUALIFY-LIST for
CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-SOURCE = "V"
CSQ-OP NOT = "NN" or "NL"

If COBOL, generate the statement:

MOVE "CV" to CSQ-VAR-cc-ii
where cv = CSQ-VARY1
cc = CS-NDML-NO
ii = CSQ-INDEX

If FORTRAN, generate the statement:

CSQ-VAR-cc-ii = 'CV'
where cv = CSQ-VARY1
cc = CS-NDML-NO
ii = CSQ-INDEX

2.2 Scan CS-QUALIFY-LIST for

CSQ-ECNOR = 0
CSQ-AUCR = 0
CSQ-ES-PTR NOT = 0

If COBOL, generate the "MOVE" statement

MOVE {var
value TO ESQ-VAR-ccc-nn
'constant'}

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT OR ES-VALUE
in ES-ACTION-LIST

ES-INDEX = CS-ES-PTR

If FORTRAN, generate the assign statement:

ESQ-VAR-ccc-nn = {var,
value,
'constant'}

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT or ES-VALUE in
ES-ACTION-LIST

3. Generate "MOVE" statements for modify values (CS-ACTION = "M").

Scan CS-ACTION-LIST for

CS-ES-PTR NOT = 0
CS-SOURCE NOT = G

- 3.1 If COBOL, generate the "MOVE" statement.

```
      {var  
MOVE  value          TO      ES-VAR-ccc-nn  
      'constant'}
```

where

ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT or ES-VALUE
in ES-ACTION-LIST using CS-ES-PTR as
ES-INDEX

If FORTRAN, generate the assign statement:

```
ES-VAR-ccc-nn = {var  
                  value  
                  'constant'}
```

where ccc = CS-NDML-NO
nn = CSQ-ES-PTR

var, value, constant is determined from
ES-LOCAL-VARIABLE, ES-SUBSCRIPT or ES-VALUE
in ES-ACTION-LIST

35.5 Outputs

1. Error status of the function

RET-STATUS

2. Code generated into the parcels of the Application Program.

SECTION 36

FUNCTION CDP10S - Perform Query Combination

This function will:

1. Generate code into the procedure parcel of the application program that is used to assemble the input parameters required for the appropriate aggregator to satisfy an NDML query combination command.
2. Generate code into the working storage parcel of the application program to be used as input parameters for the aggregator calls.
3. Generate code into the procedure parcel of the application program to call the appropriate aggregator module to satisfy an NDML Query Combination command.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

36.1 Inputs

1. Query Combination Type

The NDML Query Combination operator:

J-INTERSECT
D-DIFFERENCE

U-UNION

- 01 QUERY-TYPE PIC X.
- 2. ES-NDML Identifier for first Results
 - 01 ES-NDML-ID1 PIC X(4).
- 3. CS-NDML Identifier for first Results
 - 01 CS-NDML-ID1 PIC X(6).
- 4. ES-NDML Identifier for second Results
 - 01 ES-NDML-ID2 PIC X(4).
- 5. CS-NDML Identifier for second Results
 - 01 CS-NDML-ID2 PIC X(6).
- 6. Working Storage Work File Name
 - 01 WORK-FILE1 PIC X(30).
- 7. Procedure Division Work File Name
 - 01 WORK-FILE2 PIC X(30).
- 8. Source Language Indicator of the Application Program
 - 01 LANG-NAME PIC X(10).
- 9. Conceptual Schema Action List
 - CS-ACTION-LIST
- 10. Result Field Table
 - RFT
- 11. Join Query Graph Attribute Pair List
 - JQG-ATTRIBUTE-PAIR-LIST
- 12. Fortran Variable Association Table

FORTRAN-VARIABLE-TABLE

13. Target Host Name PIC XXX.

TARGET-HOST

36.2 CDM Requirements

None

36.3 Internal Requirements

1. Next Identifier Counters

01 NEXT-ID-CTR PIC 9(3).

01 NEXT-ID-CTR1 PIC 9(5).

36.4 Processing

1. Generate working storage variables required for the Query Combination.

1.1 Increment the next identifier counters for Query Combination.

1.2 Construct the external NEXT-QRY-ID variable using the constant "Q" and NEXT-ID-CTR.

Construct the conceptual NEXT-QRY-ID variable using the constant "Q" and NEXT-ID-CTR1.

1.3 Generate working storage variables to hold name of the results file and the result count for the Query Combination. If language is COBOL, generate:

01	CDM-CS-RESULTS-xxxx	PIC X(80).
01	CDM-CS-COUNT-xxxx	PIC 9(6).

else, generate:

CHARACTER*80	CDM-CS-RESULTS-xxxx
CHARACTER*6	CDM-CS-COUNT-xxxx

where

xxxx = NEXT-CS-QRY-ID

- 1.4 Generate copy statements to include the Attribute Pair List for the aggregation process. If language is COBOL, generate:

```
COPY APL OF IISSCLIB REPLACING
==JQG-ATTRIBUTE-PAIR-LIST==BY
==CDM-APL-xxxx==
```

else, generate:

```
CHARACTER*tt          CDM-API-xxxx
DATA CDM-APL-xxxx     /'rr0022'/
```

where

```
tt    = APL-ENTRY * number non-deleted CS
        entries + APL-HEAD
xxxx  = NEXT-CS-QRY-ID
rr    = actual number of non-deleted CS
        entries.
```

- 1.5 Generate copy statements to include the Conceptual Schema Action Table and the Result Field Table for the aggregation process.

Generate if COBOL:

```
COPY CSAL OF IISSCLIB REPLACING
==CS-ACTION-LIST==BY
==CDM-CSAL-xxxx-cccc==
```

```
COPY RFTABLE OF IISSCLIB REPLACING
==01 RFT== BY
==01 CDM-RFT-xxxx-cccc==.
```

else, generate:

If CS-USED > 0

```
CHARACTER*tt          CDM-CSAL-xxxx-cccc
DATA CDM-CSAL-xxxx-cccc /'uu'/
```

else,

CHARACTER*4 CDM-CSAL-xxxx-cccc
 /'0000'/

If RFT-USED > 0

CHARACTER**ee CDM-RFT-xxxx-cccc
DATA CDM-RFT-xxxx-cccc /'ff000024'/

else,

CHARACTER*6 CDM-RFT-xxxx-cccc
DATA CDM-RFT-xxxx-cccc /'000000'/

where

tt = CS-ENTRY * CS-USED + CS-HEAD
uu = CS-USED
xxxx = NEXT-QRY-ID
cccc = NEXT-CS-QRY-ID
ee = RFT-ROW * RFT-USED + RFT-HEAD
ff = RFT-USED

2. Generate procedure division code to satisfy the NDML
Query Combination command.

2.1 Generate code to populate the Result Field
Table, Conceptual Schema Action List and the
Attribute Pair List required for the aggregation
process.

2.1.1 If COBOL, call "CDMACR" utility with the
following:

Library Name - COBOL
Macro Name - RFTBUILD
Parameters
 P1 = ES-NDML-ID1
 P2 = CS-NDML-ID1
 P3 = ES-NDML-ID2
 P4 = CS-NDML-ID2
 P5 = NEXT-QRY-ID
 P6 = NEXT-CS-QRY-ID
File Name = WORK-FILE2

else, generate FORTRAN code to

correspond to COBOL RFTBUILD MACRO.

2.2 Generate code to call the appropriate aggregator to perform the Query Combination:

2.2.1 If Query Combination Type = "I", call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - J01CALL
Parameters
P1 = ES-NDML-ID1
P2 = ES-NDML-ID2
P3 = NEXT-CS-QRY-ID
P4 = CS-NDML-ID1
P5 = CS-NDML-ID2
File Name = WORK-FILE2

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - J01CALL
Parameters
P1 = CDM-CS-RESULTS-cc1
P2 = CDM-CS-RESULTS-cc2
P3 = CDM-APL-cq
P4 = CDM-RFT-ee1-cc1
P5 = CDM-RFT-ee2-cc2
P6 = CDM-CS-COUNT-cq
P7 = CDM-CS-RESULTS-cq

where cc1 = CS-NDML-ID1
cc2 = CS-NDML-ID2
ee1 = ES-NDML-ID1
ee2 = ES-NDML-ID2
cq = NEXT-CS-QRY-ID

2.2.2 If Query Combination Type = "U", call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - U01CALL
Parameters
P1 = CS-NDML-ID1
P2 = CS-NDML-ID2
P3 = NEXT-CS-QRY-ID
P4 = ES-NDML-ID1

Library Name - VAXFORTRAN or IBMFORTRAN

Macro Name - U01CALL

Parameters

P1 = CDM-CS-RESULTS-cc1
 P2 = CDM-CS-RESULTS-cc2
 P3 = CDM-CS-RESULTS-cq
 P4 = CDM-RFT-eel-cc1

where cc1 = CS-NDML-ID1
 cc2 = CS-NDML-ID2
 cq = NEXT-CS-QUERY-ID

- 2.2.3 If Query Combination Type = "D", call "CDMACR" utility with the following:

Library Name - COBOL

Macro Name - N01CALL

Parameters

P1 = ES-NDML-ID1
 P2 = ES-NDML-ID2
 P3 = NEXT-CS-QRY-ID
 P4 = CS-NDML-ID1
 P5 = CS-NDML-ID2

Library Name - VAXFORTRAN or IBMFORTRAN

Macro Name - N01CALL

Parameters

P1 = CDM-CS-RESULTS-cc1
 P2 = CDM-CS-RESULTS-cc2
 P3 = CDM-APL-cq
 P4 = CDM-RFT-eel-cc1
 P5 = CDM-RFT-ee2-cc2
 P6 = CDM-CS-COUNT-cq
 P7 = CDM-CS-RESULTS-cq

where cc1 = CS-NDML-ID1
 cc2 = CS-NDML-ID2
 eel = ES-NDML-ID1
 ee2 = ES-NDML-ID2
 cq = NEXT-CS-QRY-ID

- 2.3 Generate a closing label for the Query Combination. If COBOL, generate:

END-NDML-xxxx.

else, generate:

99ii CONTINUE

where

xxxx = NEXT-QRY-ID
ii = NEXT-ID-CTR

36.5 Outputs

1. Next External Identifier for Query Combination results.

01 NEXT-QRY-ID PIC X(4)

2. Next Conceptual Identifier for Query Combination results file.

01 NEXT-CS-QRY-ID PIC X(6)

3. Return Status

01 RET-STATUS PIC X(5)

CDP10S MACROS

Library Name - COBOL

Macro Name - J01CALL

Parameters - P1
 P2
 P3
 P4
 P5

*
* Call the Join aggregator to Perform the Intersect
*

CALL "CDJS1" USING
 CDM-CS-RESULTS-P4
 CDM-CS-RESULTS-P5
 CDM-APL-P3
 CDM-RFT-P1-P4
 CDM-RFT-P2-P5
 CDM-RFT-P1-P4

*
 CDM-CS-COUNT-P3
 CDM-CS-RESULTS-P3
 NDML-STATUS.

Library name - COBOL

Macro Name - U01CALL

Parameters - P1
 P2
 P3
 P4

*

* Call the Union Aggregator to Perform the UNION.

CALL "CDUS1" USING
 CDM-CS-RESULTS-P1
 CDM-CS-RESULTS-P2
 CDM-CS-RESULTS-P3
 CDM-RFT-P4-P1
 NDML-STATUS.

Library Name - COBOL

Macro Name - N01CALL

Parameters - P1
P2
P3
P4
P5

*

* Call the Not In Set Aggregator to Perform the Difference

CALL "CDNS1" USING
CDM-CS-RESULTS-P4
CDM-CS-RESULTS-P5
CDM-APL-P3
CDM-RFT-P1-P4
CDM-RFT-P2-P5

*

CDM-CS-COUNT-P3
CDM-CS-RESULTS-P3
NDML-STATUS.

Library Name - COBOL

Macro Name - RFTBUILD

Parameters - P1
P2
P3
P4
P5
P6

```

MOVE 0 TO NDML-CS-COUNT.
MOVE 0 TO NDML-RFT-COUNT.
NDML-TEMP-LOOP-P5.
  ADD 1 TO NDML-CS-COUNT.
  IF NDML-CS-COUNT > CS-USED OF CDM-CSAL-P1-P2
    MOVE NDML-RFT-COUNT TO CS-USED OF CDM-CSAL-P5-P6
                        RFT-USED OF CDM-RFT-P1-P2
                        APL-USED OF CDM-APL-P6

  MOVE 0 TO NDML-CS-COUNT
  MOVE 0 TO NDML-RFT-COUNT
  GO TO NDML-LOOP-PL-END-P5.
  IF NOT-CS-DELETED OF CDM-CSAL-P1-P2 (NDML-CS-COUNT)
    ADD 1 TO NDML-RFT-COUNT
    MOVE CS-AUC OF CDM-CSAL-P1-P2 (NDML-CS-COUNT) TO
      RFT-ATTR OF CDM-RFT-P1-P2 (NDML-RFT-COUNT)
    JQG-ATTRL OF CDM-APL-P6 (NDML-RFT-COUNT)
    CS-AUC OF CDM-CSAL-P5-P6 (NDML-RFT-COUNT)
    MOVE CS-TYPE OF CDM-CSAL-P1-P2 (NDML-CS-COUNT) TO
      RFT-TYPE OF CDM-RFT-P1-P2 (NDML-RFT-COUNT)
    CS-TYPE OF CDM-CSAL-P5-P6 (NDML-RFT-COUNT)
    MOVE CS-SIZE OF CDM-CSAL-P1-P2 (NDML-CS-COUNT) TO
      RFT-SIZE OF CDM-RFT-P1-P2 (NDML-RFT-COUNT)
    CS-SIZE OF CDM-CSAL-P5-P6 (NDML-RFT-COUNT)
    MOVE "=" TO JQG-OP OF CDM-APL-P6 (NDML-RFT-COUNT)
    MOVE 1 TO
      JQG-SUBTRANSL OF CDM-APL-P6 (NDML-RFT-COUNT)
      RFT-SUBTRANS OF CDM-RFT-P1-P2 (NDML-RFT-COUNT)
    MOVE CS-ND OF CDM-CSAL-P1-P2 (NDML-CS-COUNT) TO
      RFT-ND OF CDM-RFT-P1-P2 (NDML-RFT-COUNT)
    CS-ND OF CDM-CSAL-P5-P6 (NDML-RFT-COUNT).
  GO TO NDML-TEMP-LOOP-PL-P5.
NDML-LOOP-PL-END-P5.
NDML-TEMP-LOOP-PR-P5.
  ADD 1 TO NDML-CS-COUNT.

```

```
IF NDML-TEMP-COUNT > CS-USED OF CDM-CSAL-P3-P4
  MOVE NDML-RFT-COUNT TO RFT-USED OF CDM-RFT-P3-P4
  MOVE 0 TO NDML-CS-COUNT
  MOVE 0 TO NDML-RFT-COUNT
  GO TO NDML-LOOPR-END-P5.
IF NOT-CS-DELETED OF CDM-CSAL-P3-P4 (NDML-CS-COUNT)
  ADD 1 TO NDML-RFT-COUNT
  MOVE CS-AUC OF CDM-CSAL-P3-P4 (NDML-CS-COUNT) TO
    RFT-ATTR OF CDM-RFT-P3-P4 (NDML-RFT-COUNT)
  JQG-ATTRR OF CDM-APL-P6 (NDML-RFT-COUNT)
  MOVE CS-TYPE OF CDM-CSAL-P3-P4 (NDML-CS-COUNT) TO
    RFT-TYPE OF CDM-RFT-P3-P4 (NDML-RFT-COUNT)
  MOVE CS-SIZE OF CDM-CSAL-P3-P4 (NDML-CS-COUNT) TO
    RFT-SIZE OF CDM-RFT-P3-P4 (NDML-RFT-COUNT)
  MOVE 2 TO
    JQG-SUBTRANSR OF CDM-APL-P6 (NDML-RFT-COUNT)
    RFT-SUBTRANS OF CDM-RFT-P3-P4 (NDML-RFT-COUNT)
  MOVE CS-ND OF CDM-CSAL-P3-P4 (NDML-CS-COUNT) TO
    RFT-ND OF CDM-RFT-P3-P4 (NDML-RFT-COUNT).
GO TO NDML-TEMP-LOOPR-P5.
NDML-LOOPR-END-P5.
```

LIBRARY NAME - FORTRAN

MACRO NAME - J01CALL

PARAMETERS - P1
P2
P3
P4
P5
P6
P7

```
CALL CDJS1( %REF(P1), %REF(P2), %REF(P3)
*           , %REF(P4), %REF(P5)
*           , %REF(P6)
*           , %REF(P7), %REF(NDMLST))
*
```

LIBRARY NAME - FORTRAN

MACRO NAME - N01CALL

PARAMETERS - P1
P2
P3
P4
P5
P6
P7

```
CALL CDNS1( %REF(P1), %REF(P2), %REF(P3)
*          , %REF(P4), %REF(P5)
*          , %REF(P6)
*          , %REF(P7), %REF(NDMLST) )
```

LIBRARY NAME - FORTRAN

MACRO NAME - U01CALL

PARAMETERS - P1
P2
P3
P4

```
      CALL CDUS1( %REF(P1), %REF(P2)
*           , %REF(P3)
*           , %REF(P1), %REF(NDMLST))
```

SECTION 37

FUNCTION CDP10T - Generate code to perform final mapping of results from Query Combination Command.

This function will:

1. Generate code into the working storage parcel of the Application Program to be used for the final mapping of results from a Query Combination command.
2. Generate code into the procedure parcel of the Application Program to call the C/E Transform Program and map the results into user specified variables, structure or file.

NOTE: If the user's application program is written in FORTRAN, then as of release 2.3, all FORTRAN variable names will be generated with a length of six. This will be done by generating names of the convention: CDMXXX where XXX is any combination of three characters. The three character combination is determined by routine CDCREFO. This routine associates a six character FORTRAN variable with the corresponding COBOL variable. This association between the COBOL name and the generated FORTRAN name is stored in the FORTRAN-VARIABLE-TABLE. The FORTRAN-VARIABLE-TABLE is copied into modified user's application program. In this design specification, the COBOL name will be used to show how the FORTRAN code will be generated.

37.1 Inputs

1. External Schema representation of the Data

ES-ACTION-LIST (mapping SELECT)
ES-ACTION-LIST (inner SELECT)

2. Conceptual Schema representation of the Data

CS-ACTION-LIST (inner SELECT)

3. Code Generation Table

CODE-GENERATOR-TABLE

4. Application Program Parcel Names

IDFILE-NAME PIC X(30)
FDFILE-NAME
WORKFILE-NAME
PROCFILE-NAME

5. Application Program Error File Name

ERROR-FILE PIC X(30)

6. Source Language Indicator of the Application Program

SOURCE-LANGUAGE

7. Input-Output Section Indicator

IOSECTION-INDICATOR

8. Host Information about the Application

TARGET-HOST PIC X(3)
CURRENT-HOST PIC X(3)

9. Block Stack

BLOCK-STACK

10. Next Conceptual Schema Query Identification Number

NEXT-CS-QRY-ID PIC X(6)

11. Action Symbol

ACTION-SYMBOL PIC X

12. Fortran Variable Association Table

FORTTRAN VARIABLE TABLE

37.2 CDM Requirements

None

37.3 Internal Requirements

01	MOD-NAME	PIC X(10).	
01	CE-EMPTY	PIC 9999	VALUE 0.
01	BOOL-EMPTY	PIC 9999	VALUE 0.
01	CSQ-EMPTY	PIC 9999	VALUE 0.
01	CMA-FLAG	PIC 9	VALUE 0.
01	DBMS-NAME	PIC X(30).	
01	GEN-FILE-NAME	PIC X(30).	
01	NEXT-QRY-ID	PIC X.	

37.4 Processing

1. Obtain a program name for the Conceptual/External Transformation Program. Call routine "APNAME" with the following parameters to obtain a program name:

30 September 1990

DBMS-NAME - variable containing value of spaces
 MOD-NAME - output parameter containing a new
 program name
 RET-STATUS - returned status of function APNAME

- 1.1 If ES-SEMI-CURLY-IND of mapping (outer) SELECT, add another entry to the BLOCK-STACK.

Add 1 to BLOCK-INDEX
 Set MOD-NAME-STACK to MOD-NAME
 Set CS-NDML-NO-STACK to CS-NAME-NO

2. Determine if the number of result fields requested on the outer select, match those requested on the inner selects.

- 2.1 If ES-FILE-NAME or ES-STRUCTURE does not equal space, continue at step 3.

- 2.2 Count the number of projected columns of ES-ACTION-INNER. If this number does not equal ES-USED of ES-ACTION-LIST, generate an error message, set the status variable and exit.

3. Combine the ES-ACTION List of the inner selects with the ES-ACTION List of the outer (mapping) select.

- 3.1 For each used entry in the Inner ES-ACTION List, transfer the type, size and number of decimal digits to the corresponding entry in the mapping ES-ACTION List.

ES-TYPE
 ES-SIZE
 ES-ND
 ES-PROJECT-FLAG
 ES-DELETE-FLAG

4. Generate a Conceptual/External Schema Transformation Program for the final results of the Query Combination command.

- 4.1 Call function CDPRE8 with the following parameters:

TARGET-HOST	input parameter
CURRENT-HOST	input parameter
MOD-NAME	name of program obtained in step 1
ES-ACTION-LIST	combined action list from step 3
CS-ACTION-LIST	input parameter (modified to not include generated entries)
BOOL-EMPTY	indicates no entries in BOOLEAN list
CSQ-EMPTY	indicates no entries in the CS-QUALIFY list
ISQ-EMPTY	indicates no entries in the IS-QUALIFY LIST
ERROR-FILE	input parameter
CMA-FLAG	indicates no complex mapping

transformation
GEN-FILE-NAME returned file name of file
containing the Generated C/E
Transform Program

LANG-NO indicates what language the user's AP
was written in.

RET-STATUS returned status of function CDPRE8

4.2 Store the file name of the Generated C/E
Transform Program in the Code Generation Table.

5. Generate working storage variables required for
processing the final results.

5.1 Generate working storage variables to hold the
name of the file containing the final conceptual
results and table for qualify variables. If
language is COBOL, generate:

01 CDM-CSQ-TABLE-eee PIC X.

Else, generate:

CHARACTER*1 CDM-CSQ-TABLE-eee

where

eee = ES-NDML-NO

5.2 Generate file layout structure or local variables
for the final results.

5.2.1 If ES-FILE-NAME equals space, continue
at step 5.2.2.

5.2.1.1 Generate variable containing
file name in WS parcel. If
language is COBOL, generate:

01 CDM-RESULTS-NAME-eee PIC X(80).
01 CDM-RESULTS-REC-eee.

Else, generate:

CHARACTER*80 CDM-RESULTS-NAME-eee

where

eee = ES-NDML-NO

Continue at step 5.2.3.

5.2.2. Generate working storage for the
results first 01 level. If language is
COBOL, generate:

01 CDM-RESULTS-eee.

where

eee = ES-NDML-NO

5.2.3 Generate variables to hold results

Call function CDP10F with the following parameters:

LANG-NO
CS-ACTION-LIST
ES-ACTION-LIST
FDFILE-NAME
WORKFILE-NAME
FORTRAN-VARIABLE-TABLE
RET-STATUS

6. Generate code to move the results file name of last combination operation to results file name of current SELECT. If language is COBOL, generate:

MOVE CDM-CS-RESULTS-xxxx TO
CDM-CS-RESULTS-FILE-eee.

Else, generate:

CDM-CS-RESULTS-FILE-eee = CDM-CS-RESULTS-xxxx

where

eee = ES-NDML-NO
xxxx = NEXT-CS-QRY-ID

7. Generate status checking and record count determination from last combination operator and initialization of NDML-COUNT for retrieval loop. If language is COBOL, generate:

MOVE ZERO TO NDML-COUNT.

IF NOT OK
GO TO END-NDML-eee.

If ACTION-SYMBOL NOT = "U" generate:

IF CDM-CS-COUNT-xxxx = 0
GO TO END-NDML-eee.

Else, generate:

NDMLCT = 0
IF (NDMLST.NE. '00000') Go to 93eee

If ACTION-SYMBOL NOT = "U" generate:

IF (CDM-CS-COUNT-xxxx .EQ. '000000') Go to 93eee

where

eee = ES-NDML-NO
xxxx = NEXT-CS-QRY-ID

8. Generate code to call the C/E Transform Program for the first time.

- 8.1 Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 1
EE = ES-NDML-NO
MMMMM = program name from step 4.1
CC = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CECALL
Parameters
P1 = '1'
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-ee
P4 = CDM-RESULTS-ee
MMMMM = program name from step 4.1

where ee = ES-NDML-NO

- 8.2 If language is COBOL, generate:

IF NOT CDM-CE-EOF
ADD 1 TO NDML-COUNT

Else, generate:

IF (EOFFLA .NE. '1') NDMLCT = NDMLCT+1

- 8.3 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHKCE
Parameters
EE = ES-NDML-NO

9. Generate code to move the values in the result record to the named variables, structure or file.

- 9.1 If ES-FILE-NAME not equal space continue at step
9.2 else if COBOL generate:

LOOP-eee.

Else, generate:

94eee CONTINUE

where

eee = ES-NDML-NO

- 9.2 Call function "CDP10C" to generate the moves with the following parameters:

LANG-NO
PROCFE-NAME
ES-ACTION-LIST
CS-NDML-NO
FORTRAN-VARIABLE-TABLE
RET-STATUS

10. Generate code to close and delete the results file and terminate the loop structure for SELECT into variables or structure that did not have an NDML loop structure.

If ES-FILE-NAME not equal space continue at step 11. If ES-SEMI-CURLY-IND equal spaces generate code to call the C/E Transform Program to close and delete the results file. Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 3
EE = ES-NDML-NO
MMMM = program name from step 4.1
CC = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CECALL
Parameters
P1 = '3'
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-ee
P4 = CDM-RESULTS-ee
MMMM = program name from step 4.1

where ee = ES-NDML-NO

If ES-SEMI-CURLY-IND equal space generate closing loop structure.

Generate if COBOL:

END-NDML-eee.

Else, generate:

93eee CONTINUE

where

eee = ES-NDML-NO

Continue processing at step 12.

11. Generate code to save results to a users file.

11.1 Generate code to begin saving results into users file.

Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - FILSAV1
Parameters
EE = ES-NDML-NO
F1 = ES-FILE-NAME (variable or constant)

Library Name - VAXFORTRAN or IBMFORTRAN
Macro name - FILSAV1
Parameters
EE = ES-NDML-NO
F1 = 'ES-FILE-NAME'
P1 = CDM-RESULTS-NAME-ee
P2 = FCB-CDM-RESULTS-ee
P3 = CDM-RECORD-LENGTH-ee

11.2 Generate paragraph name for program loop of saving results to a file.

Generate if COBOL:

LOOP-eee.

where

eee = ES-NDML-NO

11.3 Generate code to save the null flag values for the retrieved data.

For each projected data item in the
ES-ACTION-LIST,
generate if COBOL:

MOVE FLAG-X(ii) TO ES-NULL-cc-nn.

where

ii = current index into the null flag array
cc = CS-NDML-NO
nn = ES-INDEX

Else, calculate the sum of the number of
non-deleted ES entries.

Generate:

CDM-RESULTS-REC-ee(1:rr) = FLAGAR(1:rr)

where ee = ES-NDML-NO
rr = REAL-CS-USED

11.4 Generate code to save the retrieved data:

11.4.1 If COBOL, generate:

MOVE CDM-RESULTS-ee to
CDM-RESULTS-RECORD-ee

where ee = ES-NDML-NO

11.4.2 If FORTRAN, perform steps 11.4.2.1 through 11.4.2.5 for each projected data item in the ES-ACTION-LIST. Initialize START-POS to 1 and START-POSF to REAL-ES-USED plus 1.

11.4.2.1 If ES-FCTN-NAME is "COUNT," perform steps 11.4.2.1.1 through 11.4.2.1.3.

11.4.2.1.1 Set END-POS equal
START-POS + 8.
Set END-POSF equal
START-POSF + 8.

11.4.2.1.2 Generate:

Call CONDIG
(CDM-RESULTS-ee(ep:ep),
SIGN,DIGIT,NDMLST)
CDM-RESULTS-REC-ee(sf:ef) =
CDM-RESULTS-ee(sp:ep)
where ee = ES-NDML-NO
ep = END-POS
sp = START-POS
ef = END-POSF
sf = START-POSF

11.4.2.1.3 Set START-POS equal END-POS
plus 1.
Set START-POSF equal
END-POSF plus 1.
Continue at step 4.2.12.2.

11.4.2.2 If ES-FCTN-NAME is equal to "MEAN," or "AVG," or "SUM," perform steps 11.4.2.2.1 through 11.4.2.2.3.

11.4.2.2.1 Set START-POS equal
START-POS plus 8.
Set END-POSF = START-POSF
plus 18.

11.4.2.2.2 Generate:

DECIML = 9
CALL RELFTN(DECIML,ES-RES-cc-ii,
LONG-ES-RES-cc-ii,

DS 620341200
30 September 1990
CDM-RESULTS-REC-ee
(sf:ef))
where cc = CS-NDML-NO
ii = ES-INDEX
sf = START-POSF
ef = END-POSF
ee = ES-NDML-NO

- 11.4.2.2.3 Set START-POSF equal
END-POSF plus 1.
Continue at step 11.4.2.
- 11.4.2.3 If ES-TYPE equals "I", perform
steps 11.4.2.3.1 through
11.4.2.3.3.
- 11.4.2.3.1 Set END-POSF equal
START-POSF plus 9.
- 11.4.2.3.2 Generate:

DIGIT = ES-RES-cc-ii
CALL
INTFTN(DIGIT,CDM-RESULTS-RE
C- (sf:ef))
where cc = CS-NDML-NO
ii = ES-INDEX
ee = ES-NDML-NO
sf = START-POSF
ef = END-POSF
- 11.4.2.3.3 Set START-POSF equal
END-POSF plus 1.
If ES-SIZE is greater than
4
Set START-POS equal
START-POS plus 4.
Else
Set START-POS equal
START-POS plus 2.
Continue at step 11.4.2.
- 11.4.2.4 If ES-TYPE equals "F", same
processing as step 11.4.2.2.
- 11.4.2.5 If ES-TYPE equals "C", perform
steps 11.4.2.5.1 through
11.4.2.5.3.
- 11.4.2.5.1 Set END-POSF equal
START-POSF plus ES-SIZE
minus 1.
Set END-POS equal START-POS
plus ES-SIZE minus 1.
- 11.4.2.5.2 Generate:

DS 620341200
30 September 1990
CDM-RESULTS-REC-ee(sf:ef) =
CDM-RESULTS-ee(sp:ep)
where ee = ES-NDML-NO
sf = START-POSF
ef = END-POSF
sp = START-POS
ep = END-POS

11.4.2.5.3 Set START-POS equal END-POS
plus 1.
Set START-POSF equal
END-POSF plus 1.
Continue at step 11.4.2.

11.5 Generate code to write the results to the user
specified file.

Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - UAPWR
Parameters
EE = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - UAPWR
Parameters
EE = ES-NDML-NO
P1 = FCB-CDM-RESULTS-ee
P2 = CDM-RESULTS-REC-ee
P3 = CDM-RECORD-LENGTH-ee

where ee = ES-NDML-NO

11.6 Generate code to call the C/E Transform Program
for the 2-N time.

11.6.1 Call "CDMACR" utility with the
following:

Library Name - COBOL
Macro Name - CECALL
Parameters
P1 = 2
EE = ES-NDML-NO
MMMMM = program name from step 4.1
CC = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - CECALL
Parameters
P1 = '2'
P2 = CDM-CS-RESULTS-FILE-ee
P3 = CDM-CSQ-TABLE-ee
P4 = CDM-RESULTS-ee
MMMMM = program name from step 4.1

where ee = ES-NDML-NO

11.6.2 If language is COBOL, generate:

IF NOT CDM-CE-EOF
ADD 1 TO NDML-COUNT

Else, generate:

IF (EOFFLA .NE. '1') NDMLCT = NDMLCT + 1

11.6.3 Call "CDMACR" utility with the following:

Library Name - COBOL or VAXFORTRAN or
IBMFORTRAN
Macro Name - ERRCHK
Parameters
EE = ES-NDML-NO

11.7 Generate code to move the values in the result record to the named file.

Call function "CDP10C" to generate the moves with the following parameters:

LANG-NO
PROCFILE-NAME
ES-ACTION-LIST
CS-NDML-NO
FORTRAN-VARIABLE-TABLE
RET-STATUS

11.8 Generate code for completion of the loop for saving results into a file.

Call "CDMACR" utility with the following:

Library Name - COBOL
Macro Name - FILSAV2
Parameters
EE = ES-NDML-NO

Library Name - VAXFORTRAN or IBMFORTRAN
Macro Name - FILSAV2
Parameters
EE = ES-NDML-NO
P1 = FCB-CDM-RESULTS-ee

where ee = ES-NDML-NO

12. Terminate processing.

37.5 Outputs

1. Error Status of the function

RET-STATUS

30 September 1990

2. Code generated into the parcels of the Application
Program

LIBRARY NAME - COBOL

MACRO NAME - FILSAV1

PARAMETERS - EE
 F1

```
*  
*   BEGIN SAVING RESULTS INTO USERS FILE  
*  
  IF CDM-CE-EOF  
    GO TO END-NDML-EE.  
  MOVE F1 TO CDM-RESULTS-NAME-EE.  
  MOVE "W" TO DISPOSITION.  
  CALL "OPNFIL" USING FCB-CDM-RESULTS-EE,  
                     RET-STATUS,  
                     CDM-RESULTS-NAME-EE,  
                     DISPOSITION,  
                     CDM-RECORD-LENGTH-EE,  
                     NUMBER-OF-RECORDS.  
  
  IF RET-STATUS NOT = KES-FILE-OK  
    MOVE "ERROR OPENING FILE CDM-RESULTS-NAME-EE"  
      TO MSG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO EXIT-PROGRAM.
```

LIBRARY NAME - COBOL

MACRO NAME - CECALL

PARAMETERS - P1
 EE
 MMMM

```
*  
* CALL CS-ES-TRANSFORM  
*  
  MOVE P1 TO CDM-CE-FLAG  
  CALL "MMMM" USING  
        CDM-CE-FLAG  
        CDM-CS-RESULTS-FILE-EE  
        CDM-CSQ-TABLE-CC  
        CDM-FLAG-ARRAY  
        CDM-RESULTS-EE  
        CDM-CE-EOF-FLAG  
        NDML-STATUS
```

LIBRARY NAME - COBOL

MACRO NAME - UAPWR

PARAMETERS - EE

```
CALL "OUTFIL" USING
    FCB-CDM-RESULTS-EE,
    RET-STATUS,
    CDM-RESULTS-REC-EE,
    CDM-RECORD-LENGTH-EE.
IF RET-STATUS NOT = KES-FILE-OK
    STRING "CDM-RESULTS-REC WRITE-ERROR: " RET-STATUS
        DELIMITED BY SIZE INTO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO
    END-NDML-EE.
```

LIBRARY NAME - FORTRAN

MACRO NAME - FILSAV1

PARAMETERS - EE

F1
P1
P2
P3

IF (EOFFLA .EQ. '1') GO TO 93EE

FILEST = 'W'

P1 = F1

CALL OPNFIL (%REF(P2), %REF(NDMLST), %REF(P1)

* , %REF(FILEST), %REF(P3), %REF(NUMREC))

IF (NDMLST .NE. '00000') GO TO 93EE

94EE CONTINUE

LIBRARY NAME - FORTRAN

MACRO NAME - CECALL

PARAMETERS - P1
 P2
 P3
 P4
 MMMMM

```
CEFLAG = P1
CALL MMMMM( %REF(CEFLAG), %REF(P2), %REF(P3)
*           , %REF(FLAGAR), %REF(P4), %REF(EOFFLA)
*           , %REF(NDMLST))
```

LIBRARY NAME - FORTRAN

MACRO NAME - UAPWR

PARAMETERS - EE

P1

P2

P3

```
      CALL OUTFIL ( %REF(P1), %REF(NDMLST)
*           , %REF(P2), %REF(P3))
      IF (NDMLST .NE. '00000') GO TO 93EE
```

SECTION 38

FUNCTION PRE11 - BUILD SOURCE CODE

The function of the source code builder is to combine previously constructed parcels into a modified application process capable of servicing NDML requests and compilable by the appropriate host-compiler.

38.1 Inputs

1. Identification Parcel
2. File Parcel
3. Working-Storage Parcel
4. Procedure Parcel

38.2 Processing

Concatenate parcels 1, 2, 3 and 4. Then, return to PRE12 to continue precompiling the source programs for the user AP.

38.3 Output

1. Expanded Source Code - to be input to the appropriate host-language compiler.

SECTION 39

FUNCTION PRE12 - CONTROL PRECOMPILATIONS (MAIN ROUTINE)

This function:

1. Obtains input from the user about the set of input programs to be precompiled.
2. Verifies that the logical unit of work being precompiled exists. Verifies that this logical unit of work and the software module being precompiled is not being precompiled by another user at the same time.
3. Handles error checking and commit/rollback of any changes made to the CDM software module cross references.
4. Store new cross references of generated code.
5. Deletes references and source files of all generated code made obsolete by re-precompilation.

39.1 Inputs

- | | |
|-----------------------|--|
| 1. LUW-NAME | - this identifies the logical unit of work or transaction being precompiled. |
| 2. AP-TARGET-HOST | - this identifies the host of the IISS where the application code will be executed. |
| 3. AP-FILE-IN | - this identifies the file on which the input to the precompiler is found. |
| 4. ERROR-FILE | - this contains the name of the file which will contain the error messages encountered during execution. |
| 5. CDM-USER-NAME | - this identifies the ORACLE user name and password as input by user. |
| 6. FILE-DELETE-OPTION | - this contains the user's input to delete ("FD=N") or not to delete ("FD=Y") obsolete references to user's NDML module. |

39.2 CDM Requirements

The entity classes needed are:

SOFTWARE MODULE	=	SM	(E57)
CDMP_GENERATED_MOD	=	CGM	(E293)
NDML_MODULE	=	NM	(E292)
LOG_UNIT_WORK	=	LUW	(E291)
RECORD_SET_USAGE	=	RSU	(E299)
DATA_FIELD_USAGE	=	DFU	(E300)

39.3 Internal Data Requirements

The following table is used to maintain a list of files containing obsolete source code due to re-precompilation. The files cannot be deleted at the same time as their CDM references due to the effects of ROLLBACK. They can only be deleted when and if COMMIT is done.

```
1. 01 FILE-DELETE-LIST.
    02 FDL-MAX          PIC 99.    VALUE 50.
    02 FDL-USED         PIC 99.
    02 FDL-ENTRY        OCCURS 50 TIMES.
        04 FDL-FILE-NAME PIC X(30).
        04 FDL-HOST-ID   PIC XXX.
        04 FDL-MODULE-NAME PIC X(10).
```

39.4 Processing

1. Read the input directive containing logical unit of work, target host, input file name, error file name, CDM user name and file delete option. Logon to the database, using CDM-USER-NAME. If logon is not successful, write error message and exit program.
2. Open the file as input by user in AP-FILE-IN. If open is not successful, write error message and exit program.
3. Verify the presence of the user input logical unit of work (LUW-NAME) using LUW(E291). Call routine CDVERLW to perform this operation.
 - 3.1 Call routine CDLKLWU to look for the logical unit of work and if found to lock the row to prevent updating or precompiling with the same LUW. Retrieve the LAST-CASE-NO attribute.
 - 3.2 If the logical unit of work was not found, call routine CDINSLW to insert a new occurrence of LUW(E291) with LAST CASE-NO = 0. Perform step 2.1 to obtain a lock on this LUW.
4. Execute Function PRE1 by calling routine CDPRE1. This identifies the name of the user module to be precompiled and signals the end of input conditions. In addition, CDPRE1 partitions the input application program into four parcels which will be added to by other precompiler

30 September 1990

components. On end of input, proceed to step 8. (Note: Only one logical unit of work can be specified per use of NDML precompiler).

5. Using the name of the software module being precompiled, call routine CDVERSM.

- 5.1 If a logical unit of work is not locked due to a commit or rollback after precompiling a previous module in the same batch input, execute step 2.1 to obtain the lock.
- 5.2 Select from NDML MODULE entity NM(E292) the name of the user's software module. If found, retrieve the LUW-NAME attribute. This means this module has been precompiled previously.
 - 5.2.1 Compare the LUW of (E292) to the one entered by the user in step 1. If they do not match, return an error code and terminate precompilation of this software module.
 - 5.2.2 If the module names do match, drop all existing cross references to this software module. (NOTE: in the event of a failed precompilation, these will all be restored by the ROLLBACK action.)
 - 5.2.2.1 Call routine CDRPXREF. Search the CGM(E293) for all modules generated by searching on the attribute USER-MOD-ID = the value of USER-MOD-ID returned by Function PRE1. For each row found, delete all DFU(E300), RSU(E299) and SM(E57) entries for the GENERATED-MOD-ID returned in the search of CGM(E293). Retrieve and store the module name, file name and host identifier from the CGM(E293) search in the FILE-DELETE-LIST table. These files will be deleted later upon successful precompilation.
 - 5.2.2.2 Call routine CDRPESU to delete all external schema references from ES_USAGE(E282) for the MOD_ID returned in the search of CGM(E293).
 - 5.2.2.3 Call routine CDDGAP to delete all rows from CGM(E293) for the user module being re-precompiled.

30 September 1990

- 5.3 If the module name was not found, insert a new occurrence of both SM(E57) and NM(E292). The values for SM(E57) are:

MOD_ID = the USER-MOD-ID from Function PRE1
 LANG_NAME = the SOURCE-LANGUAGE returned from Function PRE1
 LATEST_REV_DATE = the date when the software module was last revised or first implemented if never revised
 STATUS_IND = "N" to indicate an NDML user module

The values for NM(E292) are:

MOD_ID = the USER-MOD-ID
 LUW_NAME = logical unit of work from user input
 PRECOMP_DATE = latest date the NDML module was successfully precompiled.
 LAST_COMP_STAT = value "N"

Commit both inserts. This row will be saved even if the module precompiles with errors.

- 5.4 Because a commit was done in 5.3, the lock for LUW(E291) must be re-obtained. Perform step 3.1.

6. Perform precompilation of a single user module by executing Function PRE2.
 7. If precompilation was successful, combine all parcels of code generated to become a modified user module by executing Function PRE11.
 8. Perform error checking for precompilation of this user module by calling routine CDECHK.
- 8.1 If the module was precompiled successfully, for each new entry added to the CG table for the user routine, insert generated AP references.

- 8.1.1 If MOD_TYPE is not equal to "USER-MOD" insert a new row into the generated module CDM table (CGM) as follows:

GENERATED_MOD_ID	= CGT-MOD-NAME
USER_MOD_ID	= USER-MOD-ID from Function PRE1
GENERATED_BY	= CGT-GENED-BY
GEN_DATE	= today's system date
MODULE_TYPE	= CGT-MOD-TYPE
CASE_NO	= CGT-CASE-NO
IS_ACTION	= type of request processor action
FILE_NAME	= CGT-GEN-FILE-NAME
HOST_ID	= CGT-TARGET-HOST
DB_ID	= CGT-DBID
LUW_NAME	= the current LUW

30 September 1990

being precompiled

LOCAL_REMOTE

= CGT-LOCALITY

SUBTRANS_ID

= subtransaction identifier

- 8.1.2 Insert a new row into the software module table (SM) as follows:

MOD_ID

= CGT-MOD-NAME

LANG_NAME

= CGT-LANGUAGE

LATEST_REV_DATE

= system date

STATUS_IND

= "G" for generated

- 8.1.3 If MOD-TYPE is equal to "USER-MOD", store a reference to the generated, modified user module in the the generated module CDM table (CGM), the software module table (SM) and update the last case number on the previously locked LUW (E291) row.

- 8.1.4 Update NDML_MODULE (NM) to indicate a successful precompilation as follows:

LAST_COMP_STAT

= status code indicating success

PRECOMP_DATE

= system date

- 8.1.5 Delete any obsolete code files at the user's option. If file delete option is requested, delete each file found on the FILE-DELETE-LIST. Note that these are not necessarily on the same computer as the precompiler. Save the names of obsolete object code.

Open the file OBSOBJ and write the name of each module and its host of residence to the file from the FILE-DELETE-LIST. This will be used to periodically clear object libraries of obsolete code.

- 8.1.6 Increment the good precompile counter.

- 8.2 If the module was not precompiled successfully,

- 8.2.1 Increment the bad precompile counter.

- 8.2.2 Delete the files containing the generated code which were added to the CODE_GENERATOR_TABLE for this bad precompilation.

- 8.2.3 Delete the four parcels created during the bad precompile.

- 8.2.4 Rollback all changes made to the CDM.

30 September 1990

8.2.5 Update NDML MODULE (NM) to indicate an unsuccessful precompilation and commit this change to the database.

8.3 Commit all database changes and logoff from the CDM's DBMS, if end of file was encountered. If not end of file, return to step 4 to precompile the next module found on the user's input file.

9. When the end of user's input file is encountered, display a message showing number of good and bad precompiles, followed by a report of all modules successfully precompiled. If the user did not request the file delete option, a list of obsolete code is displayed.

39.5 Outputs

1. CODE-GENERATOR-TABLE - This table will maintain a single row for each file of generated code. Entries refer to the various types of generated software.

- CS-ES Transform
- CS Selector
- Request Processor Main
- Modified User Module

This information must be saved in the CDM to track all generated code.

2. OBSOBJ - This is a sequential file designed to hold the host, module, and file names for each module of generated code made obsolete due to a deleted user module containing NDML or a re-precompilation. This is designed to allow a JCL utility (not designed at this time) to be periodically executed to read the file OBSOBJ.DAT and for each entry, delete the object code from the library of generated code at each node of IISS. This is designed as an interim solution. Eventually, RCL service should use the information to dynamically remove objects from the libraries.

The file will consist of a single record description.

01	OBSOLETE-OBJECT.	
03	OBJ-HOST-ID	PIC XX.
03	FILLER	PIC XXX.
03	OBJ-FILE-NAME	PIC X(30).
03	FILLER	PIC XXX.
03	OBJ-MODULE-NAME	PIC X(10).
03	FILLER	PIC XXX.
03	OBJ-TIME-STAMP	PIC X(22).

3. GOOD-PRECOMPILES, BAD-PRECOMPILES - Counters to record the number of successful and unsuccessful routines precompiled.

DS 620341200
30 September 1990

4. Report Results - A listing of all successfully precompiled modules, displaying the module name, language, target host, DBMS, database, locality (remote or local), module type and file name.

SECTION 40

FUNCTION PRE13 - CONTROL CODE GENERATION

This function controls the generation of source code for a single request in conceptual schema terms. It also controls the selection of access paths for databases that require them and assigns unique names for RPs and CS-ES transformers. It determines the name of the AP that a request processor subroutine will be called by.

40.1 Inputs

1. The following tables and lists are simply passed on to other modules:

ES-ACTION-LIST	from PRE4 to PRE8, PRE10
ES-QUALIFY-LIST	from PRE4 to PRE10
CS-ACTION-LIST	from PRE5 to PRE8, PRE10
CS-QUALIFY-LIST	from PRE5 to PRE9, PRE10
IS-ACTION-LIST	from PRE5 to PRE6, PRE9
IS-QUALIFY-LIST	from PRE5 to PRE6, PRE9, PRE10
JQG	from PRE5 to PRE10
RFT	from PRE5 to PRE9, PRE10
SET-TABLE	from PRE5 to PRE6, PRE9
OCCURS-TABLE	from PRE5 to PRE6
COMPLEX-MAPPING- ALG-TABLE	from PRE5 to PRE6
ACCESS-PATH (several tables)	from PRE6 to PRE7
ERRFILE	
UV-ABBR-LIST	
JQG-ATTRIBUTE-PAIR-LIST	
BOOLEAN-LIST	
SUBTRANS-PROCESS-ID-TABLE	
SUBTRANS-BOOLEAN-LIST	
BLOCK-STACK	
FIRST-INNER-SELECT	

2. MY-HOST
TARGET-HOST
PARCL1
PARCL2
PARCL3
PARCL4
LUW-NAME
SOURCE-LANGUAGE
IOSECTION-INDICATOR
3. CODE-GENERATOR-TABLE, which is received from PRE12 and in which PRE13 records information about generated RPs and CS-ES transformers.
4. Logical Unit Of Work being precompiled.

5. CDM Meta Data

The entity classes needed are:

CDMP_GENERATED_MOD = CGM (E293)

40.2 Processing

1. Determine the name of the RP driver. Given the database id (DBID) that the request processor subroutine is to access and the logical unit of work (LUW-NAME) currently being precompiled:

- 1.1 Search for an entry in the CODE-GENERATOR-TABLE where:

CGT-DBID = DBID and

CGT-MOD-TYPE = "RP-MAIN"

If an entry is found, return the CGT-MOD-NAME and CGT-LOCALITY as output of this sub-function.

- 1.2 If an entry in CGT is not found, search the CDM for the locality and MOD-ID given the logical unit of work and database ID of the subtransaction.

Search CDMP_GENERATED_MOD (E293) where:

LUW NAME = LUW being precompiled
DBID = database ID of the
subtransaction
MODULE_TYPE = 'RP-MAIN'

If an entry was found, return LOCALITY and MOD-ID as output of this sub-function.

- 1.3 If an RP driver name was not located in step 1.1, execute the module name generator function APNAME to derive a new subroutine module name.

- 1.3.1 Determine if this RP-MAIN will be remote (accessed by the NTM) or local (accessed by a direct call from the DRS). Search the CDM entity class CDMP_GENERATED_MOD (E293) where:

LUW NAME = LUW being precompiled
LOCAL_REMOTE = 'L'
MODULE_TYPE = 'RP-MAIN'

- 1.3.1.1 If such a row is found, store a value of "R" in the variable LOCALITY; since there are other LOCAL RP's, this one must be remote.

- 1.3.1.2 If a row was not found and the

30 September 1990

TARGET-HOST is equal to the RP-SUB-HOST, search the entire CG Table for an RP-MAIN entry with a CGT-LOCALITY = "L". This RP can only be local if there are no locals in the CGT and it is to run on the same host as the target host of the user's AP. If one is not found, set LOCALITY = "L", else set LOCALITY = "R".

- 1.3.1.3 If a row was not found in step 1.3.1 and TARGET-HOST is not equal to RP-SUB-HOST, move "R" to LOCALITY.

- 1.4 Convert this name to an NTM application name by concatenating, in order, the NTM-DIRECTORY, the MOD-NAME and three trailing Z's if LOCALITY = "R". Note the trailing Z's are an NTM workaround. If LOCALITY, "L" use only the MOD-NAME. Create a new entry in the CGT for this RP-MAIN entry:

CGT-MOD-NAME	= the concatenated name
CGT-LANGUAGE	= "COBOL"
CGT-TARGET-HOST	= the host-id of the RP-SUB currently being generated
CGT-DBMS	= the DBMS of the RP-SUB currently being generated
CGT-DB-NAME	= the name of the database for which the RP-SUB is currently being generated
CGT-MOD-TYPE	= "RP-MAIN"
CGT-GENED-BY	= "CDP14"
CGT-DBID	= the database id of the RP-SUB currently being generated
CGT-CURRENT-HOST	= "VAX"
CGT-RCL-STATUS	= "COMP"

- 1.5 Get a file name on which the RP-MAIN will be generated at a later time.
2. Next, select a unique name for the RP subroutine to be generated for the subtransaction by executing the function APNAME.
3. Generate the COBOL code required to execute NDML subtransactions.
- 3.1 Invoke the appropriate version of PRE9 to generate an RP-SUB for the database to be accessed, passing it the selected RP name:

PRE9.2 for SQL databases
PRE9.3 for CODASYL databases
PRE9.4 for TOTAL databases
PRE9.5 for IMS databases

If the subtransaction is for a CODASYL or TOTAL database, invoke PRE6, passing it the IS-ACTION-LIST and the IS-QUALIFY-LIST, to select an access path through the database. When PRE6 is finished, invoke PRE7 to transform the access path into generic DML statements.

Save information about each routine generated in the CODE-GENERATOR-TABLE.

CGT-DBID	= database ID the RP-SUB will access
CGT-DBMS	= ORACLE, DB2, IDS-II, IDMS, VAX-11 or TOTAL
CGT-LIBRARY-NAME	= library where macros used reside
CGT-DB-NAME	= database name RP-SUB will access
CGT-TARGET-HOST	= HOST-ID
CGT-MOD-NAME	= selected RP name
CGT-GEN-FILE-NAME	= GEN-FILE-NAME
CGT-MOD-TYPE	= "RP-SUB" or "USER-MOD"
CGT-GENED-BY	= subroutine which generated the code
CGT-ACTION	= IS-ACTION
CGT-SUBTRANS-ID	= SUB-SCRIPT
CGT-CASE-NO	= CS-NDML-NO
CGT-SCHEMA	= DB-SCHEMA
CGT-SUBSCHEMA	= DB-SUBSCHEMA
CGT-DB-LOCATION	= DB-LOCATION
CGT-PASSWORD	= DB-PASSWORD
CGT-CURRENT-HOST	= name of host computer
CGT-RCL-STATUS	= "GEN"
CGT-LANGUAGE	= "COBOL" or SOURCE-LANGUAGE if MOD-TYPE = "USER-MOD"

3.2 When the PRE9 version is finished, repeat Steps 1 through 3.1 for the next subtransaction, if any.

4. Control the code generation for NDML conceptual requests.

If ES-ACTION is:

B	(BEGIN)	or
C	(COMMIT)	or
R	(ROLLBACK)	or
N	(NEXT)	or
E	(END CURLEY)	or
X	(BREAK)	or

M (MODIFY) or
D (DELETE) or
I (INSERT)

invoke PRE10 to generate code into the source program.
When PRE10 is finished, return to PRE5.

5. If ES-ACTION is:

S (SELECT) or
Q (QUERY COMBINATION) or
1 (TYPE 1 REFERENTIAL INTEGRITY) or
2 (TYPE 2 REFERENTIAL INTEGRITY) or
K (KEY UNIQUENESS)

5.1 Call routine APNAME to get a unique name for the conceptual to external transformer.

This routine maintains a buffer (Module Name Table) of 20 module names and passes out one on each call. If the buffer is empty, it sends an NTM message to the Module Name Q-server to acquire 20 new unique names. For a description of this Q-server, see the File Utilities DS, DS#620241330.

5.1.1 Upon request, increment the last used index of the module name table.

5.1.2 If the index exceeds the size of the table:

Issue a message to module name Q-server.

Wait on a reply from the Q-server.

On a successful message, store the data of the message in the module name table and set the last used index of the table to zero. Return to step 5.1.1.

5.1.3 If the index does not exceed the maximum, return the entry in the module name table pointed to by the index.
(Note, this routine is not re-entrant. The index cannot be reset on each invocation).

5.2 Perform the following depending on the contents of ES-ACTION.

5.2.1 If ES-ACTION = 'S' and is not part of an NDML query combination command, invoke PRE8 to generate a CS-ES transformer, passing it the selected CS-ES transformer name. Record it in the CODE-GENERATOR-TABLE as follows:

DS 620341200
30 September 1990

CGT-DBID	= 0
CGT-DBMS	= spaces
CGT-LIBRARY-NAME	= name of the macro library from which code is generated
CGT-DB-NAME	= spaces
CGT-TARGET-HOST	= name of the host computer on which the user AP will be run
CGT-MOD-NAME	= selected name for the CS-ES transformer
CGT-GEN-FILE-NAME	= GEN-FILE-NAME
CGT-MOD-TYPE	= "CS-ES"
CGT-GENED-BY	= "CDPRE8"
CGT-SUBTRANS-ID	= 0
CGT-CASE-NO	= 0
CGT-ACTION	= spaces
CGT-SCHEMA	= spaces
CGT-SUBSCHEMA	= spaces
CGT-DB-LOCATION	= spaces
CGT-DB-PASSWORD	= spaces
CGT-CURRENT-HOST	= name of the host computer on which the source program is being precompiled
CGT-RCL-STATUS	= "GEN"
CGT-LANGUAGE	= "COBOL"

5.2.2 If ES-ACTION = "Q" indicating an inner select, we need only transform the results to CS terms. Invoke CDPRE8C to generate a CS-CS transformer for an inner SELECT of an NDML query combination command. Record it in the CODE-GENERATOR-TABLE as follows:

CGT-DBID	= 0
CGT-DBMS	= spaces
CGT-LIBRARY-NAME	= name of the macro library from which code is generated
CGT-DB-NAME	= spaces
CGT-TARGET-HOST	= name of the host computer on which the user AP will be run
CGT-MOD-NAME	= selected name for the CS-ES transformer
CGT-GEN-FILE-NAME	= GEN-FILE-NAME
CGT-MOD-TYPE	= "CS-CS"
CGT-GENED-BY	= "CDPRE8C"
CGT-SUBTRANS-ID	= 0
CGT-CASE-NO	= 0
CGT-ACTION	= spaces
CGT-SCHEMA	= spaces

CGT-SUBSCHEMA = spaces
CGT-DB-LOCATION = spaces
CGT-DB-PASSWORD = spaces
CGT-CURRENT-HOST = name of the host computer on
which the source program is
being precompiled
CGT-RCL-STATUS = "GEN"
CGT-LANGUAGE = "COBOL"

- 5.2.3 If ES-ACTION = "2" or "1" or "K", invoke CDPRE8D to generate a CS-CS transformer for a type1 or type2 referential integrity test or a key uniqueness test, passing it the selected CS-CS transformer name. Record it in the CODE-GENERATOR-TABLE as follows:

CGT-DBID = 0
CGT-DBMS = spaces
CGT-LIBRARY-NAME = name of the macro library from
which code is generated
CGT-DB-NAME = spaces
CGT-TARGET-HOST = HOST-ID
CGT-MOD-NAME = selected name for the CS-ES
transformer
CGT-GEN-FILE-NAME = GEN-FILE-NAME
CGT-MOD-TYPE = "CS-CS"
CGT-GENED-BY = "CDPRE8D"
CGT-SUBTRANS-ID = 0
CGT-CASE-NO = 0
CGT-ACTION = spaces
CGT-SCHEMA = spaces
CGT-SUBSCHEMA = spaces
CGT-DB-LOCATION = spaces
CGT-DB-PASSWORD = spaces
CGT-CURRENT-HOST = name of the host computer on
which the source program is
being precompiled
CGT-RCL-STATUS = "GEN"
CGT-LANGUAGE = "COBOL"

- 5.3 Invoke PRE10 to generate code into the source program. When PRE10 is finished, return to PRE5.

40.3 Outputs

1. CODE-GENERATOR-TABLE, which is received from PRE12 and in which PRE13 records information about generated RPs and CS-ES transformers.
2. RET-STATUS - a status code indicating whether function was executed successfully.

SECTION 41

FUNCTION PRE14 - REQUEST PROCESSOR DRIVER GENERATOR

This function is a stand alone program used after precompile time, but before application usage or runtime. It would be executed at the same time a link is done in the normal programming scenario of edit, compile, link and run. Because the NDML Precompiler only generates request processor subroutines and many precompiles can be done separately and independently, a separate "link" step to generate a Request Processor Driver (RPD) is needed. No one usage of the precompiler can generate an RPD since information on all RP subroutines generated for a logical unit of work is necessary. (The separate precompilation feature allows different modules of the same logical unit of work to be precompiled at separate times, without re-precompiling other, unaffected modules). The CDM will act as the library or directory of all generated code. With this information, PRE14 can generate RPDs with the correct names when the user indicates precompilation of all modules in a logical unit of work is complete.

41.1 Inputs

1. Logical Unit of Work - this identifies the logical unit of work, or transaction for which RPDs must be generated. This is a user input.
2. CDM user name - this identifies the user name needed to access the database. This is user input.

41.2 CDM Requirements

The entity classes needed are:

SOFTWARE_MODULE	=	SWM	(E57)
LOG UNIT WORK	=	LUW	(E291)
NDML_MODULE	=	NM	(E292)
RP_MAIN	=	RPM	(E298)
CDMP_GENERATED_MOD	=	CGM	(E293)
RP_SUB_ROUTINE	=	RSM	(E295)
DATA_BASE	=	DB	(E24)
DB_PASSWORD	=	DBP	(E25)
SCHEMA-NAMES	=	SS	(E14)

41.3 Processing

1. Perform user interface to the function.
 - 1.1 Initiate processing by connecting with the NTM using INITIAL or INITEX service, depending on the choice of option in step 1.2 below. All options but 1.2.1 will require INITEX.
 - 1.2 Upon initiation, obtain the name of a logical unit of work from the user. Note this specification does not detail how this is to be

done. Options are:

- 1.2.1 Use of User Interface Subsystem forms.
- 1.2.2 Simple Sequential file, allowing batch usage.
- 1.2.3 COBOL DISPLAY and ACCEPT for simple prompted input.
- 1.2.4 Operating System dependent input parameter such as the UNIX "args" concept.

2. Perform generation of each Request Processor Driver.

- 2.1 Logon to the ORACLE CDM data base using the input CDM user name.
- 2.2 Verify that the logical unit of work input parameter exists in the CDM (LUW (E291)) and lock this occurrence. This is the same processing as step 3.1 of PRE12, Control Precompilation.
 - 2.2.1 If found, lock the row, preventing other user's from updating or precompiling with the same LUW.
 - 2.2.2 If the LUW is not found, set the proper error return status, terminate this function and display error message.
- 2.3 Verify that all NDML modules have been precompiled successfully for this logical unit of work by searching CDM NDML_MODULE entity for LAST_COMP_STAT not equal to "5". If any bad precompiles are found, set the proper error return status, terminate this function and display error message.
- 2.4 Search CDM entity CGM (E293) for each occurrence matching the logical unit of work input parameter and MODULE_TYPE = 'RP-MAIN'. Retrieve the MOD-ID, DB-ID, RP-MAIN-FILE and LOCAL-REMOTE attributes. For each row found:
 - 2.4.1 Determine parameters needed for the generation of the RPD data base logon, the schema section, and Data Division of the RPD. Using the DB-ID attribute from the search of step 2.4, retrieve the following attributes from the named entities using DB_ID as the search value:

DBMS-NAME	from	E24
HOST_ID	from	E24
DB_NAME	from	E24
LIBRARY_NAME	from	E14

30 September 1990

SCHEMA NAME	from E14
SUBSCHEMA NAME	from E14
DB LOCATION	from E25
DB PASSWORD	from E24
CHARACTER NULL	from E24
INTEGER NULL	from E24
NTM_DIRECTORY	from E24

(Implementation note: an OUTER-JOIN can identify these with one search. Be aware that some attributes will be null for some DBMS types).

- 2.4.2 Delete the old version, if any, of the RP-MAIN file since the macro used to generate the RPDs writes to the end of a file (OPEN EXTEND) if one by that name already exists.
- 2.4.3 Call routine CDMACR to generate code into the RPD. Using the library name and module name, along with a substitution parameter list from Step 2.4.1, macros chosen from the CDM are written to the output file, the name of which is in RP-MAIN-FILE. These macros write code containing the IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION and the beginning of the PROCEDURE DIVISION to the RPD.

The parameters used are as follows:

RP-MAIN-FILE	from Step 2.4
LIBRARY-NAME	from E14
MACRO-NAME	RPSTART if LOCAL-REMOTE = "R" else RPGO
SUBSTITUTION-LIST	contains MOD-ID DB-NAME SCHEMA-NAME DB-PASSWORD DB-LOCATION SUBSCHEMA-NAME
RET-STATUS	

- 2.4.4 Generate a call to each RP subroutine into the RPD Procedure Division. Identify each RP-SUB participating in the logical unit of work for this data base. Select the CASE_NO, SUBTRANS_ID and GENERATED_MOD_ID from CDMF GENERATED_MOD (E293) for the MODULE_TYPE of 'RP-SUB', the DB ID from Step 2.4 and the user input logical unit of work.

- 2.4.5 If no rows are returned, an "obsolete" RP-MAID has been encountered. This occurs when all RP-subroutines for a particular database have been deleted by re-precompilation of an NDML request after a change in the CS-IS mapping. (The request no longer needed that particular database). In this case:
- 2.4.5.1 Delete the RP-MAIN reference from CDMPT_GENERATED_MOD (E293) for MODULE-TYPE of 'RP-MAIN' and the MOD-ID from step 2.4.
 - 2.4.5.2 Delete the RP-MAIN reference from SOFTWARE_MODULE (E57).
 - 2.4.5.3 Delete the RP-MAIN driver partially built in step 2.4.3.
 - 2.4.5.4 The RPD is now deleted. Clear the "obsolete" error code and continue processing at step 2.4.
- 2.4.6 If a row was returned from the Select in step 2.4.4, generate the termination Procedure Division code of the RPD. Call routine CDMACR for the appropriate macro using:

```
RP-MAIN-FILE      from step 2.4
LIBRARY-NAME      from E14
MACRO-NAME        RPEND if LOCAL-REMOTE
                                     ="R"
                                     else
                                     RPSTOP
SUBSTITUTION-LIST no parameters
RET-STATUS
```

- 2.5 Record a new entry in the CODE-GENERATOR table for the RPD just established.

```
CGT-MOD-NAME      = MOD-ID retrieved in step 2.4
CGT-LANGUAGE      = "COBOL"
CGT-TARGET-HOST   = HOST-ID of E24
CGT-DBMS          = DBMS-NAME of E24
CGT-DB-NAME       = DB-NAME of E24
CGT-MOD-TYPE      = "RP-MAIN"
CGT-GEN-FILE-NAME = RP-MAIN-FILE of E293
CGT-DBID          = DB-ID of E293
CGT-LIBRARY-NAME  = LIBRARY-NAME of E14
CGT-SUBTRANS-ID   = 0
CGT-CASE-NO       = 0
CGT-CURRENT-HOST  = Current host on which
                   PRE14 is executing.
                   (NTM can be used to determine
                   the host or host can be hard
                   wired into machine dependent
```

CGT-RCL-STATUS	versions)
CGT-LOCALITY	= "GEN"
	= LOCAL-REMOTE of E293

- 2.6 When all rows have been processed, terminate generation of RPDs. Commit changes made to database.
3. Display the results of RPD generation to the user. Depending on the user interface option chosen, this may be (corresponding choices):
 - 3.1 Use of User Interface output form
 - 3.2 Output to a formatted sequential file
 - 3.3 COBOL DISPLAY's for a listing on the terminal
 - 3.4 Operating system dependent, e.g. UNIX standard output

The columns of the CODE-GENERATOR-TABLE to be output should be:

CGT-MOD-NAME
CGT-LANGUAGE
CGT-TARGET-HOST
CGT-DBMS
CGT-DB-NAME
CGT-GEN-FILE-NAME
CGT-MOD-TYPE
CGT-LOCALITY

The user must be warned to remember the module name of the "local" RPD, which is a subroutine. This name will be needed when linking the user AP (on VAX) because of COBOL dynamic calls used by the DRS.

4. Terminate the NTM connection by using the "TRMNAT" service (or "TRMNDML" if the NDML is used).

41.4 Outputs

1. One RPD program will be generated for each RP-MAIN for the given logical unit of work. There is one RPMAIN in the CDM for each data base involved in the user's transaction or logical unit of work.
2. CODE-GENERATOR-TABLE

This table tracks all generated software and holds pertinent results about all code generated or modified by the precompiler.

* CGTABLE.INC

```
01 CODE-GENERATOR-TABLE
03 CGT-USED PIC 999 VALUE 0.
03 CGT-MAX PIC 999 VALUE 189.
03 CGT-ENTRY OCCURS 190 TIMES INDEXED BY CGT-INDEX
    05 CGT-MOD-NAME PIC X(10).
    05 CGT-LANGUAGE PIC X(8).
    05 CGT-TARGET-HOST PIC XXX.
    05 CGT-DBMS PIC X(30).
    05 CGT-DB-NAME PIC X(30).
    05 CGT-MOD-TYPE PIC X(10).
        88 USER-MODULE VALUE "USER-MOD".
        88 RP-MAIN VALUE "RP-MAIN".
        88 RP-SUB VALUE "RP-SUB".
        88 CS-ES VALUE "CS-ES".
    05 CGT-ACTION PIC X.
    05 CGT-GENED-BY PIC X(10).
    05 CGT-GEN-FILE-NAME PIC X(30).
    05 CGT-PASSWORD PIC X(30).
    05 CGT-LOCALITY PIC X.
        88 CGT-LOCAL VALUE "L".
        88 CGT-REMOTE VALUE "R".
*
* THE ABOVE CAN BE SHOWN TO THE USER, THE FOLLOWING ARE FOR
* RCL AND INTERNAL USAGE:
    05 CGT-DBID PIC 9(6).
    05 CGT-LIBRARY-NAME PIC X(30).
    05 CGT-SUBTRANS-ID PIC 9(6).
    05 CGT-CASE-NO PIC 9(6).
    05 CGT-SCHEMA PIC X(30).
    05 CGT-SUBSCHEMA PIC X(30).
    05 CGT-DB-LOCATION PIC X(30).
    05 CGT-PASSWORD PIC X(30).
* THE FOLLOWING ARE REQUIRED FOR RCL FUNCTIONS
    05 CGT-LOG-FILE-NAME PIC X(30).
    05 CGT-RCL-LOG-CHAIN PIC XXX.
    05 CGT-CURRENT-HOST PIC XXX.
* NOTE THAT CURRENT HOST MAY CHANGE DURING THE RCL PROCESS
    05 CGT-RCL-STATUS PIC X(5).
        88 CODE-GEN VALUE "GEN".
        88 CODE-XFERRED VALUE "XFER".
        88 CODE-COMPILED VALUE "COMP".
        88 AP-LINKED VALUE "LINK".
        88 AP-DEFINED VALUE "NTM".
```

LIBRARY: ORACLE
MACRO: RPSTART

```
IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(10) VALUE IS "P1".
01 MSG-DESC PIC X(60).
01 RP-SUB-NAME PIC X(6).
*   REPLY TO DRS
01 MESSAGE-BODY-OUT.
    03 OUTFILE-NAME PIC X(80).
    03 REC-COUNT PIC 9(6) VALUE ZERO.
    03 QP-STATUS PIC 9(5).
01 MSG-OUT-L PIC 9(5) COMP VALUE 91.
*   NTM STUFF
01 BUFFER PIC X(4096).
01 BUFFER-SIZE PIC 9(6) VALUE 4096.
01 DATA-TYPE PIC X.
01 NTM-DESTINATION PIC X(10).
01 LOGICAL-CHANNEL PIC X(3).
01 MESSAGE-TYPE PIC X(2).
01 OUT-MESSAGE-TYPE PIC XX VALUE "RR".
01 MESSAGE-SERIAL-NUMBER PIC X(7).
01 NTM-SOURCE PIC X(10).
01 TERMINATION-STATUS PIC X VALUE SPACE.
01 TIMEOUT-VALUE PIC X(15) VALUE ZEROES.
01 WAIT-FLAG PIC 9 VALUE 1.
01 DATA-LENGTH PIC 9(5) COMP.
COPY ERRCDM OF IISSCLIB.
COPY CHKCDM OF IISSCLIB.
COPY SRVRET OF IISSCLIB.
01 SHOW-RC PIC ----9.
*   MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
    03 CASE-NO PIC X(6).
    03 SUB-ID PIC XXX.
    03 MESSAGE-PARAMETERS.
        05 USER-PW PIC X(21).
        05 FILLER PIC X(1979).
*   WS FOR ORACLE
EXEC SQL BEGIN DECLARE SECTION END-EXEC.

01 OUNAM PIC X(30) VALUE "P2".
01 OUPWD PIC X(30) VALUE "P4".
01 USER-NAME PIC X(30).
EXEC SQL END DECLARE SECTION END-EXEC.
EXEC SQL INCLUDE SQLCA END-EXEC.
PROCEDURE DIVISION.
START-HERE.
CALL "INITAL" USING BUFFER,
                  BUFFER-SIZE,
                  SYSTEM-STATE,
                  RET-CODE.
```

```
IF INITIAL-SUCCESSFUL
  NEXT SENTENCE
ELSE
  MOVE "RP CANNOT START" TO MSG-DESC
  MOVE RET-CODE TO RET-STATUS
  PERFORM PROCESS-ERROR
  GO TO PGM-END.
WAIT-HERE.
  MOVE SPACES TO OUTFILE-NAME.
  MOVE ZEROES TO REC-COUNT.
  MOVE SPACES TO LOGICAL-CHANNEL, NTM-SOURCE, MESSAGE-TYPE.
  CALL "RCV" USING LOGICAL-CHANNEL,
                    WAIT-FLAG,
                    NTM-SOURCE,
                    MESSAGE-TYPE,
                    DATA-LENGTH,
                    MESSAGE-BODY-IN,
                    ACCEPT-STATUS,
                    MESSAGE-SERIAL-NUMBER.
IF RCV-NORMAL-MSG
  NEXT SENTENCE
ELSE
  MOVE "UNABLE TO RECEIVE DRS REQUEST" TO MSG-DESC
  MOVE ACCEPT-STATUS TO RET-STATUS
  PERFORM PROCESS-ERROR
  GO TO PGM-ABORT.
  MOVE ZEROES TO RET-STATUS.

TRY-OPEN.
  IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
*   ORACLE LOGON
*

  MOVE USER-PW TO USER-NAME.
  EXEC SQL CONNECT :USER-NAME END-EXEC.
  IF SQLCODE = 0 GO TO END-CASE-CHECK.
  EXEC SQL CONNECT :OUNAM
  IDENTIFIED BY :OUPWD
  END-EXEC.
  IF SQLCODE < 0
    MOVE KES-OLOGON-ERROR TO RET-STATUS
    MOVE SQLCODE TO SHOW-RC
    STRING "BAD ORACLE LOGON:" DELIMITED BY SIZE
    SHOW-RC DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT.
  GO TO END-CASE-CHECK.
TRY-CLOSE.
  IF CASE-NOT = "CLS" GO TO TRY-BEGIN.
  GO TO END-CASE-CHECK.
TRY-BEGIN.
  IF CASE-NO NOT = "BEG" GO TO TRY-COMMIT.
  GO TO END-CASE-CHECK.
TRY-COMMIT.
  IF CASE-NO NOT = "CMT"
    GO TO TRY-ROLLBACK.
  EXEC SQL
    COMMIT WORK
```

```
        END-EXEC.
IF SQLCODE < 0
    MOVE KES-NOCOMMIT TO RET-STATUS
    MOVE SQLCODE TO SHOW-RC
    STRING "UNABLE TO COMMIT"
                                DELIMITED BY SIZE
                                SHOW-RC
                                INTO MSG-DESC
                                GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY-ROLLBACK.
    IF CASE-NO NOT = "RBK"
        GO TO END-FIXED-CASES.
    EXEC SQL ROLLBACK WORK
                                END-EXEC.
    IF SQLCODE < 0
        MOVE KES-NOROLLBACK TO RET-STATUS
        MOVE SQLCODE TO SHOW-RC
        STRING "UNABLE TO ROLLBACK"
                                DELIMITED BY SIZE
                                SHOW-RC
                                INTO MSG-DESC
                                GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
END-FIXED-CASES.
    MOVE CASE-NO TO RP-SUB-NAME.
    CALL RP-SUB-NAME USING MESSAGE-BODY-IN
                        MESSAGE-BODY-OUT
ON EXCEPTION
    MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
    STRING " CANNOT CALL: "
        RP-SUB-NAME
        " CASE/SUB: "
        CASE-NO
        SUB-ID
                                DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT.
*
*
*
MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.
```

ORACLE REQUEST PROCESSOR MACROS

LIBRARY: ORACLE
MACRO: RPGO

IDENTIFICATION DIVISION.

PROGRAM-ID. P1.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

```
01 RET-STATUS          PIC X(5).
01 MODULE-NAME         PIC X(10) VALUE IS "P1".
01 MSGG-DESC          PIC X(60).
01 RP-SUB-NAME        PIC X(6).
01 SHOW-RC            PIC ----9.
01 MSGG-OUT-L         PIC 9(5) COMP VALUE 91.
    COPY ERRCDM OF IISSCLIB.
    EXEC SQL BEGIN DECLARE SECTION END-EXEC.
01 USER-NAME          PIC X(30).
01 OUNAM              PIC X(30) VALUE "P2".
01 OUPWD              PIC X(30) VALUE "P4".
    EXEC SQL END DECLARE SECTION END-EXEC.
    EXEC SQL INCLUDE SQLCA END-EXEC.
```

LINKAGE-SECTION.

```
*    REPLY TO DRS
01 MESSAGE-BODY-OUT.
    03 OUTFILE-NAME          PIC X(80).
    03 REC-COUNT             PIC 9(6).
    03 QP-STATUS             PIC X(5).
*    MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
    03 CASE-NO              PIC X(6).
    03 SUB-ID              PIC XXX.
    03 MESSAGE-PARAMETERS.
        05 USER-PW          PIC X(21).
        05 FILLER          PIC X(1979).
01 LOGICAL-CHANNEL          PIC XXX.
01 DATA-LENGTH            PIC 9(5) COMP.
```

PROCEDURE DIVISION USING
LOGICAL-CHANNEL
DATA-LENGTH
MESSAGE-BODY-IN
MESSAGE-BODY-OUT

START-HERE.

MOVE SPACES TO OUTFILE-NAME.

MOVE ZEROES TO REC-COUNT.

MOVE ZEROES TO RET-STATUS.

TRY-OPEN.

IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.

```
*
*    ORACLE LOGON
*
```

```
MOVE USER-PW TO USER-NAME.
EXEC SQL CONNECT :USER-NAME END-EXEC.
IF SQLCODE = 0 GO TO END-CASE-CHECK.
EXEC SQL CONNECT :OUNAM
IDENTIFIED BY :OUPWD
END-EXEC.
IF SQLCODE < 0
MOVE KES-OLOGON-ERROR TO RET-STATUS
MOVE SQLCODE TO SHOW-RC
STRING "BAD ORACLE LOGON:"
SHOW-RC
INTO MSG-DESC
GO TO PGM-ABORT.
GO TO END-CASE-CHECK.
TRY-CLOSE.
IF CASE-NO NOT = "CLS" GO TO TRY-BEGIN
GO TO END-CASE-CHECK.
TRY-BEGIN.
IF CASE-NO NOT = "BEG" GO TO TRY-COMMIT.
GO TO END-CASE-CHECK.
TRY-COMMIT.
IF CASE-NO NOT = "CMT"
GO TO TRY-ROLLBACK.
EXEC SQL
COMMIT WORK
END-EXEC.
IF SQLCODE < 0
MOVE KES-NOCOMMIT TO RET-STATUS
MOVE SQLCODE TO SHOW-RC
STRING "UNABLE TO COMMIT"
SHOW-RC
INTO MSG-DESC
GO TO PGM-ABORT
ELSE
GO TO END-CASE-CHECK.
TRY-ROLLBACK.
IF CASE-NO NOT = "RBK"
GO TO END-FIXED-CASES.
EXEC SQL ROLLBACK WORK
END-EXEC.
IF SQLCODE < 0
MOVE KES-NOROLLBACK TO RET-STATUS
MOVE SQLCODE TO SHOW-RC
STRING "UNABLE TO ROLLBACK"
SHOW-RC
INTO MSG-DESC
GO TO PGM-ABORT
ELSE
GO TO END-CASE-CHECK.
END-FIXED-CASES.
MOVE CASE-NO TO RP-SUB-NAME.
CALL RP-SUB-NAME USING MESSAGE-BODY-IN
MESSAGE-BODY-OUT
ON EXCEPTION
MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
STRING " CANNOT CALL: "
RP-SUB-NAME
" CASE/SUB: "
CASE-NO
SUB-ID
DELIMITED BY SIZE
```

DS 620341200
30 September 1990

INTO MESG-DESC
GO TO PGM-ABORT.

*
* CALL WORKED --
*

MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.

ORACLE REQUEST PROCESSOR MACROS

LIBRARY: ORACLE
MACRO: RPEND

```
PGM-ABORT.  
  MOVE ZERO TO REC-COUNT.  
END-CASE-CHECK.  
  MOVE RET-STATUS TO QP-STATUS  
  IF RET-STATUS NOT = ZEROES  
    PERFORM PROCESS-ERROR.  
  MOVE NTM-SOURCE TO NTM-DESTINATION.  
  MOVE SPACES TO TIMEOUT-VALUE.  
  MOVE "N" TO DATA-TYPE.  
  CALL "NSEND" USING  NTM-DESTINATION,  
                      LOGICAL-CHANNEL,  
                      TIMEOUT-VALUE,  
                      DATA-TYPE,  
                      OUT-MESSAGE-TYPE,  
                      MSG-OUT-L  
                      MESSAGE-BODY-OUT  
                      ACCEPT-STATUS,  
  IF SEND-MSG-ACCEPTED  
    IF CASE-NO = "CLS"  
      GO TO PGM-END  
    ELSE  
      GO TO WAIT-HERE  
  ELSE  
    MOVE "RP CANNOT REPLY TO DRS" TO MESG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO PGM-END.  
PGM-END.  
*  
* TRMNAT DOES A COBOL STOP RUN  
*  
  CALL "TRMNAT"      USING  TERMINATION-STATUS.  
* INCLUDE THE ERRPRO OF IISSCLIB.  
  COPY ERRPRO OF IISSCLIB.
```

ORACLE REQUEST PROCESSOR MACROS

LIBRARY: ORACLE
MACRO: RPSTOP

PGM-ABORT.
MOVE ZERO TO REC-COUNT.
END-CASE-CHECK.
MOVE RET-STATUS TO QP-STATUS
IF RET-STATUS NOT = ZEROES
PERFORM PROCESS-ERROR.
MOVE MSG-OUT-L TO DATA-LENGTH.
PGM-END.
*
EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.

DB2 REQUEST PROCESSOR MACROS

LIBRARY: DB2
MACRO: RPSTART

```
IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(10) VALUE IS "P1".
01 MSG-DESC PIC X(60).
* REPLY TO DRS
01 MESSAGE-BODY-OUT.
    03 OUTFILE-NAME PIC X(80).
    03 REC-COUNT PIC 9(6) VALUE ZERO.
    03 QP-STATUS PIC 9(5).
01 MSG-OUT-L PIC 9(5) COMP VALUE 91.
* NTM STUFF
01 BUFFER PIC X(4096).
01 BUFFER-SIZE PIC 9(6) VALUE 4096.
01 DATA-TYPE PIC X.
01 NTM-DESTINATION PIC X(10).
01 LOGICAL-CHANNEL PIC X(3).
01 MESSAGE-TYPE PIC X(2).
01 OUT-MESSAGE-TYPE PIC XX VALUE "RR".
01 MESSAGE-SERIAL-NUMBER PIC X(7).
01 NTM-SOURCE PIC X(10).
01 TERMINATION-STATUS PIC X VALUE SPACE.
01 TIMEOUT-VALUE PIC X(15) VALUE ZEROES.
01 WAIT-FLAG PIC 9 VALUE 1.
01 DATA-LENGTH PIC 9(5) COMP.
COPY ERRCDM OF IISSCLIB.
COPY CHKCDM OF IISSCLIB.
COPY SRVRET OF IISSCLIB.
01 SHOW-RC PIC ----9.
* MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
    03 CASE-NO PIC X(6).
    03 SUB-ID PIC XXX.
    03 MESSAGE-PARAMETERS PIC X(2000).
* WS FOR DB2
01 DB2-PLAN-NAME PIC X(8) VALUE "P1".
01 DB2-RETURN-STATUS PIC X(5) VALUE SPACES.
01 DB2-TERMINATION-TYPE PIC X(4) VALUE "ABRT".
EXEC SQL INCLUDE SQLCA END-EXEC.
PROCEDURE DIVISION.
START-HERE.
    CALL "INITAL" USING BUFFER,
                        BUFFER-SIZE,
                        SYSTEM-STATE,
                        RET-CODE.
    IF INITAL-SUCCESSFUL
```

```
      NEXT SENTENCE
    ELSE
      MOVE "RP CANNOT START" TO MSG-DESC
      MOVE RET-CODE TO RET-STATUS
      PERFORM PROCESS-ERROR
      GO TO PGM-END.
WAIT-HERE.
  MOVE SPACES TO OUTFILE-NAME.
  MOVE ZEROES TO REC-COUNT.
  MOVE SPACES TO LOGICAL-CHANNEL, NTM-SOURCE, MESSAGE-TYPE.
  CALL "RCV" USING LOGICAL-CHANNEL,
                  WAIT-FLAG,
                  NTM-SOURCE,
                  MESSAGE-TYPE,
                  DATA-LENGTH,
                  MESSAGE-BODY-IN,
                  ACCEPT-STATUS,
                  MESSAGE-SERIAL-NUMBER.
  IF RCV-NORMAL-MESSAGE
    NEXT SENTENCE
  ELSE
    MOVE "UNABLE TO RECEIVE DRS REQUEST" TO MSG-DESC
    MOVE ACCEPT-STATUS TO RET-STATUS
    PERFORM PROCESS-ERROR
    GO TO PGM-ABORT.
  MOVE ZEROES TO RET-STATUS.
TRY-OPEN.
  IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
  CALL "DB2OPN" USING DB2-PLAN-NAME,
                    SQLCA,
                    DB2-RETURN-STATUS.
  IF DB2-RETURN-STATUS = KES-SUCCESSFUL
    NEXT SENTENCE
  ELSE MOVE DB2-RETURN-STATUS TO RET-STATUS
    MOVE "DB2 OPEN ERROR" TO MSG-DESC
    PERFORM PROCESS-ERROR
    GO TO PGM-ABORT.
  GO TO END-CASE-CHECK.
TRY-CLOSE.
  IF CASE-NO NOT = "CLS" GO TO TRY-BEGIN.
  CALL "DB2CLS" USING DB2-TERMINATION-TYPE,
                    DB2-RETURN-STATUS.
  IF DB2-RETURN-STATUS = KES-SUCCESSFUL
    NEXT SENTENCE
  ELSE MOVE DB2-RETURN-STATUS TO RET-STATUS
    MOVE "DB2 CLOSE ERROR" TO MSG-DESC
    PERFORM PROCESS-ERROR
    GO TO PGM-ABORT.
  GO TO END-CASE-CHECK.
TRY-BEGIN.
  IF CASE-NO NOT = "BEG" GO TO TRY-COMMIT.
  GO TO END-CASE-CHECK.
TRY-COMMIT.
  IF CASE-NO NOT = "CMT"
    GO TO TRY-ROLLBACK.
  EXEC SQL
    COMMIT WORK
  END-EXEC.
  IF SQLCODE < 0
```

```
MOVE KES-NOCOMMIT TO RET-STATUS
MOVE SQLCODE TO SHOW-RC
STRING "UNABLE TO COMMIT" DELIMITED BY SIZE
      SHOW-RC DELIMITED BY SIZE
      INTO MSG-DESC
GO TO PGM-ABORT
ELSE
GO TO END-CASE-CHECK.
TRY-ROLLBACK.
IF CASE-NO NOT = "RBK"
GO TO END-FIXED-CASES.
EXEC SQL ROLLBACK WORK
END-EXEC.
IF SQLCODE < 0
MOVE KES-NOROLLBACK TO RET-STATUS
MOVE SQLCODE TO SHOW-RC
STRING "UNABLE TO ROLLBACK" DELIMITED BY SIZE
      SHOW-RC DELIMITED BY SIZE
      INTO MSG-DESC
GO TO PGM-ABORT
ELSE
GO TO END-CASE-CHECK.
END-FIXED-CASES.
MOVE CASE-NO TO RP-SUB-NAME.
CALL RP-SUB-NAME USING MESSAGE-BODY-IN
                  MESSAGE-BODY-OUT
ON EXCEPTION
MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
STRING " CANNOT CALL: "
      RP-SUB-NAME
      " CASE/SUB: "
      CASE-NO
      SUB-ID DELIMITED BY SIZE
      INTO MSG-DESC
GO TO PGM-ABORT.
*
* CALL WORKED --
*
MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.
```

DB2 REQUEST PROCESSOR MACROS

LIBRARY: DB2
MACRO: RPGO

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(10) VALUE IS "P1".
01 MSG-DESC PIC X(60).
01 SHOW-RC PIC ----9.
01 RP-SUB-NAME PIC X(6).
01 MSG-OUT-L PIC 9(5) COMP VALUE 91.
COPY ERRCMD OF IISSCLIB.
EXEC SQL INCLUDE SQLCA END-EXEC.
01 DB2-PLAN-NAME PIC X(8) VALUE "P1".
01 DB2-RETURN-STATUS PIC X(5) VALUE SPACES.
01 DB2-TERMINATION-TYPE PIC X(4) VALUE "ABRT".
LINKAGE-SECTION.

* REPLY TO DRS

01 MESSAGE-BODY-OUT.
03 OUTFILE-NAME PIC X(80).
03 REC-COUNT PIC 9(6).
03 QP-STATUS PIC X(5).

* MESSAGE FROM DRS

01 MESSAGE-BODY-IN.
03 CASE-NO PIC X(6).
03 SUB-ID PIC XXX.
03 MESSAGE-PARAMETERS PIC X(2000).
01 LOGICAL-CHANNEL PIC XXX.
01 DATA-LENGTH PIC 9(5) COMP.

PROCEDURE DIVISION USING
LOGICAL-CHANNEL
DATA-LENGTH
MESSAGE-BODY-IN
MESSAGE-BODY-OUT.

START-HERE.

MOVE SPACES TO OUTFILE-NAME.
MOVE ZEROES TO REC-COUNT.
MOVE ZEROES TO RET-STATUS.

TRY-OPEN.

IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
CALL "DB2OPN" USING DB2-PLAN-NAME,
SQLCA,
DB2-RETURN-STATUS.

IF DB2-RETURN-STATUS = KES-SUCCESSFUL
NEXT SENTENCE

ELSE MOVE DB2-RETURN-STATUS TO RET-STATUS
MOVE "LOCAL DB2 OPEN FAILURE" TO MSG-DESC
PERFORM PROCESS-ERROR
GO TO PGM-ABORT.

GO TO END-CASE-CHECK.

```
TRY-CLOSE.
  IF CASE-NO NOT = "CLS" GO TO TRY-BEGIN.
  CALL "DB2CLS" USING DB2-TERMINATION-TYPE,
    DB2-RETURN-STATUS.
  IF DB2-RETURN-STATUS = KES-SUCCESSFUL
    NEXT SENTENCE
  ELSE MOVE DB2-RETURN-STATUS TO RET-STATUS
    MOVE "LOCAL DB2 CLOSE FAILURE" TO MESG-DESC
    PERFORM PROCESS-ERROR
    GO TO PGM-ABORT.
GO TO END-CASE-CHECK.
TRY-BEGIN.
  IF CASE-NO NOT = "BEG" GO TO TRY-COMMIT.
  GO TO END-CASE-CHECK.
TRY-COMMIT.
  IF CASE-NO NOT = "CMT"
    GO TO TRY-ROLLBACK.
  EXEC SQL
    COMMIT WORK
  END-EXEC.
  IF SQLCODE < 0
    MOVE KES-NOCOMMIT TO RET-STATUS
    MOVE SQLCODE TO SHOW-RC
    STRING "UNABLE TO COMMIT"
      SHOW-RC
      INTO MESG-DESC
    GO TO PGM-ABORT
  ELSE
    GO TO END-CASE-CHECK.
TRY-ROLLBACK.
  IF CASE-NO NOT = "RBK"
    GO TO END-FIXED-CASES.
  EXEC SQL ROLLBACK WORK
  END-EXEC.
  IF SQLCODE < 0
    MOVE KES-NOROLLBACK TO RET-STATUS
    MOVE SQLCODE TO SHOW-RC
    STRING "UNABLE TO ROLLBACK"
      SHOW-RC
      INTO MESG-DESC
    GO TO PGM-ABORT
  ELSE
    GO TO END-CASE-CHECK.
END-FIXED-CASES.
  MOVE CASE-NO TO RP-SUB-NAME.
  CALL RP-SUB-NAME USING MESSAGE-BODY-IN
    MESSAGE-BODY-OUT
  ON EXCEPTION
    MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
    STRING " CANNOT CALL: "
      RP-SUB-NAME
      " CASE/SUB: "
      CASE-NO
      SUB-ID
      INTO MESG-DESC
    GO TO PGM-ABORT.
  CALL WORKED--
```

DS 620341200
30 September 1990

*

MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.

DB2 REQUEST PROCESSOR MACROS

LIBRARY: DB2
MACRO: RPEND

```
PGM-ABORT.  
  MOVE ZERO TO REC-COUNT.  
END-CASE-CHECK.  
  MOVE RET-STATUS TO QP-STATUS  
  IF RET-STATUS NOT = ZEROES  
    PERFORM PROCESS-ERROR.  
  MOVE NTM-SOURCE TO NTM-DESTINATION.  
  MOVE SPACES TO TIMEOUT-VALUE.  
  MOVE "N" TO DATA-TYPE.  
  CALL "NSEND" USING  NTM-DESTINATION,  
                      LOGICAL-CHANNEL,  
                      TIMEOUT-VALUE,  
                      DATA-TYPE,  
                      OUT-MESSAGE-TYPE,  
                      MSG-OUT-L  
                      MESSAGE-BODY-OUT  
                      ACCEPT-STATUS,  
  IF SEND-MSG-ACCEPTED  
    IF CASE-NO = "CLS"  
      GO TO PGM-END  
    ELSE  
      GO TO WAIT-HERE  
  ELSE  
    MOVE "RP CANNOT REPLY TO DRS" TO MESG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO PGM-END.  
PGM-END.  
*  
  CALL "TRMNAT"      USING  TERMINATION-STATUS.  
  COPY ERRPRO OF IISSCLIB.
```

DB2 REQUEST PROCESSOR MACROS

LIBRARY: DB2
MACRO: RPSTOP

PGM-ABORT.
MOVE ZERO TO REC-COUNT.
END-CASE-CHECK.
MOVE RET-STATUS TO QP-STATUS
IF RET-STATUS NOT = ZEROES
PERFORM PROCESS-ERROR.
MOVE MSG-OUT-L TO DATA-LENGTH.
PGM-END.

*

EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.

TOTAL REQUEST PROCESSOR MACROS

LIBRARY: TOTAL
MACRO: RPSTART

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

```

01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(10) VALUE IS "P1".
01 MSGG-DESC PIC X(60).
01 RP-SUB-NAME PIC X(6).
* REPLY TO DRS
01 MESSAGE-BODY-OUT.
    03 OUTFILE-NAME PIC X(80).
    03 REC-COUNT PIC 9(6) VALUE ZERO.
    03 QP-STATUS PIC 9(5).
01 MSG-OUT-L PIC 9(5) COMP VALUE 91.
* NTM STUFF
01 BUFFER PIC X(4096).
01 BUFFER-SIZE PIC 9(6) VALUE 4096.
01 DATA-TYPE PIC X.
01 NTM-DESTINATION PIC X(10).
01 LOGICAL-CHANNEL PIC X(3).
01 MESSAGE-TYPE PIC X(2).
01 OUT-MESSAGE-TYPE PIC XX VALUE "RR".
01 MESSAGE-SERIAL-NUMBER PIC X(7).
01 NTM-SOURCE PIC X(10).
01 TERMINATION-STATUS PIC X VALUE SPACE.
01 TIMEOUT-VALUE PIC X(15) VALUE ZEROES.
01 WAIT-FLAG PIC 9 VALUE 1.
01 DATA-LENGTH PIC 9(5) COMP.
COPY ERRCDM OF IISSCLIB.
COPY CHKCDM OF IISSCLIB.
COPY SRVRET OF IISSCLIB.
* MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
    03 CASE-NO PIC X(6).
    03 SUB-ID PIC XXX.
    03 MESSAGE-PARAMETERS PIC X(2000).
* WS FOR TOTAL
01 CLOSX PIC X(5) VALUE "CLOSX".
01 SINON PIC X(5) VALUE "SINON".
01 SINOF PIC X(5) VALUE "SINOF".
01 ENDP PIC X(5) VALUE "END".
01 COMIT PIC X(5) VALUE "COMIT".
01 REST PIC X(5) VALUE "RESET".
01 ASGN PIC X(4) VALUE "ASGN".
*
* TOTAL STATUS VALUES
*
01 TOTAL-STATUS PIC X(4).
    88 SUCCESSFUL VALUE "*****".
    88 CONTROL-FIELD-BLANK VALUE "BCTL".

```

30 September 1990

```

      88 MASTER-NOT-FOUND          VALUE "MRNF".
      88 LINK-PATH-INVALID         VALUE "MLNF".
      88 FILE-ALREADY-OPEN        VALUE "DUPO".
      88 NO-SINOF-ISSUED          VALUE "EXSO".
01  TOTAL-ACCESS                  PIC X(6)  VALUE "UPDATE".
01  DBMOD                         PIC X(8)  VALUE "P2".
01  TASK                          PIC X(10) VALUE "P1".
01  OPTIONS                       PIC X(14) VALUE "LOGOPTS=N,END".
01  GLOBAL-REALM.
03  FILLER                        PIC X(6)  VALUE "REALM=".
03  FILLER                        PIC X(13) OCCUR 40 TIMES.
03  FILLER                        PIC X(4)  VALUE "END".
03  REALM-FILE-COUNT              PIC 99.
01  COMIT-LENGTH                  PIC X(4)  VALUE LOW-VALUES.
01  COMIT-DATA-AREA               PIC X      VALUE SPACE.
01  RESET-LENGTH                  PIC X(4)  VALUE LOW-VALUES.
01  RESET-DATA-AREA               PIC X      VALUE SPACE.
PROCEDURE DIVISION.
START-HERE.
    MOVE "REALM=END." TO GLOBAL-REALM.
    MOVE 0 TO REALM-FILE-COUNT.
    CALL "INITAL" USING BUFFER,
                        BUFFER-SIZE,
                        SYSTEM-STATE,
                        RET-CODE.

    IF INITAL-SUCCESSFUL
        NEXT SENTENCE
    ELSE
        MOVE "RP CANNOT START" TO MESG-DESC
        MOVE RET-CODE TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-END.
WAIT-HERE.
    MOVE SPACES TO OUTFILE-NAME.
    MOVE ZEROES TO REC-COUNT.
    MOVE SPACES TO LOGICAL-CHANNEL, NTM-SOURCE,
MESSAGE-TYPE.
        CALL "RCV" USING LOGICAL-CHANNEL,
                        WAIT-FLAG,
                        NTM-SOURCE,
                        MESSAGE-TYPE
                        DATA-LENGTH,
                        MESSAGE-BODY-IN,
                        ACCEPT-STATUS,
                        MESSAGE-SERIAL-NUMBER.

    IF RCV-NORMAL-MESSAGE
        NEXT SENTENCE
    ELSE
        MOVE "UNABLE TO RECEIVE DRS REQUEST" TO MESG-DESC
        MOVE ACCEPT-STATUS TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-ABORT.
    MOVE ZEROES TO RET-STATUS.
TRY-OPEN.
    IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
*  TOTAL INITIATION
*
    CALL "DATBAS" USING SINON,

```

```

                                TOTAL-STATUS,
                                TOTAL-ACCESS,
                                DBMOD,
                                TASK,
                                OPTIONS,
                                ENDP.

IF SUCCESSFUL
    GO TO END-CASE-CHECK
ELSE
    MOVE KES-TOTAL-SINON-FAILED TO RET-STATUS
    STRING "TOTAL SINON FAILED WITH STATUS-"
        DELIMITED BY SIZE
        TOTAL-STATUS DELIMITED BY SIZE
        INTO MSG-DESC
    GO TO PGM-ABORT.
TRY-CLOSE.
    IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.
    CALL "DATBAS" USING CLOSX,
        TOTAL-STATUS,
        GLOBAL-REALM,
        ENDP.

    IF SUCCESSFUL
        NEXT SENTENCE
    ELSE
        MOVE KES-TOTAL-CLOSX-FAILED TO RET-STATUS
        STRING "TOTAL CLOSE FAILED WITH STATUS-"
            DELIMITED BY SIZE
            TOTAL-STATUS DELIMITED BY SIZE
            INTO MSG-DESC
        GO TO PGM-ABORT.
    CALL "DATBAS" USING SINOF,
        TOTAL-STATUS,
        TASK,
        ENDP.

    IF SUCCESSFUL
        NEXT SENTENCE
    ELSE
        MOVE KES-TOTAL-SINOF-FAILED TO RET-STATUS
        STRING "TOTAL SINOF FAILED WITH STATUS-"
            DELIMITED BY SIZE
            TOTAL-STATUS DELIMITED BY SIZE
            INTO MSG-DESC.
TRY-COMMIT.
    IF CASE-NO NOT = "CMT"
        GO TO TRY-ROLLBACK.
    CALL "DATBAS" USING COMMIT,
        TOTAL-STATUS,
        ASGN,
        COMMIT-LENGTH,
        COMMIT-DATA-AREA,
        ENDP.

    IF SUCCESSFUL
        GO TO END-CASE-CHECK
    ELSE
        MOVE KES-TOTAL-COMIT-FAILED TO RET-STATUS
        STRING "TOTAL COMMIT FAILED WITH STATUS OF"
            DELIMITED BY SIZE
            TOTAL-STATUS DELIMITED BY SIZE
            INTO MSG-DESC
```

```
        GO TO PGM-ABORT.
TRY-ROLLBACK.
    IF CASE-NO NOT = "RBK"
        GO TO END-FIXED-CASES.
    CALL "DATBAS" USING REST,
                                TOTAL-STATUS,
                                ASGN,
                                RESET-LENGTH
                                RESET-DATA-AREA,
                                ENDP.

    IF SUCCESSFUL
        GO TO END-CASE-CHECK
    ELSE
        MOVE KES-TOTAL-RESET-FAILED TO RET-STATUS
        STRING "TOTAL RESET FAILED WITH STATUS OF"
                                DELIMITED BY SIZE
                                TOTAL-STATUS      DELIMITED BY SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT.
END-FIXED-CASES.
    MOVE CASE-NO TO RP-SUB-NAME.
    CALL RP-SUB-NAME USING MESSAGE-BODY-IN
        MESSAGE-BODY-OUT
    ON EXCEPTION
        MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
        STRING " CANNOT CALL: "
            RP-SUB-NAME
            " CASE/SUB: "
            CASE-NO
            SUB-ID      DELIMITED BY SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT.
*
* CALL WORKED--
*
    MOVE QP-STATUS TO RET-STATUS.
    GO TO END-CASE-CHECK.
```

TOTAL REQUEST PROCESSOR MACROS

LIBRARY: TOTAL
MACRO: RPGO

IDENTIFICATION DIVISION.

PROGRAM-ID. P1.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(10) VALUE IS "P1".
01 MSG-DESC PIC X(60).
01 RP-SUB-NAME PIC X(6).

COPY ERRCDM OF IISSCLIB.

* WS FOR TOTAL
01 CLOSX PIC X(5) VALUE "CLOSX".
01 SINON PIC X(5) VALUE "SINON".
01 SINOF PIC X(5) VALUE "SINOF".
01 ENDP PIC X(5) VALUE "END.".
01 COMIT PIC X(5) VALUE "COMIT".
01 REST PIC X(5) VALUE "RESET".
01 ASGN PIC X(4) VALUE "ASGN".

*
* TOTAL STATUS VALUES
*

01 TOTAL-STATUS PIC X(4).
88 SUCCESSFUL VALUE "*****".
88 CONTROL-FIELD-BLANK VALUE "BCTL".
88 MASTER-NOT-FOUND VALUE "MRNF".
88 LINK-PATH-INVALID VALUE "MLNF".
88 FILE-ALREADY-OPEN VALUE "DUPO"
88 NO-SINOF-ISSUED VALUE "EXSO".

*
01 TOTAL-ACCESS PIC X(6) VALUE "UPDATE".
01 DBMOD PIC X(8) VALUE "P2".
01 TASK PIC X(10) VALUE "P1".
01 OPTIONS PIC X(14) VALUE "LOGOPTS=N,END".
01 GLOBAL-REALM.
03 FILLER PIC X(6) VALUE "REALM=".
03 FILLER PIC X(13) OCCURS 40

TIMES.
03 FILLER PIC X(4) VALUE "END.".
03 REALM-FILE-COUNT PIC 99.
01 COMIT-LENGTH PIC X(4) VALUE LOW-VALUES.
01 COMIT-DATA-AREA PIC X VALUE SPACE.
01 RESET-LENGTH PIC X(4) VALUE LOW-VALUES.
01 RESET-DATA-AREA PIC X VALUE SPACE.
01 MSG-OUT-L PIC S9(5) COMP VALUE 91.

LINKAGE SECTION.

01 MESSAGE-BODY-IN.
03 CASE-NO PIC X(6).
03 SUB-ID PIC XXX.
03 MESSAGE-PARAMETERS PIC X(2000).
01 MESSAGE-BODY-OUT.
03 OUTFILE-NAME PIC X(80).

	03	REC-COUNT	PIC 9(6).
	03	QP-STATUS	PIC X(5).
01		DATA-LENGTH	PIC S9(5) COMP.
01		LOGICAL-CHANNEL	PIC X(3).

PROCEDURE DIVISION USING
LOGICAL-CHANNEL
DATA-LENGTH
MESSAGE-BODY-IN
MESSAGE-BODY-OUT.

START-HERE

MOVE SPACES TO OUTFILE-NAME.
MOVE ZEROES TO REC-COUNT.
MOVE ZEROES TO RET-STATUS.

TRY-OPEN.

IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.

*
*
*

TOTAL INITIALIZATION

MOVE "REALM=END." TO GLOBAL-REALM.
MOVE 0 TO REALM-FILE-COUNT.
CALL "DATBAS" USING SINON,
TOTAL-STATUS,
TOTAL-ACCESS,
DBMOD,
TASK,
OPTIONS,
ENDP.

IF SUCCESSFUL

GO TO END-CASE-CHECK

ELSE

MOVE KES-TOTAL-SINON-FAILED TO RET-STATUS
STRING "TOTAL SINON FAILED WITH STATUS-"
DELIMITED BY SIZE
TOTAL-STATUS DELIMITED BY SIZE
INTO MMSG-DESC

GO TO PGM-ABORT.

TRY-CLOSE.

IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.

CALL "DATBAS" USING CLOSX,
TOTAL-STATUS,
GLOBAL-REALM,
ENDP.

IF SUCCESSFUL

NEXT SENTENCE

ELSE

MOVE KES-TOTAL-CLOSX-FAILED TO RET-STATUS
STRING "TOTAL CLOSE FAILED WITH STATUS-"
DELIMITED BY SIZE
TOTAL-STATUS DELIMITED BY SIZE
INTO MMSG-DESC

GO TO PGM-ABORT.

CALL "DATBAS" USING SINOF,
TOTAL-STATUS,
TASK,
ENDP.

IF SUCCESSFUL

NEXT SENTENCE

ELSE

```
MOVE KES-TOTAL-SINOF-FAILED TO RET-STATUS
STRING "TOTAL SINOF FAILED WITH STATUS-"
      DELIMITED BY SIZE
      TOTAL-STATUS DELIMITED BY SIZE
      INTO MESG-DESC
GO TO PGM-ABORT.
TRY-COMMIT.
  IF CASE-NO NOT = "CMT"
    GO TO TRY-ROLLBACK.
  CALL "DATBAS" USING COMIT,
        TOTAL-STATUS
        ASGN,
        COMMIT-LENGTH,
        COMMIT-DATA-AREA,
        ENDP.
  IF SUCCESSFUL
    GO TO END-CASE-CHECK
  ELSE
    MOVE KES-TOTAL-COMIT-FAILED TO RET-STATUS
    STRING "TOTAL COMMIT FAILED WITH STATUS OF"
      DELIMITED BY SIZE
      TOTAL-STATUS DELIMITED BY SIZE
    INTO MESG-DESC
    GO TO PGM-ABORT
TRY-ROLLBACK.
  IF CASE-NO NOT = "RBK"
    GO TO END-FIXED-CASES.
  CALL "DATBAS" USING REST,
        TOTAL-STATUS,
        ASGN,
        RESET-LENGTH,
        RESET-DATA-AREA,
        ENDP.
  IF SUCCESSFUL
    GO TO END-CASE-CHECK
  ELSE
    MOVE KES-TOTAL-RESET-FAILED TO RET-STATUS
    STRING "TOTAL RESET FAILED WITH STATUS OF"
      DELIMITED BY SIZE
      TOTAL-STATUS DELIMITED BY SIZE
    INTO MESG-DESC
    GO TO PGM-ABORT.
END-FIXED-CASES.
  MOVE CASE-NO TO RP-SUB-NAME.
  CALL RP-SUB-NAME USING MESSAGE-BODY-IN
    MESSAGE-BODY-OUT
  ON EXCEPTION
    MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
    STRING " CANNOT CALL: "
      RP-SUB-NAME
      " CASE/SUB: "
      CASE-NO
      SUB-ID      DELIMITED BY SIZE
    INTO MESG-DESC
    GO TO PGM-ABORT.
*
* CALL WORKED--
```

DS 620341200
30 September 1990

*

MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.

TOTAL REQUEST PROCESSOR MACROS

LIBRARY: TOTAL
MACRO: RPEND

```
PGM-ABORT.  
  MOVE ZERO TO REC-COUNT.  
END-CASE-CHECK.  
  MOVE RET-STATUS TO QP-STATUS.  
  IF RET-STATUS NOT = ZEROES  
    PERFORM PROCESS-ERROR.  
  MOVE NTM-SOURCE TO NTM-DESTINATION.  
  MOVE SPACES TO TIMEOUT-VALUE.  
  MOVE "N" TO DATA-TYPE.  
  CALL "NSEND" USING  NTM-DESTINATION,  
                      LOGICAL-CHANNEL,  
                      TIMEOUT-VALUE,  
                      DATA-TYPE,  
                      OUT-MESSAGE-TYPE,  
                      MSG-OUT-L  
                      MESSAGE-BODY-OUT  
                      ACCEPT-STATUS.  
  IF SEND-MSG-ACCEPTED  
    IF CASE-NO = "CLS"  
      GO TO PGM-END  
    ELSE  
      GO TO WAIT-HERE  
  ELSE  
    MOVE "RP CANNOT REPLY TO DRS" TO MESSG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO PGM-END.  
PGM-END.  
*  
*   TRMNAT DOES A COBOL STOP RUN  
*  
*       CALL "TRMNAT" USING TERMINATION-STATUS.  
*  
*   INCLUDE THE ERRPRO OF IISSCLIB.  
*  
*       COPY ERRPRO OF IISSCLIB.
```

DS 620341200
30 September 1990
TOTAL REQUEST PROCESSOR MACROS

LIBRARY: TOTAL
MACRO: RPSTOP

PGM-ABORT.
MOVE ZERO TO REC-COUNT.
END-CASE-CHECK.
MOVE RET-STATUS TO QP-STATUS.
IF RET-STATUS NOT = ZEROES
PERFORM PROCESS-ERROR.
MOVE MSG-OUT-L TO DATA-LENGTH.
PGM-END.

*

EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.

LIBRARY: VAX11
MACRO: RPSTART

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.

```
*
SUB-SCHEMA SECTION.
DB P6 WITHIN P3
FOR "P5".
WORKING-STORAGE SECTION.
01  RET-STATUS                PIC X(5).
01  MODULE-NAME              PIC X(8) VALUE  "P1".
01  MSG-DESC                 PIC X(60).
01  RP-SUB-NAME              PIC X(6).
*  REPLY TO DRS
01  MESSAGE-BODY-OUT.
    03  OUTFILE-NAME          PIC X(80).
    03  REC-COUNT             PIC 9(6) VALUE ZERO.
    03  QP-STATUS             PIC 9(5).
01  MSG-OUT-L                PIC 9(5) COMP VALUE 91.
*  NTM STUFF
01  BUFFER                   PIC X(4096).
01  BUFFER-SIZE              PIC 9(6) VALUE 4096.
01  DATA-TYPE               PIC X.
01  NTM-DESTINATION          PIC X(10).
01  LOGICAL-CHANNEL          PIC X(3).
01  MESSAGE-TYPE             PIC X(2).
01  OUT-MESSAGE-TYPE         PIC XX VALUE "RR".
01  MESSAGE-SERIAL-NUMBER    PIC X(7).
01  NTM-SOURCE               PIC X(10).
01  TERMINATION-STATUS       PIC X  VALUE SPACE.
01  TIMEOUT-VALUE            PIC X(15) VALUE ZEROES.
01  WAIT-FLAG                PIC 9 VALUE 1.
01  DATA-LENGTH            PIC 9(5) COMP.
COPY CHKCDM OF IISSCLIB.
COPY SRVRET OF IISSCLIB.
*  MESSAGE FROM DRS
01  MESSAGE-BODY-IN.
    03  CASE-NO               PIC X(6).
    03  SUB-ID               PIC XXX.
    03  MESSAGE-PARAMETERS    PIC X(2000).
*  WS FOR VAX-11
01  DBMS-STATUS              PIC S9(9).
    88  EOA  VALUE 2654548.
    88  EOS  VALUE 2654548.
    88  EOC  VALUE 2654548.
    88  EOO  VALUE 2654548.
    88  OK-STATUS VALUE 1.
    88  OK VALUE 1 2654548.
    88  NON-FATAL VALUE 2654548 1.
01  DISP-STATUS              PIC -----9.
```

```
PROCEDURE DIVISION.
DECLARATIVES.
DB-DATABASE-EXCEPTIONS SECTION.
    USE FOR DB-EXCEPTION.
DB-ERROR-ROUTINE.
END DECLARATIVES.
START-PROGRAM SECTION.
START-HERE.
    CALL "LOCKEF".
    CALL "INITAL" USING BUFFER,
                        BUFFER-SIZE,
                        SYSTEM-STATE,
                        RET-CODE.

    IF INITAL-SUCCESSFUL
        NEXT SENTENCE
    ELSE
        MOVE "RP CANNOT START" TO MESG-DESC
        MOVE RET-CODE TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-END.
WAIT-HERE.
    MOVE SPACES TO OUTFILE-NAME.
    MOVE ZEROES TO REC-COUNT.
    MOVE SPACES TO LOGICAL-CHANNEL, NTM-SOURCE,
MESSAGE-TYPE.
    CALL "RCV" USING LOGICAL-CHANNEL,
                    WAIT-FLAG,
                    NTM-SOURCE,
                    MESSAGE-TYPE,
                    DATA-LENGTH,
                    MESSAGE-BODY-IN,
                    ACCEPT-STATUS,
                    MESSAGE-SERIAL-NUMBER.

    IF RCV-NORMAL-MESSAGE
        NEXT SENTENCE
    ELSE
        MOVE "UNABLE TO RECEIVE DRS REQUEST" TO MESG-DESC
        MOVE ACCEPT-STATUS TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-ABORT.
        MOVE ZEROES TO RET-STATUS.
TRY-OPEN.
    IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
*       VAX-11 INITIALIZATION
*
    READY CONCURRENT UPDATE.
    PERFORM VAX-11-STATUS.
    IF NOT OK
        MOVE KES-VAX11-READY-FAILED TO RET-STATUS
        STRING "VAX-11 READY FAILED"          DELIMITED BY SIZE
            DISP-STATUS                      DELIMITED BY SIZE
            INTO MESG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-CLOSE.
    IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.
*
    FINISH.
```

```
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-FINISH-FAILED TO RET-STATUS
    STRING "VAX-11 FINISH FAILED"      DELIMITED BY
SIZE                                     DISP-STATUS      DELIMITED BY SIZE INTO
MSG-DESC
    GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY COMMIT.
    IF CASE-NO NOT = "CMT"
        GO TO TRY-BEGIN.
    COMMIT.
    PERFORM VAX-11-STATUS.
    IF NOT OK
        MOVE KES-VAX11-COMMIT-FAILED TO RET-STATUS
        STRING "VAX-11 COMMIT FAILED"    DELIMITED BY SIZE
        DISP-STATUS                      DELIMITED BY SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-BEGIN.
    IF CASE-NO NOT = "BEG" GO TO TRY-ROLLBACK.
    READY CONCURRENT UPDATE.
    PERFORM VAX-11-STATUS.
    IF NOT OK
        MOVE KES-VAX11-READY-FAILED TO RET-STATUS
        STRING "VAX-11 READY FAILED" DISP-STATUS
        DELIMITED BY SIZE INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-ROLLBACK.
    IF CASE-NO NOT = "RBK"
        GO TO END-FIXED-CASES.
    ROLLBACK.
    PERFORM VAX-11-STATUS.
    IF NOT OK
        MOVE KES-VAX11-ROLLBACK-FAILED TO RET-STATUS
        STRING "VAX-11-ROLLBACK FAILED" DELIMITED BY SIZE
        DISP-STATUS                      DELIMITED BY SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
VAX-11-STATUS.
    MOVE DB-CONDITION TO DBMS-STATUS, DISP-STATUS.
END-FIXED-CASES.
    MOVE CASE-NO TO RP-SUB-NAME.
    CALL RP-SUB-NAME USING MESSAGE-BODY-IN
        MESSAGE-BODY-OUT
ON EXCEPTION
    MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
    STRING " CANNOT CALL: "
        RP-SUB-NAME
        " CASE/SUB: "
```

DS 620341200
30 September 1990

CASE-NO
SUB-ID DELIMITED BY SIZE
INTO MSG-DESC
GO TO PGM-ABORT.

*
*
*

CALL WORKED--

MOVE QP-STATUS TO RET-STATUS.
GO TO END-CASE-CHECK.

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RPGO

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
DATA DIVISION.

*

SUB-SCHEMA SECTION.

DB P6 WITHIN P3
FOR "P5".

WORKING-STORAGE SECTION.

01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(8) VALUE "P1".
01 MSGG-DESC PIC X(60).
01 MSGG-OUT-L PIC 9(5) COMP VALUE 91.
01 RP-SUB-NAME PIC X(6).

COPY ERRCDM OF IISSCLIB.

* WS FOR VAX-11

01 DBMS-STATUS PIC S9(9).
88 EOA VALUE 2654548.
88 EOS VALUE 2654548.
88 EOC VALUE 2654548.
88 EOO VALUE 2654548.
88 OK-STATUS VALUE 1.
88 OK VALUE 1 2654548.
88 NON-FATAL VALUE 2654548 1.

01 DISP-STATUS PIC -----9.

LINKAGE SECTION.

01 MESSAGE-BODY-OUT.

03 OUTFILE-NAME PIC X(80).
03 REC-COUNT PIC 9(6).
03 QP-STATUS PIC X(5).

01 MESSAGE-BODY-IN

03 CASE-NO PIC X(6).
03 SUB-ID PIC XXX.
03 MESSAGE-PARAMETERS PIC X(2000).

01 LOGICAL-CHANNEL PIC X(3).

01 DATA-LENGTH PIC S9(5) COMP.

PROCEDURE DIVISION USING
LOGICAL-CHANNEL
DATA-LENGTH
MESSAGE-BODY-IN
MESSAGE-BODY-OUT.

DECLARATIVES.

DB-DATABASE-EXCEPTIONS SECTION. USE FOR DB-EXCEPTION.

DB-ERROR-ROUTINE.

END DECLARATIVES.

START-PROGRAM SECTION.

START-HERE.

MOVE SPACES TO OUTFILE-NAME.

MOVE ZEROES TO REC-COUNT.

MOVE ZEROES TO RET-STATUS.

TRY-OPEN.

```
IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
* VAX-11  INITIALIZATION
*
READY          CONCURRENT UPDATE.
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-READY-FAILED TO RET-STATUS
    STRING "VAX-11 READY FAILED"          DELIMITED BY SIZE
    DISP-STATUS                          DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY-CLOSE.
IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.
*
FINISH.
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-FINISH-FAILED TO RET-STATUS
    STRING "VAX-11 FINISH FAILED"        DELIMITED BY SIZE
    DISP-STATUS                          DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY-COMMIT.
IF CASE-NO NOT = "CMT"
    GO TO TRY-BEGIN.
COMMIT.
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-COMMIT-FAILED TO RET-STATUS
    STRING "IDMS COMMIT FAILED"          DELIMITED BY SIZE
    DISP-STATUS                          DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY-BEGIN.
IF CASE-NO NOT = "BEG" GO TO TRY-ROLLBACK.
READY CONCURRENT UPDATE.
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-READY-FAILED TO RET-STATUS
    STRING "VAX-11 READY FAILED" DELIMITED BY SIZE
    DISP-STATUS                  DELIMITED BY SIZE
    INTO MSG-DESC
    GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
TRY-ROLLBACK.
IF CASE-NO NOT = "RBK"
    GO TO END-FIXED-CASES.
ROLLBACK.
PERFORM VAX-11-STATUS.
IF NOT OK
    MOVE KES-VAX11-ROLLBACK-FAILED TO RET-STATUS
    STRING "VAX-11 ROLLBACK FAILED" DELIMITED BY SIZE
```

DS 620341200
30 September 1990
DELIMITED BY SIZE

```
DISP-STATUS
INTO MSG-DESC
GO TO PGM-ABORT
ELSE
  GO TO END-CASE-CHECK.
VAX-11-STATUS.
  MOVE DB-CONDITION TO DBMS-STATUS, DISP-STATUS.
END-FIXED-CASES.
  MOVE CASE-NO TO RP-SUB-NAME.
  CALL RP-SUB-NAME USING MESSAGE-BODY-IN
    MESSAGE-BODY-OUT
  ON EXCEPTION
    MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
    STRING " CANNOT CALL: "
      RP-SUB-NAME
        " CASE/SUB: "
          CASE-NO
            SUB-ID          DELIMITED BY SIZE
              INTO MSG-DESC
                GO TO PGM-ABORT.
*  CALL WORKED--
*
  MOVE QP-STATUS TO RET-STATUS.
  GO TO END-CASE-CHECK.
```

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RPEND

```
PGM-ABORT.  
MOVE ZERO TO REC-COUNT.  
END-CASE-CHECK.  
MOVE RET-STATUS TO QP-STATUS  
IF RET-STATUS NOT = ZEROES  
    PERFORM PROCESS-ERROR.  
MOVE NTM-SOURCE TO NTM-DESTINATION.  
MOVE SPACES TO TIMEOUT-VALUE.  
MOVE "N" TO DATA-TYPE.  
CALL "NSEND" USING NTM-DESTINATION,  
                    LOGICAL-CHANNEL,  
                    TIMEOUT-VALUE,  
                    DATA-TYPE,  
                    OUT-MESSAGE-TYPE,  
                    MSG-OUT-L  
                    MESSAGE-BODY-OUT  
                    ACCEPT-STATUS.  
IF SEND-MSG-ACCEPTED  
    IF CASE-NO = "CLS"  
        GO TO PGM-END  
    ELSE  
        GO TO WAIT-HERE  
ELSE  
    MOVE "RP CANNOT REPLY TO DRS" TO MESG-DESC  
    PERFORM PROCESS-ERROR  
    GO TO PGM-END.  
PGM-END.  
*  
* TRMNAT DOES A COBOL STOP RUN  
*  
*     CALL "TRMNAT"      USING     TERMINATION-STATUS.  
*  
* INCLUDE THE ERRPRO OF IISSCLIB.  
*  
*     COPY ERRPRO OF IISSCLIB.
```

VAX-11 REQUEST PROCESSOR MACROS

LIBRARY: VAX-11
MACRO: RPSTOP

PGM-ABORT.
MOVE ZERO TO REC-COUNT.
END-CASE-CHECK.
MOVE RET-STATUS TO QP-STATUS.
IF RET-STATUS NOT = ZEROES
 PERFORM PROCESS-ERROR.
MOVE MSG-OUT-L TO DATA-LENGTH.
PGM-END.
EXIT PROGRAM.
COPY ERRPRO OF IISSCLIB.

LIBRARY: IDMS
MACRO: RPSTART

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
IDMS-CONTROL SECTION.
PROTOCOL. MODE IS BATCH DEBUG
IDMS-RECORDS WITHIN WORKING-STORAGE SECTION.
DATA DIVISION.

*

SCHEMA SECTION.
DB P6 WITHIN P3.

WORKING-STORAGE SECTION.

YANKME*01 INPUT-CARD PIC X(80).
01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(8) VALUE "P1".
01 MSG-DESC PIC X(60).
01 RP-SUB-NAME PIC X(6).
* REPLY TO DRS
01 MESSAGE-BODY-OUT.
03 OUTFILE-NAME PIC X(80).
03 REC-COUNT PIC 9(6) VALUE ZERO.
03 QP-STATUS PIC 9(5).
01 MSG-OUT-L PIC 9(5) COMP VALUE 91.
* NTM STUFF
01 BUFFER PIC X(4096).
01 BUFFER-SIZE PIC 9(6) VALUE 4096.
01 DATA-TYPE PIC X.
01 NTM-DESTINATION PIC X(10).
01 LOGICAL-CHANNEL PIC X(3).
01 MESSAGE-TYPE PIC X(2).
01 OUT-MESSAGE-TYPE PIC XX VALUE "RR".
01 MESSAGE-SERIAL-NUMBER PIC X(7).
01 NTM-SOURCE PIC X(10).
01 TERMINATION-STATUS PIC X VALUE SPACE.
01 TIMEOUT-VALUE PIC X(15) VALUE ZEROES.
01 WAIT-FLAG PIC 9 VALUE 1.
01 DATA-LENGTH PIC 9(5) COMP.
KEEPME*COPY ERRCDM OF IISSCLIB.
KEEPME*COPY CHKCDM OF IISSCLIB.
KEEPME*COPY SRVRET OF IISSCLIB..
* MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
03 CASE-NO PIC X(6).
03 SUB-ID PIC XXX.
03 MESSAGE-PARAMETERS PIC X(2000).
* WS FOR IDMS
01 DBMS-STATUS PIC (4).
88 EOA VALUE "0307".
88 EOS VALUE "0307".
88 EOC VALUE "0364" "0326" "0332".
88 EOO VALUE "0307".
88 OK-STATUS VALUE "0000".

```

88 OK VALUES "0307" "0364" "0326" "0332" "0000".
88 NON-FATAL VALUES "0307" "0364" "0326" "0332"
"0000".
01 DISP-STATUS          PIC -----9.
PROCEDURE DIVISION.
START-HERE.
    CALL "INITAL" USING  BUFFER,
                        BUFFER-SIZE,
                        SYSTEM-STATE,
                        RET-CODE.

    IF INITAL-SUCCESSFUL
        NEXT SENTENCE
    ELSE
        MOVE "RP CANNOT START" TO MESG-DESC
        MOVE RET-CODE TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-END.
    COPY IDMS SUBSCHEMA-BINDS.
WAIT-HERE.
    MOVE SPACES TO OUTFILE-NAME.
    MOVE ZEROES TO REC-COUNT.
    MOVE SPACES TO LOGICAL-CHANNEL, NTM-SOURCE,
MESSAGE-TYPE.
    CALL "RCV" USING LOGICAL-CHANNEL,
                    WAIT-FLAG,
                    NTM-SOURCE,
                    MESSAGE-TYPE,
                    DATA-LENGTH,
                    MESSAGE-BODY-IN,
                    ACCEPT-STATUS,
                    MESSAGE-SERIAL-NUMBER.

    IF RCV-NORMAL-MESSAGE
        NEXT SENTENCE
    ELSE
        MOVE "UNABLE TO RECEIVE DRS REQUEST" TO MESG-DESC
        MOVE ACCEPT-STATUS TO RET-STATUS
        PERFORM PROCESS-ERROR
        GO TO PGM-ABORT.
YANKME*    DISPLAY "WAITING FOR NEXT INPUT MESSAGE:"
YANKME*    ACCEPT INPUT-CARD.
YANKME*    MOVE SPACES TO MESSAGE-BODY-IN.
YANKME*    MOVE INPUT-CARD TO MESSAGE-BODY-IN.
YANKME*    DISPLAY "==>" MESSAGE-BODY-IN.
            MOVE ZEROES TO RET-STATUS.
TRY-OPEN.
    IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
*    IDMS    INITIALIZATION
*
    READY          USAGE-MODE IS UPDATE.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-READY-FAILED TO RET-STATUS
        STRING "IDMS READY FAILED"          DELIMITED BY
SIZE
        DISP-STATUS          DELIMITED BY
SIZE
        INTO MESG-DESC
        GO TO PGM-ABORT

```

```
ELSE
    GO TO END-CASE-CHECK.
TRY-CLOSE.
    IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.
    FINISH.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-FINISH-FAILED TO RET-STATUS
        STRING "IDMS FINISH FAILED" DELIMITED BY
SIZE
        DISP-STATUS DELIMITED BY
SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-COMMIT.
    IF CASE-NO NOT = "CMT"
        GO TO TRY-BEGIN.
    COMMIT.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-COMMIT-FAILED TO RET-STATUS
        STRING "IDMS COMMIT FAILED" DELIMITED BY
SIZE
        DISP-STATUS DELIMITED BY
SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CHECK.
TRY-BEGIN.
    IF CASE-NO NOT = "BEG" GO TO TRY-ROLLBACK
    READY USAGE-MODE IS UPDATE.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-READY-FAILED TO RET-STATUS
        STRING "IDMS READY FAILED" DISP-STATUS
        DELIMITED BY SIZE INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-ROLLBACK.
    IF CASE-NO NOT = "RBK"
        GO TO END-FIXED-CASES.
    ROLLBACK.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-ROLLBACK-FAILED TO RET-STATUS
        STRING "IDMS ROLLBACK FAILED" DELIMITED BY
SIZE
        DISP-STATUS DELIMITED BY
SIZE
        INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
IDMS-STATUS.
    MOVE ERROR-STATUS TO DNMS-STATUS, DISP-STATUS.
```

DS 620341200
30 September 1990

END-FIXED-CASES.

MOVE CASE-NO TO RP-SUB-NAME.

CALL RP-SUB-NAME USING MESSAGE-BODY-IN
MESSAGE-BODY-OUT

ON EXCEPTION

MOVE KES-NO-RSUB-ERROR TO RET-STATUS

SRING " CANNOT CALL: "

RP-SUB-NAME

" CASE/SUB: "

CASE-NO

SUB-ID

DELIMITED BY SIZE

INTO MESG-DESC

GO TO PGM-ABORT.

*

*

*

CALL WORKED--

MOVE QP-STATUS TO RET-STATUS.

GO TO END-CASE-CHECK.

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RPGO

IDENTIFICATION DIVISION.
PROGRAM-ID. P1.
ENVIRONMENT DIVISION.
IDMS-CONTROL SECTION.
PROTOCOL. MODE IS BATCH DEBUG
IDMS-RECORDS WITHIN WORKING-STORAGE SECTION.

DATA DIVISION.
*
SCHEMA SECTION.
DB P6 WITHIN P3.

WORKING-STORAGE SECTION.
01 RET-STATUS PIC X(5).
01 MODULE-NAME PIC X(8) VALUE "P1".
01 MSGG-DESC PIC X(60).
01 MSGG-OUT-L PIC 9(5) COMP VALUE 91.
01 RP-SUB-NAME PIC X(6).
COPY ERRCDM OF IISSCLIB.
* WS FOR IDMS
01 DBMS-STATUS PIC X(4).
88 EOA VALUE "0307".
88 EOS VALUE "0307".
88 EOC VALUE "0364" "0326" "0332".
88 EOO VALUE "0307".
88 OK-STATUS VALUE "0000".
88 OK VALUES "0307" "0364" "0326" "0332" "0000".
88 NON-FATAL VALUES "0307" "0364" "0326" "0332" "0000".
01 DISP-STATUS PIC -----9.

LINKAGE SECTION.
* MESSAGE FROM DRS
01 MESSAGE-BODY-IN.
03 CASE-NO PIC X(6).
03 SUB-ID PIC XXX.
03 MESSAGE-PARAMETERS PIC X(2000).
* REPLY TO DRS
01 MESSAGE-BODY-OUT.
03 OUTFILE-NAME PIC X(80).
03 REC-COUNT PIC 9(6).
03 QP-STATUS PIC X(5).
01 LOGICAL-CHANNEL PIC X(3).
01 DATA-LENGTH PIC 9(5) COMP.

PROCEDURE DIVISION USING
LOGICAL-CHANNEL
DATA-LENGTH
MESSAGE-BODY-IN
MESSAGE-BODY-OUT.

START-HERE.
MOVE SPACES TO OUTFILE-NAME.
MOVE ZEROES TO REC-COUNT.
MOVE ZEROES TO RET-STATUS.

```
COPY IDMS SUBSCHEMA-BINDS.
TRY-OPEN.
    IF CASE-NO NOT = "000000" GO TO TRY-CLOSE.
*
*   IDMS   INITIALIZATION
*
    READY          USAGE-MODE IS UPDATE.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-READY-FAILED TO RET-STATUS
        STRING "IDMS READY FAILED"          DELIMITED BY SIZE
            DISP-STATUS                      DELIMITED BY SIZE
            INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-CLOSE.
    IF CASE-NO NOT = "CLS" GO TO TRY-COMMIT.
    FINISH.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-FINISH-FAILED TO RET-STATUS
        STRING "IDMS FINISH FAILED"          DELIMITED BY SIZE
            DISP-STATUS                      DELIMITED BY SIZE
            INTO MSG-DESC.
TRY-COMMIT.
    IF CASE-NO NOT = "CMT"
        GO TO TRY-BEGIN.
    COMMIT.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-COMMIT-FAILED TO RET-STATUS
        STRING "IDMS COMMIT FAILED"          DELIMITED BY SIZE
            DISP-STATUS                      DELIMITED BY SIZE
            INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-BEGIN.
    IF CASE-NO NOT = "BEG" GO TO TRY-ROLLBACK.
    READY          USAGE-MODE IS UPDATE.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-READY-FAILED TO RET-STATUS
        STRING "IDMS READY FAILED" DISP-STATUS
        DELIMITED BY SIZE INTO MSG-DESC
        GO TO PGM-ABORT
    ELSE
        GO TO END-CASE-CHECK.
TRY-ROLLBACK.
    IF CASE-NO NOT = "RBK"
        GO TO END-FIXED-CASES.
    ROLLBACK.
    PERFORM IDMS-STATUS.
    IF NOT OK
        MOVE KES-IDMS-ROLLBACK-FAILED TO RET-STATUS
        STRING "IDMS ROLLBACK FAILED"          DELIMITED BY SIZE
            DISP-STATUS                      DELIMITED BY SIZE
            INTO MSG-DESC
```

```
        GO TO PGM-ABORT
ELSE
    GO TO END-CASE-CHECK.
IDMS-STATUS.
    MOVE ERROR-STATUS TO DBMS-STATUS, DISP-STATUS.
END-FIXED-CASES.
    MOVE CASE-NO TO RP-SUB-NAME.
    CALL RP-SUB-NAME USING MESSAGE-BODY-IN
                          MESSAGE-BODY-OUT
    ON EXCEPTION
        MOVE KES-NO-RPSUB-ERROR TO RET-STATUS
        STRING " CANNOT CALL: "
            RP-SUB-NAME
            " CASE/SUB: "
            CASE-NO
            SUB-ID          DELIMITED BY SIZE
        INTO MESG-DESC
        GO TO PGM-ABORT.
*
* CALL WORKED--
*
    MOVE QP-STATUS TO RET-STATUS.
    GO TO END-CASE-CHECK.
```

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RPEND

```
PGM-ABORT.  
    MOVE ZERO TO REC-COUNT.  
END-CASE-CHECK.  
    MOVE RET-STATUS TO QP-STATUS  
    IF RET-STATUS NOT = ZEROES  
        PERFORM PROCESS-ERROR.  
    MOVE NTM-SOURCE TO NTM-DESTINATION.  
    MOVE SPACES TO TIMEOUT-VALUE.  
    MOVE "N" TO DATA-TYPE.  
YANKME*    DISPLAY 'REPORTING STATUS'.  
YANKME*    DISPLAY MESSAGE-BODY-OUT.  
YANKME*    DISPLAY '-----'.  
    CALL "NSEND" USING NTM-DESTINATION,  
                        LOGICAL-CHANNEL,  
                        TIMEOUT-VALUE,  
                        DATA-TYPE,  
                        OUT-MESSAGE-TYPE,  
                        MSG-OUT-L  
                        MESSAGE-BODY-OUT  
                        ACCEPT-STATUS.  
    IF SEND-MSG-ACCEPTED  
        IF CASE-NO = "CLS"  
            GO TO PGM-END  
        ELSE  
            GO TO WAIT-HERE  
    ELSE  
        MOVE "RP CANNOT REPLY TO DRS" TO MESG-DESC  
        PERFORM PROCESS-ERROR  
        GO TO PGM-END.  
PGM-END.  
*  
* TRMNAT DOES A COBOL STOP RUN  
*  
    CALL "TRMNAT" USING TERMINATION-STATUS.  
YANKME* STOP RUN.  
*  
* INCLUDE THE ERRPRO OF IISSCLIB.  
*  
    COPY ERRPRO OF IISSCLIB.  
YANKME*PROCESS-ERROR.  
YANKME*    DISPLAY 'ERRPRO*****'.  
YANKME*    DISPLAY RET-STATUS.  
YANKME*    DISPLAY MODULE-NAME.  
YANKME*    DISPLAY MESG-DESC.  
YANKME*    DISPLAY '*****'.  
/*
```

IDMS REQUEST PROCESSOR MACROS

LIBRARY: IDMS
MACRO: RPSTOP

PGM-ABORT.
 MOVE ZERO TO REC-COUNT.
END-CASE-CHECK.
 MOVE RET-STATUS TO QP-STATUS.
 IF RET-STATUS NOT = ZEROES
 PERFORM PROCESS-ERROR.
 MOVE MSG-OUT-L TO DATA-LENGTH.
PGM-END.
 EXIT PROGRAM.
 COPY ERRPRO OF IISSCLIB.

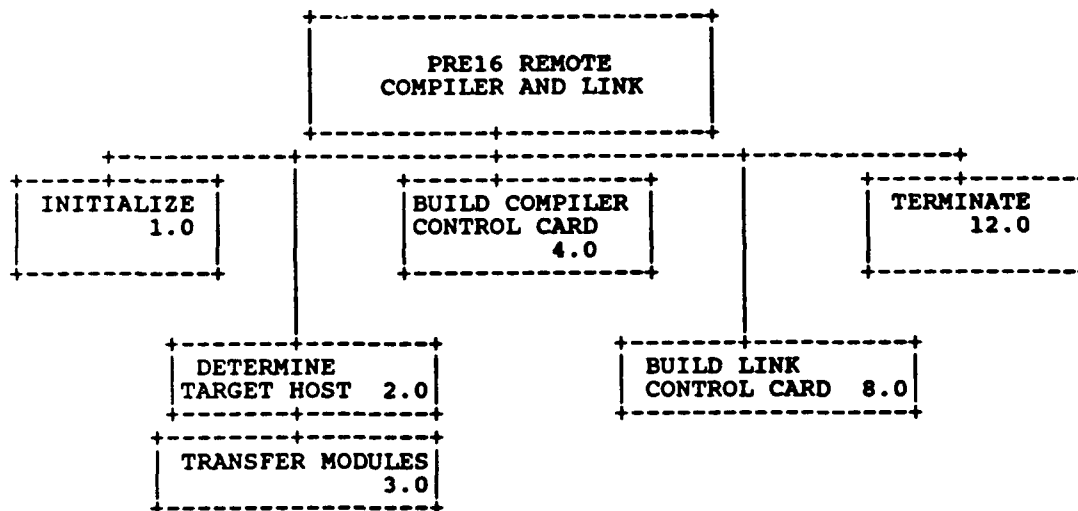
SECTION 42

Function PRE16 - Precompiler Remote Compile and Link

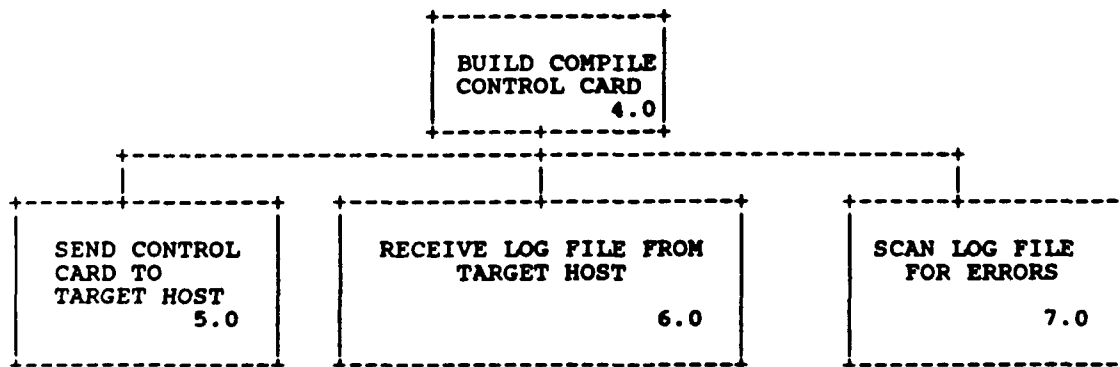
This function requests the Network Transaction Manager (NTM) to compile and link all modules generated from precompilation. PRE16 will transfer all files to the appropriate host computer, build the control cards needed to compile and link the modules and access the NTM service "SNDRCLE" to execute these control cards.

The following structure charts illustrate the major functions to be accomplished by PRE16.

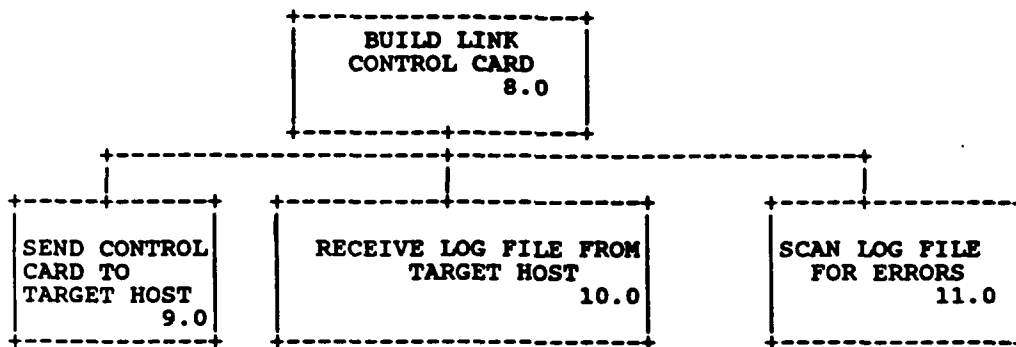
REMOTE COMPILE AND LINK



REMOTE COMPILER AND LINK



REMOTE COMPILE AND LINK (CONT'D)



42.1 Inputs:

1. Code Generator Table

The Code Generator Table contains the results about all code generated or modified by the precompiler. This table is passed to CDRCL from module NDML. The fields used by CDRCL are CGT-CURRENT-HOST, CGT-DBMS, CGT-GEN-FILE-NAME, CGT-INDEX, CGT-LOG-FILE-NAME, CGT-MOD-NAME, CGT-TARGET-HOST, and CGT-USED.

* CGTABLE.INC

```
01 CODE-GENERATOR-TABLE
03 CGT-USED PIC 999 VALUE 0.
03 CGT-MAX PIC 999 VALUE 189.
03 CGT-ENTRY OCCURS 190 TIMES INDEXED BY CGT-INDEX
    05 CGT-MOD-NAME PIC X(10).
    05 CGT-LANGUAGE PIC X(8).
    05 CGT-TARGET-HOST PIC XXX.
    05 CGT-DBMS PIC X(30).
    05 CGT-DB-NAME PIC X(30).
    05 CGT-MOD-TYPE PIC X(10).
    88 USER-MODULE VALUE "USER-MOD".
    88 RP-MAIN VALUE "RP-MAIN".
    88 RP-SUB VALUE "RP-SUB".
    88 CS-ES VALUE "CS-ES".
    05 CGT-ACTION PIC X.
    05 CGT-GENED-BY PIC X(10).
    05 CGT-GEN-FILE-NAME PIC X(30).
    05 CGT-PASSWORD PIC X(30).
    05 CGT-LOCALITY PIC X.
    88 CGT-LOCAL VALUE "L".
    88 CGT-REMOTE VALUE "R".
*
* THE ABOVE CAN BE SHOWN TO THE USER, THE FOLLOWING ARE FOR
* RCL AND INTERNAL USAGE:
    05 CGT-DBID PIC 9(6).
    05 CGT-LIBRARY-NAME PIC X(30).
    05 CGT-SUBTRANS-ID PIC 9(6).
    05 CGT-CASE-NO PIC 9(6).
    05 CGT-SCHEMA PIC X(30).
    05 CGT-SUBSCHEMA PIC X(30).
    05 CGT-DB-LOCATION PIC X(30).
    05 CGT-PASSWORD PIC X(30).
* THE FOLLOWING ARE REQUIRED FOR RCL FUNCTIONS
    05 CGT-LOG-FILE-NAME PIC X(30).
    05 CGT-RCL-LOG-CHAIN PIC XXX.
    05 CGT-CURRENT-HOST PIC XXX.
* NOTE THAT CURRENT HOST MAY CHANGE DURING THE RCL PROCESS
    05 CGT-RCL-STATUS PIC X(5).
    88 CODE-GEN VALUE "GEN".
    88 CODE-XFERRED VALUE "XFER".
    88 CODE-COMPILED VALUE "COMP".
    88 AP-LINKED VALUE "LINK".
    88 AP-DEFINED VALUE "NTM".
```

42.2 Internal Requirements:

1. The Control Card Area Table contains the control cards needed to compile and link program modules on various host computers for various DBMs. It also contains the error keywords used to search the log files to determine if errors occurred during the compilation and linking of modules.

```

*
01 CONTROL-CARD-AREA-TABLE.
03 CGT-USED PIC 999 VALUE 0.
03 CGT-MAX PIC 999 VALUE 20.
05 FILLER.
    07 FILLER PIC X(3) VALUE "VAX".
    07 FILLER PIC X(30) VALUE "COBOL".
        09 FILLER PIC X(70) VALUE.
        " @RPLIB P1".
        09 FILLER PIC XX.
    07 FILLER PIC X(10) VALUE "%COBOL".
05 FILLER
    07 FILLER PIC X(3) VALUE "VAX".
    07 FILLER PIC X(30) VALUE.
        "VAX-11"
    07 FILLER.
        09 FILLER PIC X(70) VALUE.
        "LKVAX11 P1 NODEBUG".
        09 FILLER PIC XX.
    07 FILLER PIC X(10) VALUE "%LINK".

03 CONTROL-CARD-TABLE REDEFINES CONTROL-CARD-AREA.
05 CONTROL-CARD-REC OCCURS 20 TIMES
    INDEXED BY CGT-INDEX.
    07 CGT-HOST PIC XXX.
    07 CGT-FUNC-DBMS PIC X(30).
    07 CGT-CTL-CARD
        09 CGT-CONTROL-CARD PIC X(70).
        09 CGT-TERMINATOR PIC XX.
    07 CGT-ERROR-KEYWORD PIC X(10).

*
* TERMINATOR CONTAINS A 'IE' IN HEXADECIMAL ENTERED
* WITH A CONTROL/6 ON A DEC VT100 SERIES TERMINAL
* CONSTANT, TO BE MOVED TO CGT-TERMINATOR IN PROCESS
*
03 TERMINATOR
    05 FILLER PIC X VALUE ' '.
    05 FILLER PIC X VALUE ' '.

```

42.3 Constraints:

1. None

42.4 Outputs:

1. A status code indicating whether function CDRCL was successful.

01 RET-STATUS PIC X(5)

42.5 PROCESSING:

Remote Compile and Link PRE16, module CDRCL transfers all generated routines from the precompiler that need to be transferred to a host computer. When all routines are transferred, CDRCL calls BLDCC to build the control cards to compile the routines and then sends the control cards to the target host computer. A log file is returned for each compiled routine and is scanned by LOGANA for error conditions. If no errors exist in any of the compiles, CDRCL calls BLDCC to generate the control cards to link the main routines. CDRCL sends these cards to the target host, a log file is returned for each, and then LOGANA is called to scan the log files for error conditions.

Modules that are called by CDRCL other than BLDCC and LOGANA are:
FILXFR to transfer files; SNDRCLE to send control cards; RCV to receive the log file name; DELFIL to delete the log file; ERRPRO to process error codes. SNDRCLE and RCV are part of the NTM services. FILXFR and DELFIL are part of the CDM File Utilities Configuration Item. ERRPRO is part of the COMM subsystem.

1. Initialize CDRCL.
 - 1.1 Set internal flags and RET-STATUS to default values.
2. Check for generated source code that needs to be transferred to different host.
 - 2.1 Compare each CGT-TARGET-HOST to CGT-CURRENT-HOST, if the values are different, set up parameters to transfer the routine to the target host.
3. Transfer each routine identified in Step 2.1
 - 3.1 Call FILXFR passing the parameters needed to transfer the routine from the current host to the target host.
4. Build compile control card for each entry in the Code Generator Table.
 - 4.1 Call "BLDCC" to build the compile control card from the CONTROL-CARD-TABLE using CGT-LANGUAGE in the selection.

5. Send control card to target host for each entry found in Step 4.
 - 5.1 Call "SNDRCLE" to send the compile control card to the target host as specified by CGT-TARGET-HOST.
6. Obtain log file from target host for each entry found in Step 4.
 - 6.1 Call "RCV" to obtain the log file name.
 - 6.2 If the log file is on a host other than the current host, call "FILXFR" to transfer the log file to the current host as specified by CGT-CURRENT-HOST.
7. Scan each log file for errors.
 - 7.1 Call LOGANA to scan the log file for an error keyword as specified by CCT-ERR-KEYWORD for the compile indicated by CGT-LANGUAGE.
8. Build link control card for each entry in the Code Generator Table that is a main program, and is not a Local request processor (CGT-LOCAL).
 - 8.1 Call "BLDCC" to build the link control card from CONTROL-CARD-TABLE using CGT-DBMS in the selection.
9. Send control card to target host for each entry found in Step 8.
 - 9.1 Sets up parameters and calls SNDRCLE to send the link control card to the target host.
10. Obtain log file from target host for each entry found in Step 8.
 - 10.1 Call "RCV" to obtain the log file name.
 - 10.2 If the log file is on a host other than the current host, call "FILXFR" to transfer the log file to the current host as specified by CGT-CURRENT-HOST.
11. Scan each log file for errors.
 - 11.1 Call LOGANA to scan the log file for an error keyword as specified by CCT-ERROR-KEYWORD for the link indicated by CGT-DBMS.
12. Terminate.
 - 12.1 Terminates CDRCL.

DS 620341200
30 September 1990

SECTION 43

QUALITY ASSURANCE PROVISION

In preparation for describing requirements for quality assurance provisions it is appropriate to define the terms "test" and "debug" which are often used interchangeably. "Testing" is a systematic process that may be preplanned and explicitly scheduled. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error. To start with, the concept of "antibugging" is recommended in the construction of the software modules. In his text on software development (Techniques of Program Structure and Design, Prentice-Hall, 1975), Yourdon defines antibugging as "the philosophy of writing programs in such a way as to make bugs less likely to occur, and when they do occur (which is inevitable), to make them more noticeable to the programmer and the user." That is, do as much error checking as is practical and possible in each routine.

Among the tests that should be incorporated into all software are:

1. input data checks
2. interface data checks, i.e., tests to determine validity of data passed from calling routine
3. database verification
4. operator command checks
5. output data checks

Not all tests are required in all routines, but error checking is an essential part of all software.

The CI quality assurance provisions must consist of three levels of test, validation and qualification of the constructed application software.

- A. The initial level can consist of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests will be performed by the design team which will be organized in a manner similar to that discussed by Weinberg in his text on software development team organization (The Psychology of Computer Programming, New York: Van Nostrand Reinhold, 1971). Essentially a team is assigned to work on a subsystem or CI. This approach has been referred to as "adaptive teams" and "egoless teams". Members of the team are involved in the overall design of the subsystem. There is better control and members are exposed to each other's design. The specific advantages from a quality assurance point is

the formalized critique of design walk-throughs which are a preventive measure for desing errors and program "bugs". Structured design, design walk-throughs and the incorporation of "antibugging" facilitate this level of testing by exposing and addressing problem areas before they become coded "bugs."

- B. Preliminary qualification tests of the CI are performed to highlight the special functions of the CI from an integrated point of view. Certain functional requirements may require the cooperative execution of one or more modules to achieve an intermediate or special function of the CI. Specific test plans will be provided for the validation of this type of functional requirement including preparation of appropriate test data. (Selected functions from 3.2 must be listed).
- C. Formal Qualification Test will verify the functional performance of all the modules, within the CI as an integrated unit, that accept the specified input, perform the specified processes and deliver the specified outputs. Special consideration must be given to test data to ensure verification that proper interface of modules has been constructed.

SECTION 44

PREPARATION FOR DELIVERY

The implementation site for the constructed software will be the ICAM Integrated Support System (IISS) Test Bed site located at Arizona State University, Tempe AZ. The required computer equipment will have been installed. The constructed software will be transferred to the IISS system via appropriate storage media.