

LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

AD-A228 061

DTIC ACCESSION NUMBER

LEVEL

DTIC FILE COPY

INVENTORY

WRDC-TR-90-8027 Vol. III
DOCUMENT IDENTIFICATION
NOV 1990

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS	GRA&I
DTIC	TRAC
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION/	
AVAILABILITY CODES	
DISTRIBUTION	AVAILABILITY AND/OR SPECIAL
A-1	

DISTRIBUTION STAMP

DTIC
ELECTE
NOV 02 1990

DATE ACCESSIONED

DATE RETURNED

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC

H
A
N
D
L
E
W
I
T
H
C
A
R
E

AD-A228 061

WRDC-TR-90-8027
Volume III



GEOMETRIC MODELING APPLICATIONS INTERFACE PROGRAM

GMAP/PDDI SYSTEM COMPONENT PRODUCT SPECIFICATION (AS BUILT)

VOL. III - Model Access Software Listings

United Technologies Corporation
Pratt and Whitney
Government Products Division
P.O. Box 9600
West Palm Beach, Florida 33410-9600

NOVEMBER 1990

Final Report For Period August 1985 - March 1989

Approved for public release; distribution is unlimited

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

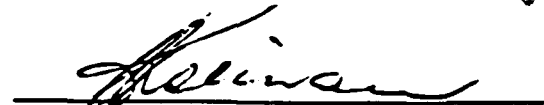
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

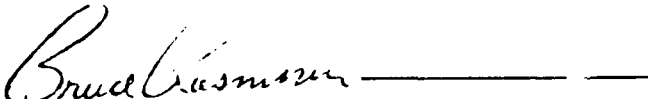
This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

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


Charles Gilman
Project Manager


Walter H. Reimann, Chief
Computer-Integrated Mfg. Branch

FOR THE COMMANDER


BRUCE A. RASMUSSEN
Chief, Integration Technology Division
Manufacturing Technology Directorate

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/MTT, WPAFB, 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

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) FR 20889		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8027, Vol. III	
6a. NAME OF PERFORMING ORGANIZATION United Technologies Corporation Pratt & Whitney Government Products Division	6b. OFFICE SYMBOL (If applicable) (P&W)	7a. NAME OF MONITORING ORGANIZATION Wright Research and Development Center Manufacturing Technology Directorate (WRDC/MTI)	
6c. ADDRESS (City, State and ZIP Code) P.O. Box 9600 West Palm Beach, Florida 33410-9600		7b. ADDRESS (City, State and ZIP Code) Wright-Patterson Air Force Base, OH 45433-6533	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-5122	
8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS.	
		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT NO.
11. TITLE (Include Security Classification) GEOMETRIC MODELING APPLICATIONS INTERFACE PROGRAM (GMAP)		78011F	MTPI 06 84
12. PERSONAL AUTHOR(S) R. Disa, C. Van Wie, K. Arnold, J. Altemueller, A. Whelan, G. White, J. Purses			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 1 Aug 85 TO 31 MAR 89	14. DATE OF REPORT (Yr., Mo., Day) November 1990	15. PAGE COUNT 360
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	Geometric Modeling Applications Interface Program	
		Product Definition Data Interface	
		Turbine Blades and Disks	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			
This "As-Built" Product Specification establishes the "as-built" Computer Program Configuration Item (CPCI) identified as the GMAP/PDDI System Components under U.S. Air Force Contract F33615-85-C-5122. It includes descriptions of the structure, functions, language, database requirements, interfaces, and quality assurance provisions of the primary GMAP system components: the System Translator, Model Access Software with Name/Value Interface, and the Schema Manager.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL David Judson		22b. TELEPHONE NUMBER (Include Area Code) (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

18. Subject Terms (Continued)

Product Life Cycle
Engineering
Manufacturing
Interface
Exchange Format
CAD
CAM
CIM
IBIS
RFC
System Translator
Schema Manager
Model Access Software
Name/Value Interface

FOREWORD

This As-built Product Specification, divided into four volumes, covers work performed under Air Force Contract F33615-85-C-5122, Geometric Modeling Applications Interface Program (GMAP), covering the period 1 August 1985 to 31 March 1989. The document addresses the GMAP/PDDI System Components developed or enhanced under this contract which is sponsored by the Computer Integrated Manufacturing Branch, Materials Laboratory, Air Force Systems Command, Wright Air Force Base, Ohio 45433-6533. The GMAP Project Manager for the Air Force is Mr. Charles Gilman.

The primary contractor is Pratt & Whitney, an operating unit of United Technologies Corporation. Mr. Richard Lopatka is managing the GMAP project at Pratt & Whitney. Ms. Linda Phillips is the Program Integrator. Mr. John Hamill is the Deputy Program Manager.

McDonnell Aircraft Company was the subcontractor responsible for the PDDI System Component work. Mr. Jerry Weiss is the GMAP Program Manager at McDonnell Aircraft and Mr. Herb Ryan is the Deputy Program Manager.

Volume III of this Product Specification provides the Model Access Software with Name/Value Interface routine listings.

NOTE: The number and date in the upper right corner of each page in this document indicate that it has been prepared in accordance to the ICAM CM Life Cycle Documentation requirements for a Configuration Item (CI).

TABLE OF CONTENTS

Volume I

		<u>Page</u>
SECTION 1.	SCOPE.....	1-1
1.1	Identification.....	1-1
1.2	Functional Summary.....	1-1
1.3	Approach.....	1-2
SECTION 2.	REFERENCES.....	2-1
2.1	Reference Documents.....	2-1
2.1.1	Military.....	2-1
2.1.2	Commercial.....	2-4
2.1.3	Standards Organizations.....	2-5
2.2	Terms and Acronyms.....	2-6
2.2.1	Glossary A -- Terms Used In GMAP.....	2-6
2.2.2	Glossary B -- Terms Used In IDEF0 Diagrams....	2-19
2.2.3	Acronyms Used In GMAP.....	2-24
SECTION 3.	DETAIL DESIGN.....	3-1
3.1	System Overview.....	3-1
3.1.1	Physical Schemas.....	3-1
3.1.2	Software Packages.....	3-1
3.2	IDEF0 Function Models.....	3-1
3.2.1	Schema Manager.....	3-3
3.2.1.1	A-0 Manage Schema Data.....	3-3
3.2.1.2	A0 - Manage Schema Data.....	3-3
3.2.2	Model Access Software.....	3-7
3.2.2.1	A-0 Perform Model Access.....	3-7
3.2.2.2	A0 Perform Model Access.....	3-7
3.2.3	System Translator.....	3-10
3.2.3.1	A0 Exchange PDD.....	3-10
3.2.3.2	A1 Preprocess PDD.....	3-10
3.2.3.3	A12 Create Data Section.....	3-13
3.2.3.4	A2 Postprocess PDD.....	3-13
3.2.3.5	A23 Process Data Section.....	3-16
3.2.3.6	A24 Resolve Forward References.....	3-16
3.3	Application Interfaces.....	3-19
3.3.1	Interfaces Between GMAP/PDDI	
	System Components.....	3-19
3.3.1.1	Application Program/Working Form.....	3-20
3.3.1.2	User Interface/System Translator.....	3-20
3.3.1.3	Working Form/Exchange Format.....	3-22
3.3.1.4	Working Form/Database	
	Management System.....	3-22

TABLE OF CONTENTS (Continued)

	<u>Page</u>
3.3.1.5 Exchange Format.....	3-22
3.3.1.6 Schema Manager.....	3-23
3.3.2 Interfaces Between Applications.....	3-23
3.3.2.1 Working Form as Exchange Mechanism.....	3-24
3.3.2.2 Working Form as the Native Form.....	3-24
3.3.2.3 Exchange Format and Working Form.....	3-24
3.3.2.4 Transfer Using the Working Form, Exchange Format, and Direct Translator to the Exchange Format.....	3-28
3.4 Program Interrupts.....	3-29
3.4.1 Schema Manager.....	3-29
3.4.2 Model Access Software with Name Value Interface.....	3-31
3.5 Timing and Sequencing Description.....	3-33
3.6 Data Dictionary.....	3-34
3.6.1 Schema Manager.....	3-34
3.6.2 Model Access Software Data Dictionary.....	3-64
3.6.3 Name Value Interface Data Dictionary.....	3-92
3.6.4 System Translator Data Dictionary.....	3-104
3.7 Object Code Creation.....	3-107
3.7.1 Schema Manager.....	3-107
3.7.2 Model Access Software.....	3-107
3.7.3 System Translator.....	3-107
3.7.3.1 Use Portable Higher Order Language.....	3-107
3.7.3.2 Use Model Access Software.....	3-107
3.7.3.3 Interface to Exchange Format.....	3-108
3.7.3.4 Interface to Native System.....	3-108
3.8 Adaptation Data.....	3-108
3.9 Detail Design Description.....	3-108
3.9.1 Schema Manager Hierarchy.....	3-109
3.9.2 Model Access Software Hierarchy.....	3-120
3.9.3 Name/Value Interface Hierarchy.....	3-162
3.9.4 System Translator.....	3-168
3.9.4.1 Postprocessor.....	3-168
3.9.4.2 Preprocessor.....	3-170

TABLE OF CONTENTS (Continued)

		<u>Page</u>
Volume II		
3.10	Routine Listings.....	3-171
3.10.1	Schema Manager.....	3-171
3.10.1.1	Index.....	3-171
3.10.1.2	Listings.....	3-178
Volume III		
3.10.2	Model Access Software.....	3-625
3.10.2.1	Index.....	3-625
3.10.2.2	Listings.....	3-631
3.10.3	N/VI.....	3-913
3.10.3.1	Index.....	3-913
3.10.3.2	Listings.....	3-914
Volume IV		
3.10.4	System Translator.....	3-979
3.10.4.1	Index.....	3-979
3.10.4.2	Listings.....	3-980
SECTION 4	QUALITY ASSURANCE.....	4-1
4.1	Quality Assurance (QA) Requirements.....	4-1

3.10.2 Model Access Software

3.10.2.1 Index

<u>Routine Name</u>	<u>Function</u>
ADCRBM	- Adds a new CRB entry.
ADRLSM	- Adds an entity after a relative position in a system list.
ADSCH	- Connects an internal item to the correct portion of the NDS superstructure.
ADSCHR	- Connects an internal item to the schema root.
ADTLSM	- Adds an entity to a system list.
ADTNM	- Adds an entity to the end of an application list.
BIGCREMM	- Create a user-constituent relation between entities with a constituent list of a given size.
CHKDEL	- Checks deletability of an entity relative to its users.
CHKTDEL	- Checks deletability of an entity relative to its users.
CMPCRB	- Compresses the CRB.
CNNODM	- Connects two entities.
CNNODMN	- Connect two entities and create a constituent list of a given size.
CNVOSP	- Converts out of space system error code to user recognizeable error code.
CNVRR	- Gets the external return code corresponding to the internal format.
CYPAUDB	- Stores the value of an application entity block in an uninitialized system UDB.
CPYCST	- Adds the entities in a constituent list into a list.
CPYLSM	- Copies the non-vacant elements of LIST FROM to LIST TO.
CPYNM	- Creates a new list which contains a copy of the entities referenced by KEYL.
CRCLST	- Creates relations between a user entity and a list of constituents.
CRCLSTN	- Create relations between a user entity and a list of constituents.
CRCNM	- Creates relations between a user entity and a list of constituents.
CRDLST	- Creates a sorted inclusive list of an entity or a list of entities and their direct and indirect constituents.
CREMM	- Creates a user-constituent relation between entities.
CRURUL	- Creates the user's rules for deletability.
DELCNST	- Determines deletability of entity's constituents.
DELCRBE	- Deletes a CRB entry.
DELEMM	- Deletes all references to this entity from all application lists and disposes of the entity.
DELPLST	- Removes an entity from a specified position in a system list.

DELPNLA	- Deletes all non-"locked" APPL lists after a specified position in the LIST_OF_LISTS.
DELRLSM	- Removes an entity from a system list.
DELRUL	- Deletes an entity according to the delete rules.
DELSCH	- Disconnects an internal item from the correct portion of the NDS superstructure.
DELTLSM	- Removes the last non-vacant entity reference in a list.
DETCNST	- Checks deletability of entity's constituents.
DETRUL	- Tests delete of an entity according to the delete rules.
DIFLSM	- Creates a system's list consisting of all entities in LIST1 that are not in LIST2.
DISPCRB	- Disposes of CRB.
DISPEMM	- Releases all space allocated to an entity.
DISPLSM	- Deletes space allocated to a system list.
DISPNM	- Removes all entities from the list and free the allocated space.
ELDNM	- Creates a list with all duplicate entities eliminated.
ELMNODM	- Returns an ENTBLOCK corresponding to a key.
EXCRBE	- Exchanges two entries in the CRB.
EXPCLSM	- Expands list with all of its constituents and places this expanded list in LISTOUT.
EXPCRB	- Expands the CRB.
EXPSUDB	- Expands a system UDL.
EXPULSM	- Places the expanded list with all of its users in LISTOUT.
EXPULSMI	- Place the expanded list with all of its users in LISTOUT.
FDSCH	- Finds a Schema_Instance_Collector or Schema_Class entity on the specified Schema_Root's constituent list.
FNDCRBE	- Finds a specific entry in the CRB.
FNDSKIND	- Builds an array of kind value collected by a class or instance collector in the schema.
GTCRBE	- Gets an entry in the CRB.
INDLSM	- Locates an entity in a system list.
INITMGR	- Initialize the memory manager.
INNM	- Indicates whether a list references an entity.
INTLSM	- Creates a list which is the intersection of two lists.
LSTLNM	- Returns the number of non-vacant entities in a system list.
LSTMXLNM	- Returns the number of entries allocated to a system list.
MABRST	- Reset the process and application flags for all entities.
MACPDT	- Update a specified application flag for the constituents of an entity or list of entities.
MAEA	- Activates an entity.
MAEAI	- Activates an entity or a list of entities and their inclusive constituents.
MAEAV	- Finds the present value of the activation setting for an entity.
MAEC	- Creates an application list of constituent entities.
MAECI	- Creates an application list of inclusive constituent entities.

MAECIK	- Creates a list of inclusive constituents by kind.
MAECMP	- Determines which of it's constituents an entity compresses with.
MAECQY	- Determines if an entity's user should compress with it.
MAECR	- Creates an entity.
MAECRN	- Create an entity with a constituent list of a given size.
MAECTK	- Returns the number of "KIND" values in the WF model.
MAECXQ	- Execute a procedure on a list, creating an output list.
MAED	- Deletes an entity or list of entities.
MAEDI	- Deletes inclusively an entity or list of entities.
MAEDT	- Tests delete an entity or list of entities.
MAEDTI	- Tests for inclusive deletion of an entity or list of entities and their direct and indirect constituents.
MAEDTS	- Tests delete an entity or list of entities, and return three lists.
MAEGKN	- Retrieves the KIND value of an entity.
MAEGTK	- Retrieves the entity block which corresponds to KEYE.
MAEKND	- Returns a "KIND" value from the list of KINDS in the WF model.
MAERST	- Reset the specified flag in all entities in the WF model.
MAESCI	- Set or reset the process flag for the inclusive constituents of an entity or list of entities.
MAESVL	- Finds the current binary switch setting of an entity.
MAESWA	- Sets the process bit "off" in all entities in the model.
MAESWT	- Sets an entity switch or the switches for each entity in a list as requested by the user.
MAEU	- Creates a list of user entity references.
MAEUD	- Updates the entity block corresponding to a key.
MAEUI	- Creates an application list of inclusive user entities.
MAEUIK	- Creates a list of inclusive users by kind.
MAEUSR	- Determines if an entity has any users.
MAEUXQ	- Executes a procedure on the users of an entity.
MAEXEQ	- Executes a procedure on an entity, or a list of entities.
MAINIT	- Initializes the MAS network.
MAKCNT	- Determines the number of entities of a specified kind in the WF model.
MAKILL	- Deletes the WF model.
MAKXEQ	- Executes a procedure on all entities of a specified kind.
MAL	- Creates an empty list.
MALAND	- Creates an application list of entities common to two input lists.
MALATC	- Appends an entity of list (LIST2) to an entity or list (KEY1).
MALCPY	- Makes a copy of a list.
MALD	- Deletes an application list.
MALDA	- Deletes all application lists that are not "locked".
MALDI	- Deletes an application list and all lists after it that are not locked.

April 1990

MALFND	- Finds the position of an entity (KEY2) in an applicaiton list (KEY1).
MALGTK	- Gets the Nth key from the list.
MALINS	- Inserts an entity or list into a list.
MALK	- Creates a list of all entities of a specified kind.
MALKC	- Create a list of constituesnts of a specified kind.
MALKL	- Creates a list of entity kinds which are found within another list.
MALKU	- Create a list of users of a specified kind.
MALN	- Creates an empty list of a specified size.
MALNO	- Counts the entities on the list.
MALNOT	- Creates an application list of entities in KEY1 but not in KEY2.
MALOCK	- Sets an application list for delete or non-delete status.
MALOR	- Creates an application list from a BOOLEAN "OR" on two input lists.
MALRD	- Reads the next entry in a directed list.
MALRDE	- Removes duplicate entries in a list.
MALREP	- Replaces a list.
MALRMV	- Removes an entity from a list.
MALROR	- Reorders an application list in user-constituent order.
MALRORI	- Put the application list in inclusive user to constituent order.
MALRPL	- Replaces an entity in a list.
MALRRI	- Put the application list in inclusive user to constituent order.
MALRST	- Reset an application list for reuse.
MALRVS	- Reverses the order of an application list.
MALSRT	- Sorts an application list.
MALSTF	- Initializes for reading a directed list in forward order.
MALSTR	- Initializes for reading a directed list in reverse order.
MALXEQ	- Executes a procedure on an entity or a list of entities.
MAQURY	- Determines the value of the specified flag for an entity.
MARDLT	- Delete the run-time subschema entry for the specified entity kind.
MARSGT	- Locate the run-time subschema entry for the specified entity kind.
MASALOC	- MAS memory management runtime.
MASDSP	- Disposes of a MAS dynamically allocated memory area.
MASMSZ	- Returns the actual model space used and the amount of the free space in the allocated memory blocks of the model.
MASNEW	- Allocates a new dynamic memory area for MAS elements.
MASOVR	- MAS Memory management runtime.
MAUPDT	- Update the specified flag for an entity.
MIDBD	- Deletes an entity without checking delete rules.
MIDBRV	- Removes an entity from a list without checking delete rules.
MOVRLSM	- Moves entities between system lists.

MRGTLSM	- Concatenates the entities in LIST2 to LIST1.
MRGTNM	- Concatenates the entities in LIST2 to LIST1.
MRKNM	- Marks the stack of lists so that the next release list will only destroy lists created after this mark operation.
MRSCR	- Store a run-time subschema entry for the specified entity kind.
MSINIT	- Initialize the Working Form with a specified minimum size.
MSTART	- Generates start statistics.
MSTOP	- Generates stop statistics.
NDSCTM	- Defines dummy program.
NDSFCT	- Computes the amount of used model space and the amount of free space in the allocated memory blocks.
NDSGBM	- Dummy procedure for compile time initialization of NDS global area.
NDSRML	- Releases all memory blocks allocated to the WF.
NEWCRB	- Creates a CRB.
NEWETM	- Creates a new NDS object.
NEWIIM	- Creates a new entity and copies into it the application ENTDATA.
NEWLSM	- Initializes LISTREF and allocates enough space to hold size entities.
NEWNDM	- Creates a new empty model in memory.
NEWNM	- Creates an empty application list.
NEWNMT	- Create an empty application list without adding it to the list of lists.
NEWNODE	- Creates a new entity in the NDS and copies into it the application ENTDATA.
NEWNSI	- Creates an empty schema instance collector attached to the schema root.
NEWNSR	- Creates a new null schema root and attaches it to the NDS.
NEWSADB	- Creates a new application data block.
NEWSCHI	- Creates an empty schema instance collector entity attached to the schema root.
NEWSCHR	- Creates an empty root collector entity attached to the NDS.
NODECNM	- Creates a list which contains a copy of the entity's constituent list.
NODECNN	- Create a copy of the entity's constituent list without adding it to the list of lists.
NODEUNM	- Creates a list which contains a copy of the entity's user list.
OCOUNT	- MAS memory management runtime.
ORDRLST	- Reorders an application list.
ORDRLSTI	- Put the application list in inclusive user to constituent order.
OSTART	- MAS memory management runtime.
PASASM	- Links to a user defined procedure.
RDLSM	- Reads a system list as a first-in first-out order.

RDRLSM	- Reads the last entity key from LISTREF.
RDTLSM	- Reads the last entity key from LISTREF.
REVAADB	- Assigns the value of a system UDB to an application ENTBLOCK.
REVNODM	- Revises an entity's user data block.
REVRLSM	- Changes an entity in a system list.
REVSADB	- Replaces the value of a system ENTBLOCK with the value of ENTDEF.
RLSNM	- Releases all the lists on the current list of lists.
RSTLSM	- Resets position to indicate the beginning of a list.
RSTSFLG	- Resets the requested position in the internal MAS process flag (MAPROB) in the IIT to the requested BOOLEAN value.
RVRLSM	- Copies an application list in the reverse order.
SETRULS	- Sets delete flags according to user's dependence and strength rules.
SORTDLST	- Gives an application list of entities to be deleted, DEL_LST returns a system list sorted in user_constituent order in SRT_LST.
SORTLSM	- Sorts a system list.
SRTBYCNT	- Creates an application list of inclusive constituents in constituent-user order.
UPDCRBE	- Updates an entry in the CRB.
VERAPN	- Verifies legality of appending an entity or list of entities (KEY2) to an entity or list of entities (KEY1).
VERCN	- Verifies legality of connecting each entity on a list of users to each entity on a list of constituents.
VERCR	- Verifies legality of creating an entity with the user-supplied entity data block and list of constituents.
VERDEL	- Verifies legality of deleting an entity.
VERGT	- Verifies legality of retrieving an entity with the user-supplied entity key.
VERUD	- Verifies legality of updating an entity with the user-supplied entity key using the user-supplied entity data block and list of constituents.
XIEMM	- Deletes an entity.

3.10.2.2 Listings

```
(* %INCLUDE ADCRBM *)
(**)
  PROCEDURE ADCRBM(VAR CRB:CRBPNTN; CONST EKEY:ENTKEY;
    CONST POS:LISTPSTN; CONST DIR:LISTDIR; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:  B. A. ULMER           FRMI   CREATED: 85/02/07  CC??*)
(*   VERSION: XXXX                   REVISED: YY/MM/DD  CC  *)
(*
(*   FUNCTION:
(*     ADD A NEW CRB ENTRY
(*
(*   ENVIRONMENT:
(*     IBM PASCAL LANGUAGE
(*     IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(*   EXECUTION PROCEDURE:
(*     HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(*   DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     CRB        I/O  CONSTITUENT READ BLOCK ADDRESS
(*     EKEY        I   ENTITY CONTAINING THE CONSTITUENT LIST BEING
(*                     READ
(*     POS         I   LIST POSITION SETTING
(*     DIR         I   DIRECTION TO READ THE LIST (FORWARD OR REVERSE)
(*     RR          0   ERROR CONDITION RETURN CODE
(*                     = 0  OK RETURN CODE
(*                     = 1  YOU BLEW IT
(*                     = 2  THE ROUTINE BLEW IT
(*
(*   COMMONS:
(*     COM1
(*       VAR1      I   VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*                     MUST BE PROVIDED
(*       VAR2      I   VAR2 MUST BE SPECIFIED
(*     COM2
(*       VAR3      I   CHARACTER DATA MUST BE SPECIFIED
(*
(*   PROCESSING DESCRIPTION:
(*     DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*     FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
```

```
(*      COMMENTS:                                     *)
(*      TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*      THE FUNCTION/EXECUTION OF THIS ROUTINE.         *)
(*                                                     *)
(*      CHANGE CONTROL:                                 *)
(*      YY/MM/DD  CCZZ  I. M. THECHANGER                *)
(*      DESCRIPTION OF LATEST CHANGE MADE.              *)
(*      YY/MM/DD  CCYY  I. M. THEPROGRAMMER            *)
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE *)
(*      NARRATION ON THE NEXT LINE.                    *)
(*      YY/MM/DD  CCXX  I. M. APERSON                  *)
(*      DESCRIPTION OF FIRST CHANGE MADE.              *)
(*      -----*)
(**)
(* END %INCLUDE ADCRBM *)
```

```
(* %INCLUDE ADRLSM. *)
(**)
  PROCEDURE ADRLSM(CONST INCREMENT:LISTSIZE;CONST POSITION:LISTPSTN;
    CONST KEYE:ENTKEY;VAR LISTREF:LISTPNTR;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   ADD AN ENTITY AFTER A RELATIVE POSITION IN A SYSTEM LIST.
(*   A POSITION OF ZERO INDICATES THE TOP OF THE LIST. IF THE
(*   LIST REQUIRES EXPANSION TO HOLD THE NEW ENTITY, IT IS
(*   EXPANDED BY INCREMENT ENTRIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   INCREMENT  I   NUMBER OF ENTITIES BY WHICH A LIST IS
(*                   EXPANDED AT A TIME
(*   POSITION    I   RELATIVE POSITION AFTER WHICH THE NEW
(*                   ENTRY IS ADDED
(*   KEYE        I   ENTITY TO BE ADDED.
(*   LISTREF     I   POINTER TO THE SYSTEM LIST TO WHICH KEYE
(*                   WAS ADDED
(*   RC          O   EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   = 1  YOU BLEW IT
(*                   = 2  THE ROUTINE BLEW IT
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*   ADDS KEYE TO THE SYSTEM LIST LISTREF AT POSITION AFTER THE
(*   THE RELATIVE POSITION GIVEN
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 12/30/85          B. A. ULMER          FRMI

```

```
(*      ADD PROCESSING FOR LARGE LISTS                                *)
(*)
(*)      REVISED: 12/24/84      R. A. MCCLUSKEY      FRMI      *)
(*)      ADDED SYSTEM LIST CURRENT LENGTH INDICATOR -- LSTLNM      *)
(*)
(*)      REVISED: 10/01/84      E. D. SHREVE      FRMI      *)
(*)      CORRECT THE MOVE OF ENTRIES FROM OLD TO NEW LIST WHEN OLD LIST *)
(*)      MUST BE EXPANDED      *)
(*)      *)
```

```
(* %INCLUDE ADSCH. *)
(**)
PROCEDURE ADSCH(CONST KEYE:ENTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(* CONNECT AN INTERNAL ITEM TO THE CORRECT PORTION OF THE
(* NDS SUPERSTRUCTURE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ===
(* SCH_ROOT I KEY OF THE SCHEMA_ROOT TO WHICH THE
(* INTERNAL ITEM WILL BE ATTACHED
(* KEYE I KEY OF THE INTERNAL ITEM TO BE ATTACHED
(* RR O EXTERNAL RETURN CODE
(* =0 OK
(* >0 CRITICAL ERROR
(* <0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(* DESCRIPTION OF HOW THIS ROUTINE WORKS (INTERNAL ACTIONS)
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* REVISED: 06/19/86 B. A. ULMER FRMI
(* CHANGE CALLING PARAMETERS TO CRURUL - NEW DELETE RULES
(*
(* REVISED: 09/09/85 B. A. ULMER FRMI
(* ADD TWO NEW PARAMETERS TO FNDURUL
(*
(* REVISED: 02/18/85 B. A. ULMER FRMI
(* CHANGED STRUCTURE OF THE INTERNAL ITEM FOT IMPLEMENTATION OF
(* THE CRB
```



```
(*)
(*) REVISIED: 10/04/84          E. D. SHREVE          FRMI      *)
(*) TO CHANGE LIST INCREMENT WHEN ADDING TO THE INSTANCE COLLECTOR *)
(*) CONSTITUENT LIST                                     *)
(*)
(*) REVISIED: 05/14/84          E. D. SHREVE          FRMI      *)
(*) TO RESET THE SCH_INST 'KIND' TO 'SCH_INST' AFTER THE ENTITY  *)
(*) KIND IS PUT INTO THE STANDARD ARRAY OF THE SCHEMA_ROOT      *)
(*)
```

```
(* %INCLUDE ADSCHR. *)
(**)
  PROCEDURE ADSCHR(CONST KEYE:ENTKEY;VAR POSITION:LISTPSTN;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*  FUNCTION                                          *)
(*    CONNECT AN INTERNAL ITEM TO THE SCHEMA ROOT.  *)
(*                                          *)
(*  LANGUAGE                                          *)
(*    PASCAL.                                          *)
(*                                          *)
(*  PACKAGE                                          *)
(*    SCHEMA PACKAGE.                                *)
(*                                          *)
(*  ARGUMENTS                                          *)
(*    INPUT                                          *)
(*      KEYE      - KEY OF THE INTERNAL ITEM TO BE ATTACHED. *)
(*    OUTPUT                                          *)
(*      POSITION   - RELATIVE POSITION OF THIS SCHEMA INSTANCE *)
(*                  OR CLASS ENTITY IN THE SCHEMA ROOT'S *)
(*                  CONSTITUENT LIST.                  *)
(*      RR        - THE FUNCTION RETURN RECORD.        *)
(*-----*)
(**)
(* END %INCLUDE ADSCHR. *)
```

```
(* %INCLUDE ADTLSM. *)
(**)
  PROCEDURE ADTLSM(CONST INCREMENT:LISTSIZE;CONST KEYE:ENTKEY;
    VAR LISTREF:LISTPNTR;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      ADD AN ENTITY TO A SYSTEM LIST. IF LISTREF IS NIL, THEN
(*      THE LIST IS EMPTY. IF NO ROOM IS AVAILABLE, THEN THE LIST
(*      IS EXPANDED BY INCREMENT ENTITIES.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      INCREMENT - THE NUMBER OF ENTITIES BY WHICH A LIST IS
(*      EXPANDED AT A TIME.
(*      KEYE      - KEY OF THE ENTITY TO BE ADDED.
(*      LISTREF   - A POINTER TO A SYSTEM LIST.
(*      OUTPUT
(*      LISTREF   - POINTER TO THE SYSTEM LIST TO WHICH KEYE
(*      WAS ADDED.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE ADTLSM. *)
```

```

(*) %INCLUDE ADTNM *)
(**)
  PROCEDURE ADTNM(CONST KEYE:ENTKEY;VAR KEYL:LISTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*)   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC  (*)
(*)   VERSION: MAS VER 2           REVISED: 84/10/11 CC  (*)
(*)
(*)   FUNCTION:
(*)       ADD AN ENTITY TO THE END OF AN APPLICATION LIST.
(*)
(*)   ENVIRONMENT:
(*)       IBM PASCAL LANGUAGE
(*)       IBM 30XX, 43XX, DEC VAX 11/780
(*)
(*)   DESCRIPTION OF ARGUMENTS:
(*)       NAME      I/O  DESCRIPTION
(*)       KEYE      I    KEY OF ENTITY TO BE ADDED.
(*)       KEYL      I    KEY OF THE APPLICATION LIST TO WHICH THE
(*)                   ENTITY IS ADDED.
(*)       KEYL      O    THE KEY OF THE LIST WITH THE ENTITY ADDED TO
(*)                   THE END.
(*)       RR        O    ERROR CONDITION RETURN CODE.
(*)                   = 0  NORMAL RETURN CODE.
(*)
(*)   COMMONS:
(*)
(*)   PROCESSING DESCRIPTION:
(*)
(*)   COMMENTS:
(*)
(*)   CHANGE CONTROL:
(*)       84/10/11  MAS VER 2  D. J. KERCHNER
(*)       UPDATED DOCUMENTATION.
(*)       84/10/04  MAS VER 2  E. D. SHREVE
(*)       CHANGED DECLARATION OF KEYL TO VAR.
(*)
(*-----*)
(**)
(*) END %INCLUDE ADTNM *)

```

```
(* %INCLUDE BIGCREMM. *)
(**)
  PROCEDURE BIGCREMM(CONST KEYEU:ENTKEY;CONST KEYEC:ENTKEY;
    CONST NUM:INTEGER;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A USER-CONSTITUENT RELATION BETWEEN ENTITIES
(*      WITH A CONSTITUENT LIST OF A GIVEN SIZE.
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      ENTITY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYEU      - KEY OF ENTITY TO BE THE USER.
(*      KEYEC      - KEY OF ENTITY TO BE THE CONSTITUENT.
(*      NUM        - LENGTH OF THE CONSTITUENT LIST.
(*      OUTPUT
(*      RR         - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE BIGCREMM. *)
```

```

(*) %INCLUDE CHKDEL *)
(**)
  PROCEDURE CHKDEL(CONST KEYE:ENTKEY; VAR TEMP_DEL_LIST:LISTPNTR;
    VAR MARK_LIST:LISTKEY; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)   CHECK DELETABILITY OF A GIVEN ENTITY BASED ON THE RELATION- (*)
(*)   SHIP BETWEEN ITS USERS AND ITSELF (*)
(*) (*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ====          ==  ===== (*)
(*)   KEYE           I    ENTITY WHOSE DELETABILITY IS TO BE (*)
(*)                   CHECKED (*)
(*)   TEMP_DEL_LIST I/O  LIST WHICH CONTAINS ENTITIES THAT ARE (*)
(*)                   ELIGIBLE FOR DELETE (*)
(*)   MARK_LIST     I/O  LIST WHICH CONTAINS ENTITIES THAT ARE (*)
(*)                   MARKED (*)
(*)   RR            O    RETURN CODE (*)
(*) (*)
(*) $COMMONS: (*)
(*) (*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*) (*)
(*) $EXECUTION PROCEDURE: (*)
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE (*)
(*) (*)
(*) $PROCESSING DESCRIPTION: (*)
(*) (*)
(*) $COMMENTS: (*)
(*) (*)
(*) $CHANGE CONTROL: (*)
(*) (*)

```

```

(*) %INCLUDE CHKTDEL *)
(**)
  PROCEDURE CHKTDEL(CONST KEYE:ENTKEY; VAR MARK_LIST:LISTKEY;
    VAR TEMP_DEL_LIST:LISTKEY; VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) CHECK DELETABILITY OF A GIVEN ENTITY BASED ON THE RELATION- *)
(*) SHIP BETWEEN ITS USERS AND ITSELF *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ==== === ===== *)
(*) KEYE I ENTITY WHOSE DELETABILITY IS TO BE *)
(*) CHECKED *)
(*) MARK_LIST I/O LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*) TO BE MARKED BY MAED, MAEDI *)
(*) TEMP_DEL_LIST I/O LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*) ELIGIBLE FOR DELETE BY MAED, MAEDI *)
(*) RR O RETURN CODE *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)

```

```

(*) %INCLUDE CMPCRB *)
(**)
  PROCEDURE CMPCRB(VAR CRB:CRBPNT; VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*)  AUTHOR:  B. A. ULMER          FRMI    CREATED: 85/02/08  CC??*)
(*)  VERSION: XXXX                  REVISED: YY/MM/DD  CC  *)
(*)
(*)  FUNCTION:
(*)    COMPREE THE CRB
(*)
(*)  ENVIRONMENT:
(*)    IBM PASCAL LANGUAGE
(*)    IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*)
(*)  EXECUTION PROCEDURE:
(*)    HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*)
(*)  DESCRIPTION OF ARGUMENTS:
(*)    NAME      I/O  DESCRIPTION
(*)    CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*)    RR        0    ERROR CONDITION RETURN CODE
(*)                = 0  OK RETURN CODE
(*)                = 1  YOU BLEW IT
(*)                = 2  THE ROUTINE BLEW IT
(*)
(*)  COMMONS:
(*)    COM1
(*)      VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*)                MUST BE PROVIDED
(*)      VAR2      I    VAR2 MUST BE SPECIFIED
(*)    COM2
(*)      VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*)
(*)  PROCESSING DESCRIPTION:
(*)    DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*)    FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*)
(*)  COMMENTS:
(*)    TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*)    THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
(*)  CHANGE CONTROL:
(*)    YY/MM/DD  CCZZ  I. M. THECHANGER
(*)    DESCRIPTION OF LATEST CHANGE MADE.
(*)    YY/MM/DD  CCYY  I. M. THEPROGRAMMER
(*)

```


CI PS560240032U
April 1990

```
(*          DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE  *)
(*          NARRATION ON THE NEXT LINE.                             *)
(*          YY/MM/DD  CCXX  I. M. APERSON                            *)
(*          DESCRIPTION OF FIRST CHANGE MADE.                        *)
(*          -----*)
(**)
(* END %INCLUDE CMPCRB *)
```

```
(* %INCLUDE CNNODM. *)
(**)
  PROCEDURE CNNODM(CONST KEYEU:ENTKEY;CONST KEYEC:ENTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*      FUNCTION                                          *)
(*      CONNECT TWO ENTITIES.                          *)
(*                                          *)
(*      LANGUAGE                                          *)
(*      PASCAL.                                          *)
(*                                          *)
(*      PACKAGE                                          *)
(*      ENTITY PACKAGE.                                *)
(*                                          *)
(*      ARGUMENTS                                        *)
(*      INPUT                                           *)
(*      KEYEU      - THE KEY OF THE ENTITY TO BE THE USER. *)
(*      KEYEC      - THE KEY OF THE ENTITY TO BE THE CONSTITUENT. *)
(*      OUTPUT                                           *)
(*      RR         - THE FUNCTION RETURN CODE.          *)
(*                                          *)
(*-----*)
(**)
(* END %INCLUDE CNNODM. *)
```

```
(* %INCLUDE CNNODMN. *)
(**)
  PROCEDURE CNNODMN(CONST KEYEU:ENTKEY;CONST KEYEC:ENTKEY;
                    VAR NUM:INTEGER;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*  FUNCTION                                          *)
(*    CONNECT TWO ENTITIES AND CREATE A CONSTITUENT LIST OF A  *)
(*    GIVEN SIZE                                          *)
(*                                          *)
(*  LANGUAGE                                          *)
(*    PASCAL.                                          *)
(*                                          *)
(*  PACKAGE                                          *)
(*    ENTITY PACKAGE.                                  *)
(*                                          *)
(*  ARGUMENTS                                          *)
(*    INPUT                                          *)
(*      KEYEU      - THE KEY OF THE ENTITY TO BE THE USER.  *)
(*      KEYEC      - THE KEY OF THE ENTITY TO BE THE CONSTITUENT. *)
(*    OUTPUT                                          *)
(*      RR         - THE FUNCTION RETURN CODE.              *)
(*-----*)
(**)
(* END %INCLUDE CNNODMN. *)
```

```
(* %INCLUDE CNVOSP. *)
(**)
  PROCEDURE CNVOSP(VAR RR:RET_REC;CONST ID:INTEGER;
    CONST THIS_ROUTINE:PGMNAME; VAR RC:EXT_RET_CODE);EXTERNAL;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    CONVERT THE OUT OF CORE SPACE CONDITION TO A APPLICATION
(*    USER RECOGNIZEABLE FORM
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ====          ==  =====
(*    RR            I    RETURN RECORD TO BE CONVERTED
(*    ID            I    INTEGER ID OF THE MAS INTERFACE ROUTINE
(*                    THAT ISSUED THE RETURN CODE
(*    THIS_ROUTINE  I    CHARACTER REPRESENTATION OF THE INTERFACE
(*                    ROUTINE THAT ISSUED THE RETURN CODE
(*    RC            O    EXTERNAL RETURN CODE
(*                    = 0  OK
(*                    > 0  CRITICAL ERROR
(*                    < 0  WARNING
(*
(*  $COMMONS:
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(*  $PROCESSING DESCRIPTION:
(*    CONVERTS THE OUT OF CORE SPACE CONDITION TO APPLICATION USER*)
(*    RECOGNIZEABLE FORM
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
```

```
(* %INCLUDE CNVRR. *)
(**)
PROCEDURE CNVRR(CONST RR:RET_REC;CONST PGM_ID:INTEGER;
CONST PGM_NAME: PGMNAME; VAR RC:EXT_RET_CODE);EXTERNAL;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* GET THE EXTERNAL RETURN CODE CORRESPONDING TO THE INTERNAL *)
(* FORMAT. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* RR I RETURN RECORD TO BE CONVERTED *)
(* PGM_ID I INTEGER ID OF THE MAS INTERFACE ROUTINE *)
(* THAT ISSUED THE RETURN CODE *)
(* RC O EXTERNAL RETURN CODE *)
(* = 0 OK *)
(* > 0 CRITICAL ERROR *)
(* < 0 WARNING *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* CONVERTS THE INTERNAL RETURN CODE TO EXTERNAL RETURN CODE *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
(* REVISED: 85/07/11 B. A. ULMER FRMI *)
(* CHANGED TO ADD ERROR MESSAGE AND PROGRAM NAME TO MSTATUS COMMON*)
(* WHEN AN INTERFACE GETS A NONE ZERO RETURN CODE *)
(* *)
```

```
(* %INCLUDE CPYAUDB *)
(**)
  PROCEDURE CPYAUDB(VAR ENTBPNTNTR:ENTPNTR;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC
(*   VERSION:  MAS VER 2         REVISED: 84/10/11 CC
(*
(*   FUNCTION:
(*       STORE THE VALUE OF AN APPLICATION ENTITY BLOCK IN AN
(*       UNINITIALIZED SYSTEM UDB.
(*
(*   ENVIRONMENT:
(*       IBM PASCAL LANGUAGE
(*       IBM 30XX, 43XX, DEC VAX 11/780
(*
(*   DESCRIPTION OF ARGUMENTS:
(*       NAME      I/O  DESCRIPTION
(*       ENTDEF    I    ENTBLOCK CONTAINING THE VALUES TO STORE.
(*       ENTBPNTNTR O    POINTER TO ENTBLOCK CREATED.
(*       RR        O    ERROR CONDITION RETURN CODE.
(*                      = 0  NORMAL RETURN CODE.
(*
(*   COMMONS:
(*
(*   PROCESSING DESCRIPTION:
(*       CPYAUDB USES AMPXMOVE A SYSTEM ROUTINE. AMPXMOVE MOVES
(*       DATA FROM MEMORY TO MEMORY (THE NUMBER OF BYTES TO MOVE
(*       MUST BE SPECIFIED).
(*
(*   COMMENTS:
(*
(*   CHANGE CONTROL:
(*       84/10/11  MAS VER 2  D. J. KERCHNER
(*       UPDATED DOCUMENTATION.
(*       84/10/04  MAS VER 2  E. D. SHREVE
(*       CHANGED DECLARATION OF ENTDEF TO VAR.
(*-----*)
(**)
(* END %INCLUDE CPYAUDB *)
```

```

(*) %INCLUDE CPYCST. *)
(**)
  PROCEDURE CPYCST(CONST SCH_KEY   : ENTKEY;
                   VAR  KEY1      : LISTKEY;
                   VAR  LIST LENG  : LISTSIZE;
                   VAR  RR        : RET_REC); EXTERNAL;

(**)
(*-----*)
(*)
(*) FUNCTION
(*)   ADD THE ENTITIES IN A CONSTITUENT LIST INTO A LIST.
(*)
(*) LANGUAGE
(*)   PASCAL
(*)
(*) PACKAGE
(*)   LIST PACKAGE.
(*)
(*) ARGUMENTS
(*)   INPUT
(*)     SCH-KEY  - KEY OF A CLASS OR ENTITY COLLECTOR.
(*)     KEY1    - KEY OF THE LIST ONTO WHICH THE ENTITIES
(*)               WILL BE ADDED.
(*)
(*)   OUTPUT
(*)     LIST LENG - TOTAL LENGTH OF ALL CNST ADDED TO LIST.
(*)     RR       - THE FUNCTION RETURN RECORD.
(*)
(*) METHOD
(*)   IF SCH_KEY IS AN ENTITY COLLECTOR, THEN ALL CONSTITUENTS
(*)   ARE ADDED TO THE LIST. IF SCH_KEY IS A CLASS COLLECTOR,
(*)   'CPYCST' IS CALLED RECURSIVELY TO PROCESS THE ENTITY
(*)   COLLECTORS THAT ARE CONSTITUENTS OF SCH_KEY. LIST LENG IS
(*)   ACCUMULATED FOR ALL RECURSIONS.
(*)-----*)
(**)
(*) END %INCLUDE CPYCST *)

```

```
(* %INCLUDE CPYLSM. *)
(**)
  PROCEDURE CPYLSM(CONST LISTFROM : LISTPNTR;
                   VAR   POSITION : LISTPSTN;
                   VAR   LISTTO  : LISTPNTR;
                   VAR   RR      : RET_REC);  EXTERNAL;
(**)
(*-----*)
(*                                           *)
(* FUNCTION                                           *)
(*   COPY THE NON-VACANT ELEMENTS OF LISTFROM TO LISTTO.  IF   *)
(*   LISTTO WAS INITIALIZED, IT IS DELETED PRIOR TO COPYING. *)
(*   POSITION IS SET TO THE BEGINNING OF LISTTO.  CURRENT LENGTH *)
(*   OF LISTTO IS SET TO CURRENT LENGTH OF LISTFROM.         *)
(*                                           *)
(* LANGUAGE                                           *)
(*   PASCAL.                                           *)
(*                                           *)
(* PACKAGE                                           *)
(*   LIST PACKAGE.                                     *)
(*                                           *)
(* ARGUMENTS                                           *)
(*   INPUT                                           *)
(*     LISTFROM - POINTER TO SYSTEM LIST TO BE COPIED.      *)
(*   OUTPUT                                           *)
(*     LISTTO   - POINTER TO SYSTEM LIST TO WHICH COPY IS MADE. *)
(*     POSITION  - SET TO INDICATE THE BEGINNING OF LISTTO.   *)
(*     RR      - THE FUNCTION RETURN RECORD.                 *)
(*                                           *)
(*-----*)
(**)
(* END %INCLUDE CPYLSM. *)
```



```
(* %INCLUDE CPYNM. *)
(**)
  PROCEDURE CPYNM(CONST KEYL:LISTKEY;VAR KEYLOUT:LISTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A NEW LIST WHICH CONTAINS A COPY OF THE ENTITIES
(*      REFERENCED BY KEYL.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYL      - KEY OF THE LIST TO BE COPIED.
(*      OUTPUT
(*      KEYLOUT   - KEY OF THE NEW LIST WHICH IS A COPY OF THE
(*                  INPUT LIST.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE CPYNM. *)
```

```
(* %INCLUDE CRCLST. *)
(**)
  PROCEDURE CRCLST(CONST KEYE:ENTKEY;CONST LISTREF:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE RELATIONS BETWEEN A USER ENTITY AND A LIST OF
(*      CONSTITUENTS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF THE USER ENTITY OF THE RELATIONS.
(*      LISTREF   - POINTER TO SYSTEM LIST OF CONSTITUENTS.
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE CRCLST. *)
```

```
(* %INCLUDE CRCLSTN. *)
(**)
  PROCEDURE CRCLSTN(CONST KEYE:ENTKEY;CONST LISTREF:LISTPNTR;
    VAR NUM:INTEGER; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(* *)
(* FUNCTION *)
(* CREATE RELATIONS BETWEEN A USER ENTITY AND A LIST OF *)
(* CONSTITUENTS. *)
(* *)
(* LANGUAGE *)
(* PASCAL. *)
(* *)
(* PACKAGE *)
(* LIST PACKAGE. *)
(* *)
(* ARGUMENTS *)
(* INPUT *)
(* KEYE - KEY OF THE USER ENTITY OF THE RELATIONS. *)
(* LISTREF - POINTER TO SYSTEM LIST OF CONSTITUENTS. *)
(* NUM - THE LENGTH OF THE CONSTITUENT LIST. *)
(* OUTPUT *)
(* RR - THE FUNCTION RETURN RECORD. *)
(* *)
(*-----*)
(**)
(* END %INCLUDE CRCLSTN. *)
```

```
(* %INCLUDE CRCNM. *)
(**)
  PROCEDURE CRCNM(CONST KEYE:ENTKEY;CONST KEYL:LISTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE RELATIONS BETWEEN A USER ENTITY AND A LIST OF
(*      CONSTITUENTS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF THE USER ENTITY OF THE RELATIONS.
(*      KEYL      - KEY OF LIST OF CONSTITUENT ENTITIES.
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE CRCNM. *)
```

```

(*) %INCLUDE CRDLST *)
(**)
  PROCEDURE CRDLST(CONST KEY1:ANYKEY;VAR CNSTS_SRTLST:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   CREATE A SORTED INCLUSIVE LIST OF AN ENTITY OR A LIST OF
(*)   ENTITIES AND THEIR DIRECTAND INDIRECT CONSTITUENTS.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KEY1           I    AN ENTITY OR LIST OF ENTITIES TO BE PUT
(*)                   ON A LIST WITH THEIR CONSTITUENTS.
(*)   CNSTS_SRTLST   O    AN INCLUSIVE LIST OF AN ENTITY OR LIST
(*)                   AND THIER DIRECT AND INDIRECT CNSTS. IN
(*)                   USER-CONSTITUENT ORDER.
(*)   RC             O    EXTERNAL RETURN CODE
(*)                   = 0  OK RETURN CODE
(*)                   > 0  CRITICAL ERROR
(*)                   < 0  WARNING
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   IF KEY1 IS AN ENTKEY THEN
(*)     AN INCLUSIVE LIST OF THE ENTITIES CONSTITUENTS IS BUILT
(*)     INCLUDING KEY1.
(*)   IF KEY1 IS A LIST THEN
(*)     A LIST OF THE INCLUSIVE CONSTITUENTS OF THE ENTITIES ON
(*)     KEY1 IS CREATED, INCLUDING THE ENTITIES ON KEY1.
(*)
(*) $COMMENTS:
(*)   THE OUTPUT LIST IS SORTED IN USER-CONSTITUENT ORDER.
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(*  REVISED: 04/26/85          E. D. SHREVE          W315      *)
(*    CHANGED TO USE THE INTER PROCESS FLAG (MAPROB).      *)
(*                                                              *)
(*  REVISED: 02/18/85          B. A. ULMER           W315      *)
(*    CHANGED TO IMPLEMENT THE CONST. READ BLOCK.          *)
(*                                                              *)
(*  REVISED: 11/01/84          E. D. SHREVE          W315      *)
(*    REMOVE CALL TO DISPLSM                                *)
(*                                                              *)
(*  ORIGINATED: 08/23/84        C. J. SAMPLE          W315      *)
(*                                                              *)
(*-----*)
(*END-----*)
(* END %INCLUDE PROG_ID *)
```

```
(* %INCLUDE CREMM. *)
(**)
  PROCEDURE CREMM(CONST KEYEU:ENTKEY;CONST KEYEC:ENTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*  FUNCTION
(*    CREATE A USER-CONSTITUENT RELATION BETWEEN ENTITIES.
(*
(*  LANGUAGE
(*    PASCAL.
(*
(*  PACKAGE
(*    ENTITY PACKAGE.
(*
(*  ARGUMENTS
(*    INPUT
(*      KEYEU      - KEY OF ENTITY TO BE THE USER.
(*      KEYEC      - KEY OF ENTITY TO BE THE CONSTITUENT.
(*    OUTPUT
(*      RR         - THE FUNCTION RETURN RECORD.
(*-----*)
(**)
(* END %INCLUDE CREMM. *)
```

```
(* %INCLUDE CRURUL. *)
(**)
PROCEDURE CRURUL(CONST ENTITY_TYPE:ORD_KIND;VAR GROUP:T_GROUP_ARRAY;
VAR NUM_GROUP:LISTPSTN; VAR MIN_CNST:LISTPSTN);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     CREATES THE USER'S RULES.  RULES OF CONNECTIVITY USED TO
(*     DETERMINE DELETABILITY OF ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ===  =====
(*     ENTITY_TYPE    I    ENTITY KIND VALUE WHICH WILL HAVE THE
(*                       DELETE RULE
(*     GROUP          0    ARRAY THAT WILL BE FILLED WITH THE RULES
(*                       AND NUMBER OF CONSTITUENTS OF EACH
(*                       DIFFERENT RELATIONSHIP THAT THIS ENTITY
(*                       KIND CAN HAVE WITH ITS CONSTITUENTS
(*     NUM_GROUP      0    NUMBER OF DIFFERENT RELATIONSHIPS THIS
(*                       ENTITY CAN HAVE WITH ITS CONSTITUENTS
(*     MIN_CNST       0    MINIMUM NUMBER OF CONSTITUENTS THAT THIS
(*                       ENTITY CAN HAVE WHEN IT HAS A GROUP OF
(*                       CONSTITUENTS THAT ARE "SECONDARY"
(*     RC             0    EXTERNAL RETURN CODE
(*                       = 0  OK RETURN CODE
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*     ?????ARE SET TO INDICATE IF THE RELATIONSHIP BETWEEN THE
(*     USER AND ITS CONSTITUTES IS DEPENDENT OR INDEPENDENT AND
(*     STRONG OR WEAK.
(*     DEFAULT RULE IS DEPENDENT/STRONG.
(*
(* $COMMENTS:
(*
```



```
(* $CHANGE CONTROL: *)
(*) *)
(*) REVISED: 06/19/86 B. A. ULMER FRMI *)
(*) REDO LOGIC OF HOW CRURUL WORKS BASED ON THE NEW DELETE RULES *)
(*) *)
(*) REVISED: 09/ /85 B. A. ULMER FRMI *)
(*) ADD ENTITY KINDS SO AS TO TEST THE NEW DELETE RULES (2070, *)
(*) 2080, 2090) *)
(*) *)
(*) REVISED: 09/ /85 B. A. ULMER FRMI *)
(*) ADD PARAMETERS TO HANDLE THE TWO NEW DELETE RULES *)
(*) *)
(*) REVISED: 09/18/84 D. J. KERCHNER FRMI *)
(*) ADDED I/S RULE FOR THE PICK ENTITY *)
(*) *)
```

```
(* %INCLUDE DELCNST *)
(**)
  PROCEDURE DELCNST(CONST KEYE:ENTKEY; VAR TEMP_DEL_LIST:LISTPNTR;
    VAR MARK_LIST:LISTKEY; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   DETERMINES THE DELETABILITY OF GIVEN ENTITY'S CONSTITUENTS *)
(*   BASED ON THE RELATIONSHIP THE CONSTITUENT HAS WITH ITS USERS*)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME          I/O  DESCRIPTION *)
(*   ===          ==  ===== *)
(*   KEYE          I    ENTITY WHOSE CONSTITUENTS WILL HAVE THEIR*)
(*                   DELETABILITY DETERMINED *)
(*   TEMP_DEL_LIST I/O  LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*                   ELIGIBLE FOR DELETE *)
(*   MARK_LIST     I/O  LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*                   MARKED *)
(*   RR           O    RETURN CODE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```
(* %INCLUDE DELCRBE *)
(**)
  PROCEDURE DELCRBE(VAR CRB:CRBPNTN; CONST EKEY:ENTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:  B. A. ULMER          FRMI   CREATED: 85/02/08  CC??*)
(*   VERSION: XXXX                  REVISED: YY/MM/DD  CC  *)
(*
(*   FUNCTION:
(*       DELETE A CRB ENTRY
(*
(*   ENVIRONMENT:
(*       IBM PASCAL LANGUAGE
(*       IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(*   EXECUTION PROCEDURE:
(*       HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(*   DESCRIPTION OF ARGUMENTS:
(*       NAME      I/O  DESCRIPTION
(*       CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*       EKEY      I    ENTITY KEY OF ENTRY TO BE DELETED
(*       RR        O    ERROR CONDITION RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   = 1  YOU BLEW IT
(*                   = 2  THE ROUTINE BLEW IT
(*
(*   COMMONS:
(*       COM1
(*       VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*                   MUST BE PROVIDED
(*       VAR2      I    VAR2 MUST BE SPECIFIED
(*       COM2
(*       VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*
(*   PROCESSING DESCRIPTION:
(*       DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*       FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(*   COMMENTS:
(*       TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*       THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
(*   CHANGE CONTROL:
(*       YY/MM/DD  CCZZ  I. M. THECHANGER
(*)
```

```
(*      DESCRIPTION OF LATEST CHANGE MADE.      *)
(*      YY/MM/DD CCYY I. M. THEPROGRAMMER      *)
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE  *)
(*      NARRATION ON THE NEXT LINE.      *)
(*      YY/MM/DD CCXX I. M. APERSON      *)
(*      DESCRIPTION OF FIRST CHANGE MADE.      *)
(*      -----*)
(**)
(* END %INCLUDE DELCRBE *)
```

```
(* %INCLUDE DELEMM. *)
(**)
  PROCEDURE DELEMM(VAR KEYE:ENTKEY;VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*  FUNCTION                                          *)
(*    DELETE ALL REFERENCES TO THIS ENTITY FROM ALL APPLICATION *)
(*    LISTS AND DISPOSE OF THE ENTITY.  TO COMPLETE DELETE ACTION *)
(*    REQUIRES BREAKING ALL USER AND CONSTITUENT CONNECTIONS. *)
(*                                          *)
(*  LANGUAGE                                          *)
(*    PASCAL.                                          *)
(*                                          *)
(*  PACKAGE                                          *)
(*    ENTITY PACKAGE.                                          *)
(*                                          *)
(*  ARGUMENTS                                          *)
(*    INPUT                                          *)
(*      KEYE      - KEY OF THE ENTITY TO BE DELETED. *)
(*    OUTPUT                                          *)
(*      RR        - THE FUNCTION RETURN RECORD. *)
(*                                          *)
(*  METHOD                                          *)
(*    AN ENTRY IN AN APPLICATION LIST HAS A FORM OF INT_ITEM. *)
(*    ALL REFERENCES TO IT WILL BE DELETED. THE USER WILL NEVER *)
(*    DIRECTLY DELETE ENTITIES OF FORM INT_ROOT. THESE ARE ONLY *)
(*    DELETED AS A RESULT OF THE CLEANUP ASSOCIATED WITH THE *)
(*    DELETION OF AN NDS. *)
(*-----*)
(**)
(* END %INCLUDE DELEMM. *)
```

```

(*) %INCLUDE DELPLST. *)
(**)
  PROCEDURE DELPLST(CONST INCREMENT:LISTSIZE;CONST IPOS:LISTINDX;
    VAR POSITION:LISTPSTN;VAR LISTREF:LISTPNTR;VAR RR:RET_REC);
    EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) REMOVE AN ENTITY FROM A SPECIFIED POSITION IN A SYSTEM LIST *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) = == === ===== *)
(*) INCREMEN I NUMBER OF ENTITIES BY WHICH SYSTEM LIST *)
(*) LIST IS EXPANDED OR REDUCED *)
(*) IPOS I POSITION IN THE LIST FROM WHICH THE *)
(*) ENTITY WILL BE REMOVED *)
(*) POSITION I/O LAST LOCATION ON THE SYSTEM LIST THAT WAS*)
(*) PROCESSED *)
(*) LISTREF I POINTER TO SYSTEM LIST FROM WHICH ENTITY *)
(*) WILL BE REMOVED *)
(*) RC O EXTERNAL RETURN CODE *)
(*) = 0 OK RETURN CODE *)
(*) = 1 YOU BLEW IT *)
(*) = 2 THE ROUTINE BLEW IT *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*) SHIFT ALL FOLLOWING ENTITIES UP UNTIL ALL VACANT ENTITIES *)
(*) ARE AT THE END OF THE LIST. RECALCULATE THE POSITION IF *)
(*) IT WAS AFFECTED BY THIS REMOVAL. IF MORE THAN INCREMENT *)
(*) ENTITIES ARE VACANT, THEN COMPRESS THE LIST BY REMOVING *)
(*) THE INCREMENT ENTITIES. *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)

```

(*	REVISED: 12/30/85	B. A. ULMER	FRMI	*)
(*	ADD PROCESSING FOR LARGE LISTS			*)
(*				*)
(*	REVISED: 02/06/85	E. D. SHREVE	FRMI	*)
(*	TEST FOR NIL POINTER			*)
(*				*)
(*	REVISED: 12/24/84	R. A. MCCLUSKEY	FRMI	*)
(*	ADDED SYSTEM LIST CURRENT LENGTH INDICATOR -- LSTLNM			*)
(*				*)

```

(*) %INCLUDE DELPNLA. *)
(**)
  PROCEDURE DELPNLA(VAR POSITION:LISTPSTN; VAR LISTA:LISTPNTR;
                    VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*
(*  $FUNCTION
(*    DELETE ALL APPL LISTS AFTER A SPECIFIED POSITION IN THE
(*    LIST_OF_LISTS EXCEPT THOSE THAT ARE 'LOCKED'.
(*
(*  $DESCRIPTION OF ARGUMENTS
(*
(*    NAME          I/O    DESCRIPTION
(*    ====          ===    =====
(*    POSITION        I      POSITION IN LISTA TO START DELETE.
(*    LISTA          I      LIST_OF_LISTS SYSTEM LIST
(*    RR             0      RETURN CODE
(*                          =0  GOOD RETURN
(*                          >0  CRITICAL ERROR
(*                          <0  WARNING
(*
(*  $COMMONS
(*    NONE
(*
(*  $ENVIRONMENT:
(*    LANGUAGE:  IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE OF THE MODEL ACCESS SOFTWARE
(*
(*  $PROCESSING DESCRIPTION:
(*    STARTING WITH THE INPUT POSITION, EACH APPL LIST ON THE
(*    INPUT LIST_OF_LISTS (LISTA) IS PROCESSED.  IF THE LIST
(*    IS 'LOCKED' (DELTF LG = NODEL), THE LISTKEY IS PLACED ON A
(*    TEMPORARY LIST; ELSE, THE LIST IS DELETED.  AFTER ALL
(*    ENTRIES ARE PROCESSED, THE TEMPORARY LIST IS MERGED WITH
(*    ANY ENTRIES STILL REMAINING ON LISTA.
(*
(*  $CHANGE CONTROL:
(*
(*    ORIGINATED:  04/23/85    E. D. SHREVE    W315
(*-----*)
(**)
(*END %INCLUDE DELPNLA. *)

```



```
(* %INCLUDE DELRLSM. *)
(**)
PROCEDURE DELRLSM(CONST INCREMENT:LISTSIZE;CONST KEYE:ENTKEY;
  VAR POSITION:LISTPSTN;VAR LISTREF:LISTPNTR;VAR RR:RET_REC);
EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   REMOVE AN ENTITY FROM A SYSTEM LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   INCREMENT      I   NUMBER OF ENTITIES BY WHICH A SYSTEM LIST*)
(*                   IS EXPANDED OR REDUCED
(*   KEYE           I   KEY OF THE ENTITY TO BE REMOVED FROM THE *)
(*                   LIST
(*   POSITION        I/O  LOCATION ON THE SYSTEM LIST OF ENITY
(*                   TO BE PROCESSED -- UPDATED LOCATION OF
(*                   ENTITY ORIGINALLY INDICATED BY POSITION
(*   LISTREF        I   POINTER TO SYSTEM LIST FROM WHICH ENTITY *)
(*                   WILL BE REMOVED
(*   RC             0   EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   = 1  YOU BLEW IT
(*                   = 2  THE ROUTINE BLEW IT
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*   SHIFT ALL FOLLOWING ENTITIES UP UNTIL ALL VACANT ENTITIES
(*   ARE AT THE END OF THE LIST. RECALCULATE THE POSITION IF
(*   IT WAS AFFECTED BY THIS REMOVAL. IF MORE THAN INCREMENT
(*   ENTITIES ARE VACANT, THEN COMPRESS THE LIST BY REMOVING
(*   THE INCREMENT ENTITIES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

CI PS560240032U
April 1990

(*				*)
(*	REVISED: 12/30/85	B. A. ULMER	FRMI	*)
(*	ADD PROCESSING FOR LARGE LISTS			*)
(*				*)
(*	REVISED: 12/24/84	R. A. MCCLUSKEY	FRMI	*)
(*	ADDED SYSTEM LIST CURRENT LIST INDICATOR -- LSTLNM			*)
(*				*)

```
(* %INCLUDE DELRUL *)
(**)
  PROCEDURE DELRUL(VAR KEYE:ENTKEY;VAR DEL_LIST:LISTPNTR;VAR
    MARK_LIST:LISTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DELETE AN ENTITY ACCORDING TO THE DELETE RULES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   KEYE       I   ENTITY TO BE DELETED OR MARKED FOR
(*                 DELETION
(*   DEL_LST    I   LIST OF KEYS THAT ARE ELIGIBLE FOR
(*                 DELETION
(*   MARK_LIST  O   LIST OF ENTITIES WHICH HAVE BEEN MARKED
(*   RC         O   EXTERNAL RETURN CODE
(*                 = 0  OK RETURN CODE
(*                 > 0  CRITICAL ERROR
(*                 < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*   ??????? TY'S USER LIST IS READ AND THE DELETE RULES FOR
(*   EACH USER ARE CHECKED TO DETERMINE IF ENTITY CAN BE DELETED.
(*   IF UNABLE TO DELETE THE ENTITY THEN CHECK IF THE USER IS ON
(*   THE DELETE LIST.  IF ON THE LIST THEN DELETE THE ENTITY ELSE
(*   MARK IT FOR DELETE.  IF UNABLE TO MARK FOR DELETE THEN ADD
(*   ENTITY TO THE EXCEPTION LIST.
(*
(* $COMMENTS:
(*   THE DELETE RULES ARE STORED IN THE INSTANCE COLLECTOR OF AN
(*   ENTITY'S USER AS DEPENDENCE AND STRENGTH FLAGS.  DEPENDENCE
(*   IS DEFINED AS DEPENDENT (TRUE) OR INDEPENDENT (FALSE).
(*   STRENGTH IS DEFINED AS DEPENDENT (TRUE) OR INDEPENDENT
(*   (FALSE).
(*   IF THERE EXISTS A DEPENDENT/STRONG USER CONNECTION, THEN
```

```
(*      THE ENTITY MAY NOT BE DELETED.      *)
(*      IF THERE EXISTS A DEPENDENT/WEAK USER CONNECTION, BUT NO      *)
(*      DEPENDENT/STRONG CONNECTION THEN THE ENTITY IS MARKED FOR      *)
(*      DELETION AND IF ANY OF ITS USER CONNECTIONS WERE                *)
(*      INDEPENDENT/WEAK, THEN IT IS DISCONNECTED FROM THOSE          *)
(*      INDEPENDENT/WEAK USER CONNECTIONS.                             *)
(*      IF THERE ARE NO DEPENDENT/STRONG NOR DEPENDENT/WEAK            *)
(*      USER CONNECTIONS OR NO USERS AT ALL, THEN THE ENTITY IS        *)
(*      DELETED AND ITS CONSTITUENTS ARE PROCESSED THE SAME AS         *)
(*      THE ENTITY WAS.                                                 *)
(*      $CHANGE CONTROL:                                               *)
(*      REVISED: 09/02/86      B. A. ULMER      DBMA      *)
(*      REMOVE DUPLICATE ENTITIES FROM DELETE LIST - CAUSES A PROBLEM *)
(*      WHEN AN ENTITY HAS THE SAME CNST TWICE                          *)
(*      REVISED: 06/19/86      B. A. ULMER      FRMI      *)
(*      MAJOR REWRITE DUE TO NEW DELETE RULES                          *)
(*      REVISED: 12/17/85      B. A. ULMER      FRMI      *)
(*      FIX PROBLEM WITH CODE FOR NEW DELETE RULES                     *)
(*      REVISED: 09/05/85      B. A. ULMER      FRMI      *)
(*      ADD CODE TO HANDLE THE TWO NEW DELETE RULES                    *)
(*      ORIGINATED: 06/15/84    C. J. SAMPLE      FRMI      *)
(*      -----*)
(*      %PAGE      *)
(*      -----*)
(*      DATA STRUCTURES/MAJOR VARIABLES:      *)
(*      -----*)
(*      *END-----*)
(**)
(* END %INCLUDE DELRUL. *)
```

```
(* %INCLUDE DELSCH. *)
(**)
  PROCEDURE DELSCH(CONST KEYE:ENTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      DISCONNECT AN INTERNAL ITEM FROM THE CORRECT PORTION OF
(*      THE NDS SUPERSTRUCTURE.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      SCHEMA PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF THE INTERNAL ITEM TO BE DETACHED.
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL
(*      CHANGED: 12/14/84  E. SHREVE - TO DELETE THE INSTANCE
(*      COLLECTOR IF ALL IT'S CNSTS ARE DELETED.
(*-----*)
(**)
(* END %INCLUDE DELSCH. *)
```

```

(*) %INCLUDE DELTSLM. *)
(**)
  PROCEDURE DELTSLM(CONST INCREMENT:LISTSIZE;VAR LISTREF:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   REMOVES THE LAST NON-VACANT ENTITY REFERENCE IN A LIST. *)
(*)   IF THIS REMOVAL PRODUCES MORE THAN INCREMENT VACANT *)
(*)   ENTITIES AT THE BOTTOM OF THE LIST, THEN THE VACANT *)
(*)   ENTITIES ARE ELIMINATED. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   LISTREF       I    LIST WHOSE LAST ENTITY IS TO BE REMOVED *)
(*)   INCREMENT     I    MAXIMUM NUMBER OF VACANT ENTITIES THE *)
(*)                   LAST MIGHT CONTAIN *)
(*)   RC            0    EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   = 1  YOU BLEW IT *)
(*)                   = 2  THE ROUTINE BLEW IT *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 12/30/85          B. A. ULMER          FRMI *)
(*)   ADD PROCESSING FOR LARGE LISTS *)
(*) *)
(*)   REVISED: 12/24/84          R. A. MCCLUSKEY      FRMI *)
(*)   ADDED SYSTEM LIST CURRENT LENGTH INDICATOR -- LSTLNM *)
(*) *)

```

```
(* %INCLUDE DETCNST *)
(**)
  PROCEDURE DETCNST(CONST KEYE:ENTKEY; VAR MARK_LIST:LISTKEY;
    VAR TEMP_DEL_LIST:LISTKEY; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(*   DETERMINES THE DELETABILITY OF GIVEN ENTITY'S CONSTITUENTS *)
(*   BASED ON THE RELATIONSHIP THE CONSTITUENT HAS WITH ITS USERS*)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(*   NAME      I/O  DESCRIPTION *)
(*   ====      ==  =====*)
(*   KEYE       I   ENTITY WHOSE CONSTITUENTS WILL HAVE THEIR*)
(*               DELETABILITY DETERMINED *)
(*   MARK_LIST  I/O  LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*               MARKED *)
(*   TEMP_DEL_LIST I/O LIST WHICH CONTAINS ENTITIES THAT ARE *)
(*               ELIGIBLE FOR DELETE *)
(*   RR         O   RETURN CODE *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(*   LANGUAGE: IBM PASCAL *)
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

```

(*) %INCLUDE DETRUL *)
(**)
  PROCEDURE DETRUL(CONST KEYE:ENTKEY;VAR MARK_LIST:LISTKEY;
    VAR DEL_LIST:LISTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*) TEST DELETE OF AN ENTITY ACCORDING TO THE DELETE RULES.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*) ====
(*) KEYE I ENTITY TO TESTED FOR DELETION OR MARK FOR*)
(*) DELETION
(*) MLIST I/O LIST OF ENTITIES WHICH MAY BE MARKED
(*) DLIST I/O LIST OF ENTITIES WHICH MAY BE DELETED
(*) RC 0 EXTERNAL RETURN CODE
(*) = 0 OK RETURN CODE
(*) > 0 CRITICAL ERROR
(*) < 0 WARNING
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*) ????? TITY'S USERS LIST IS READ AND THE DELETE RULES FOR
(*) EACH USER ARE CHECKED TO DETERMINE IF THE ENTITY CAN BE
(*) DELETED.
(*) THE DELETE RULES ARE STORED IN THE INSTANCE COLLECTOR OF AN
(*) ENTITY'S USER AS DEPENDENCE AND STRENGTH FLAGS. DEPENDENCE
(*) IS DEFINED AS DEPENDENT (TRUE) OR INDEPENDENT (FALSE).
(*) STRENGTH IS DEFINED AS DEPENDENT (TRUE) OR INDEPENDENT
(*) (FALSE).
(*) IF THERE EXISTS A DEPENDENT/STRONG USER CONNECTION, THEN
(*) THE ENTITY MAY NOT BE DELETED AND IT IS ADDED TO THE
(*) EXCEPTION LIST.
(*) IF THERE EXISTS A DEPENDENT/WEAK USER CONNECTION, BUT NO
(*) DEPENDENT/STRONG CONNECTION THEN THE ENTITY CAN BE
(*) MARKED FOR DELETION AND ADDED TO THE MARK LIST.
(*) IF THERE ARE NO DEPENDENT/STRONG USER CONNECTIONS,

```



```

(*)          NO DEPENDENT/WEAK USER CONNECTIONS,          *)
(*)          NO USERS AT ALL, OR                             *)
(*)          ALL USERS ARE ON THE DELETE LIST,              *)
(*)      THEN                                                *)
(*)          THE ENTITY IS DELETABLE AND ADDED TO THE DELETE LIST. *)
(*)          IF THE ENTITY IS MARKED FOR DELETION OR         *)
(*)              IS ON THE MARK LIST,                        *)
(*)      THEN                                                *)
(*)          ITS CONSTITUENTS ARE PROCESSED THE SAME AS THE ENTITY. *)
(*)
(*) $COMMENTS:                                              *)
(*)
(*) $CHANGE CONTROL:                                         *)
(*)
(*)  REVISED: 06/19/86          B. A. ULMER                FRMI *)
(*)    MAJOR REWRITE DUE TO THE NEW DELETE RULES             *)
(*)
(*)  REVISED: 04/18/86          E. D. SHREVE                FRMI *)
(*)    TO SET DELETE RULES ONLY WHEN USER IS NOT IN DELIST  *)
(*)
(*)  REVISED: 09/06/85          B. A. ULMER                FRMI *)
(*)    ADDED CODE TO HANDLE THE TWO NEW DELETE RULES         *)
(*)
(*)  ORIGINATED: 06/28/84        C. J. SAMPLE                FRMI *)
(*)
(*)-----*)
(*)%PAGE                                                    *)
(*)-----*)
(*)  DATA STRUCTURES/MAJOR VARIABLES:                      *)
(*)-----*)
(*)
(*)
(*)END-----*)
(**)
(*) END %INCLUDE DETRUL. *)

```

```
(* %INCLUDE DIFLSM. *)
(**)
  PROCEDURE DIFLSM(CONST LIST1:LISTPNTR;CONST LIST2:LISTPNTR;
    VAR POSITION:LISTPSTN;VAR LISTOUT:LISTPNTR;VAR RR:RET_REC);
    EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A SYSTEMS LIST CONSISTING OF ALL ENTITIES IN LIST1
(*      THAT ARE NOT IN LIST2.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LIST1, LIST2 - THE LISTS WHOSE DIFFERENCE IS TO BE FOUND.
(*      OUTPUT
(*      LISTOUT - LIST CONTAINING THE DIFFERENCE OF THE
(*      TWO LISTS.
(*      POSITION - INTEGER INDICATING BEGINNING OF LISTOUT.
(*      RR - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE DIFLSM. *)
```

```
(* %INCLUDE DISPCRB *)
(**)
PROCEDURE DISPCRB(VAR CRB:CRBPNT; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*)
(*)  AUTHOR:  B. A. ULMER          FRMI    CREATED: 85/02/08  CC??*)
(*)  VERSION: XXXX                REVISED: YY/MM/DD  CC  *)
(*)
(*)  FUNCTION:
(*)    DISPOSE OF CRB
(*)
(*)
(*)  ENVIRONMENT:
(*)    IBM PASCAL LANGUAGE
(*)    IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*)
(*)  EXECUTION PROCEDURE:
(*)    HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*)
(*)  DESCRIPTION OF ARGUMENTS:
(*)    NAME      I/O  DESCRIPTION
(*)    CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*)    RR        0    ERROR CONDITION RETURN CODE
(*)                = 0  OK RETURN CODE
(*)                = 1  YOU BLEW IT
(*)                = 2  THE ROUTINE BLEW IT
(*)
(*)  COMMONS:
(*)    COM1
(*)      VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*)                MUST BE PROVIDED
(*)      VAR2      I    VAR2 MUST BE SPECIFIED
(*)    COM2
(*)      VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*)
(*)  PROCESSING DESCRIPTION:
(*)    DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*)    FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*)
(*)  COMMENTS:
(*)    TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*)    THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
(*)  CHANGE CONTROL:
(*)    YY/MM/DD  CCZZ  I. M. THECHANGER
(*)    DESCRIPTION OF LATEST CHANGE MADE.
(*)    YY/MM/DD  CCYY  I. M. THEPROGRAMMER
(*)
```

CI PS560240032U
April 1990

```
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE  *)
(*      NARRATION ON THE NEXT LINE.                               *)
(*      YY/MM/DD  CCXX  I. M. APERSON                             *)
(*      DESCRIPTION OF FIRST CHANGE MADE.                         *)
(*-----*)
(**)
(* END %INCLUDE DISPCRB *)
```

```
(* %INCLUDE DISPEMM. *)
(**)
  PROCEDURE DISPEMM(VAR KEYE:ENTKEY;VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      RELEASE ALL SPACE ALLOCATED TO AN ENTITY. NO DANGLING
(*      REFERENCES TO THIS ENTITY SHOULD EXIST IN AN NDS OR
(*      Nodelist.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      ENTITY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF THE ENTITY TO BE DISPOSED.
(*      OUTPUT
(*      KEYE      - SET TO NIL.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL
(*      CHANGED:  12/10/84  J. JOHNSON  - TO CALL 'MASDSP'
(*-----*)
(**)
(* END %INCLUDE DISPEMM. *)
```

```
(* %INCLUDE DISPLSM. *)
(**)
  PROCEDURE DISPLSM(VAR POSITION:LISTPSTN;VAR LISTREF:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      DELETE SPACE ALLOCATED TO A SYSTEM LIST.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LISTREF    - POINTER TO A SYSTEM LIST WHOSE SPACE IS TO
(*                  BE DEALLOCATED.
(*      OUTPUT
(*      LISTREF    - POINTER TO A SYSTEM LIST WITH ZERO SIZE.
(*      POSITION    - POSITION IS SET TO ZERO INDICATING START OF
(*                  SYSTEM LIST.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL:
(*      CHANGED:  12/10/84  J. JOHNSON  - CALL MASDSP.
(*-----*)
(**)
(* END %INCLUDE DISPLSM. *)
```

```
(* %INCLUDE DISPNM. *)
(**)
  PROCEDURE DISPNM(VAR KEYL:LISTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*      FUNCTION                                          *)
(*      REMOVE ALL ENTITIES FROM THE LIST AND FREE THE ALLOCATED *)
(*      SPACE. THE EMPTY LIST IS ALSO DELETED AND REMOVED FROM THE *)
(*      LIST OF LISTS.                                          *)
(*                                          *)
(*      LANGUAGE                                          *)
(*      PASCAL.                                          *)
(*                                          *)
(*      PACKAGE                                          *)
(*      LIST PACKAGE.                                          *)
(*                                          *)
(*      ARGUMENTS                                          *)
(*      INPUT                                          *)
(*      KEYL      - KEY OF THE LIST WHOSE ENTITIES ARE TO *)
(*                  BE REMOVED.                                          *)
(*      OUTPUT                                          *)
(*      RR      - THE FUNCTION RETURN RECORD.                                          *)
(*                                          *)
(*      METHOD                                          *)
(*      THE STACK_OF_LISTS IS READ.  FOR EACH LIST_OF_LISTS ON THE *)
(*      STACK_OF_LISTS, KEYL IS REMOVED FROM THE LIST.  WHEN ALL *)
(*      LISTS HAVE BEEN SEARCHED, KEYL IS DISPOSED.                                          *)
(*-----*)
(**)
(* END %INCLUDE DISPNM. *)
```

```
(* %INCLUDE ELDNM. *)
(**)
PROCEDURE ELDNM(VAR KEYL:LISTKEY;VAR RR:RET_REC) EXTERNAL;
(**)
(*-----*) 66
(* 76
(* $FUNCTION: 1
(* CREATE A LIST WITH ALL DUPLICATE ENTITIES ELIMINATED. 95
(* THE FIRST REFERENCE IS MAINTAINED AND ALL SUBSEQUENT
(* ENTITIES ARE REMOVED.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* KEYL I - KEY OF THE LIST WHICH MAY CONTAIN DUPLICATE
(* ENTITIES. THE LIST WILL HAVE ALL DUPLICATES
(* REMOVED.
(* RR O - THE FUNCTION RETURN RECORD.
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE: IBM 360/370/43XX
(*
(* $PROCESSING DESCRIPTION:
(* EACH ENTRY ON THE SYSTEM LIST IS READ. THE ADB.PROBIT IS
(* SET ON, AND THE ENTITY KEY IS PLACED ON THE NEW SYSTEM LIST.
(* IF THE ADB.PROBIT IS ALREADY SET ON, THEN THE ENTITY IS A
(* DUPLICATE AND NOT PLACED ON THE NEW LIST.
(* THE NEW LIST REPLACES THE OLD SYSTEM LIST IN THE APPL-
(* ICATION LIST. ALL PROBITS ARE THEN RESET TO 'OFF'.
(*
(* CHANGE CONTROL:
(* REVISED: 09/02/86 B. A. ULMER W315
(* TO USE MAS INTERNAL PROCESS FLAG (MAPROB2) INSTEAD OF
(* MAPROB (CONFLICT WITH DELRUL)
(*
(* REVISED: 04/26/85 E. D. SHREVE W315
(* TO USE MAS INTERNAL PROCESS FLAG (MAPROB)
(*
(* REVISED: 02/07/85 E. D. SHREVE W315
(* REWRITTEN TO PROCESS MORE EFFICIENTLY.
(*-----*)
(**)
(* END %INCLUDE ELDNM. *)
```



```
(* %INCLUDE ELMNODM. *)
(**)
  PROCEDURE ELMNODM(CONST KEYE:ENTKEY;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      RETURN AN ENTBLOCK CORRESPONDING TO A KEY.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      ENTITY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - THE KEY OF THE ENTITY.
(*      OUTPUT
(*      ENTDEF    - THE CORRESPONDING ENTBLOCK.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE ELMNODM. *)
```

```

(*) %INCLUDE EXCRBE *)
(**)
  PROCEDURE EXCRBE(CONST CRB:CRBPNTN; CONST POS1:RDBSIZE;
    CONST POS2:RDBSIZE; VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*)  AUTHOR:  B. A. ULMER          FRMI    CREATED: 85/02/08  CC??*)
(*)  VERSION: XXXX                      REVISED: YY/MM/DD  CC  *)
(*)
(*)  FUNCTION:
(*)    EXCHANGE TWO ENTRIES IN THE CRB
(*)
(*)  ENVIRONMENT:
(*)    IBM PASCAL LANGUAGE
(*)    IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*)
(*)  EXECUTION PROCEDURE:
(*)    HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*)
(*)  DESCRIPTION OF ARGUMENTS:
(*)    NAME      I/O  DESCRIPTION
(*)    CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*)    POS1      I    POSITION OF FIRST ENTRY TO EXCHANGE
(*)    POS2      I    POSITION OF SECOND ENTRY TO EXCHANGE
(*)    RR        O    ERROR CONDITION RETURN CODE
(*)                  = 0  OK RETURN CODE
(*)                  = 1  YOU BLEW IT
(*)                  = 2  THE ROUTINE BLEW IT
(*)
(*)  COMMONS:
(*)    COM1
(*)      VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*)                  MUST BE PROVIDED
(*)      VAR2      I    VAR2 MUST BE SPECIFIED
(*)    COM2
(*)      VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*)
(*)  PROCESSING DESCRIPTION:
(*)    DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*)    FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*)
(*)  COMMENTS:
(*)    TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*)    THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
(*)  CHANGE CONTROL:
(*)

```

```
(*      YY/MM/DD  CCZZ  I. M. THECHANGER      *)
(*      DESCRIPTION OF LATEST CHANGE MADE.      *)
(*      YY/MM/DD  CCYY  I. M. THEPROGRAMMER     *)
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE *)
(*      NARRATION ON THE NEXT LINE.             *)
(*      YY/MM/DD  CCXX  I. M. APERSON           *)
(*      DESCRIPTION OF FIRST CHANGE MADE.        *)
(*      -----*)
(**)
(* END %INCLUDE EXCRBE *)
```

```

(*) %INCLUDE EXPCLSM. *)
(**)
  PROCEDURE EXPCLSM(CONST LISTIN:LISTPNTR;VAR LISTOUT:LISTPNTR;
    VAR LSTFLG:BOOLEAN; VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   EXPAND LIST WITH ALL OF ITS CONSTITUENTS AND PLACE THIS
(*)   EXPANDED LIST IN LISTOUT.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME      I/O      DESCRIPTION
(*)   ====      ==      =====
(*)   LISTIN     I        LIST CONTAINING ENTITIES TO BE
(*)                   EXPANDED.
(*)   LISTOUT    O        LIST OF INCLUSIVE CONSTITUENTS
(*)   LSTFLG     I        FLAG TO TELL IF FIRST TIME THRU
(*)   RR         O        FUNCTION RETURN CODE.
(*)                   = 0 GOOD RETURN
(*)                   > 0 CRITICAL ERROR
(*)                   < 0 WARNING
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360,370,43XX
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THIS ROUTINE INVOKES ITSELF RECURSIVELY AND FILLS LISTOUT
(*)   BY ADDING EACH NEST OF CONSTITUENTS DIRECTLY AFTER THE
(*)   PARENT ENTITY.
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 01/10/86   B. A. ULMER           W315
(*)                   FIX BUG DEALING WITH PREVIOUS FIX
(*)
(*)   REVISED: 05/21/85   B. A. ULMER           W315
(*)                   FIX INCONSISTENCY IN OUTPUT LIST PROCESSING
(*)
(*)   REVISED: 04/26/85   E.D. SHREVE          W315
(*)                   TO USE INTERNAL MAS PROCESS FLAG MAPROB
(*)

```

CI PS560240032U
April 1990

```
(*
(*      REVISED: 02/18/85    B.A. ULMER                W315      *)
(*      IMPLEMENT THE CNST READ BLOCK                    *)
(*
(*      CREATED: 06/13/84    D.J. KERCHNER              W315      *)
(*-----*)
(**)
(* END %INCLUDE EXPCLSM. *)
```

```
(* %INCLUDE EXPCRB *)
(**)
PROCEDURE EXPCRB(VAR CRB:CRBPNT; VAR RR:RET_REC);EXTERNAL;
(**)
-----*)
(*)
(*)
(*)  AUTHOR:  B. A. ULMER          FRMI    CREATED: 85/02/08  CC??*)
(*)  VERSION: XXXX                REVISED: YY/MM/DD   CC  *)
(*)
(*)  FUNCTION:
(*)    EXPAND THE CRB
(*)
(*)  ENVIRONMENT:
(*)    IBM PASCAL LANGUAGE
(*)    IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*)
(*)  EXECUTION PROCEDURE:
(*)    HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*)
(*)  DESCRIPTION OF ARGUMENTS:
(*)    NAME      I/O  DESCRIPTION
(*)    CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*)    RR        0    ERROR CONDITION RETURN CODE
(*)              = 0  OK RETURN CODE
(*)              = 1  YOU BLEW IT
(*)              = 2  THE ROUTINE BLEW IT
(*)
(*)  COMMONS:
(*)    COM1
(*)      VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*)                MUST BE PROVIDED
(*)      VAR2      I    VAR2 MUST BE SPECIFIED
(*)    COM2
(*)      VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*)
(*)  PROCESSING DESCRIPTION:
(*)    DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*)    FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*)
(*)  COMMENTS:
(*)    TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*)    THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
(*)  CHANGE CONTROL:
(*)    YY/MM/DD  CCZZ  I. M. THECHANGER
(*)    DESCRIPTION OF LATEST CHANGE MADE.
(*)    YY/MM/DD  CCYY  I. M. THEPROGRAMMER
(*)
```

CI PS560240032U
April 1990

```
(*          DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE  *)
(*          NARRATION ON THE NEXT LINE.                             *)
(*          YY/MM/DD  CCXX  I. M. APERSON                            *)
(*          DESCRIPTION OF FIRST CHANGE MADE.                        *)
(*          -----*)
(**)
(* END %INCLUDE EXPCRB *)
```

```
(* %INCLUDE EXPSUDB *)
(**)
  PROCEDURE EXPSUDB(VAR ENTBPNTR:ENTPNTR;CONST OLDSIZE:ENTSIZE;
    CONST NEWSIZE:ENTSIZE;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   EXPAND A SYSTEM UDB (USER DATA BLOCK)
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   OLDSIZE        I    SIZE OF THE AREA TO BE EXPANDED
(*   NEWSIZE        I    SIZE OF THE OUTPUT DATA AREA FOR THE
(*                       EXPANDED ENTBLOCK
(*   ENTBPNTR       I    POINTER TO THE ENTBLOCK TO BE EXPANDED
(*   ENTBPNTR       O    POINTER TO THE EXPANDED ENTBLOCK
(*   RC             O    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*   EXPAND THE USER DATA BLOCK (UDB)
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 07/09/85          B. A. ULMER          FRMI
(*   CHANGE TO MAKE THIS ROUTINE MORE VAX COMPATIBLE - TAKE OUT THE
(*   MIN FUNCTION
(*
(*   REVISED: 12/10/84          J. JOHNSON
(*   TO CALL MASDSP
(*
```



```

(*) %INCLUDE EXPULSM. *)
(**)
  PROCEDURE EXPULSM(CONST LISTIN:LISTPNTR; VAR LISTOUT:LISTPNTR;
    VAR LSTFLG:BOOLEAN; VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   PLACE THE EXPANDED LIST WITH ALL OF ITS USERS IN LISTOUT. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O      DESCRIPTION *)
(*)   ====      ==      ===== *)
(*)   LISTIN     I        LIST TO BE EXPANDED. *)
(*)   LISTOUT     O        EXPANDED LIST. *)
(*)   LSTFLG      I        FLAG TO TELL IF FIRST TIME THRU *)
(*)   RR           O        FUNCTION RETURN RECORD. *)
(*)                               = 0   GOOD RETURN *)
(*)                               > 0   CRITICAL ERROR *)
(*)                               < 0   WARNING *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE:  IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360,370,43XX *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE. *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   THIS ROUTINE INVOKES ITSELF RECURSIVELY AND FILLS LISTOUT *)
(*)   BY ADDING EACH NEST OF USERS DIRECTLY AFTER ITS USER *)
(*)   REFERENCE. *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 01/10/86   B. A. ULMER           W315 *)
(*)               FIX BUG DEALING WITH PREVIOUS FIX *)
(*) *)
(*)   REVISED: 05/21/85   B. A. ULMER           W315 *)
(*)               FIX INCONSISTENCY IN OUTPUT LIST PROCESSING *)
(*) *)
(*)   REVISED: 04/26/85   E. D. SHREVE         W315 *)
(*)               TO USE INTERNAL MAS PROCESS FLAG MAPROB *)
(*) *)

```

CI PS560240032U
April 1990

(* ORIGINATED: 06/13/84 D. J. KERCHNER W315 *)
(* *)
(*-----*)
(**)
(* END %INCLUDE EXPULSM. *)

```

(*) %INCLUDE EXPULSMI.*)
(**)
  PROCEDURE EXPULSMI(CONST ENTITY:ENTKEY; VAR LISTOUT:LISTPNTR;
                    VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   PLACE THE EXPANDED LIST WITH ALL OF ITS USERS IN LISTOUT. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O      DESCRIPTION *)
(*)   ====      ==      ===== *)
(*)   LISTIN     I        LIST TO BE EXPANDED. *)
(*)   LISTOUT    O        EXPANDED LIST. *)
(*)   LSTFLG     I        FLAG TO TELL IF FIRST TIME THRU *)
(*)   RR         O        FUNCTION RETURN RECORD. *)
(*)                       = 0   GOOD RETURN *)
(*)                       > 0   CRITICAL ERROR *)
(*)                       < 0   WARNING *)
(*)
(*) $COMMONS: *)
(*)   NONE *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE:  IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360,370,43XX *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE. *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   THIS ROUTINE INVOKES ITSELF RECURSIVELY AND FILLS LISTOUT *)
(*)   BY ADDING EACH NEST OF USERS DIRECTLY AFTER ITS USER *)
(*)   REFERENCE. *)
(*)
(*) $CHANGE CONTROL: *)
(*)   CHANGED:      11/19/86      K. M. ROSS *)
(*)   REASON:       INCORRECT ORDER ON OUTPUT *)
(*)   CHANGE:       CHECK IF FIRST ELEMENT IN LIST ALREADY *)
(*)                  PROCESSED I.E. DELETED FROM USER LIST *)
(*)
(*)   CHANGED:      11/10/86      K. M. ROSS *)
(*)   REASON:       INCORRECT ORDER ON OUTPUT *)
(*)   CHANGE:       CHECK LISTIN LENGTH AFTER LOOP TO RE *)
(*)                  MOVE USERS ALREADY PROCESSED *)
(*)

```

```
(*  CHANGED:          11/06/86      K M ROSS                      *)
(*  REASON :          TOO SLOW                      *)
(*  CHANGE :          REDESIGN USER LIST CREATION, BY BREADTH *)
(*                      WISE INSTEAD OF LENGTHWISE      *)
(*                      *)
(*  REVISED: 01/10/86   B. A. ULMER              W315          *)
(*                      FIX BUG DEALING WITH PREVIOUS FIX      *)
(*                      *)
(*  REVISED: 05/21/85   B. A. ULMER              W315          *)
(*                      FIX INCONSISTENCY IN OUTPUT LIST PROCESSING *)
(*                      *)
(*  REVISED: 04/26/85   E. D. SHREVE            W315          *)
(*                      TO USE INTERNAL MAS PROCESS FLAG MAPROB *)
(*                      *)
(*  ORIGINATED: 06/13/84 D. J. KERCHNER          W315          *)
(*                      *)
(*-----*)
(**)
(* END %INCLUDE EXPULSMI. *)
```

```

(** %INCLUDE FDSCH. *)
(**)
PROCEDURE FDSCH(CONST SCH_ROOT:ENTKEY;CONST KIND:ORD_KIND;
  VAR SCH_PTR:ENTKEY;VAR POSITION:LISTPSTN;
  VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*
(* FUNCTION
(* FIND A SCHEMA_INSTANCE_COLLECTOR OR SCHEMA_CLASS ENTITY ON
(* THE SPECIFIED SCHEMA_ROOT'S CONSTITUENT LIST.
(*
(* LANGUAGE
(* PASCAL.
(*
(* PACKAGE
(* SCHEMA PACKAGE.
(*
(* ARGUMENTS
(* INPUT
(* NDSREM - THE NETWORK TO BE SEARCHED.
(* KIND - VALUE TO BE SEARCHED FOR IN THE ENTBLOCK OF
(* THE CLASS OR INSTANCE COLLECTOR NODE. THIS
(* IS THE KIND OF THE COLLECTED INSTANCES FOR
(* INSTANCE COLLECTORS.
(*
(* OUTPUT
(* SCH_PTR - POINTER TO THE FOUND ENTITY WITH SPECIFIED
(* DATA.KIND.
(* POSITION - POSITION IN THE CONSTITUENT LIST OF THE LAST
(* SCHEMA CLASS OR INSTANCE COLLECTOR ENTITY
(* WITH HEADER.KIND LESS THAN OR EQUAL TO THE
(* SPECIFIED KIND.
(* RR - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE FDSCH. *)

```

```
(* %INCLUDE FNDCRBE *)
(**)
PROCEDURE FNDCRBE(CONST CRB:CRBPNTN; CONST EKEY:ENTKEY;
VAR CRBPOS:RDBSIZE;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* AUTHOR: B. A. ULMER FRMI CREATED: 85/02/08 CC??*)
(* VERSION: XXXX REVISED: YY/MM/DD CC *)
(*
(* FUNCTION:
(* FIND A SPECIFIC ENTRY IN THE CRB
(*
(* ENVIRONMENT:
(* IBM PASCAL LANGUAGE
(* IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(* EXECUTION PROCEDURE:
(* HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(* DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* CRB I/O CONSTITUENT READ BLOCK ADDRESS
(* EKEY I ENTITY KEY WHICH IS TO BE FOUND IN THE CRB
(* CRBPOS 0 POSITION IN CRB WHERE EKEY WAS FOUND
(* RR 0 ERROR CONDITION RETURN CODE
(* = 0 OK RETURN CODE
(* = 1 YOU BLEW IT
(* = 2 THE ROUTINE BLEW IT
(*
(* COMMONS:
(* COM1
(* VAR1 I VAR1 NAME MUST BE FILLED, CHARACTER DATA
(* MUST BE PROVIDED
(* VAR2 I VAR2 MUST BE SPECIFIED
(* COM2
(* VAR3 I CHARACTER DATA MUST BE SPECIFIED
(*
(* PROCESSING DESCRIPTION:
(* DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(* FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(* COMMENTS:
(* TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(* THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
```

```
(*  CHANGE CONTROL: *)
(*  YY/MM/DD  CCZZ  I. M. THECHANGER *)
(*  DESCRIPTION OF LATEST CHANGE MADE. *)
(*  YY/MM/DD  CCYY  I. M. THEPROGRAMMER *)
(*  DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE *)
(*  NARRATION ON THE NEXT LINE. *)
(*  YY/MM/DD  CCXX  I. M. APERSON *)
(*  DESCRIPTION OF FIRST CHANGE MADE. *)
(*  ----- *)
(**)
(* END %INCLUDE FNDCRBE *)
```

```
(* %INCLUDE FNDSKIND *)
(**)
  PROCEDURE FNDSKIND(CONST SCHKEY:ENTKEY;VAR KINDARY:KIND_ARRAY;
    VAR NUMKIND:INTEGER;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*  FUNCTION
(*    BUILD AN ARRAY OF KIND VALUE COLLECTED BY A CLASS OR
(*    INSTANCE COLLECTOR IN THE SCHEMA.
(*
(*  LANGUAGE
(*    PASCAL
(*
(*  PACKAGE
(*    LIST PACKAGE.
(*
(*  ARGUMENTS
(*    INPUT
(*      SCHKEY    - KEY OF THE CLASS OR INSTANCE COLLECTOR NODE.
(*      KINDARY   - ARRAY TO STORE THE COLLECTED KINDS.
(*    OUTPUT
(*      NUMKIND   - NUMBER OF KIND VALUES PUT INTO KINDARY.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*  METHOD
(*    1. IF SCHKEY IS AN INSTANCE COLLECTOR, THE KIND VALUE FROM
(*    THE 1ST CONSTITUENT'S ADB IS PUT INTO KINDARY.
(*    2. IF SCHKEY IS A CLASS COLLECTOR, ALL INCLUSIVE INSTANCE
(*    COLLECTORS ARE FOUND AND THEIR KINDS PUT IN KINDARY.
(*    THIS IS ACCOMPLISHED BY RECURSIVE CALLS TO FNDSKIND.
(*-----*)
(**)
(* END %INCLUDE FNDSKIND *)
```



```
(* %INCLUDE GTCRBE *)
(**)
PROCEDURE GTCRBE(CONST CRB:CRBPNTN; VAR CRBPOS: RDBSIZE;
CONST EKEY:ENTKEY; VAR POS:LISTPSTN; VAR DIR:LISTDIR;
VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* AUTHOR:  B. A. ULMER          FRMI    CREATED: 85/02/08  CC??*)
(* VERSION: XXXX                  REVISED: YY/MM/DD  CC  *)
(*
(* FUNCTION:
(*   GET AN ENTRY IN THE CRB
(*
(* ENVIRONMENT:
(*   IBM PASCAL LANGUAGE
(*   IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(* EXECUTION PROCEDURE:
(*   HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(* DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   CRB        I/O  CONSTITUENT READ BLOCK ADDRESS
(*   CRBPOS     I    POSITION IN CRB OF ENTRY REQUESTED
(*   EKEY       O    KEY OF ENTITY CONTAINING THE CONSTITUENT LIST
(*   POS        O    LIST POSITION SETTING
(*   DIR        O    DIRECTION TO READ OF LIST (FORWARD OR REVERSE)
(*   RR         O    ERROR CONDITION RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   = 1  YOU BLEW IT
(*                   = 2  THE ROUTINE BLEW IT
(*
(* COMMONS:
(*   COM1
(*     VAR1     I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*               MUST BE PROVIDED
(*     VAR2     I    VAR2 MUST BE SPECIFIED
(*   COM2
(*     VAR3     I    CHARACTER DATA MUST BE SPECIFIED
(*
(* PROCESSING DESCRIPTION:
(*   DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*   FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(* COMMENTS:
```

```
(*      TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(*      THE FUNCTION/EXECUTION OF THIS ROUTINE.                      *)
(*                                                                    *)
(*      CHANGE CONTROL:                                              *)
(*      YY/MM/DD  CCZZ  I. M. THECHANGER                            *)
(*      DESCRIPTION OF LATEST CHANGE MADE.                          *)
(*      YY/MM/DD  CCYY  I. M. THEPROGRAMMER                        *)
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE      *)
(*      NARRATION ON THE NEXT LINE.                                *)
(*      YY/MM/DD  CCXX  I. M. APERSON                              *)
(*      DESCRIPTION OF FIRST CHANGE MADE.                          *)
(*      -----*)
(**)
(* END %INCLUDE GTCRBE *)
```

```
(* %INCLUDE INDLSM *)
(**)
  PROCEDURE INDLSM(CONST KEYE:ENTKEY;CONST LISTREF:LISTPNTR;
    VAR POSITION:LISTPSTN;VAR INLST:BOOLEAN;VAR RR:RET_REC);
    EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC
(*   VERSION:  MAS VER 2           REVISED: 84/10/11 CC
(*
(*   FUNCTION:
(*     LOCATE AN ENTITY IN A SYSTEM LIST.
(*
(*   ENVIRONMENT:
(*     IBM PASCAL LANGUAGE
(*     IBM 30XX, 43XX, DEC VAX 11/780
(*
(*   DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     KEYE      I    ENTITY TO BE LOCATED.
(*     LISTREF   I    LIST TO BE SEARCHED.
(*     POSITION   0    POSITION OF ENTITY IN SYSTEM LIST.
(*     INLST     0    TRUE IF AN ENTITY IN THE LIST CORRESPONDS
(*                   TO KEYE ELSE FALSE.
(*     RR        0    ERROR CONDITION RETURN CODE.
(*                   = 0  NORMAL RETURN CODE.
(*
(*   COMMONS:
(*
(*   PROCESSING DESCRIPTION:
(*
(*   COMMENTS:
(*
(*   CHANGE CONTROL:
(*     84/10/11  MAS VER 2  D. J. KERCHNER
(*     UPDATED DOCUMENTATION.
(*     84/10/04  MAS VER 2  E. D. SHREVE
(*     CHANGED DECLARATION OF KEYL TO VAR.
(*-----*)
(**)
(* END %INCLUDE INDLSM *)
```

%INCLUDE PCMGT

DEF

\$PCMGR: T_\$PCMGR;

VALUE

\$PCMGR.SIZE :=32768;

\$PCMGR.PTR :=NIL;

\$PCMGR.OVERFLOW :=NIL;

\$PCMGR.INIT.SIZE:=32768;

\$PCMGR.MAP :=0;

(* *)
(* ADDED A VALUE STATEMENT FOR THE OVERFLOW BLOCK USAGE FLAG *)
(* *)

```
(* %INCLUDE INNM. *)
(**)
  FUNCTION INNM(CONST KEYE:ENTKEY;CONST KEYL:LISTKEY;
    VAR RR:RET_REC):BOOLEAN; EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      INDICATE WHETHER A LIST REFERENCES AN ENTITY.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY TO LOOK FOR IN THE LIST.
(*      KEYL      - THE KEY OF THE LIST TO EXAMINE.
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*      FUNCTION VALUE - TRUE IF ENTITY IS IN LIST ELSE FALSE.
(*
(*-----*)
(**)
(* END %INCLUDE INNM. *)
```

```
(* %INCLUDE INTLSM. *)
(**)
PROCEDURE INTLSM(CONST LIST1:LISTPNTR;CONST LIST2:LISTPNTR;
VAR POSITION:LISTPSTN;VAR LISTOUT:LISTPNTR;
VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(* CREATE A LIST WHICH IS THE INTERSECTION OF TWO LISTS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ====
(* LIST1 I LIST TO BE INTERSECTED WITH THE SECOND
(* LIST2 I LIST TO BE INTERSECTED WITH THE FIRST
(* POSITION I INTEGER INDICATING THE POSITION ON
(* LISTOUT LIST CONTAINING COMMON ENTITIES TO THE
(* INPUT LISTS
(* RC 0 EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(* FIND THOSE ENTITIES WHICH ARE COMMON TO BOTH INPUT LISTS
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* REVISED: 07/01/85 B. A. ULMER FRMI
(* ELIMINATE THE MIN FUNCTION TO IMPROVE COMPATABILITY WITH VAX
(*
(* REVISED: 02/22/85 B. A. ULMER FRMI
(* FIXED EMPTY LIST ELEMENT PROBLEM
```

CI PS560240032U
April 1990

(*				*)
(*	REVISED: 12/24/85	B. A. ULMER	FRMI	*)
(*	ADDED SYSTEM LIST CURRENT LENGTH INDICATOR --	LSTLNM		*)
(*				*)

```
(* %INCLUDE LSTLNM. *)
(**)
  FUNCTION LSTLNM(CONST LISTREF:LISTPNTR;VAR RR:RET_REC):LISTSIZE;
    EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*  FUNCTION                                          *)
(*    RETURN THE NUMBER OF NON-VACANT ENTITIES IN A SYSTEM LIST. *)
(*                                          *)
(*  LANGUAGE                                          *)
(*    PASCAL.                                          *)
(*                                          *)
(*  PACKAGE                                          *)
(*    LIST PACKAGE.                                          *)
(*                                          *)
(*  ARGUMENTS                                          *)
(*    INPUT                                          *)
(*      LISTREF    - POINTER TO A SYSTEM LIST.          *)
(*    OUTPUT                                          *)
(*      RR         - THE FUNCTION RETURN RECORD.        *)
(*      FUNCTION VALUE - NUMBER OF ENTITIES IN THE SYSTEM LIST. *)
(*-----*)
(**)
(* END %INCLUDE LSTLNM. *)
```



```
(* %INCLUDE LSTMXLNM. *)
(**)
  FUNCTION LSTMXLNM(CONST LISTREF:LISTPNTR;VAR RR:RET_REC):LISTSIZE;
    EXTERNAL;
(**)
(*-----*)
(*
(* FUNCTION
(* RETURN THE NUMBER OF ENTRIES ALLOCATED TO A SYSTEM LIST.
(*
(* LANGUAGE
(* PASCAL.
(*
(* PACKAGE
(* LIST PACKAGE.
(*
(* ARGUMENTS
(* INPUT
(* LISTREF - POINTER TO A SYSTEM LIST.
(* OUTPUT
(* FUNCTION VALUE - SIZE OF SYSTEM LIST.
(* RR - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE LSTMXLNM. *)
```

```

(*) %INCLUDE MABRST  *)
(**)
  PROCEDURE MABRST(VAR RC:EXT_RET_CODE);
    SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   RESET THE PROCESS AND APPLICATION FLAGS FOR ALL ENTITIES IN *)
(*)   THE WORKING FORM MODEL. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ===== *)
(*)   RC            0    EXTERNAL RETURN CODE *)
(*)                   = 0  OK *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING *)
(*) *)
(*) $COMMONS: *)
(*)   NDSREM *)
(*)   VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA *)
(*)                   MUST BE PROVIDED *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)   DDNAMES USED WITH STANDARD FILES: *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   ORIGINATED: 03/07/87      K. M. ROSS      DBMA *)
(*) *)
(*)-----*)
(*)   DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*) *)
(*)END-----*)
(*) END %INCLUDE MABRST  *)

```

```

(*) %INCLUDE MACPDT  *)
(**)
  PROCEDURE MACPDT(CONST KEY1:ANYKEY; CONST FLGNAME:NAMTYP; CONST
    FLGVAL:INTEGER; VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   UPDATE A SPECIFIED APPLICATION ACCESSIBLE FLAG VALUE *)
(*)   FOR THE CONSTITUENTS OF AN ENTITY OR A LIST OF ENTITIES *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ===          ===  ===== *)
(*)   KEY1          I    ENTITY OR LIST OF ENTITIES WHOSE *)
(*)                   SPECIFIED FLAG VALUE IS TO BE UPDATED *)
(*)   FLGNAME        I    FLAG NAME  (STRING(6)) *)
(*)   FLGVAL         I    VALUE TO BE USED WHEN UPDATING THE FLAG *)
(*)                   = 1 TRUE *)
(*)                   = 0 FALSE *)
(*)   RC             0    EXTERNAL RETURN CODE *)
(*)                   = 0 OK RETURN CODE *)
(*)                   < 0 WARNING *)
(*)                   > 0 CRITICAL ERROR *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   DETERMINE WHICH OF THE APPLICATION ACCESSIBLE FLAGS IS TO BE *)
(*)   UPDATED AND THEN UPDATE IT WITH THE INPUT VALUE *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)
(*)   ORIGINATED: 03/07/87      K. M. ROSS      DBMA *)
(*)
(*)-----*)
(*) END %INCLUDE MACPDT  *)

```

```
(* %INCLUDE MAEA. *)
(**)
  PROCEDURE MAEA(CONST KEY1:ANYKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
%PAGE
(**)
(*-----*)
(*
(* $FUNCTION:
(*   ACTIVATE AN ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   KEY1      I   KEY OF THE ENTITY OR LIST OF ENTITIES TO
(*               BE ACTIVATED
(*   RC        O   EXTERNAL RETURN CODE
(*               = 0  OK
(*               > 0  CRITICAL ERROR
(*               < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   FOR EACH KEY, AS AN ENTITY OR A MEMBER OF A LIST
(*   RESET THE DELETE FLAG
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 04/30/86      B. A. ULMER      FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85      B. A. ULMER      FRMI
(*   ADD NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
(*   REVISED: 08/14/86      K. M. ROSS      DBMA
(*
```

```
(*      ADDED A CHECK FOR NIL POINTER ON KEY1      *)
(*      PURPOSES                                     *)
(*      *)                                           *)
(*      ORIGINATED: 07/25/84      D. J. KERCHNER      FRMI      *)
(*      *)                                           *)
(*-----*)
%PAGE                                           *)
(*-----*)
(*      LATA STRUCTURES/MAJOR VARIABLES:           *)
(*-----*)
(*      *)                                           *)
(*END-----*)
(**)
(* END %INCLUDE MAEA. *)
(**)
```

April 1990

```

(*) %INCLUDE MAEAI *)
(**)
  PROCEDURE MAEAI(CONST KEY1:ANYKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   ACTIVATE AN ENTITY OR A LIST OF ENTITIES AND THEIR *)
(*)   INCLUSIVE CONSTITUENTS. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O      DESCRIPTION *)
(*)   ====      ==      ===== *)
(*)   KEY1      I       KEY OF THE ENTITY OR LIST OF ENTITIES TO *)
(*)                   BE ACTIVATED. *)
(*)   RC        O       THE FUNCTION RETURN CODE. *)
(*)                   = 0   GOOD RETURN *)
(*)                   > 0   CRITICAL ERROR *)
(*)                   < 0   WARNING *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360,370,43XX *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   IF KEY1 IS AN ENTITY, THEN THAT ENTITY AND ITS INCLUSIVE *)
(*)   CONSTITUENT LIST WILL BE ACTIVATED. *)
(*)   IF KEY1 IS A LIST KEY, THEN THE INCLUSIVE CONSTITUENT LISTS *)
(*)   OF EACH ENTITY WILL BE ACTIVATED. *)
(*)   NOW USES THE INTERNAL MAS PROCESS FLAG (MAPROB) IN THE *)
(*)   T_ELEMENT.IIT. *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 05/01/86      B. A. ULMER      W315 *)
(*)                   ADDED A CALL CNVOSP TO CONVERT AN "OUT OF MEMORY" *)
(*)                   CONDITION TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 07/11/85      B. A. ULMER      W315 *)
(*)                   ADD NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND *)
(*)                   DEBUGGING PURPOSES *)

```

```
(*      REVISED: 04/26/85      E. D. SHREVE      W315      *)
(*      TO USE THE INTERNAL MAS PROCESS FLAG AND TO CALL  *)
(*      EXPCLST INSTEAD OF EXPALST                          *)
(*      *)
(*      REVISED: 02/18/85      B. A. ULMER      W315      *)
(*      STRUCTURE CHANGE FOR THE CNST. READ BLOCK.          *)
(*      *)
(*      REVISED: 08/14/86      K. M. ROSS      W315      *)
(*      ADDED NIL POINTER CHECK FOR KEY1.                  *)
(*      *)
(*      ORIGINATED: 07/26/84    D. J. KERCHNER      W315    *)
(*-----*)
(**)
(* END %INCLUDE MAEAI *)
```

```

(*) %INCLUDE MAEAV *)
(**)
  PROCEDURE MAEAV(CONST KEY1:ENTKEY;VAR IVAL:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   FIND THE PRESENT VALUE OF THE ACTIVATION SETTING FOR AN *)
(*)   ENTITY. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   KEY1           I    KEY OF THE ENTITY WHOSE ACTIVATION *)
(*)                   SETTING IS TO BE CHECKED *)
(*)   IVAL           0    VALUE OF THE SWITCH *)
(*)                   = 1  TRUE *)
(*)                   = 0  FALSE *)
(*)   RC             0    EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   = 1  YOU BLEW IT *)
(*)                   = 2  THE ROUTINE BLEW IT *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   THE ACTIVITY STATUS OF THE ENTITY IS TO BE CHECKED. *)
(*) *)
(*)   IF THE ENTITY IS ACTIVE (NOT MARKED FOR DELETE), THEN *)
(*)   THE ACTIVITY STATUS IS TRUE AND AN INTEGER FLAG VALUE *)
(*)   OF (1) WILL BE RETURNED. *)
(*) *)
(*)   IF THE ENTITY IS INACTIVE (MARKED FOR DELETE), THEN THE *)
(*)   ACTIVITY STATUS IS FALSE AND AN INTEGER FLAG VALUE OF *)
(*)   (0) WILL BE RETURNED. *)
(*) *)
(*) $COMMENTS: *)
(*) *)

```



```
(* $CHANGE CONTROL: *)
(*)
(*) REVISD: 05/01/86 B. A. ULMER FRMI *)
(*) ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF MEMORY" CONDITION *)
(*) TO USER RECOGNIZEABLE FORM *)
(*)
(*) REVISD: 07/11/85 B. A. ULMER FRMI *)
(*) ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*) PURPOSES *)
(*)
(*) ORIGINATED: 07/27/85 D. J. KERCHNER FRMI *)
(*)
(*)-----*)
%PAGE *)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*) *)
(*)END-----*)
(**)
(*) END %INCLUDE MAEAV *)
(**)
```

```
(* %INCLUDE MAEC *)
(**)
  PROCEDURE MAEC(CONST KEY1:ANYKEY;VAR KEY2:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
%PAGE
(*-----*)
(*)
(*) $FUNCTION:
(*)   CREATE AN APPLICATIONS LIST OF CONSTITUENT ENTITIES.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ===  =====
(*)   KEY1           I    KEY OF AN ENTITY OR A LIST.
(*)   KEY2           O    RETURNED KEY OF THE APPLICATION LIST.
(*)   RC             O    EXTERNAL RETURN CODE
(*)                   < 0  WARNING
(*)                   = 0  OK
(*)                   > 0  CRITICAL ERROR
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   KEY2 IS CREATED (EMPTY LIST).
(*)   IF KEY1 IS AN ENTITY, THEN THE CONSTITUENT LIST OF KEY1
(*)   WILL BE COPIED INTO KEY2.
(*)   IF KEY1 IS A LIST KEY, THEN THE CONSTITUENT LISTS OF EACH
(*)   ENTITY WILL BE COPIED INTO KEY2.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 05/01/86          B. A. ULMER          W315
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF MEMORY" CONDITION
(*)   TO USER RECOGNIZEABLE FORM
(*)
(*)   REVISED: 07/11/85          B. A. ULMER          W315
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*)   PURPOSES
(*)
```

```
(*  REVISED: 05/15/85          B. A. ULMER          W315          *)
(*  FIX INCONSISTENCY IN OUTPUT LIST PROCESSING      *)
(*  *)
(*  REVISED: 02/18/85          B. A. ULMER          W315          *)
(*  CHANGED THE STRUCUTRE OF THE INTERNAL ITEM FOR IMPLEMENTATION *)
(*  OF THE CRB                                         *)
(*  *)
(*  REVISED: 08/14/86          K. M. ROSS           W315          *)
(*  ADDED A CHECK FOR NIL POINTER FOR KEY1           *)
(*  *)
(*  ORIGINATED: 06/08/84       D. J. KERCHNER       W315          *)
(*  *)
(*-----*)
%PAGE
(* END %INCLUDE MAEC      *)
```

```
(* %INCLUDE MAECI *)
(**)
  PROCEDURE MAECI(CONST KEY1:ANYKEY;VAR KEY2:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE AN APPLICATION LIST OF INCLUSIVE CONSTITUENT
(*   ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O      DESCRIPTION
(*   =====
(*   KEY1      I        KEY OF AN ENTITY OR A LIST.
(*   KEY2      O        KEY OF THE CREATED APPLICATION LIST.
(*   RC        O        FUNCTION RETURN CODE.
(*                       = 0  GOOD RETURN
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360, 370, 43XX
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   KEY2 IS CREATED (EMPTY LIST).
(*   IF KEY1 IS AN ENTITY, THEN THE INCLUSIVE CONSTITUENT LIST
(*   OF KEY1 WILL BE COPIED INTO KEY2.
(*   IF KEY1 IS A LIST KEY, THEN THE INCLUSIVE CONSTITUENT LISTS
(*   OF EACH ENTITY WILL BE COPIED INTO KEY2.
(*   IT IS ASSUMED THAT THE MAPROB FLAG IS INITIALLY SET TO
(*   FALSE. AFTER PROCESSING, MAPROB FLAG IS RESET.
(*
(* $CHANGE CONTROL:
(*   REVISED: 05/01/86      B.A. ULMER      W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF MEMORY"
(*   CONDITION TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 01/20/85      B.A. ULMER      W315
```

```
(*          FIX BUG DEALING WITH PREVIOUS FIX          *)
(*)
(*) REVISD: 11/04/85      B.A. ULMER              W315      *)
(*) NOT ALLOW ENITIES THAT ARE ON THE APPLICATION INPUT*)
(*) LIST TO BE ON THE APPLICATION OUTPUT LIST (FIX THE *)
(*) INCONSISTENCY IN THE PROCESSING)                  *)
(*)
(*) REVISD: 07/11/85      B.A. ULMER              W315      *)
(*) ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING    *)
(*) AND DEBUGGING PURPOSES                            *)
(*)
(*) REVISD: 05/15/85      B.A. ULMER              W315      *)
(*) FIX INCONSISTENCY IN OUTPUT LIST PROCESSING        *)
(*)
(*) REVISD: 04/29/85      E.D. SHREVE            W315      *)
(*) TO USE THE INTERNAL MAPROB FLAG                    *)
(*)
(*) REVISD: 02/18/85      B.A. ULMER              W315      *)
(*) IMPLEMENT CRB STRUCTURE CHANGE                     *)
(*)
(*) REVISD: 08/14/86      K.M. ROSS              W315      *)
(*) ADDED A NIL POINTER CHECK FOR KEY1                 *)
(*)
(*) ORIGINATED: 07/26/84  D.J. KERCHNER          W315      *)
(*)-----*)
(**)
(* END %INCLUDE MACRO *)
```

```
(* %INCLUDE MAECIK *)
(**)
  PROCEDURE MAECIK(CONST KEY1:ANYKEY;CONST ENTKIND:ORD_KIND;
    VAR KEY2:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE A LIST OF INCLUSIVE CONSTITUENTS BY KIND.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O      DESCRIPTION
(*   ====      ==      =====
(*   KEY1      I        THE KEY OF AN ENTITY OR A LIST OF ENTITIES
(*                   WHOSE INCLUSIVE CONSTITUENTS ARE TO BE
(*                   SEARCHED FOR THE SPECIFIED KIND.
(*   KIND      I        THE KIND CODE OF AN ENTITY OR AN ENTITY
(*                   CLASS.
(*   KEY2      O        THE KEY OF THE LIST WHICH WILL CONTAIN ALL
(*                   ENTITIES OF THE SPECIFIED KIND FOUND WITHIN
(*                   THE INCLUSIVE CONSTITUENTS OF KEY1.
(*   RC        O        THE FUNCTION RETURN CODE.
(*                   = 0   GOOD RETURN
(*                   > 0   CRITICAL ERROR
(*                   < 0   WARNING
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360, 370, 43XX
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   A NEW LIST IS CREATED TO CONTAIN THE INCLUSIVE CONSTITUENTS,
(*   OR LIST MEMBERS.  FOR EACH LIST MEMBER WHOSE KIND MATCHES
(*   THE GIVEN KIND, THAT MEMBER IS ADDED TO THE OUTPUT LIST
(*   POINTED TO BY KEY2.
(*
(* $CHANGE CONTROL:
(*   REVISED: 05/01/86      B.A. ULMER      W315
(*                   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF MEMORY"
(*                   CONDITION TO USER RECOGNIZEABLE FORM
```

```
(*      REVISED: 07/11/85      B.A. ULMER      W315      *)
\*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING      *)
(*      AND DEBUGGING PURPOSES      *)
(*      REVISED: 05/15/85      B.A. ULMER      W315      *)
(*      FIX INCONSISTENCY IN OUTPUT LIST PROCESSING      *)
(*      REVISED: 04/29/85      E.D. SHREVE      W315      *)
(*      TO USE INTERNAL MAS PROCESS FLAG (MAPROB)      *)
(*      REVISED: 02/18/85      B.A. ULMER      W315      *)
(*      TO IMPLEMENT NEW CRB STRUCTURE      *)
(*      REVISED: 09/11/84      R. A. MCCLUSKEY      W315      *)
(*      CHANGED PROCESSING OF SYSUSE FLAG. DROPPED      *)
(*      ROUTINE EXPCLSTK TO USE EXPCLST INSTEAD.      *)
(*      REVISED: 08/14/86      K. M. ROSS      W315      *)
(*      ADDED A NIL POINTER CHECK FOR KEY1      *)
(*      ORIGINATED: 08/20/84      R.A. MCCLUSKEY      W315      *)
(*-----*)
(**)
(* END %INCLUDE MAECIK *)
```

```
(* %INCLUDE MAECMP *)
(**)
PROCEDURE MAECMP(CONST KEY1:ENTKEY;VAR KEY2:LISTKEY;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   GIVEN AN ENTITY DETERMINE WHICH OF ITS CONSTITUENTS IT
(*   COMPRESSES WITH
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   KEY1          I    USER ENTITY WHOSE COMPRESSIBILITY IS
(*                   DETERMINED BY THE CONSTITUENT ENTITY
(*   KEY2          I    CONSTITUENT ENTITY BEING COMPRESSED
(*   RC            O    EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   < 0  WARNING
(*                   > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```



```
(* %INCLUDE MAECQY *)
(**)
PROCEDURE MAECQY(CONST KEY1:ENTKEY;CONST KEY2:ENTKEY;VAR CMPFLG:
    INTEGER;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     GIVEN AN ENTITY AND ITS USER DETERMINE IF THE USER SHOULD BE
(*     COMPRESSED WITH THE ENTITY WHEN THE ENTITY IS COMPRESSED
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ===          ==  =====
(*     KEY1          I    USER ENTITY WHOSE COMPRESSIBILITY IS
(*                      DETERMINED BY THE CONSTITUENT ENTITY
(*     KEY2          I    CONSTITUENT ENTITY BEING COMPRESSED
(*     CMPFLG        O    FLAG WHICH TELLS IF THE USER IS
(*                      COMPRESSED WITH THE CONSTITUENT
(*     RC            O    EXTERNAL RETURN CODE
(*                      = 0  OK RETURN CODE
(*                      < 0  WARNING
(*                      > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*) 1
```

```
(* %INCLUDE MAECR *)
(**)
  PROCEDURE MAECR(VAR ENTDEF:ENTBLOCK;CONST KEYC:ANYKEY;
    VAR KEYE:ENTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE AN ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   ENTDEF    I   APPLICATION DATA DEFINING THE ENTITY TO
(*                BE CREATED
(*   KEYC      I   CONSTITUENT OR LIST OF CONSTITUENTS TO
(*                BE CONNECTED TO THE ENTITY
(*   KEYE      O   KEY OF CREATES ENTITY
(*   RC        O   EXTERNAL RETURN CODE
(*                = 0  OK
(*                > 0  CRITICAL ERROR
(*                < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86      B. A. ULMER      FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85      B. A. ULMER      FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
(*   REVISED: 10/11/84      D. J. KERCHNER    FRMI
(*
```

CI PS560240032U
April 1990

```
(*  UPDATE DOCUMENTATION                                *)
(*  *)                                                    *)
(*  REVISED: 10/04/84          E. D. SHREVE          FRMI  *)
(*  INPUT PARAMETER ENTDEF CHANGED TO VAR FROM CONST FOR COMPATAB- *)
(*  ABILITY WITH THE DEC VAC SYSTEM                      *)
(*  *)                                                    *)
```

```

(*) %INCLUDE MAECRN *)
(**)
  PROCEDURE MAECRN(VAR ENTDEF:ENTBLOCK;CONST KEYC:ANYKEY;
    VAR KEYE:ENTKEY;VAR NUM:INTEGER;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*  $FUNCTION: *)
(*    CREATE AN ENTITY WITH A CONSTITUENT LIST OF A GIVEN SIZE. *)
(* *)
(*  $DESCRIPTION OF ARGUMENTS: *)
(*    NAME      I/O  DESCRIPTION *)
(*    ====      ==  ===== *)
(*    ENTDEF     I   APPLICATION DATA DEFINING THE ENTITY TO *)
(*                   BE CREATED *)
(*    KEYC        I   CONSTITUENT OR LIST OF CONSTITUENTS TO *)
(*                   BE CONNECTED TO THE ENTITY *)
(*    KEYE        O   KEY OF CREATED ENTITY *)
(*    NUM         I   THE LENGTH OF THE CONSTITUENT LIST *)
(*    RC          O   EXTERNAL RETURN CODE *)
(*                   = 0  OK *)
(*                   > 0  CRITICAL ERROR *)
(*                   < 0  WARNING *)
(* *)
(*  $COMMONS: *)
(* *)
(*  $ENVIRONMENT: *)
(*    LANGUAGE: IBM PASCAL *)
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(*  $EXECUTION PROCEDURE: *)
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(* *)
(*  $PROCESSING DESCRIPTION: *)
(* *)
(*  $COMMENTS: *)
(* *)
(*  $CHANGE CONTROL: *)
(* *)
(*    ORIGINATED: 03/07/87      K. M. ROSS      DBMA *)
(* *)
(*-----*)
(*  DATA STRUCTURES/MAJOR VARIABLES: *)
(*-----*)
(* *)
(*END-----*)
(**)
(* END %INCLUDE MAECRN *)
(**)

```

```

(*) %INCLUDE MAECTK *)
(**)
PROCEDURE  MAECTK(VAR KNDCNT:LISTSIZE;VAR RC:EXT_RET_CODE);
          SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   TO RETURN THE NUMBER OF 'KIND' VALUES IN THE
(*)   WORKING-FORM MODEL.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KNDCNT         0    COUNT OF THE NUMBER OF ENTITIES IN THIS
(*)                   WORKING FORM MODEL OF A SPECIFIC KIND
(*)   RC             0    EXTERNAL RETURN CODE
(*)                   = 0  OK
(*)                   > 0  CRITICAL ERROR
(*)                   < 0  WARNING
(*)
(*) $COMMONS:
(*)   NDSREM
(*)   KEY            I    KEY OF THE ROOT ELEMENT - MUST BE
(*)                   PROVIDED
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   RETRIEVES THE VALUE OF THE STD_ARY_USED_LENGTH IN THE
(*)   ADB OF THE SCHEMA_ROOT ELEMENT.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)   TO USER RECOGNIZEABLE FORM
(*)
(*)   REVISED: 07/11/85          B. A. ULMER          FRMI

```

```
(*      ADD A NEW PARMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING      *)
(*      PURPOSES                                                              *)
(*                                                                           *)
(*      ORIGINATED: 10/26/84          E. D. SHREVE          FRMI          *)
(*                                                                           *)
(*-----*)
%PAGE                                                                    *)
(*-----*)
(*      DATA STRUCTURES/MAJOR VARIABLES:                                  *)
(*-----*)
(*                                                                           *)
(*END-----*)
(**)
(* END %INCLUDE MAECTK *)
```

```
(* %INCLUDE MAECXQ *)
(**)
  PROCEDURE MAECXQ(CONST KEY1:ANYKEY;VAR DATAREC:BLKDATA;
    CONST PROCNAME:ROUTINE;VAR KEY2:LISTKEY;VAR RCC:EXT_RET_CODE;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(**)
(* $FUNCTION:*)
(* EXECUTE A PROCEDURE ON THE CONSTITUENTS OF AN ENTITY, OR LIST*)
(* OF ENTITIES. IF AN OUTPUT LIST IS NOT PASSED, CONSTRUCT ONE *)
(* IN ORDER TO PUT ENTITIES ON IT AS DETERMINED BY THE *)
(* APPLICATION PROCEDURE. *)
(* $DESCRIPTION OF ARGUMENTS:*)
(* NAME I/O DESCRIPTION *)
(* ==== === *)
(* KEY1 I ENITIY OR LIST OF ENTITIES WHOSE CONSTIT-*)
(* UENTS ARE TO BE PROCESSED *)
(* DATAREC I/O APPLICATION DEFINED DATA STRUCTURE WHICH *)
(* EITHER SUPPLIES OR RECIEVES VALUES *)
(* OPERATED ON BY THE APPLICATION PROCEDURE *)
(* PROC I ENTRY POINT OF APPLICATION DEFINED *)
(* PROCEDURE *)
(* KEY2 O KEY OF THE LIST CREATED *)
(* FOR THIS ROUTINE *)
(* RCC O USER DEFINED PROCEDURE RETURN CODE *)
(* = 0,1 OK RETURN CODE *)
(* = 2-7 PROCEDURE WARNING CODE *)
(* = 8-15 PROCEDURE ERROR CODE *)
(* RC O EXTERNAL RETURN CODE *)
(* = 0 OK RETURN CODE *)
(* < 0 WARNING *)
(* > 0 CRITICAL ERROR *)
(* $COMMONS:*)
(* $ENVIRONMENT:*)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* $EXECUTION PROCEDURE:*)
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(* $PROCESSING DESCRIPTION:*)
(* THE USER SENDS IN THE NECESSARY INFORMATION, THEN THIS *)
```

```
(* %INCLUDE MAED. *)
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DELETE AN ENTITY OR LIST OF ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEY1          I    ENTITY OR LIST OF ENTITIES TO BE DELETED
(*   KEYL          O    LIST OF ENTITIES UNABLE TO DELETE
(*   RC            O    EXTERNAL RETURN CODE
(*                       = 0  OK RETURN CODE
(*                       < 0  WARNING
(*                       > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   IF KEY1 IS AN ENTKEY THEN
(*     TRY TO DELETE THE ENTITY ACCORDING TO IT'S USER'S RULES.
(*   IF KEY1 IS A LISTKEY THEN
(*     SORT THE LIST IN A DELETABLE ORDER.
(*     TRY TO DELETE EACH ENTITY ON THE LIST ACCORDING TO ITS
(*     USER'S DELETE RULES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 5/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 4/11/86          E. D. SHREVE          W315
(*   CHANGED TO TEST FOR NIL LIST POINTER BEFORE READING SORTLST..
(*
(*   REVISED: 12/30/85          B. A. ULMER          W315
(*   CHANGE TO READ THE SORT LIST IN REVERSE ORDER - REMOVE THE
```



```
(* CALLS TO ELDNL AND CPYNL (NO LONGER NECESSARY SINCE SORTDLST *)
(* HAS BEEN IMPROVED FOR EFFICIENCY ) *)
(* *)
(* REVISED: 09/ /85 B. A. ULMER W315 *)
(* ADD CODE TO HANDLE THE TWO NEW DELETE RULES *)
(* *)
(* REVISED: 08/ /85 L. J. BEHAN W315 *)
(* ADD A NEW PARAMETER TO DELRUL, DELENTY TO HANDLE APPLICATION *)
(* LIST POSITION PROBLEM *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER W315 *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* REVISED: 05/15/85 B. A. ULMER W315 *)
(* FIX INCONSISTENCY IN OUTPUT LIST PROCESSING *)
(* *)
(* ORIGINATED: 03/08/84 C. J. SAMPLE W315 *)
(* *)
(*-----*)
(**)
(* END %INCLUDE MAED. *)
```

```

(*) %INCLUDE MAEDI. *)
(**)
  PROCEDURE MAEDI(CONST KEY1:ANYKEY; VAR KEY2:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   DELETE INCLUSIVELY AN ENTITY OR LIST OF ENTITIES. *)
(*)   ENTITIES AND THEIR DIRECT AND INDIRECT CONSTITUENTS WILL *)
(*)   BE DELETED. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   KEY1           I    ENTITY OR LIST OF ENTITIES TO BE *)
(*)                   INCLUSIVELY DELETED *)
(*)   KEY2           O    LIST OF ENTITIES UNABLE TO DELETE *)
(*)   RC             O    EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   < 0  WARNING *)
(*)                   > 0  CRITICAL ERROR *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   IF KEY1 IS AN ENTKY THEN *)
(*)     AN INCLUSIVE LIST OF THE ENTITY'S CONSTITUENTS IS CREATED *)
(*)     AND THE ENTITY IS ALSO PLACED ON THE INCLUSIVE LIST. *)
(*)
(*)   IF KEY1 IS A LISTKEY THEN *)
(*)     AN INCLUSIVE LIST OF THE LIST OF ENTITIES' CONSTITUENTS *)
(*)     IS CREATED AND THE LIST OF ENTITIES ARE ALSO PLACED ON THE *)
(*)     INCLUSIVE LIST. *)
(*)
(*)   THE INCLUSIVE LIST IS SORTED IN A USER-CONSTITUENT ORDER. *)
(*)
(*)   FOR EACH ENTITY ON THE INCLUSIVE LIST, AN ATTEMPT IS MADE *)
(*)   TO DELETE THE ENTITY ACCORDING TO THE DELETE RULES OF *)
(*)   THEIR USERS. *)

```

```
(* *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* REVISED: 05/01/86 B. A. ULMER W315 *)
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(* TO USER RECOGNIZEABLE FORM *)
(* *)
(* REVISED: 12/30/85 B. A. ULMER W315 *)
(* CHANGE TO READ SORT LIST IN REVERSE ORDER *)
(* *)
(* REVISED: 09/ /85 B. A. ULMER W315 *)
(* ADD CODE TO HANDLE THE TWO NEW DELETE RULES *)
(* PURPOSES *)
(* *)
(* REVISED: 08/ /85 L. J. BEHAN W315 *)
(* ADD A NEW PARAMETER TO DELRUL, DELENTY TO HANDLE APPLICATION *)
(* LIST POSITION PROBLEM *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER W315 *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* REVISED: 05/15/85 B. A. ULMER W315 *)
(* FIX INCONSISTENCY IN OUTPUT LIST PROCESSING *)
(* *)
(* ORIGINATED: 08/20/84 C. J. SAMPLE W315 *)
(* *)
(*-----*)
%PAGE
(**)
(* END %INCLUDE MAEDI. *) 1
```

April 1990

```

(*) %INCLUDE MAEDT. *)
(**)
  PROCEDURE MAEDT(CONST KEY1:ANYKEY;VAR KEYDL:LISTKEY;
    VAR KEYML:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   TEST DELETE AN ENTITY OR LIST OF ENTITIES. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   KEY1           I    ENTITY OR LIST OF ENTITIES TO BE TEST *)
(*)                   DELETED *)
(*)   KEYDL           O    LIST OF ENTITIES WHICH WOULD BE DELETED *)
(*)                   OR MARKED FOR DELETE BY MAED *)
(*)   KEYML           O    LIST OF ENTITIES WHICH WOULD BE MARKED *)
(*)                   MAED *)
(*)   RC              O    EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   < 0  WARNING *)
(*)                   > 0  CRITICAL ERROR *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   SIMILAR TO MAED, EXCEPT NO DELETION NOR MARK FOR DELETION *)
(*)   IS PERFORMED. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 06/19/86          B. A. ULMER          W315 *)
(*)   CHANGE DETRUL CALLING PARAMETERS & EXCEPTION LIST TO MARK LIST *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 05/01/86          B. A. ULMER          W315 *)
(*) *)

```

```
(*      ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION  *)
(*      TO USER RECOGNIZEABLE FORM                                     *)
(*                                                                 *)
(*      REVISED: 12/30/85          B. A. ULMER          W315        *)
(*      CHANGE TO READ THE SORT LIST IN REVERSE ORDER - REMOVE THE  *)
(*      CALLS TO ELDNL AND CPYNL (NOT NECESSARY SORTDLST HAS BEEN    *)
(*      IMPROVED FOR EFFICIENCY)                                       *)
(*                                                                 *)
(*      REVISED: 07/11/85          B. A. ULMER          W315        *)
(*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*      PURPOSES                                                        *)
(*                                                                 *)
(*      REVISED: 05/15/85          B. A. ULMER          W315        *)
(*      FIX INCONSISTENCY IN OUTPUT LIST PROCESSING                  *)
(*                                                                 *)
(*      ORIGINATED: 06/27/84        C. J. SAMPLE          W315        *)
(*                                                                 *)
(*-----*)
%PAGE                                                                    *)
(**)
(* END %INCLUDE MAEDT. *)
```

```
(* %INCLUDE MAEDTI. *)
(**)
PROCEDURE MAEDTI(CONST KEY1:ANYKEY;VAR KEYDL:LISTKEY;
  VAR KEYML:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* TEST FOR INCLUSIVE DELETION OF AN ENTITY OR LIST OF ENTITIES
(* ENTITIES AND THEIR DIRECT AND INDIRECT CONSTITUENTS WILL BE
(* TESTED FOR DELETION.
(*
(*
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* === == =====
(* KEY1 I ENTITY OR LIST OF ENTITIES TO BE
(* INCLUSIVELY TEST DELETED
(* KEYDL O LIST OF ENTITIES WHICH WOULD BE DELETED
(* BY MAEDI
(* KEYML O LIST OF ENTITIES WHICH WOULD BE MARKED BY
(* MAEDI
(* RC O EXTERNAL RETURN CODE
(* = 0 OK RETURN CODE
(* < 0 WARNING
(* > 0 CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* IF KEY1 IS AN ENTKEY THEN
(* AN INCLUSIVE LIST OF THE ENTITY'S CONSTITUENTS IS CREATED
(* AND THE ENTITY IS ALSO PLACED ON THE INCLUSIVE LIST.
(*
(* IF KEY1 IS A LISTKEY THEN
(* AN INCLUSIVE LIST OF THE LIST OF ENTITIES' CONSTITUENTS
(* IS CREATED AND THE LIST OF ENTITIES ARE ALSO PLACED ON THE
(* INCLUSIVE LIST.
(* THE INCLUSIVE LIST IS SORTED IN A USER-CONSTITUENT ORDER.
(*)
```

```
(*)
(*)      FOR EACH ENTITY ON THE INCLUSIVE LIST, AN ATTEMPT IS MADE (*)
(*)      TO TEST DELETE THE ENTITY ACCORDING TO THE DELETE RULES (*)
(*)      OF THEIR USERS. (*)
(*)
(*)      THE LIST OF MARKABLE ENTITIES IS MERGED WITH THE LIST OF (*)
(*)      NON DELETABLE ENTITIES. (*)
(*)
(*)      $COMMENTS: (*)
(*)
(*)      $CHANGE CONTROL (*)
(*)
(*)      REVISED: 05/01/86          B. A. ULMER          W315 (*)
(*)      ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION (*)
(*)      TO USER RECOGNIZEABLE FORM (*)
(*)
(*)      REVISED: 06/19/86          B. A. ULMER          W315 (*)
(*)      CHANGE PARAMETERS TO DETRUL AND EXCEPTION LIST TO MARK LIST (*)
(*)
(*)      REVISED: 01/13/85          E. D. SHREVE          W315 (*)
(*)      CHANGED TO INITIALIZE A LIST POSITION VARIABLE (*)
(*)
(*)      REVISED: 12/30/85          B. A. ULMER          W315 (*)
(*)      CHANGE TO READ SORT LIST IN REVERSE ORDER (*)
(*)
(*)      REVISED: 07/11/85          B. A. ULMER          W315 (*)
(*)      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING (*)
(*)      PURPOSES (*)
(*)
(*)      REVISED: 05/15/85          B. A. ULMER          W315 (*)
(*)      FIX INCONSISTENCY IN OUTPUT LIST PROCESSING (*)
(*)
(*)      ORIGINATED: 08/21/84        C. J. SAMPLE          W315 (*)
(*)
(*)-----(*)
%PAGE (*)
(**)
(*) END %INCLUDE MAEDTI. *)
```

```
(* %INCLUDE MAEDTS. *)
(**)
PROCEDURE MAEDTS(CONST KEY1:ANYKEY;VAR KEYDL:LISTKEY;
VAR KEYEL:LISTKEY;VAR KEYML:LISTKEY;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* TEST DELETE AN ENTITY OR LIST OF ENTITIES, AND RETURN THREE
(* LISTS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* === ===
(* KEY1 I ENTITY OR LIST OF ENTITIES TO BE TEST
(* DELETED
(* KEYDL O LIST OF ENTITIES WHICH WOULD BE DELETED
(* OR MARKED FOR DELETE BY MAED
(* KEYEL O LIST OF ENTITIES WHICH WOULD NOT BE
(* DELETED BY MAED
(* KEYML O LIST OF ENTITIES WHICH WOULD BE MARKED_
(* FOR_DELETE BY MAED
(* RC O EXTERNAL RETURN CODE
(* = 0 OK RETURN CODE
(* < 0 WARNING
(* > 0 CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* SIMILAR TO MAEDT, EXCEPT THREE LISTS ARE RETURNED. KEYDL AND
(* KEYML CAN BE SUBMITTED TO DIRECTLY DELETE AND MARK ENTITIES
(* USING MAS DELETE ROUTINES THAT DO NOT CHECK THE DELETE RULES.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```



```
(*  REVISED: 05/01/86      B. A. ULMER      W315      *)
(*      ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*      TO USER RECOGNIZEABLE FORM      *)
(*      *)
(*  ORIGINATED: 04/22/86      E. D. SHREVE      W315      *)
(*      *)
(*-----*)
%PAGE      *)
(**)
(* END %INCLUDE MAEDTS *)
```

```

(*) %INCLUDE MAEGKN *)
(**)
  PROCEDURE MAEGKN(CONST KEYE:ENTKEY;VAR KIND:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*)
(*) $FUNCTION:
(*)   RETRIEVE THE KIND VALUE OF AN ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KEYE           I    KEY OF AN ENTITY
(*)   KIND           O    KIND VALUE OF THE ENTITY (INTEGER)
(*)   RC             O    EXTERNAL RETURN CODE
(*)                      = 0  OK RETURN CODE
(*)                      > 0  CRITICAL ERROR
(*)                      < 0  WARNING
(*)
(*) $COMMONS:
(*)   NONE
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   ACCESS THE KIND VALUE FROM THE ENTITY ADB AND RETURN IT.
(*)
(*) $COMMENTS:
(*)   NONE
(*)
(*) $CHANGE CONTROL:
(*)
(*)   REVISED: 05/01/86          B. A. ULMER          W315
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)   TO USER RECOGNIZEABLE FORM
(*)
(*)   REVISED: 07/11/85          B. A. ULMER          W315
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*)   PURPOSES
(*)
(*)

```

CI PS560240032U
April 1990

(* ORIGINATED: 03/25/85 E. D. SHREVE W315 *)
(*
(*END-----*)
(* END %INCLUDE MAEGKN *)

```
(* %INCLUDE MAEGTK *)
(**)
PROCEDURE MAEGTK(CONST KEYE:ENTKEY;VAR ENTDEF:ENTBLOCK;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   RETRIEVE THE ENTITY BLOCK WHICH CORRESPONDS TO KEYE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEYE          I    KEY OF THE ENTITY TO BE RETRIEVED
(*   ENTDEF        O    APPLICATION DATA ASSOCIATED WITH KEYE
(*   RC            O    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   APPLICATION PROVIDES ENTITY KEY.  MAS WILL RETRIEVE THE
(*   ENTITY
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 08/14/86          K. M. ROSS          DBMA
(*   ADDED A NIL POINTER CHECK FOR KEY1
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
(*   REVISED: 11/15/84          D. J. KERCHNER        FRMI
(*   CHECK FOR VALID ENTITY KEY   IF NOT RETURN RC < 0
(*
```

```

(*) %INCLUDE MAEKND *)
(**)
PROCEDURE  MAEKND(CONST KNDPOS:LISTINDX;VAR KNDVAL:ORD_KIND;
                VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   TO RETURN A 'KIND' VALUE FROM THE LIST OF KINDS IN THE
(*)   WORKING-FORM MODEL.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KNDPOS         I    SEQUENCE # OF THE KIND VALUE REQUESTED
(*)   KNDVAL         O    KIND VALUE AT THE 'KNDPOS' POSITION
(*)   RC             O    EXTERNAL RETURN CODE
(*)                   = 0  OK RETURN CODE
(*)                   = 1  YOU BLEW IT
(*)                   = 2  THE ROUTINE BLEW IT
(*)
(*) $COMMONS:
(*)   NDSREM
(*)   KEY           I    KEY OF THE ROOT ELEMENT
(*)                   MUST BE PROVIDED
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   RETRIEVES THE 'KIND' VALUE STORED AT THE 'KNDPOS' POSITION
(*)   IN THE STD_ARRAY OF THE SCH_ROOT ADB.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*)   REVISED: 05/01/86          B. A. ULMER          W315
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)   TO USER RECOGNIZEABLE FORM
(*)
(*)   ORIGINATED: 10/26/84      E. D. SHREVE          FRMI
(*)-----*)

```

```
(*      DATA STRUCTURES/MAJOR VARIABLES:                                *)  
(*-----*)  
(*                                          *)  
(*END-----*)  
(**)  
(* END %INCLUDE MAEKND *)  
(**)
```

```
(* %INCLUDE MAERST *)
(**)
PROCEDURE MAERST(CONST FLGNAME:NAMTYP; VAR RC:EXT_RET_CODE);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     RESET THE GIVEN FLAG IN ALL ENTITIES IN THE WORKING FORM
(*     MODEL
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     ====      ==  =====
(*     FLGNAME    I   THE NAME OF THE FLAG WHICH WILL BE RESET
(*     RC          0   EXTERNAL RETURN CODE
(*                     = 0  OK
(*                     > 0  CRITICAL ERROR
(*                     < 0  WARNING
(*
(* $COMMONS:
(*     NDSREM
(*     VAR1      I   VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*                     MUST BE PROVIDED
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*     DDNAMES USED WITH STANDARD FILES:
(*
(* $EXECUTION PROCEDURE:
(*     MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*     REVISED: 05/01/86      B. A. ULMER      FRMI
(*     ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*     TO USER RECOGNIZEABLE FORM
(*
(*     ORIGINATED: 08/12/85    B. A. ULMER      FRMI
(*
(*-----*)
```

CI PS560240032U
April 1990

```
(*-----*)  
(* DATA STRUCTURES/MAJOR VARIABLES: *)  
(*-----*)  
(* *)  
(*END-----*)  
(* END %INCLUDE MAERST *)
```



```

(*) %INCLUDE MAESCI. *)
(**)
  PROCEDURE MAESCI(CONST KEY1:ANYKEY;CONST ISWT:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   SET OR RESET THE PROCESS FLAG FOR THE INCLUSIVE CONSTITUENTS*)
(*)   OF AN ENTITY OR A LIST ENTITIES. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   KEY1           I    KEY OF THE ENTITY WHOSE SWITCH IS TO BE *)
(*)                   SET OR KEY OF THE LIST ALL OF WHOSE *)
(*)                   ENTITY SWITCHES ARE TO BE SET *)
(*)   ISWT           I    SWITCH VALUE REQUESTED *)
(*)   RC             0    EXTERNAL RETURN CODE *)
(*)                   = 0  OK *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   THE TYPE OF KEY IS CHECKED FOR. *)
(*)   THE INCLUSIVE CONSTITUENTS ARE COLLECTED FOR AN ENTITY OR *)
(*)   A LIST OF ENTITIES. FOR THE COLLECTED ENTITIES THE SWITCH *)
(*)   IS SET OR RESET. *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)
(*)   ORIGINATED: 03/07/87          K. M. ROSS          DBMA *)
(*)
(*)-----*)

```

CI PS560240032U
April 1990

```
(*      DATA STRUCTURES/MAJOR VARIABLES:                      *)
(*-----*)
(*-----*)
(*END-----*)
(* END %INCLUDE MAESCI. *)
(**)
```

```
(* %INCLUDE MAESVL. *)
(**)
PROCEDURE MAESVL(CONST KEY1:ENTKEY;VAR ISET:INTEGER;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* FIND THE CURRENT BINARY SWITCH SETTING OF AN ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ===
(* KEY1 I KEY OF THE ENTITY WHOSE SETTING IS TO BE
(* DETERMINED
(* RC 0 EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* THE INPUT KEY MUST BE AN ENTITY KEY. IF THE SWITCH IS
(* TRUE, THEN THE VALUE "1" IS RETURNED. IF THE SWITCH IS
(* FALSE, THEN THE VALUE "0" IS RETURNED.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* REVISED: 05/01/86 B. A. ULMER FRMI
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(* TO USER RECOGNIZEABLE FORM
(*
(* REVISED: 07/11/85 B. A. ULMER FRMI
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(* PURPOSES
(*
(* REVISED: 08/14/86 K. M. ROSS DBMA
(* ADDED A CHECK FOR NIL POINTER FOR KEY1
```

```

(*) %INCLUDE MAESWA *)
(**)
  PROCEDURE MAESWA(VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   SETS THE PROCESS BIT 'OFF' IN ALL ENTITIES IN THE MODEL. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   RC            0    EXTERNAL RETURN CODE *)
(*)                   = 0  OK *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING *)
(*) *)
(*) $COMMONS: *)
(*)   NDSREM *)
(*)   KEY      I    KEY OF THE MODEL ROOT ELEMENT *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   EACH ENTRY IN THE SCHEMA-ROOT CONSTITUENT LIST IS READ. *)
(*)   IF IT IS AN INSTANCE_COLLECTOR NODE, THEN EACH ENTITY *)
(*)   ON THE CONSTITUENT LIST OF THE COLLECTOR NODE IS READ *)
(*)   AND THE ADB.SYSUSE FIELD IS SET TO 'TRUE'. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDIGN *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 07/11/85          B. A. ULMER          FRMI *)
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)   PURPOSES *)
(*) *)

```

CI PS560240032U
April 1990

```
(*   ORIGINATED: 02/06/85 CCWW  E. D. SHREVE                FRMI   *)
(*)                                                           *)
(*)-----*)
(*)   DATA STRUCTURES/MAJOR VARIABLES:                      *)
(*)-----*)
(*)                                                           *)
(*END-----*)
(* END %INCLUDE MAESWA *)
(**)
```

```

(*) %INCLUDE MAESWT. *)
(**)
  PROCEDURE MAESWT(CONST KEY1:ANYKEY;CONST ISWT:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*) $FUNCTION: *)
(*)   SET AN ENTITY SWITCH OR THE SWITCHES FOR EACH ENTITY IN A *)
(*)   LIST AS REQUESTED BY THE USER. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   KEY1          I    KEY OF THE ENTITY WHOSE SWITCH IS TO BE *)
(*)                   SET OR KEY OF THE LIST ALL OF WHOSE *)
(*)                   ENTITY SWITCHES ARE TO BE SET *)
(*)   ISWT          I    SWITCH VALUE REQUESTED *)
(*)   RC            O    EXTERNAL RETURN CODE *)
(*)                   = 0  OK *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   THE TYPE OF KEY IS CHECKED FOR. *)
(*)   IF AN ENTITY, THEN THE ENTITY'S SWITCH IS RESET. *)
(*)   IF A LIST, THEN EACH ENTITY ON THE LIST HAS ITS SWITCH *)
(*)   RESET. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 07/11/85          B. A. ULMER          FRMI *)
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)   PURPOSES *)

```

```

(*) %INCLUDE MAEU. *)
(**)
  PROCEDURE MAEU(CONST KEY1:ANYKEY;VAR KEY2:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   CREATE A LIST OF USER ENTITY REFERENCES. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   KEY1           I   ENTITY OR LIST OF ENTITIES FOR WHICH A *)
(*)                   LIST OF DIRECT USERS IS REQUESTED *)
(*)   PARM2           O   LIST OF USER REFERENCES *)
(*)   RC              O   EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   < 0  WARNING *)
(*)                   > 0  CRITICAL ERROR *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   A NEW LIST, KEY2, IS CREATED THAT CONTAINS THE LIST OF *)
(*)   DIRECT USERS. IF KEY1 IS AN ENTITY KEY, THE DIRECT USERS *)
(*)   OF KEY1 ARE PLACED IN THE LIST. IF KEY1 IS A LISTKEY, THE *)
(*)   DIRECT USERS OF ALL ENTITIES IN THE LIST ARE PLACED INTO *)
(*)   KEY2. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 05/01/86          B. A. ULMER          W315 *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 07/11/85          B. A. ULMER          W315 *)
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)

```

```
(*      PURPOSES                                     *)
(*)
(*)  REVISED: 05/15/85          B. A. ULMER          W315      *)
(*)    FIX INCONSISTENCY IN OUTPUT LIST PROCESSING      *)
(*)
(*)  REVISED: 08/14/86          K. M. ROSS           W315      *)
(*)    ADDED A CHECK FOR NIL POINTER FOR KEY1          *)
(*)
(*)  ORIGINATED: 06/21/84       D. J. KERCHNER        W315      *)
(*)
(*)-----*)
%PAGE                                             *)
(**)
(* END %INCLUDE MAEU. *)
```



```
(* %INCLUDE MAEUD *)
(**)
PROCEDURE MAEUD(VAR KEYE:ENTKEY;VAR ENTDEF:ENTBLOCK;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   UPDATE THE ENTITY BLOCK CORRESPONDING TO A KEY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ===          ==  =====
(*   KEYE          I    KEY OF THE ENTITY TO BE UODATED
(*   ENTDEF        I    APPLICATION DATA ASSOCIATED WITH KEYE
(*   RC            0    EXTERNAL RETURN CODE
(*                      = 0  OK
(*                      > 0  CRITICAL ERROR
(*                      < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   CALL REVNODM
(*
(* $COMMENTS:
(*   IT IS ILLEGAL FOR THE APPLICATION TO CHANGE KIND ON UPDATE.
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
(*   REVISED: 10/11/84          D. J. KERCHNER        FRMI
(*
```

(*	UPDATED THE INCLUDE DOCUMENTATION		*)
(*			*)
(*	REVISED: 10/04/84	E. D. SHREVE	FRMI
(*	CHANGED THE DECLARATION FOR KEYE AND ENTDEF TO VAR		*)
(*			*)
(*	REVISED: 08/14/86	K. M. ROSS	DBMA
(*	ADDED A CHECK FOR NIL POINTER FOR KEY1		*)
(*			*)

April 1990

```

(*) %INCLUDE MAEUI *)
(**)
  PROCEDURE MAEUI(CONST KEY1:ANYKEY;VAR KEY2:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   CREATE AN APPLICATION LIST OF INCLUSIVE USER ENTITIES. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O      DESCRIPTION *)
(*)   ====      ==      ===== *)
(*)   KEY1       I       KEY OF AN ENTITY OR A LIST. *)
(*)   KEY2       0       RETURNED KEY OF THE APPLICATION LIST. *)
(*)   RC         0       FUNCTION RETURN CODE. *)
(*)                       = 0   GOOD RETURN *)
(*)                       > 0   CRITICAL ERROR *)
(*)                       < 0   WARNING *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360, 370, 43XX *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE PROCEDURE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   KEY2 IS CREATED (EMPTY LIST). *)
(*)   IF KEY1 IS AN ENTITY, THEN THE INCLUSIVE USER LIST OF KEY1 *)
(*)   WILL BE COPIED INTO KEY2. *)
(*)   IF KEY1 IS A LIST KEY, THEN THE INCLUSIVE USER LISTS OF *)
(*)   EACH ENTITY WILL BE COPIED INTO KEY2. *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 05/01/86      B. A. ULMER      W315 *)
(*)           ADDED TO CNVOSP TO CONVERT AN "OUT OF SPACE" *)
(*)           CONDITION TO USER RECOGNIZEABLE FORM *)
(*)   REVISED: 11/08/85      B. A. ULMER      W315 *)
(*)           FIX BUG DEALING WITH PREVIOUS FIX *)
(*)   REVISED: 11/08/85      B. A. ULMER      W315 *)
(*)           NOT ALLOW ENTITIES THAT ARE ON THE APPLICATION *)
(*)           INPUT LIST TO BE ON THE APPLICATION OUTPUT LIST *)
(*)           (FIX THE INCONSISTENCIES IN THE PROCESSING) *)
(*)   REVISED: 07/11/85      B. A. ULMER      W315 *)
(*)           ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING *)

```

```
(*          AND DEBUGGING PURPOSES          *)
(*  REVISED: 05/15/85      B. A. ULMER      W315      *)
(*          FIX INCONSISTENCY IN OUTPUT LIST PROCESSING      *)
(*  REVISED: 04/29/85      E. D. SHREVE      W315      *)
(*          TO USE MAS INTERNAL PROCESS FLAG (MAPROB)      *)
(*  REVISED: 08/14/86      K. M. ROSS      W315      *)
(*          ADDED A NIL POINTER CHECK FOR KEY1      *)
(*  ORIGINATED: 06/13/84    D. J. KERCHNER      W315      *)
(**)
(* END %INCLUDE MAEUI *)
```

```

(*) %INCLUDE MAEUIK *)
(**)
  PROCEDURE MAEUIK(CONST KEY1:ANYKEY;CONST ENTKind:ORD_KIND;
    VAR KEY2:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   CREATE A LIST OF INCLUSIVE USERS BY KIND. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O      DESCRIPTION *)
(*)   ====      ===      ===== *)
(*)   KEY1       I       THE KEY OF AN ENTITY OR A LIST OF ENTITIES *)
(*)                       WHOSE INCLUSIVE USERS ARE TO BE SEARCHED *)
(*)                       FOR THE SPECIFIED KIND. *)
(*)   KIND       I       THE KIND CODE OF AN ENTITY OR AN ENTITY *)
(*)                       CLASS. *)
(*)   KEY2       O       THE KEY OF THE LIST WHICH WILL CONTAIN ALL *)
(*)                       ENTITIES OF THE SPECIFIED KIND FOUND WITHIN *)
(*)                       THE INCLUSIVE USERS OF KEY1. *)
(*)   RC         O       THE FUNCTION RETURN CODE. *)
(*)                       = 0   GOOD RETURN *)
(*)                       > 0   CRITICAL ERROR *)
(*)                       < 0   WARNING *)
(*) *)
(*) $COMMONS: *)
(*)   NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360, 370, 43XX *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE. *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   FOR THE GIVEN ENTKEY OR LISTKEY EXPAND ITS USERS INCLU *)
(*)   SIVLEY. FOR EACH MEMBER OF THE EXPANDED LIST WHOSE KIND *)
(*)   MATCHES THE KIND VALUE DESIRED ADD IT TO THE LIST POINTED *)
(*)   TO BY KEY2 *)
(*) *)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 05/01/86      B.A. ULMER      W315 *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE"*)
(*)   CONDITION TO USER RECOGNIZEABLE FORM *)

```

```
(*      REVISED: 07/11/85      B.A. ULMER      W315      *)
(*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING      *)
(*      AND DEBUGGING      *)
(*      REVISED: 05/15/85      B.A. ULMER      W315      *)
(*      FIX INCONSISTENCY IN OUTPUT LIST PROCESSING      *)
(*      REVISED: 04/29/85      E.D. SHREVE      W315      *)
(*      TO USE THE INTERNAL MAS PROCESS FLAG (MAPROB)      *)
(*      REVISED: 08/14/86      K.M. ROSS      W315      *)
(*      ADDED A NIL POINTER CHECK FOR KEY1      *)
(*      ORIGINATED: 08/20/84      R. A. MCCLUSKEY      W315      *)
(*-----*)
(**)
(* END %INCLUDE MAEUIK *)
```

```
(* %INCLUDE MAEUSR *)
(**)
PROCEDURE MAEUSR(CONST KEYE:ENTKEY; VAR UEXIST:INTEGER;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     DETERMINES IF AN ENTITY HAS ANY USERS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ===          ===  =====
(*     KEYE          I    ENTITY KEY
(*     UEXIST        0    INTEGER VALUE INDICATING IF KEYE HAS
(*                      ANY USERS.
(*                      = 0 NO USERS EXIST
(*                      = 1 USERS EXIST
(*     RC            0    EXTERNAL RETURN CODE
(*                      = 0 OK RETURN CODE
(*                      > 0 CRITICAL ERROR
(*                      < 0 WARNING MESSAGE
(*
(* $COMMONS:
(*     NONE
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*
(* $PROCESSING DESCRIPTION:
(*     EVALUATES THE USER POINTER IN THE ENTITY BLOCK FOR A NIL.
(*     IF NIL, THEN NO USERS EXIST.
(*
(* $COMMENTS:
(*     THIS PROCEDURE DEVELOPED SPECIFICALLY FOR THE IDB PACKAGE
(*     BUT IS FUNCTIONAL FOR ALL APPLICATIONS.
(*
(* $CHANGE CONTROL:
(*
(*     REVISED: 05/01/86          B. A. ULMER
(*     ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*     TO USER RECOGNIZEABLE FORM
```

```
(*  REVISED: 04/07/85          B. A. ULMER          *)
(*  CHANGED TO CHECK FOR THE SYSTEM LIST HAVING NO ENTRIES - IF IT*)
(*  DOES, THEN NO USERS EXIST          *)
(*  *)
(*  REVISED: 07/11/86          B. A. ULMER          *)
(*  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*  PURPOSES          *)
(*  *)
(*  ORIGINATED: 03/25/85        E. D. SHREVE        *)
(*  *)
(*  *)
(*END-----*)
(* END %INCLUDE MAEUSR *)
```



```
(* %INCLUDE MAEUXQ *)
(**)
PROCEDURE MAEUXQ(CONST KEY1:ANYKEY;VAR DATAREC:BLKDATA;
CONST PROCNAME:ROUTINE;VAR KEY2:LISTKEY;VAR RCC:EXT_RET_CODE;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) EXECUTE A PROCEDURE ON THE USERS OF AN ENTITY, OR LIST (*)
(*) OF ENTITIES. IF AN OUTPUT LIST IS NOT PASSED, CONSTRUCT ONE (*)
(*) IN ORDER TO PUT ENTITIES ON IT AS DETERMINED BY THE (*)
(*) APPLICATION PROCEDURE. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) KEY1 I ENITIY OR LIST OF ENTITIES WHOSE USERS (*)
(*) ARE TO BE PROCESSED (*)
(*) DATAREC I/O APPLICATION DEFINED DATA STRUCTURE WHICH (*)
(*) EITHER SUPPLIES OR RECIEVES VALUES (*)
(*) OPERATED ON BY THE APPLICATION PROCEDURE (*)
(*) PROC I ENTRY POINT OF APPLICATION DEFINED (*)
(*) PROCEDURE (*)
(*) KEY2 O KEY OF THE LIST CREATED (*)
(*) FOR THIS ROUTINE (*)
(*) RCC O USER DEFINED PROCEDURE RETURN CODE (*)
(*) = 0,1 OK RETURN CODE (*)
(*) = 2-7 PROCEDURE WARNING CODE (*)
(*) = 8-15 PROCEDURE ERROR CODE (*)
(*) RC O EXTERNAL RETURN CODE (*)
(*) = 0 OK RETURN CODE (*)
(*) < 0 WARNING (*)
(*) > 0 CRITICAL ERROR (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*) THE USER SENDS IN THE NECESSARY INFORMATION, THEN THIS (*)
(*) ROUTINE REFERENCES THE USER'S SPECIFIED PROCEDURE TO ACT (*)
```

```
(*      UPON THE INFORMATION HE HAS SUPPLIED TO THE PROCEDURE.      *)
(*)                                                                    *)
(*) $COMMENTS:                                                         *)
(*)                                                                    *)
(*) $CHANGE CONTROL:                                                  *)
(*)                                                                    *)
(*)   REVISED: 09/09/86          B. A. ULMER          DBMA          *)
(*)   FIX PROBLEM WITH DELETING EMPTY PASSED IN APPL LIST          *)
(*)                                                                    *)
(*)   ORIGINATED: 06/16/86       B. A. ULMER          W315          *)
(*)                                                                    *)
(*)-----*)
%PAGE                                                                    *)
(**)
(* END %INCLUDE MAEUXQ *)
```

```
(* %INCLUDE MAEXEQ *)
(**)
PROCEDURE MAEXEQ(CONST KEY1:ANYKEY;VAR DATAREC:BLKDATA;
CONST PROCNAME:ROUTINE;VAR RCC:EXT_RET_CODE;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(* NAME I/O DESCRIPTION
(*
(* ====
(* KEY1 I THE KEY OF THE ENTITY OR APPLICATION LIST*)
(* OF ENTITIES TO BE PROCESSED
(*
(* DATAREC I/O THE APLPLICATION DEFINED DATA STRUCTURE
(* WHICH EITHER SUPPLIES OR RECEIVES VALUES
(* OPERATED ON BY THE APPLICATION USER
(* DEFINED PROCEDURE
(*
(* PROCNAME I THE NAME OF THE USER DEFINED PROCEDURE
(*
(* RCC 0 THE USER DEFINED PROCEDURE'S RETURN CODE
(* RCC < 0 & RCC > 15 PROC_OUT_OF_RANGE
(* RCC=> 0 & RCC=< 7 CONTINUE PROCESSING
(* RCC=> 8 & RCC=< 15 PROC_CODE_ERROR
(*
(* RC 0 EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* THE USER SENDS IN THE NECESSARY INFORMATION, THEN THIS
(* ROUTINE REFERENCES THE USER'S SPECIFIED PROCEDURE TO ACT
(* UPON THE INFORMATION HE HAS SUPPLIED TO THE PROCEDURE.
(* THE PROCEDURE RETURNS ITS OWN RETURN CODE TO THE USER.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
```

```
(*
(*  REVISED: 05/01/86          B. A.ULMER          FRMI      *)
(*    ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION  *)
(*    TO USER RECOGNIZEABLE FORM                                     *)
(*                                                                    *)
(*  REVISED: 01/20/86          B. A.ULMER          FRMI      *)
(*    ADD NEW CAPABILITY TO ALLOW READING LIST IN REVERSE IN ORDER  *)
(*    TO PROCESS                                                     *)
(*                                                                    *)
(*  REVISED: 07/11/85          B. A.ULMER          FRMI      *)
(*    ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING  *)
(*    PURPOSES                                                       *)
(*                                                                    *)
(*  REVISED: 03/06/85          B. A.ULMER          FRMI      *)
(*    FIXED APPLICATION LIST PROBLEM                                  *)
(*                                                                    *)
(*  REVISED: 11/28/84          D. J. KERCHNER       FRMI      *)
(*    CHANGED MANNER OF ACCESSING USER DEFINED PROCEDURE - NOW      *)
(*    ACCESSED VIA ASSEMBLER CSECT PASASM                             *)
(*                                                                    *)
(*  ORIGINATED: 04/11/84        D. J. KERCHNER       FRMI      *)
(*                                                                    *)
(*-----*)
%PAGE                                                                *)
(*-----*)
(*  DATA STRUCTURES/MAJOR VARIABLES:                               *)
(*-----*)
(*                                                                    *)
(*END-----*)
(* END %INCLUDE MAEXEQ *)
(**)
```

```

(*) %INCLUDE MAINIT. *)
(**)
  PROCEDURE MAINIT(VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*)  $FUNCTION:
(*)    INITIALIZE THE MAS NETWORK.
(*)
(*)  $DESCRIPTION OF ARGUMENTS:
(*)    NAME          I/O  DESCRIPTION
(*)    ====          ==  =====
(*)    RC            0    EXTERNAL RETURN CODE
(*)                      = 0  OK RETURN CODE
(*)                      < 0  WARNING
(*)                      > 0  CRITICAL ERROR
(*)
(*)  $COMMONS:
(*)
(*)  $ENVIRONMENT:
(*)    LANGUAGE: IBM PASCAL
(*)    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*)  $EXECUTION PROCEDURE:
(*)    MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*)  $PROCESSING DESCRIPTION:
(*)
(*)  $COMMENTS:
(*)
(*)  $CHANGE CONTROL:
(*)    REVISED: 05/01/86          B. A. ULMER          W315
(*)    ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)    TO USER RECOGNIZEABLE FORM
(*)    REVISED: 07/11/85          B. A. ULMER          W315
(*)    ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*)    PURPOSES
(*)    REVISED: 05/21/85          B. A. ULMER          W315
(*)    ADD CALL TO INITIALIZE THE APPLICATION ACCESSIBLE FLAC TABLE
(*)
(*)    ORIGINATED: 03/08/84        D. J. KERCHNER        W315
(*)-----*)
%PAGE
%PRINT ON
(*) END %INCLUDE MAINIT *)

```

```

(*) %INCLUDE MAKCNT *)
(**)
PROCEDURE MAKCNT(CONST KINDX:INTEGER;VAR COUNT:INTEGER;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) DETERMINE THE NUMBER OF ENTITIES IN THE WORKING FORM MODEL *)
(*) OF A SPECIFIED KIND *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) === === *)
(*) KINDX I KIND VALUE FOR WHICH A COUNT IS TO BE *)
(*) DETERMINED *)
(*) COUNT 0 NUMBER OF ENTITIES IN THE MODEL OF THE *)
(*) SPECIFIED KIND *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK RETURN CODE *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*) *)
(*) $COMMONS: *)
(*) NONE *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) DETERMINE IF THE KIND SPECIFIED IS IN THE WORKING FORM *)
(*) MODEL. IF SO, RETURN THE LENGTH VALUE OF THE CONSTITUENT *)
(*) LIST FOR THE COUNT. IF NOT, RETURN A VALUE OF ZERO FOR *)
(*) THE COUNT. *)
(*) *)
(*) $COMMENTS: *)
(*) NONE *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*) REVISED: 05/01/86 B. A. ULMER W315 *)
(*) ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*) TO USER RECOGNIZEABLE FORM *)

```

CI PS560240032U
April 1990

```
(*
(*   REVISED: 07/11/85           B. A. ULMER           W315           *)
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*   PURPOSES                     *)
(*
(*   ORIGINATED: 05/10/85       B. A. ULMER           W315           *)
(*
(*END-----*)
(* END %INCLUDE MAKCNT *)
```

```
(* %INCLUDE MAKILL. *)
(**)
  PROCEDURE MAKILL(VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DELETE THE WORKING FORM MODEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   RC             0    EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   < 0  WARNING
(*                   > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*   NDSGVR
(*   LIST_OF_ROOTS  0    POINTER TO LIST OF ROOTS
(*   STACK_OF_LISTS 0    POINTER TO STACK_OF_ROOTS
(*   NDSREM
(*   KEY            0    POINTER TO THE WORKING FORM ROOT NODE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*   THIS VERSION USED WITH THE MAS MEMORY MANAGER.
(*
(* $PROCESSING DESCRIPTION:
(*   DELETES THE WORKING FORM USING PROCEDURE 'NDSREL'.
(*   RESETS POINTERS IN THE COMMONS TO NIL.
(*
(* $COMMENTS:
(*   THIS VERSION FOR USE WITH THE MAS MEMORY MANAGER. THE OLD
(*   VERSION MUST BE USED IF THE PASCAL MEMORY MANAGER IS USED.
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*
```



```
(*  REVISED: 07/11/85          B. A. ULMER          W315      *)
(*  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING  *)
(*  PURPOSES                                                           *)
(*  REVISED: 04/05/85          E. D. SHREVE          W315      *)
(*  CHANGED TO DELETE THE WORKING FORM USING 'NDSREL'.              *)
(*  ORIGINATED: 02/02/84        D. KERCHNER            K315      *)
(*  -----*)
(*END-----*)
(**)
%PRINT ON
(* END %INCLUDE MAKILL *)
```

```
(* %INCLUDE MAKXEQ *)
(**)
  PROCEDURE MAKXEQ(CONST KIND:ORD_KIND;VAR DATAREC:BLKDATA;
    CONST PROCNAME:ROUTINE;VAR RCC:EXT_RET_CODE;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION
(*   EXECUTE A PROCEDURE ON ALL ENTITIES OF A SPECIFIED KIND.
(*
(* $DESCRIPTION OF ARGUMENTS
(*   NAME      I/O      DESCRIPTION
(*   ====      ==      =====
(*   KIND      I       KIND VALUE OF THE ENTITIES TO BE PROCESSED.
(*   DATAREC    I       THE APPLICATION DEFINED DATA STRUCTURE WHICH
(*                       EITHER SUPPLIES OR RECEIVES VALUES OPERATED
(*                       ON BY THE APPLICATION DEFINED PROCEDURE.
(*   PROCNAME   I       THE NAME OF THE USER DEFINED PROCEDURE.
(*   DATAREC    O       THE DATA STRUCTURE THAT RESULTS FROM USING
(*                       THE USER DEFINED PROCEDURE.
(*   RCC        O       THE USER DEFINED PROCEDURE'S RETURN CODE.
(*   RC         O       THE FUNCTION RETURN CODE.
(*                       =0 EXPECTED RESULT
(*                       >0 CRITICAL ERROR
(*                       <0 WARNING
(*
(* $COMMONS:
(*   NONE
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*
(* $PROCESSING DESCRIPTION:
(*   THE CONSTITUENT LIST OF THE INPUT 'KIND' INSTANCE COLLECTOR
(*   IS READ IN LIFO ORDER AND THE INPUT PROCEDURE IS CALLED
(*   WITH EACH ENTRY IN THE CONSTITUENT LIST.
(*
(* $COMMENTS:
(*   THE ROUTINE PASASM IS CALLED TO PROVIDE A METHOD OF PASSING
(*   ARGUMENTS FROM A FORTRAN ROUTINE.
(*   THE LIST IS READ IN LIFO ORDER IN CASE THE INPUT PROCEDURE
(*   DELETES ENTITIES THAT AFFECT THE LIST BEING READ. WITH THE
(*   LIFO ORDER, THE LIST POSITION IS NOT AFFECTED.
```

```
(* *)
(* $CHANGE CONTROL: *)
(* REVISED: 05/01/86 B. A. ULMER *)
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" *)
(* CONDITION TO USER RECOGNIZEABLE FORM *)
(* REVISED: 07/29/85 B. A. ULMER *)
(* FIX LOCAL LIST PROBLEM *)
(* REVISED: 07/11/85 B. A. ULMER *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND *)
(* DEBUGGING PURPOSES *)
(* REVISED: 03/27/85 E. SHREVE *)
(* TO READ THE LIST IN LIFO ORDER *)
(* REVISED: 03/11/85 B. ULMER *)
(* FIX PROBLEM OF LIST POSITION. *)
(* REVISED: 02/18/85 B. ULMER *)
(* CHANGED THE STRUCTURE OF THE INTERNAL ITEM FOR IMPLEMENT- *)
(* ATION OF THE CRB *)
(* ORIGINATED: 01/20/85 E. SHREVE *)
(*-----*)
(**)
(* END %INCLUDE MAKXEQ *)
```

```
(* %INCLUDE MAL *)
(**)
  PROCEDURE MAL(VAR KEYL:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE AN EMPTY LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   RC            0    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A.ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
```

```
(* %INCLUDE MALAND. *)
(**)
  PROCEDURE MALAND(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
    VAR KEY3:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE AN APPLICATION LIST OF ENTITIES COMMON TO TWO INPUT
(*   LISTS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   KEY1          I    ENTITY OR LIST OF ENTITIES WHICH WILL BE
(*                   'ANDED' - IF ENTITY, USE CONSTITUENT LIST*)
(*   KEY2          I    ENTITY OR LIST OF ENTITIES WHICH WILL BE
(*                   'ANDED' - IF ENTITY, USE CONSTITUENT LIST*)
(*   KEY3          O    LIST OF ENTITIES COMMON TO KEY1 AND KEY2
(*   RC            O    EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   < 0  WARNING
(*                   > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   THE INPUT LIST KEY1 WILL BE COMPARED WITH THE INPUT LIST
(*   KEY2.  THOSE ENTITIES WHICH APPEAR ON KEY1 AND KEY2 WILL
(*   BE PUT ON THE OUTPUT LIST KEY3.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*   REVISED: 05/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          W315
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
```

CI PS560240032U
April 1990

(* *)
(* REVISED: 08/14/86 K. M. ROSS W315 *)
(* ADDED A NIL POINTER CHECK FOR KEY1 *)
(* *)
(* ORIGINATED: 03/09/84 D. J. KERCHNER W315 *)
(* *)
(*-----*)
%PAGE *)
(**)
(* END %INCLUDE MALAND. *)

```

(*) %INCLUDE MALATC *)
(**)
  PROCEDURE MALATC(VAR KEY1:ANYKEY;CONST KEY2:ANYKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) APPEND AN ENTITY OR LIST (KEY2) TO AN ENTITY OR LIST (KEY1). *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) === === *)
(*) KEY1 I THE KEY OF THE ENTITY OR LIST OF ENTITIES*)
(*) TO WHICH KEY2 IS APPENDED *)
(*) KEY2 I THE KEY OF THE ENTITY OR LIST OF ENTITIES*)
(*) TO BE APPENDED RO KEY1 *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK RETURN CODE *)
(*) = 1 YOU BLEW IT *)
(*) = 2 THE ROUTINE BLEW IT *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*) IF KEY1 AND KEY2 ARE BOTH ENTITIES, THEN *)
(*) KEY2 IS ADDED TO THE CONSTITUENT LIST OF KEY1. *)
(*) IF KEY1 IS AN ENTITY AND KEY2 IS A LIST, THEN *)
(*) ALL ENTITIES OF KEY2 ARE ADDED TO THE CONSTITUENT LIST *)
(*) OF KEY1. *)
(*) IF KEY1 IS A LIST AND KEY2 IS AN ENTITY, THEN *)
(*) KEY2 IS ADDED TO THE END OF KEY1. *)
(*) IF KEY1 AND KEY2 ARE BOTH LISTS, THEN *)
(*) ALL ENTITIES OF KEY2 ARE ADDED TO THE END OF KEY1. *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)

```

```
(*  REVISED: 05/01/86          B. A. ULMER          FRMI      *)
(*  ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*  TO USER RECOGNIZEABLE FORM                                     *)
(*                                                                    *)
(*  REVISED: 07/11/85          B. A. ULMER          FRMI      *)
(*  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*  PURPOSES                                                         *)
(*                                                                    *)
(*  REVISED: 10/11/84          D. J. KERCHNER        FRMI      *)
(*  INPUT PARAMETER KEY2 CHANGED TO VAR FROM CONST FOR COMPATA-    *)
(*  BILITY WITH DEC VAX SYSTEM  - UPDATE DOC                       *)
(*                                                                    *)
```



```
(* %INCLUDE MALCPY. *)
(**)
PROCEDURE MALCPY(CONST KEY1:LISTKEY;VAR KEY2:LISTKEY;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   MAKE A COPY OF A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEY1           I    THE KEY OF THE LIST TO BE COPIED
(*   KEY2           I    THE KEY OF THE NEW LIST THAT WILL CONTAIN*
(*                       A COPY OF KEY1
(*   RC             O    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
```

```
(* %INCLUDE MALD. *)
(**)
PROCEDURE MALD(CONST KEY1:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* DELETE AN APPLICATION LIST. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* KEY1 I THE KEY OF THE APPLICATION LIST TO BE *)
(* DELETED *)
(* RC O EXTERNAL RETURN CODE *)
(* = 0 OK *)
(* > 0 CRITICAL ERROR *)
(* < 0 WARNING *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* 1. KEY1 MUST BE A LISTKEY. *)
(* 2. KEY1 IS DELETED AND CAN NOT BE RECOVERED. *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* REVISED: 05/01/86 B. A. ULMER FRMI *)
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(* TO USER RECOGNIZEABLE FORM *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER FRMI *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* REVISED: 08/14/86 K. M. ROSS DBMA *)
(* ADDED A NIL POINTER CHECK FOR KEY1 *)
(* PURPOSES *)
(* *)
```

```

(*) %INCLUDE MALDA *)
(**)
  PROCEDURE MALDA(VAR  RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*)  $FUNCTION:
(*)    DELETE ALL APPLICATION LISTS THAT ARE NOT 'LOCKED'.
(*)
(*)  $DESCRIPTION OF ARGUMENTS:
(*)    NAME          I/O  DESCRIPTION
(*)    ===          ==  =====
(*)    RC            0    EXTERNAL RETURN CODE
(*)                   = 0  OK RETURN CODE
(*)                   > 0  CRITICAL ERROR
(*)                   < 0  WARNING MESSAGE
(*)
(*)  $COMMONS:
(*)    NDSGVR
(*)    STACK_OF_LISTS  I    KEY OF STACK_OF_LISTS
(*)
(*)  $ENVIRONMENT:
(*)    LANGUAGE: IBM PASCAL
(*)    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*)  $EXECUTION PROCEDURE:
(*)    MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*)
(*)  $PROCESSING DESCRIPTION:
(*)    READS THE STACK_OF_LISTS AND CALLS THE APPROPRIATE ROUTINE
(*)    TO DELETE ALL LISTS FROM THE LIST_OF_LISTS.  IF THE LIST_
(*)    OF_LISTS IS EMPTY, THE SYSTEM LIST IS DISPOSED.
(*)
(*)  $COMMENTS:
(*)    ONLY APPLICATION LISTS THAT ARE NOT LOCKED (DELTF LG = 0)
(*)    ARE DELETED.
(*)
(*)  $CHANGE CONTROL:
(*)
(*)    REVISED: 05/01/86          B. A. ULMER          W315
(*)    ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION*
(*)    TO USER RECOGNIZEABLE FORM
(*)
(*)    REVISED: 07/11/85          B. A. ULMER          W315
(*)    ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING*
(*)    PURPOSES

```

CI PS560240032U
April 1990

(* REVISED: 04/23/85 E.D. SHREVE W315 *)
(* TO DELETE ONLY UN_LOCKED APPLICATION LISTS. *)
(* ORIGINATED: 03/21/84 R. A. MCCLUSKEY W315 *)
(* *)
(* *)
(*END-----*)
(* END %INCLUDE MALDA *)

```
(* %INCLUDE MALDI *)
(**)
PROCEDURE MALDI(CONST KEY1:ANYKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DELETE AN APPLICATION LIST AND ALL LISTS AFTER IT THAT ARE
(*   NOT LOCKED.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ===          ==  =====
(*   KEY1          I    LIST TO START THE DELETE
(*   RC            0    EXTERNAL RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING MESSAGE
(*
(* $COMMONS:
(*   NDSGVR
(*   STACK_OF_LISTS  I    KEY OF STACK_OF_LISTS
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*
(* $PROCESSING DESCRIPTION:
(*   READS THE STACK_OF_LISTS AND CALLS THE APPROPRIATE ROUTINE
(*   TO DELETE ALL LISTS FROM THE LIST_OF_LISTS AFTER A SPECI-
(*   FIED LIST.
(*
(* $COMMENTS:
(*   ONLY APPLICATION LISTS THAT ARE NOT LOCKED (DELTF LG = 0)
(*   ARE DELETED.
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION*)
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          W315
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING*)
(*   PURPOSES
(*)
```

```
(*  REVISED: 04/23/85          E.D. SHREVE          W315      *)
(*      TO DELETE ONLY UN_LOCKED APPLICATION LISTS.      *)
(*  REVISED: 84/09/27          D. KERCHNER          *)
(*      CHG TO DECREMENT POSITION FOR READ FROM LIST, CHG TO *)
(*      CHECK FOR VALID POSITION NUMBER, CHG TO DELETE EACH *)
(*      EACH ENTITY FROM LIST_OF_LISTS.                  *)
(*  REVISED: 86/08/14          K. ROSS              *)
(*      ADDED A NIL POINTER CHECK FOR KEY1                *)
(*  ORIGINATED: 03/21/84        R. A. MCCLUSKEY        W315    *)
(*                                                              *)
(*END-----*)
(* END %INCLUDE MALDI *)
```

```
(* %INCLUDE MALFND. *)
(**)
PROCEDURE MALFND(CONST KEY1:ANYKEY;CONST KEY2:ENTKEY;
CONST IFIRST:LISTPSTN;VAR IPOS:LISTPSTN;VAR RC:EXT_RET_CODE);
SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION
(* FIND THE POSITION OF AN ENTITY (KEY2) IN AN APPLICATION
(* LIST (KEY1). IF KEY1 IS AN ENTITY THEN FIND ITS POSITION
(* IN THE CONSTITUENT LIST OF THAT ENTITY.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(* NAME I/O DESCRIPTION
(*
(* ==== === =====
(* KEY1 I THE KEY OF THE LIST IN WHICH KEY2 IS TO BE
(* FOUND.
(* KEY2 I THE KEY OF THE ENTITY TO BE FOUND IN KEY1.
(* IFIRST I THE POSITION IN KEY1 WHERE THE FIND
(* OPERATION IS TO START.
(* IPOS O THE POSITION IN KEY1 FWHERE KEY2 IS FOUND.
(* RC O THE FUNCTION RETURN CODE.
(*
(* $COMMONS
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL MODEL ACCESS SOFTWARE PROCEDURE
(*
(* $PROCESSING DESCRIPTION:
(* KEY1 IS EITHER AN ENTITY KEY OR A LIST KEY. IF KEY1 IS A
(* LIST, THEN KEY2 IS FOUND IN THE LIST. IF KEY1 IS AN ENTITY
(* THEN KEY2 IS FOUND IN THE CONSTITUENT LIST OF KEY1. KEY2
(* IS AN ENTITY KEY THAT IS TO BE MATCHED.
(* THE SEARCH STARTS AT POSITION IFIRST. EACH ENTITY IN KEY1
(* IS CHECKED FOR A MATCH WITH KEY2. IF MATCHED, THEN THE
(* POSITION IS RETURNED IN IPOS. IF NO MATCH, THEN IPOS IS
(* RETURNED AS ZERO AND THE RETURN CODE SIGNALS AN ERROR.
(*
(* $CHANGE CONTROL:
(* REVISED: 05/01/86 B. A. ULMER W315
```

```
(*      ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE"      *)
(*      CONDITION TO USER RECOGNIZEABLE FORM                      *)
(*                                                                  *)
(*      REVISED:  07/11/85      B. A. ULMER                      W315 *)
(*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND      *)
(*      DEBUGGING PURPOSES                                       *)
(*                                                                  *)
(*      REVISED:  03/25/85      E.D. SHREVE                      W315 *)
(*      TO CALL RDLST FROM OUTSIDE THE WHILE LOOP TO SET THE EOL. *)
(*                                                                  *)
(*      REVISED:  08/14/86      K.M. ROSS                        W315 *)
(*      ADDED A NIL POINTER CHECK FOR KEY1                       *)
(*                                                                  *)
(*      ORIGINATED: 05/07/85      D. KERCHNER                     W315 *)
(*-----*)
(**)
(* END %INCLUDE MALFND. *)
```



```

(*) %INCLUDE MALGTK. *)
(**)
  PROCEDURE MALGTK(CONST KEY1:ANYKEY;CONST IPOS:INTEGER;
    VAR KEY2:ENTKEY; VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   GET THE NTH KEY FROM A LIST. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ===          ===  ===== *)
(*)   KEY1          I    THE KEY OF ENTITY OF LIST OF ENTITIES *)
(*)                   WHOSE NTH KEY IS TO BE GOTTEN *)
(*)   IPOS          I    POSITION IN THE LIST WHERE THE TARGET *)
(*)                   ENTRY IS LOCATED *)
(*)   KEY2          O    THE KSY OF THE ENTITY AT THE NTH POSITION*)
(*)   RC            O    EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   = 1  YOU BLEW IT *)
(*)                   = 2  THE ROUTINE BLEW IT *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   1. IF KEY1 IS A LIST, GET THE IPOS ENTRY FROM THE LIST. *)
(*)   2. IF KEY2 IS AN ENTITY, GET THE IPOS ENTRY IN THE *)
(*)       CONSTITUENT LIST OF KEY1. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI *)
(*)   ADDED A CALL TO CCNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 08/28/85          B. A. ULMER          FRMI *)

```

(* CHANGE WHEN KEY2 IS SET TO NIL - BU FIX FOR HANDLING 1ST AND *)
(* 3RD PARAMETERS AS SAME KEY *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER FRMI *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)

```
(* %INCLUDE MALINS. *)
(**)
PROCEDURE MALINS(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
CONST IPOS:INTEGER;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* INSERT AN ENTITY OR LIST INTO A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* === ===
(* KEY1 I THE KEY OF ENTITY OR LIST OF ENTITIES
(* INTO TO WHICH KEY2 IS TO BE INSERTED
(* KEY2 I THE KEY OF ENTITY OR LIST OF ENTITIES TO
(* BE INSERTED INTO KEY1
(* IPOS I THE POSITION IN KEY1 TO INSERT KEY2
(* (NOTE: THE INSERT BEGINS AT IPOS-1)
(* RC O EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* 1. KEY1 AND KEY2 MAY BE LIST OR ENTITY KEYS.
(* 2. IF KEY1 IS AN ENTITY KEY, KEY2 IS INSERTED INTO THE
(* CONSTITUENT LIST OF KEY1.
(* 3. IF KEY2 IS A LIST KEY, ALL ENTITIES IN THE LIST ARE
(* INSERTED INTO KEY1.
(* 4. THE INSERT TAKES PLACE STARTING AT THE POSITION 'BEFORE'
(* IPOS IN KEY1.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
```

(* REVISED: 05/01/86 B. A. ULMER FRMI *)
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(* TO USER RECOGNIZEABLE FORM *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER FRMI *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)

```

(*) %INCLUDE MALK. *)
(**)
  PROCEDURE MALK(CONST KIND:ORD_KIND;VAR KEY1:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*)   CREATE A LIST OF ALL ENTITIES OF A SPECIFIED KIND.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ===          ==  =====
(*)   KIND          I    KIND CODE OF A CLASS COLLECTOR OR AN
(*)                   INSTANCE COLLECTOR
(*)   KEY1          O    KEY OF THE CREATED LIST OF ENTITIES
(*)   RC            O    EXTERNAL RETURN CODE
(*)                   = 0  OK RETURN CODE
(*)                   < 0  WARNING
(*)                   > 0  CRITICAL ERROR
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   THE ELEMENTS OF THE LIST WILL BE A CONCATENATION OF THE
(*)   CONTENT OF EACH ENTITY CLASS AS THEY ARE ENCOUNTERED IN
(*)   THE ENTITY CLASS STRUCTURE.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 05/01/86          B. A. ULMER          W315
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)   TO USER REOCGNIZEABLE FORM
(*)
(*)   REVISED: 07/11/85          B. A. ULMER          W315
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*)   PURPOSES
(*)
(*)

```

CI PS560240032U
April 1990

```
(*  REVISED: 05/15/85          B. A. ULMER          W315          *)
(*    FIX INCONSISTENCY IN OUTPUT LIST PROCESSING          *)
(*                                *)
(*  ORIGINATED: 04/24/84        D. J. KERCHNER        W315        *)
(*                                *)
(*-----*)
%PAGE
(**)
(* END %INCLUDE MALK. *)
```

```

(*) %INCLUDE MALKC. *)
(**)
  PROCEDURE MALKC(CONST KEY1:ANYKEY;CONST KIND:ORD_KIND;
    VAR KEY2:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   CREATE A LIST OF AN ENTITY KIND WHICH ARE FOUND WITHIN
(*)   THE CONSTITUENT LIST OF AN ENTITY OR A LIST OF ENTITIES.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ===          ==  =====
(*)   KEY1          I    THE KEY OF ENTITY OR LIST OF ENTITIES
(*)                   WHOSE USER LISTS ARE TO BE SEARCHED
(*)
(*)   KIND          I    THE KIND VALUE OF AN ENTITY OR CLASS
(*)   KEY2          O    THE KEY OF THE LIST THAT CONTAINS THE
(*)                   SELECTED ENTITIES
(*)   RC            O    EXTERNAL RETURN CODE
(*)                   = 0  OK
(*)                   > 0  CRITICAL ERROR
(*)                   < 0  WARNING
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   1. IF KEY1 IS AN ENTKEY, THEN ALL USERS OF KEY1 ARE PUT
(*)      INTO KEY2 THAT ARE OF THE GIVEN KIND.
(*)   2. IF KEY1 IS A LISTKEY, THEN ALL THE USERS OF EACH ENTITY
(*)      ON THE LIST IS PUT INTO KEY2 THAT ARE OF THE GIVEN KIND
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*)   ORIGINATED: 03/07/87      K. M. ROSS      DBMA
(*)

```

CI PS560240032U
April 1990

```
(-----*)  
(* DATA STRUCTURES/MAJOR VARIABLES: *)  
(-----*)  
(* *)  
(*END-----*)  
(* END %INCLUDE MALKC. *)  
(**)
```



```

(*) %INCLUDE MALKL. *)
(**)
  PROCEDURE MALKL(CONST KEY1:ANYKEY;CONST KIND:ORD_KIND;
    VAR KEY2:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   CREATE A LIST OF AN ENTITY KIND WHICH ARE FOUND WITHIN
(*)   ANOTHER LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ===          ==  =====
(*)   KEY1          I    THE KEY OF ENTITY OR LIST OF ENTITIES
(*)                   WHOSE IMMEDIATE CONSTITUENTS ARE TO BE
(*)                   SEARCHED
(*)   KIND          I    THE KIND VALUE OF AN ENTITY OR CLASS
(*)   KEY2          O    THE KEY OF THE LIST THAT CONTAINS THE
(*)                   SELECTED ENTITIES
(*)   RC            O    EXTERNAL RETURN CODE
(*)                   = 0  OK
(*)                   > 0  CRITICAL ERROR
(*)                   < 0  WARNING
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   1. IF KEY1 IS AN ENTKEY, THEN ALL CONSTITUENTS OF KEY1
(*)      THAT MATCH ON KIND ARE PUT INTO KEY2.
(*)   2. IF KEY1 IS A LISTKEY, THEN ALL ENTITIES ON KEY1 THAT
(*)      MATCH ON KIND ARE PUT INTO KEY2.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION

```

```
(*      TO USER RECOGNIZEABLE FORM                      *)
(*)
(*) REVISD: 12/10/85          B. A. ULMER                FRMI  (*)
(*) RETURN WARNING WHEN OUTPUT LIST IS NIL              (*)
(*)
(*) REVISD: 07/11/85          B. A. ULMER                FRMI  (*)
(*) ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND  (*)
(*) PURPOSES                                              (*)
(*)
```

```
(* %INCLUDE MALKU. *)
(**)
PROCEDURE MALKU(CONST KEY1:ANYKEY;CONST KIND:ORD_KIND;
VAR KEY2:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* CREATE A LIST OF AN ENTITY KIND WHICH ARE FOUND WITHIN
(* THE USER LIST OF AN ENTITY OR A LIST OF ENTITIES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ===
(* KEY1 I THE KEY OF ENTITY OR LIST OF ENTITIES
(* WHOSE USER LISTS ARE TO BE SEARCHED
(*
(* KIND I THE KIND VALUE OF AN ENTITY OR CLASS
(* KEY2 O THE KEY OF THE LIST THAT CONTAINS THE
(* SELECTED ENTITIES
(* RC O EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* 1. IF KEY1 IS AN ENTIKEY, THEN ALL USERS OF KEY1 ARE PUT
(* INTO KEY2 THAT ARE OF THE GIVEN KIND.
(* 2. IF KEY1 IS A LISTKEY, THEN ALL THE USERS OF EACH ENTITY
(* ON THE LIST IS PUT INTO KEY2 THAT ARE OF THE GIVEN KIND.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(* ORIGINATED: 03/07/87 K. M. ROSS DBMA
(*)
```

```
(*      DATA STRUCTURES/MAJOR VARIABLES:                                *)  
(*-----*)  
(*-----*)  
(*END-----*)  
(* END %INCLUDE MALKU. *)  
(**)
```

```
(* %INCLUDE MALN *)
(**)
  PROCEDURE MALN(CONST LSIZE:INTEGER;VAR KEYL:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   CREATE AN EMPTY LIST OF A SPECIFIED SIZE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   LSIZE          I   NUMBER OF ENTITIES IN THE LIST
(*   KEYL           O   INITIALIZED TO KEY OF EMPTY LIST
(*   RC             O   EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   A NEW APPLICATION LIST WILL BE CREATED, WITH SUFFICIENT
(*   SPACE TO ACCOMODATE 'LSIZE' ENTRIES. ALL ENTRIES ARE
(*   INITIALIZED TO NIL.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
```

April 1990

```

(*) %INCLUDE MALNO. *)
(**)
  PROCEDURE MALNO(CONST KEY1:ANYKEY;VAR KOUNT:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   COUNT THE ENTITIES ON A LIST. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ===          ===  ===== *)
(*)   KEY1          I    THE LIST WHOSE ENTRIES ARE TO BE COUNTED *)
(*)   KOUNT          O    THE NUMBER OF ENTRIES IN KEY1 *)
(*)   PC            O    EXTERNAL RETURN CODE *)
(*)                      = 0  OK *)
(*)                      > 0  CRITICAL ERROR *)
(*)                      < 0  WARNING *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   IF KEY1 IS A LIST, RETURN THE NUMBER ON THE LIST. IF KEY1 *)
(*)   IS AN ENTITY, RETURN THE NUMBER OF CONSTITUENTS. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 05/01/86          B. A. ULMER          FRMI *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TO USER RECOGNIZEABLE FORM *)
(*) *)
(*)   REVISED: 07/11/85          B. A. ULMER          FRMI *)
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)   PURPOSES *)
(*) *)

```

```
(* %INCLUDE MALNOT. *)
(**)
PROCEDURE MALNOT(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
VAR KEY3:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* CREATE AN APPLICATION LIST OF ENTITIES IN KEY1 BUT NOT IN
(* KEY2.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ====
(* KEY1 I ENTITY OR LIST OF ENTITIES WHICH WILL BE
(* 'NOTED' ~ IF ENTITY, USE CONSTITUENT LIST
(* KEY2 I ENTITY OR LIST OF ENTITIES WHICH WILL BE
(* 'NOTED' ~ IF ENTITY, USE CONSTITUENT LIST
(* KEY3 O LIST OF ENTITIES WHICH KEY1 HAS BUT KEY2
(* DOES NOT
(* RC O EXTERNAL RETURN CODE
(* = 0 OK RETURN CODE
(* = 1 YOU BLEW IT
(* = 2 THE ROUTINE BLEW IT
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* THE KEY1 LIST IS COMPARED TO THE KEY2 LIST. IF AN ENTITY
(* IS IN THE KEY1 LIST, THEN IT IS PUT ON THE OUTPUT KEY3
(* LIST. THE OUTPUT LIST WILL CONSIST OF ONLY THOSE ENTITIES
(* FOUND IN KEY1 BUT NOT IN KEY2.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(* REVISED: 05/01/86 B. A. ULMER W315
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(* TO USER RECOGNIZEABLE FORM
```

```
(*
(*) REVISD: 07/11/85          B. A. ULMER          W315          *)
(*)  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)  PURPOSES                                                         *)
(*)
(*) REVISD: 05/15/85          B. A. ULMER          W315          *)
(*)  FIX INCONSISTENCY IN OUTPUTLIST PROCESSING                      *)
(*)
(*) REVISD: 08/14/86          K. M. ROSS           W315          *)
(*)  ADDED NIL POINTER CHECK FOR KEY1                                *)
(*)
(*) ORIGINATED: 03/09/84      D. J. KERCHNER       W315          *)
(*)
(*)-----*)
%PAGE
(**)
(* END %INCLUDE MALNOT. *)
```



```

(*) %INCLUDE MALOCK *)
(**)
  PROCEDURE MALOCK(VAR LKEY:LISTKEY;CONST LOCK:INTEGER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   SET AN APPLICATION LIST FOR DELETE OR NON-DELETE STATUS. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME      I/O  DESCRIPTION *)
(*)   ====      ==  ===== *)
(*)   LKEY       I   LISTKEY *)
(*)   LOCK       I   INTEGER VALUE INDICATING LOCK SETTING *)
(*)                   = 0 SET TO 'DELETE' *)
(*)                   = 1 SET TO 'NON-DELETE' *)
(*)   RC         0   EXTERNAL RETURN CODE *)
(*)                   = 0  OK RETURN CODE *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING MESSAGE *)
(*)
(*) $COMMONS: *)
(*)   NONE *)
(*)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE. *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)   SETS A FLAG IN THE INPUT LIST TO DELETE OR NON-DELETE. *)
(*)
(*) $COMMENTS: *)
(*)   THE DELETE/NON-DELETE STATUS AFFECTS ONLY THE MALDA AND *)
(*)   MALDI INTERFACE ROUTINES. THESE ROUTINES WILL CHECK THE *)
(*)   STATUS AND NOT DELETE THE LIST IF STATUS = 1. ALL OTHER *)
(*)   DELETE FUNCTIONS (EG. MALD) DO NOT CHECK THE STATUS WHEN *)
(*)   DELETING. *)
(*)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 05/01/86      B. A. ULMER *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TO USER RECOGNIZEABLE FORM *)

```

```
(* *)
(* REVISD: 07/11/85          B. A. ULMER *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* ORIGINATED: 04/23/85      E. D. SHREVE *)
(* *)
(* *)
(*END-----*)
(* END %INCLUDE MALOCK *)
```

April 1990

```

(*) %INCLUDE MALOR *)
(**)
  PROCEDURE MALOR(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
    VAR KEY3:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   CREATE AN APPLICATION LIST FROM A BOOLEAN 'OR' ON TWO
(*)   INPUT LISTS.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KEY1           I    ENTITY OR LIST OF ENTITIES WHICH WILL BE
(*)                   'ORED' - IF ENTITY, USE CONSTITUENT LIST
(*)   KEY2           I    ENTITY OR LIST OF ENTITIES WHICH WILL BE
(*)                   'ORED' - IF ENTITY, USE CONSTITUENT LIST
(*)   KEY3           O    LIST OF ENTITIES WHICH ARE EITHER IN KEY1
(*)                   OR KEY2
(*)   RC             O    EXTERNAL RETURN CODE
(*)                   = 0  OK RETURN CODE
(*)                   < C  WARNING
(*)                   > 0  CRITICAL ERROR
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   KEY1 AND KEY2 MAY BE EITHER ENTKEYS OR LISTKEYS.
(*)   IF KEY1 IS AN ENTITY KEY, THEN ITS CONSTITUENT LIST WILL BE
(*)   'OR'ED WITH KEY2.
(*)   IF KEY2 IS AN ENTITY KEY, THEN ITS CONSTITUENT LIST WILL BE
(*)   'OR'ED WITH KEY1.
(*)   CREATE AN APPLICATION LIST, KEY3, CONTAINING ALL ENTITIES
(*)   IN EITHER OR BOTH OF TWO INPUT LISTS.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:

```

April 1990

```
(*  REVISED: 05/01/86          B. A. ULMER          W315          *)
(*  ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION  *)
(*  TO USER RECOGNIZEABLE FORM                                     *)
(*                                                                  *)
(*  REVISED: 07/11/85          B. A. ULMER          W315          *)
(*  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING  *)
(*  PURPOSES                                                       *)
(*                                                                  *)
(*  REVISED: 05/15/85          B. A. ULMER          W315          *)
(*  FIX INCONSISTENCY IN OUTPUT LIST PROCESSING                   *)
(*                                                                  *)
(*  REVISED: 08/14/86          K. M. ROSS           W315          *)
(*  ADDED A NIL POINTER CHECK FOR KEY1                             *)
(*                                                                  *)
(*  ORIGINATED: 03/09/85        D. J. KERCHNER        W315          *)
(*  -----*)
%PAGE                                                                *)
(**)
(* END %INCLUDE MALOR *)
```

```
(* %INCLUDE MALRD *)
(**)
  PROCEDURE MALRD(CONST KEY1:ANYKEY;VAR KEY2:ENTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(*
(*  $FUNCTION:
(*    READ THE NEXT ENTRY IN A DIRECTED LIST.
(*
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*
(*    NAME          I/O  DESCRIPTION
(*    ====          ==  =====
(*    KEY1           I    THE KEY OF THE DIRECTED LIST TO BE READ
(*    KEY2           O    KEY OF THE ENTITY READ
(*    RC             O    EXTERNAL RETURN CODE
(*                        = 0  OK
(*                        > 0  CRITICAL ERROR
(*                        < 0  WARNING
(*
(*
(*  $COMMONS:
(*
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*
(*  $EXECUTION PROCEDURE:
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(*
(*  $PROCESSING DESCRIPTION:
(*    THE LIST IS READ IN THE DIRECTION AS SET BY MALSTF OR
(*    MALSTR. IF KEY1 IS AN ENTKEY THEN THE NEXT CONSTITUENT
(*    IS READ. IF KEY1 IS AN APPLICATION LIST THE NEXT ENTITY
(*    IS READ.
(*
(*
(*  $COMMENTS:
(*
(*
(*  $CHANGE CONTROL:
(*
(*
(*    REVISED: 05/01/86          B. A. ULMER          FRMI
(*    ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*    TO USER RECOGNIZEABLE FORM
(*
(*
(*    REVISED: 07/11/85          B. A. ULMER          FRMI
(*    ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*    PURPOSES
(*
```

```
(* %INCLUDE MALRDE. *)
(**)
PROCEDURE MALRDE(CONST KEYL:LISTKEY;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   REMOVE DUPLICATE ENTRIES IN A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEYL          I    THE KEY OF THE LIST WHOSE DUPLICATE
(*                   ENTITIES WILL BE REMOVED
(*   RC            0    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   ANY DUPLICATE ENTITIES FOUND IN THE INPUT LIST WILL BE
(*   REMOVED.  THE CHANGE IS MADE IN-PLACE.
(*   CALLS ELDNL IN THE NDS PACKAGE.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
(*
```

```

(*) %INCLUDE MALREP *)
(**)
PROCEDURE MALREP(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) REPLACE A LIST. IF KEY1 IS AN ENTITY, THEN REPLACE THE *)
(*) CONSTITUENT LIST OF THAT ENTITY. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ---- === ===== *)
(*) KEY1 I THE KEY OF THE ENTITY OR LIST OF ENTITIES*)
(*) TO BE REPLACED - IF AN ENTITY, THEN *)
(*) USE THE CONSTITUENT LIST OF KEY1 *)
(*) KEY2 I THE KEY OF THE ENTITY OR LIST OF ENTITIES*)
(*) TO REPLACE KEY1 - IF AN ENTITY, THEN *)
(*) USE THE CONSTITUENT LIST OF KEY1 *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) KEY1 MAY BE EITHER AN ENTITY KEY OR A LIST KEY. *)
(*) IF KEY1 IS A LIST KEY, THEN KEY2 REPLACES KEY1. *)
(*) IF KEY1 IS AN ENTITY, THEN THE CONSTITUENT LIST OF KEY1 IS *)
(*) REPLACED BY KEY2. *)
(*) KEY2 MAY BE EITHER AN ENTITY KEY OR A LIST KEY. *)
(*) IF KEY2 IS A LIST KEY, THEN KEY2 REPLACES KEY1. *)
(*) IF KEY2 IS AN ENTITY, THEN THE CONSTITUENT LIST OF KEY2 *)
(*) REPLACES KEY1. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

(* REVISED: 05/01/86 B. A. ULMER FRMI *)
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(* TO USER RECOGNIZEABLE FORM *)
(* *)
(* REVISED: 10/30/85 B. A. ULMER FRMI *)
(* TAKE OUT CHECK OF DELETE RULES *)
(* *)
(* REVISED: 09/05/85 B. A. ULMER FRMI *)
(* ADDED NEW PARAMETERS TO FNDURUL FOR THE TWO NEW DELETE RULES. *)
(* *)
(* REVISED: 07/11/85 B. A. ULMER FRMI *)
(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* REVISED: 08/14/86 K. M. ROSS DBMA *)
(* ADDED A NIL POINTER CHECK KEY1 *)
(* *)


```

(*) %INCLUDE MALRMV *)
(**)
  PROCEDURE MALRMV(CONST KEY1:ANYKEY;CONST IPOS:LISTINDX;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   REMOVE AN ENTITY FROM A LIST. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ==  ===== *)
(*)   KEY1          I    THE KEY OF ENTITY OF LIST OF ENTITIES *)
(*)                   WHICH AN ENITTY WILL BE REMOVED *)
(*)   IPOS          I    THE POSITION IN KEY1 LIST WHICH THE *)
(*)                   ENTITY WILL BE REMOVED *)
(*)   RC            O    EXTERNAL RETURN CODE *)
(*)                   = 0  OK *)
(*)                   > 0  CRITICAL ERROR *)
(*)                   < 0  WARNING *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*)   1. KEY1 MAY BE AN ENTITY OR LIST KEY. *)
(*)   2. IF KEY1 IS A LIST KEY, THEN AN ENTITY IS REMOVED *)
(*)     FROM THE LIST. *)
(*)   3. IF KEY1 IS AN ENTITY KEY, THEN AN ENTITY IS REMOVED *)
(*)     FROM THE CONSTITUENT LIST OF KEY1. THE DELETE RULES *)
(*)     FOR KEY1 ARE TESTED TO INSURE THAT THE REMOVAL FROM *)
(*)     KEY1 IS PERMITTED. *)
(*)   4. IPOS IS THE POSITION NUMBER OF THE ENTITY TO BE *)
(*)     REMOVED. *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)

```

```
(*   REVISED: 05/01/86           B. A. ULMER           FRMI   *)
(*   ADDED A CALL TO CNVOSP TO CONVERT AAN "OUT OF SPACE" CONDITION *)
(*   TO USER RECOGNIZEABLE FORM                                     *)
(*                                                                    *)
(*   REVISED: 09/05/85           B. A. ULMER           FRMI   *)
(*   ADDED NEW PARAMETERS TO FNDURUL FOR THE TWO NEW DELETE RULES. *)
(*                                                                    *)
(*   REVISED: 07/11/85           B. A. ULMER           FRMI   *)
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*   PURPOSES                                                         *)
(*                                                                    *)
(*   REVISED: 10/31/84           D. J. KERCHNER         FRMI   *)
(*   INITIALIZED THE POSITION TO AN ARBITRARY #100 FOR THE DELRLST *)
(*   AND DELPLST CALLS                                               *)
(*                                                                    *)
(*   REVISED: 02/06/85           E. D. SHREVE          FRMI   *)
(*   TEST FOR INVALID IPOS ARGUMENT                                   *)
(*                                                                    *)
(*   ORIGINATED: 06/28/84         E. D. SHREVE          FRMI   *)
(*                                                                    *)
(*-----*)
%PAGE                                                                    *)
(*-----*)
(*   DATA STRUCTURES/MAJOR VARIABLES:                               *)
(*-----*)
(*                                                                    *)
(*END-----*)
(* END %INCLUDE MALRMV *)
(**)
```

```

(*) %INCLUDE MALROR *)
(**)
  PROCEDURE MALROR(VAR KEYL:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) REORDER THE APPLICATION LIST IN USER CONSTITUENT ORDER *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) === === *)
(*) KEYL I/O LIST TO BE REORDERED *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)

```

```

(*) %INCLUDE MALRORI *)
(**)
  PROCEDURE MALRORI(VAR KEYL:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION:
(*) REORDER THE APPLICATION LIST IN INCLUSIVE USER TO
(*) CONSTITUENT ORDER
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*) ===
(*) KEYL I/O LIST TO BE REORDERED
(*) RC 0 EXTERNAL RETURN CODE
(*) = 0 OK
(*) > 0 CRITICAL ERROR
(*) < 0 WARNING
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*) REVISED: MM/DD/YY CCRR I. M. THECHANG'R GROUP_ID
(*) DESCRIPTION OF LATEST CHANGE MADE.
(*)
(*) ORIGINATED: 10/14/86 K.M. ROSS DBMA
(*)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES:
(*)-----*)
(*)
(*)END-----*)
(*) END %INCLUDE MALRORI *)
(**)

```

```
(* %INCLUDE MALRPL. *)
(**)
  PROCEDURE MALRPL(CONST KEY1:ANYKEY;CONST KEY2:ENTKEY;
    CONST IPOS:LISTPSTN;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   REPLACE AN ENTITY IN A LIST.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEY1           I    THE KEY OF AN ENTITY OR LIST OF ENTITIES
(*                   WHICH WILL BE REPLACED
(*   KEY2           I    KEY OF THE ENTITY TO BE MOVED INTO KEY1
(*   IPOS           I    THE POSITION IN KEY1 WHERE KEY2 IS TO BE
(*                   PLACED
(*   RC             O    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   1. KEY1 MAY BE AN ENTITY OR LIST KEY.
(*
(*   2. IF KEY1 IS AN ENTITY KEY, THEN KEY2 WILL REPLACE THE
(*      ENTITY AT IPOS IN KEY1'S CONSTITUENT LIST.
(*
(*   3. THE KEY AT IPOS POSITION IN THE LIST IS REPLACED.
(*
(* $COMMENTS:
(*   IF THE ENTITY BEING REPLACED IN A CONSTITUENT LIST IS
(*   'MARKED FOR DELETE', THEN AN ATTEMPT WILL BE MADE TO
(*   DELETE THE ENTITY.
(*
(* $CHANGE CONTROL:
```

```
(*
(*)  REVISED: 05/01/86          B. A. ULMER          FRMI      *)
(*)  ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION  *)
(*)  TO USER RECOGNIZEABLE FORM                                     *)
(*)
(*)  REVISED: 03/20/86          B. A. ULMER          FRMI      *)
(*)  CHANGE DELRLST TO INDLST AND DELPLST WHEN TRYING TO REMOVE    *)
(*)  THE USER KEY FROM THE REPLACED ENTITY'S USER LIST            *)
(*)
(*)  REVISED: 08/ /85           L. J. BEHAN          FRMI      *)
(*)  ADD NEW PARAMETER TO DELRUL, DELENTY TO HANDLE APPLICATION    *)
(*)  LIST POSITION PROBLEM                                           *)
(*)
(*)  REVISED: 07/11/85          B. A. ULMER          FRMI      *)
(*)  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)  PURPOSES                                                        *)
(*)
```

```

(*) %INCLUDE MALRRI *)
(**)
  PROCEDURE MALRRI(VAR KEYL:LISTKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) REORDER THE APPLICATION LIST IN INCLUSIVE USFR TO *)
(*) CONSTITUENT ORDER *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) ===: ===== *)
(*) KEYL I/O LIST TO BE REORDERED *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*)
(*) $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)
(*) REVISED: MM/DD/YY CCRR I. M. THECHANGER GROUP_ID *)
(*) DESCRIPTION OF LATEST CHANGE MADE. *)
(*)
(*) ORIGINATED: 10/14/86 K.M. ROSS DBMA *)
(*)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*)
(*)END-----*)
(*) END %INCLUDE MALRRI *)
(**)

```

```
(* %INCLUDE MALRST *)
(**)
  PROCEDURE MALRST(CONST KEYE:LISTKEY;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    REMOVES ALL ENTITIES FROM AN APPLICATION LIST
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ===          ===  =====
(*    KEYE          I    LIST KEY
(*    RC            O    EXTERNAL RETURN CODE
(*                      = 0  OK RETURN CODE
(*                      > 0  CRITICAL ERROR
(*                      < 0  WARNING MESSAGE
(*
(*  $COMMONS:
(*    NONE
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE.
(*
(*  $PROCESSING DESCRIPTION:
(*    GIVEN AN APPLICATION LIST ALL ENTITIES ARE REMOVED LEAVING
(*    THE LIST SIZE INTACT
(*
(*  $COMMENTS:
(*    THIS PROCEDURE DEVELOPED SPECIFICALLY FOR THE IDB PACKAGE
(*    BUT IS FUNCTIONAL FOR ALL APPLICATIONS.
(*
(*  $CHANGE CONTROL:
(*
(*
(*    ORIGINATED: 08/14/87      K. M. ROSS
(*
(*
(*END-----*)
(* END %INCLUDE MALRST *)
```



```

(*) %INCLUDE MALRVS.*)
(**)
  PROCEDURE MALRVS(VAR KEYA:ANYKEY; VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*) REVERSE THE ORDER OF THE INPUT LIST. (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*) NAME I/O DESCRIPTION (*)
(*) ==== === ===== (*)
(*) KEYA I/O A LIST OR ENTITY KEY (*)
(*) RC 0 EXTERNAL RETURN CODE (*)
(*) = 0 OK RETURN CODE (*)
(*)
(*) $COMMONS: (*)
(*) NONE (*)
(*)
(*) $ENVIRONMENT: (*)
(*) LANGUAGE: IBM PASCAL (*)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*) IF THE INPUT KEY IS AN APPLICATION LIST, THE LIST IS (*)
(*) REVERSED. IF THE INPUT IS AN ENTITY, THE CONSTITUENT (*)
(*) LIST OF THE ENTITY IS REVERSED. (*)
(*)
(*) $COMMENTS: (*)
(*) NONE (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)
(*) REVISED: 05/01/86 B. A. ULMER W315 (*)
(*) ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION (*)
(*) TO USER RECOGNIZEABLE FORM (*)
(*)
(*) ORIGINATED: 04/11/86 MAS2 E. D. SHREVE W315 (*)
(*)
(*)-----*)
(*END-----*)
(*) END %INCLUDE MALRVS. *)

```

```
(* %INCLUDE MALSRT *)
(**)
PROCEDURE MALSRT(CONST KEY:ANYKEY; CONST PROCNAME:ROUTINE;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION: GIVEN THE USER DEFINED ORDER FUNCTION THE LIST PASSED*)
(* IN AS INPUT WILL BE SORTED USING THIS FUNCTION *)
(*
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* KEY I THE KEY OF THE ENTITY OR APPLICATION LIST*)
(* OF ENTITIES TO BE SORTED *)
(* PROCNAME I THE NAME OF THE USER DEFINED FUNCTION *)
(* FOR THE ORDERING OF THE LIST *)
(* RC O EXTERNAL RETURN CODE *)
(* = 0 OK *)
(* > 0 CRITICAL ERROR *)
(* < 0 WARNING *)
(*
(* $COMMONS: *)
(*
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*
(* $EXECUTION PROCEDURE: *)
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*
(* $PROCESSING DESCRIPTION: *)
(* THE USER SENDS IN THE ORDER FUNCTION, THEN THIS ROUTINES *)
(* REFERENCES THE USER DEFINED FUNCTION TO ACT UPON THE ENTITIES*)
(* BEING SORTED. *)
(*
(* $COMMENTS: *)
(*
(* $CHANGE CONTROL: *)
(*
(* REVISED: MM/DD/YY I M THECHANGER GROUP *)
(* REASON FOR CHANGING THE ROUTINE *)
(*
(* ORIGINATED: 04/ /86 B. A. ULMER FRMI *)
(*
(*-----*)
```

CI PS560240032U
April 1990

```
( *-----*)  
(*   DATA STRUCTURES/MAJOR VARIABLES:   *)  
(*-----*)  
(*                                       *)  
(*END-----*)  
(* END %INCLUDE MALSRT *)  
(**)
```

```
(* %INCLUDE MALSTF *)
(**)
  PROCEDURE MALSTF(CONST KEY1:ANYKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    INITIALIZE FOR READING A DIRECTED LIST IN FORWARD ORDER.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ===          ==  =====
(*    KEY1          I    THE KEY OF AN ENTITY OR LIST OF ENTITIES
(*                     WHOSE READ DIRECTION WILL BE SET TO
(*                     FORWARD
(*    RC            O    EXTERNAL RETURN CODE
(*                     = 0  OK
(*                     > 0  CRITICAL ERROR
(*                     < 0  WARNING
(*
(*  $COMMONS:
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(*  $PROCESSING DESCRIPTION:
(*    IF KEY1 IS AN ENTITY THEN THE CONSTITUENT LIST OF KEY1
(*    WILL BE INITIALIZED. IF KEY1 IS A LISTKEY THEN THE LIST
(*    POINTED TO WILL BE INITIALIZED. IN EITHER CASE THE
(*    <.POSITION> ELEMENT IS SET TO THE VALUE 1 AND THE
(*    <.DIRECTION> ELEMENT IS SET TO THE VALUE <FORWARD>.
(*
(*  $COMMENTS:
(*    USES NDS FUNCTION LSTLNM.
(*
(*  $CHANGE CONTROL:
(*
(*    REVISED: 05/01/86          B. A. ULMER          FRMI
(*    ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*    TO USER RECOGNIZEABLE FORM
(*
(*    REVISED: 07/11/85          B. A. ULMER          FRMI
(*)
```

(* ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(* PURPOSES *)
(* *)
(* REVISED: 08/14/86 K. M. ROSS DBMA *)
(* ADDED A NIL POINTER CHECK FOR KEY1 *)
(* *)

```
(* %INCLUDE MALSTR *)
(**)
PROCEDURE MALSTR(CONST KEY1:ANYKEY;VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   INITIALIZE FOR READING A DIRECTED LIST IN REVERSE ORDER.
(*   MAS INTERFACE PACKAGE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   KEY1          I    THE KEY OF AN ENTITY OR LIST OF ENTITIES
(*                   WHSOE READ DIRECTION WILL BE SET TO
(*                   REVERSE
(*   RC            O    EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   IF KEY1 IS AN ENTITY THEN THE CONSTITUENT LIST OF KEY1
(*   WILL BE INITIALIZED. IF KEY1 IS A LISTKEY THEN THE LIST
(*   POINTED TO WILL BE INITIALIZED. IN EITHER CASE THE
(*   <.POSITION> ELEMENT IS SET TO THE LENGTH OF THE LIST AND
(*   THE <.DIRECTION> ELEMENT IS SET TO THE VALUE <REVERSE>.
(*
(* $COMMENTS:
(*   USES NDS FUNCTION LSTLNM.
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          FRMI
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 08/14/86          K. M. ROSS          DBMA
(*   ADDED A NIL POINTER CHECK FOR KEY1
```

```
(* %INCLUDE MALXEQ *)
(**)
PROCEDURE MALXEQ(CONST KEY1:ANYKEY;VAR DATAREC:BLKDATA;
CONST PROCNAME:ROUTINE;VAR KEY2:LISTKEY;VAR RCC:EXT_RET_CODE;
VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* EXECUTE A PROCEDURE ON AN ENTITY, OR A LIST OF ENTITIES.
(* CONSTRUCT AN OUTPUT LIST OF ENTITIES AS DETERMINED BY THE
(* APPLICATION PROCEDURE.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ==== === =====
(* KEY1 I ENITIY OR LIST OF ENTITIES TO BE
(* PROCESSED
(* DATAREC I/O APPLICATION DEFINED DATA STRUCTURE WHICH
(* EITHER SUPPLIES OR RECIEVES VALUES
(* OPERATED ON BY THE APPLICATION PROCEDURE
(* PROC I ENTRY POINT OF APPLICATION DEFINED
(* PROCEDURE
(* KEY2 O KEY OF THE LIST CREATED
(* FOR THIS ROUTINE
(* RCC O USER DEFINED PROCEDURE RETURN CODE
(* = 0,1 OK RETURN CODE
(* = 2-7 PROCEDURE WARNING CODE
(* = 8-15 PROCEDURE ERROR CODE
(* RC O EXTERNAL RETURN CODE
(* = 0 OK RETURN CODE
(* < 0 WARNING
(* > 0 CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(* THE USER SENDS IN THE NECESSARY INFORMATION, THEN THIS
(* ROUTINE REFERENCES THE USER'S SPECIFIED PROCEDURE TO ACT
(*)
```

```
(*      UPON THE INFORMATION HE HAS SUPPLIED TO THE PROCEDURE.      *)
(*)
(*) $COMMENTS:                                                         *)
(*)
(*) $CHANGE CONTROL:                                                  *)
(*)   REVISED: 05/01/86          B. A. ULMER          W315          *)
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)   TTO USER RECOGNIZEABLE FORM                                     *)
(*)
(*)   REVISED: 01/20/86          B. A. ULMER          W315          *)
(*)   ADD CAPABILITY TO READ THE INPUT LIST IN REVERSE IN ORDER    *)
(*)   TO PROCESS                                                       *)
(*)
(*)   REVISED: 07/11/85          B. A. ULMER          W315          *)
(*)   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)   PURPOSES                                                         *)
(*)
(*)   REVISED: 05/15/85          B. A. ULMER          W315          *)
(*)   FIX INCONSISTENCY IN OUTPUT LIST PROCESSING                   *)
(*)
(*)   REVISED: 03/06/85          B. A. ULMER          W315          *)
(*)   FIX APPLICATION LIST PROBLEM                                    *)
(*)
(*)   REVISED: 11/28/84          D. J. KERCHNER        W315          *)
(*)   MALXEQ MADE FROTRAN CALLABLE BY USING INTERMEDIATE ASSEMBLER *)
(*)   ROUTINE (PASASM)                                                 *)
(*)
(*)   ORIGINATED: 04/24/84        D. J. KERCHNER        W315          *)
(*)
(*)-----*)
%PAGE                                                                    *)
(**)
(*) END %INCLUDE MALXEQ *)
```



```
(* %INCLUDE MAQURY *)
(**)
  PROCEDURE MAQURY(CONST KEY1:ENTKEY; CONST FLGNAME:NAMTYP; VAR
    FLGVAL:INTEGER; VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DESCRIPTION OF WHAT THIS ROUTINE DOES.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   KEY1          I    ENTITY WHOSE SPECIFIED FLAG VALUE IS TO
(*                   DETERMINED
(*   FLGNAME       I    FLAG NAME (STRING(6))
(*   FLGVAL        O    VALUE OF THE SPECIFIED FLAG
(*                   =1 TRUE
(*                   =0 FALSE
(*   RC            O    EXTERNAL RETURN CODE
(*                   = 0 OK RETURN CODE
(*                   < 0 WARNING
(*                   > 0 CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DETERMINE WHICH APPLICATION ACCESSIBLE FLAG'S VALUE IS TO
(*   BE GOTTEN AND THEN GET THE FLAG VALUE
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 05/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 07/11/85          B. A. ULMER          W315
```

CI PS560240032U
April 1990

```
(*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING  *)
(*      PURPOSES                                                         *)
(*                                                                           *)
(*      ORIGINATED: 05/21/85      B. A. ULMER      W315      *)
(*                                                                           *)
(*-----*)
%PAGE
(* END %INCLUDE MAQURY *)
```

```
(* %INCLUDE MARDLT *)
(**)
  PROCEDURE MARDLT(CONST KIND:ORD_KIND;VAR RC:EXT_RET_CODE);SUBPROGRAM;
%PAGE
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)   THIS SUBPROGRAM REMOVES THE RUNTIME SUBSCHEMA FROM A (*)
(*)   PARTICULAR KIND IN THE WORKING FORM (*)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ====          ==  ===== (*)
(*)   KIND          I    KIND VALUE FOR WHICH THE SUBSCHEMA (*)
(*)                   DEFINITION WILL BE DELETED (*)
(*)   RC            0    EXTERNAL RETURN CODE (*)
(*)                   < 0  WARNING (*)
(*)                   = 0  OK (*)
(*)                   > 0  CRITICAL ERROR (*)
(*)
(*) $COMMONS: (*)
(*) (*) (*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*) (*) (*)
(*) $EXECUTION PROCEDURE: (*)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)
(*)   ORIGINATED: 03/07/87      K. M. ROSS      W315 (*)
(*)
(*)-----*)
(* END %INCLUDE MARDLT *)
```

```

(*) %INCLUDE MARSGT *)
(**)
  PROCEDURE MARSGT(CONST KIND:ORD_KIND; VAR SCHPTR:T_SCHEMA_POINTER;
    VAR RC:EXT_RET_CODE);SUBPROGRAM;
%PAGE
(*)-----*)
(*)
(*) $FUNCTION:
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KIND          I    KIND VALUE FOR WHICH THE SUBSCHEMA
(*)                   DEFINITION WILL BE CREATED
(*)   SCHPTR        I/O  KEY TO THE DATA DEFINING THE SUBSCHEMA TO
(*)                   BE CREATED
(*)   RC            O    EXTERNAL RETURN CODE
(*)                   < 0  WARNING
(*)                   = 0  OK
(*)                   > 0  CRITICAL ERROR
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 05/01/86          B. A. ULMER          W315
(*)   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*)   TO USER RECOGNIZEABLE FORM
(*)
(*)   ORIGINATED: 10/18/85        B. A. ULMER          W315
(*)
(*)-----*)
(*) END %INCLUDE MARSGT *)

```

```
(* %INCLUDE MASALOC *)  
(**)  
  PROCEDURE MASALOC(CONST SIZE:INTEGER; VAR REGVAL:POINTER;  
    VAR RC:INTEGER);FORTRAN;  
(**)  
(* END %INCLUDE MASALOC *)
```

```

(** %INCLUDE MASDSP *)
(**)
PROCEDURE MASDSP(  VAR ENT_PTR:  POINTER;
                   CONST TYPE_SIZE:  INTEGER);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   DISPOSE OF A MAS DYNAAMICALLY ALLOCATED MEMORY AREA.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   ====      ==  =====
(*   TYPE_SIZE  I   THE SIZE OF THE AREA TO BE DISPOSED
(*   ENT_PTR    I   POINTER TO THE MEMORY AREA TO BE DISPOSED
(*   RC         O   EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(* $COMMONS:
(*   $PCMGR     HOLDS THE DESCRIPTORS FOR THE MAS MEMORY AREAS.
(*
(* $ENVIRONMENT:
(*   LANGUAGE:  IBM PASCAL
(*   HARDWARE SYSTEM:  IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DELETE A BLOCK AND COMBINE IT WITH ANY CONTIGIOUS BLOCKS
(*   OF FREED MEMORY.
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 02/06/86      B. A. ULMER      FRMI
(*   ADDED CODE TO HANDLE WHEN THE 8K OVERFLOW BLOCK NEEDS FREED
(*   (JUST REMOVE IF FROM THE BLOCK CHAIN AND SET OVERFLOW FLAG TO
(*   FALSE)
(*
(*   REVISED: 08/ /85      B. A. ULMER      FRMI
(*   FIX BUG DEALING WITH THE PRESENCE OF AN INFINITE LOOP
(*

```

```
(*  REVISED: 07/11/85          B. A. ULMER          FRMI      *)
(*  ELIMINATE THE LEAVE AND MAX FUNCTIONS FOR BETTER COMPATABILITY *)
(*  WITH THE DEC VAX          *)
(*  *)
(*  ORIGINATED: 12/10/84      J. J. JOHNSON          FRMI      *)
(*  *)
(*  -----*)
%PAGE                                     *)
(*  -----*)
(*  DATA STRUCTURES/MAJOR VARIABLES:          *)
(*  -----*)
(*  *)
(*END-----*)
(* END %INCLUDE MASDSP *)
(**)
```

```
(* %INCLUDE MASMSZ *)
(**)
PROCEDURE MASMSZ(VAR MODSIZ:INTEGER; VAR FRESIZ:INTEGER;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(* RETURNS THE ACTUAL MODEL SPACE USED AND THE AMOUNT OF
(* FREE SPACE IN THE ALLOCATED MEMORY BLOCKS OF THE MODEL.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ===
(* MODSIZ 0 TOTAL BYTES OF USED MODEL SPACE
(* FRESIZ 0 TOTAL BYTES OF FREE SPACE IN THE
(* ALLOCATED MODEL BLOCKS.
(* RC 0 EXTERNAL RETURN CODE
(* = 0 OK RETURN CODE
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(* NONE
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(* USED ONLY WITH THE MAS MEMORY MANAGER. CAN NOT BE USED
(* WITH THE PASCAL MEMORY MANAGER.
(*
(* $PROCESSING DESCRIPTION:
(* CALLS THE INTERNAL MAS ROUTINES THAT CALCULATE FREESPACE
(* AND MODEL SPACE USING THE MAS MEMORY MANAGER CONTROL BLOCKS.
(*
(* $COMMENTS:
(* IF THIS PROCEDURE IS TO BE USED WITH THE PASCAL MEMORY
(* MANAGER, THEN A SPECIAL PROCEDURE 'NDSFCT' IS REQUIRED.
(*
(* $CHANGE CONTROL:
(*
(* REVISED: 05/01/86 B. A. ULMER W315
(* ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
```


CI PS560240032U
April 1990

```
(*      TO USER RECOGNIZEABLE FORM                      *)
(*)                                                    (*)
(*)  ORIGINATED: 04/09/85      E. SHREVE                  W315  (*)
(*)                                                    (*)
(*)-----(*)
(*)                                                    (*)
(*END-----*)
(* END %INCLUDE MASMSZ *)
```

```
(* %INCLUDE MASNEW *)
(**)
PROCEDURE MASNEW(  VAR ENT_PTR:  POINTER;
                   CONST TYPE_SIZE:  INTEGER;
                   VAR  RR:  RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    ALLOCATES A NEW DYNAMIC MEMORY AREA FOR MAS ELEMENTS.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME          I/O  DESCRIPTION
(*    ===          ==  =====
(*    TYP_SIZE      I    THE SIZE OF THE MEMORY REGION REQUIRED
(*    ENT_PTR       O    POINTER TO THE AREA OBTAINED
(*    RR            O    EXTERNAL RETURN CODE
(*                      = 0  OK
(*                      > 0  CRITICAL ERROR
(*                      < 0  WARNING
(*
(*  $COMMONS:
(*    $PCMGR        HOLDS THE DESCRIPTORS FOR THE MAS MEMORY SPACE.
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(*  $PROCESSING DESCRIPTION:
(*    ATTEMPTS TO LOCATE A FREE SPACE, STARTING AT THE FIRST
(*    ALLOCATED REGION, AND CONTINUES THRU ALL ALLOCATED REGIONS.
(*    IF FOUND, IT REMOVES THE REGION FROM THE FREE SPACE CHAIN.
(*    IF NO SPACE EXISTS, IT ALLOCATES A NEW REGION AND CONNECTS
(*    THE NEW REGION TO THE LAST.
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
(*    REVISED: 02/06/86          B. A. ULMER          FRMI
(*    ADDED CODE TO HANDLE A FAILURE ON GETMAIN IN ROUTINE MASALOC
(*    AND PROCESSING OF 8K OVERFLOW BLOCK
(*
```

CI PS560240032U
April 1990

```
(*  REVISED: 07/11/85          B. A. ULMER          FRMI      *)
(*  ELIMINATE THE LEAVE AND MAX FUNCTIONS TO BETTER COMPATIBILITY *)
(*  WITH THE DEC VAX          *)
(*  *)
(*  ORIGINATED: 12/10/84      J. J. JOHNSON          FMRI      *)
(*  *)
(*-----*)
%PAGE                                          *)
(*-----*)
(*  DATA STRUCTURES/MAJOR VARIABLES:          *)
(*END-----*)
(* END %INCLUDE MASNEW *)
(**)
```

```

(*) %INCLUDE MASOVR *)
(**)
PROCEDURE MASOVR( CONST $SIZE: INTEGER; VAR ENT_PTR: POINTER;
                  VAR OSPACE: $CBP);EXTERNAL;
(**)
(*)-----*)
(*)*)
(*) $FUNCTION: *)
(*) *)
(*) *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) === === *)
(*) RC 0 EXTERNAL RETURN CODE *)
(*) = 0 OK *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*) *)
(*) $COMMONS: *)
(*) $PCMGR HOLDS THE DESCRIPTORS FOR THE MAS MEMORY AREAS. *)
(*) *)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*) REVISED: 07/11/85 B. A. ULMER FRMI *)
(*) ELIMINATE THE LEAVE AND MAX FUNCTIONS FOR BETTER COMPATABILITY *)
(*) WITH THE DEC VAX *)
(*) *)
(*) ORIGINATED: 3/21/86 B. A. ULMER FRMI *)
(*) *)
(*)-----*)
%PAGE *)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*) *)
(*)END-----*)
(*) END %INCLUDE MASDSP *)
(**)

```

```
(* %INCLUDE MAUPDT *)
(**)
PROCEDURE MAUPDT(VAR KEY1:ANYKEY; CONST FLGNAME:NAMTYP; CONST
  FLGVAL:INTEGER; VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   UPDATE A SPECIFIED APPLICATION ACCESSIBLE FLAG VALUE
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   =====
(*   KEY1           I    ENTITY OR LIST OF ENTITIES WHOSE
(*                       SPECIFIED FLAG VALUE IS TO BE UPDATED
(*   FLGNAME        I    FLAG NAME (STRING(6))
(*   FLGVAL         I    VALUE TO BE USED WHEN UPDATING THE FLAG
(*                       = 1 TRUE
(*                       = 0 FALSE
(*   RC             0    EXTERNAL RETURN CODE
(*                       = 0 OK RETURN CODE
(*                       < 0 WARNING
(*                       > 0 CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   DETERMINE WHICH OF THE APPLICATION ACCESSIBLE FLAGS IS TO BE
(*   UPDATED AND THEN UPDATE IT WITH THE INPUT VALUE
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*   REVISED: 08/21/85          B. A. ULMER          FRMI
(*   CHANGED TO NOT ALLOW APPLICATION TO SET AN ENTITY FOR MARK
(*   DELETE
(*
(*   REVISED: 07/11/85          B. A. ULMER          FRMI
(*   ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*   PURPOSES
```

CI PS560240032U
April 1990

(* *)
(* ORIGINATED: 05/21/85 B. A. ULMER FRMI *)
(* *)
(* ----- *)
%PAGE *)
(* END %INCLUDE MAUPDT *)

```
(* %INCLUDE MIDBD. *)
(**)
PROCEDURE MIDBD(VAR KEY1:ANYKEY; VAR RC:EXT_RET_CODE);SUBPROGRAM;

(**)
(*-----*)
(*
(* WARNING:  FOR IDB USE ONLY
(* MAY CONTAMINATE MODEL IF USING DELETE WITH NO DELETE RULES
(*
(* $FUNCTION:
(*   DELETE AN ENTITY OR LIST OF ENTITIES BUT DO NOT CONSIDER
(*   THE DELETE RULES
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   KEY1          I    ENTITY OR LIST OF ENTITIES TO BE DELETED
(*   RC            O    EXTERNAL RETURN CODE
(*                      = 0  OK RETURN CODE
(*                      < 0  WARNING
(*                      > 0  CRITICAL ERROR
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(* $PROCESSING DESCRIPTION:
(*   IF KEY1 IS AN ENTKEY THEN
(*     DELETE THE ENTITY
(*   IF KEY1 IS A LISTKEY THEN
(*     DELETE EACH ENTITY ON THE LIST
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*   REVISED: 05/01/86          B. A. ULMER          W315
(*   ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION
(*   TO USER RECOGNIZEABLE FORM
(*
(*   REVISED: 04/22/86          E. D. SHREVE          W315
```

```
(*      CHANGED TO CALL XIELM INSTEAD OF DELENTY TO PERFORM THE DELETE *)
(*      AND CHANGE INPUT TO VAR.                                         *)
(*                                                                           *)
(*      REVISED: 08/04/85          L. J. BEHAN          W315             *)
(*      ADD NEW PARAMETER TO DELENTY FOR HANDLING OF APPLICATION         *)
(*      LIST POSITION PROBLEM                                             *)
(*                                                                           *)
(*      REVISED: 07/11/85          B. A. ULMER          W315             *)
(*      ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING    *)
(*      PURPOSES                                                         *)
(*                                                                           *)
(*      ORIGINATED: 06/17/85       B. A. ULMER          W315             *)
(*                                                                           *)
(*-----*)
(**)
(* END %INCLUDE MIDBD. *)
```



```
(* %INCLUDE MIDBRV *)
(**)
PROCEDURE MIDBRV(CONST KEY1:ANYKEY;CONST IPOS:LISTINDX;
  VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*  WARNING: FOR IDB USE ONLY                                *)
(*  MAY CONTAMINATE MODEL IF USING REMOVE WITHOUT DELETE RULES *)
(*  *)
(*  $FUNCTION:                                                *)
(*  REMOVE AN ENTITY FROM A LIST WITHOUT CONSIDERING THE    *)
(*  DELETE RULES                                             *)
(*  *)
(*  $DESCRIPTION OF ARGUMENTS:                                *)
(*  NAME      I/O  DESCRIPTION                                *)
(*  ===      ==  =====*)
(*  KEY1      I   THE KEY OF AN ENTITY OR LIST OF ENTITIES *)
(*              FROM WHICH AN ENTITY WILL BE REMOVED      *)
(*  IPOS      I   THE POSITION IN KEY1 FROM WHICH THE      *)
(*              ENTITY WILL BE REMOVED                     *)
(*  RC        O   EXTERNAL RETURN CODE                    *)
(*              = 0  OK                                     *)
(*              > 0  CRITICAL ERROR                        *)
(*              < 0  WARNING                                *)
(*  *)
(*  $COMMONS:                                                *)
(*  *)
(*  $ENVIRONMENT:                                            *)
(*  LANGUAGE: IBM PASCAL                                     *)
(*  HARDWARE SYSTEM: IBM 360/370/4341/4381                 *)
(*  *)
(*  $EXECUTION PROCEDURE:                                    *)
(*  MODEL ACCESS SOFTWARE INTERFACE ROUTINE                *)
(*  *)
(*  $PROCESSING DESCRIPTION:                                  *)
(*  1. KEY1 MAY BE AN ENTITY OR LIST KEY.                   *)
(*  2. IF KEY1 IS A LIST KEY, THEN AN ENTITY IS REMOVED    *)
(*     FROM THE LIST.                                       *)
(*  3. IF KEY1 IS AN ENTITY KEY, THEN AN ENTITY IS REMOVED *)
(*     FROM THE CONSTITUENT LIST OF KEY1.                  *)
(*  4. IPOS IS THE POSITION NUMBER OF THE ENTITY TO BE      *)
(*     REMOVED.                                             *)
(*  *)
(*  $COMMENTS:                                              *)
(*  *)
(*  $CHANGE CONTROL:                                         *)
```

```

(*)  REVISED: 05/01/86          B. A. ULMER          FRMI      *)
(*)  ADDED A CALL TO CNVOSP TO CONVERT AN "OUT OF SPACE" CONDITION *)
(*)  TO USER RECOGNIZEABLE FORM                                     *)
(*)                                                                    *)
(*)                                                                    *)
(*)  REVISED: 12/30/85          B. A. ULMER          FRMI      *)
(*)  CHANGE CALL FROM DELENTY TO DELRUL FOR THE CASE WHEN ENTITY IS *)
(*)  MARKED FOR DELETE                                              *)
(*)                                                                    *)
(*)  REVISED: 08/ /85           L. J. BEHAN          FRMI      *)
(*)  ADD NEW PARAMETERS TO DELENTY FOR HANDLING OF APPLICATION      *)
(*)  LIST POSITION PROBLEM                                           *)
(*)                                                                    *)
(*)  REVISED: 07/11/85          B. A. ULMER          FRMI      *)
(*)  ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING *)
(*)  PURPOSES                                                       *)
(*)                                                                    *)
(*)  ORIGINATED: 06/19/85        B. A. ULMER          FRMI      *)
(*)                                                                    *)
(*)-----*)
%PAGE                                                                *)
(*)-----*)
(*)  DATA STRUCTURES/MAJOR VARIABLES:                             *)
(*)-----*)
(*)                                                                    *)
(*)-----*)
(*END-----*)
(*) END %INCLUDE MIDBRV *)
(**)

```

```

(*) %INCLUDE MOVRLSM *)
(**)
  PROCEDURE MOVRLSM(CONST FROM_LIST:LISTPNTR;
    CONST FROM_POSITION:LISTPSTN;VAR TO_LIST:LISTPNTR;
    CONST TO_POSITION:LISTPSTN;CONST ENTCount:LISTSIZE;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*)      AUTHOR:  UNKNOWN          CADD      CREATED: YY/MM/DD CC  *)
(*)      VERSION: MAS VER 2              REVISED: 84/10/11 CC  *)
(*)
(*)      FUNCTION:
(*)      MOVE ENTITIES BETWEEN SYSTEM LISTS.
(*)
(*)      ENVIRONMENT:
(*)      IBM PASCAL LANGUAGE
(*)      IBM 30XX, 43XX, DEC VAX 11/780
(*)
(*)      DESCRIPTION OF ARGUMENTS:
(*)      NAME      I/O  DESCRIPTION
(*)      FROM_LIST
(*)      I  POINTER TO A SYSTEM LIST.
(*)      FROM_POSITION
(*)      I  THE RELATIVE POSITION OF THE FIRST ENTITY TO
(*)      BE MOVED.
(*)      TO_LIST  I  POINTER TO A SYSTEM LIST.
(*)      TO_POSITION
(*)      I  THE RELATIVE POSITION IN THE LIST TO WHICH
(*)      THE ENTITIES WILL BE MOVED.
(*)      ENTCount I  THE NUMBER OF ENTITIES TO MOVE.
(*)      RR      0   ERROR CONDITION RETURN CODE.
(*)      = 0   NORMAL RETURN CODE.
(*)      = 14  BAD_LIST_POSITION
(*)      = 16  BAD_LIST_MOVE_COUNT
(*)      = 17  BAD_LIST_REFERENCE
(*)
(*)      COMMONS:
(*)
(*)      PROCESSING DESCRIPTION:
(*)      MOVRLSM USES AMPXMOVE A SYSTEM ROUTINE. AMPXMOVE MOVES
(*)      DATA FROM MEMORY TO MEMORY (NUMBER OF BYTES TO BE MOVED
(*)      HAS TO BE SPECIFIED).
(*)
(*)      COMMENTS:
(*)

```

```
(*      CHANGE CONTROL:                                *)
(*      84/10/11  MAS VER 2  D. J. KERCHNER            *)
(*      UPDATED DOCUMENTATION.                          *)
(*      84/10/04  MAS VER 2  E. D. SHREVE              *)
(*      CHANGED DECLARATION OF 'TO_LIST' TO VAR.        *)
(*      -----*)
(**)
(* END %INCLUDE MOVRLSM *)
```

```
(* %INCLUDE MRGTLSM. *)
(**)
  PROCEDURE MRGTLSM(VAR LIST1:LISTPNTR;CONST LIST2:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CONCATENATE THE ENTITIES IN LIST2 TO LIST1.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LIST1, LIST2 - TWO LIST POINTERS.
(*
(*      OUTPUT
(*      RR      - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE MRGTLSM. *)
```

```
(* %INCLUDE MRGTNM. *)
(**)
  PROCEDURE MRGTNM(CONST KEYL1:LISTKEY;CONST KEYL2:LISTKEY;
    VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CONCATENATE THE ENTITIES IN LIST2 TO LIST1.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE1      - KEY OF THE APPLICATION LIST.  IF ENTITY KEY,
(*                  THEN USE CONSTITUENT LIST.
(*      KEYE2      - KEY OF THE APPLICATION LIST TO BE
(*                  CONCATENATED.  IF ENTITY KEY, THEN USE
(*                  CONSTITUENT LIST.
(*
(*      OUTPUT
(*      RR          - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE MRGTNM. *)
```

```
(* %INCLUDE MRKNM. *)
(**)
  PROCEDURE MRKNM(VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                          *)
(*      FUNCTION                                          *)
(*      MARK THE STACK OF LISTS SO THAT THE NEXT RELEASE LIST  *)
(*      WILL ONLY DESTROY LISTS CREATED AFTER THIS MARK OPERATION. *)
(*                                          *)
(*      LANGUAGE                                          *)
(*      PASCAL.                                          *)
(*                                          *)
(*      PACKAGE                                          *)
(*      LIST PACKAGE.                                  *)
(*                                          *)
(*      ARGUMENTS                                          *)
(*      INPUT                                          *)
(*      NONE      -                                          *)
(*                                          *)
(*      OUTPUT                                          *)
(*      RR      - THE FUNCTION RETURN RECORD.          *)
(*-----*)
(**)
(* END %INCLUDE MRKNM. *)
```

```

(*) %INCLUDE MRSCR *)
(**)
  PROCEDURE MRSCR(CONST KIND:ORD_KIND; VAR SCH_SIZE:INTEGER;
    VAR RTSS:T_SCHEMA_POINTER;VAR RC:EXT_RET_CODE);SUBPROGRAM;
%PAGE
(*)-----*)
(*)
(*) $FUNCTION:
(*) THIS SUBPROGRAM IS GIVEN A COPY OF THE RUNTIME SUBSCHEMA FOR
(*) A PARTICULAR KIND THIS COPY IS APPENDED TO THE ADB FOR THE
(*) KIND COLLECTOR AND ITS OFFSET INTO IT IS RETURNED
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*) NAME I/O DESCRIPTION
(*)
(*) === === =====
(*) KIND I KIND VALUE FOR WHICH THE SUBSCHEMA
(*) DEFINITION WILL BE CREATED
(*) SCH_SIZE I SIZE OF THE RUN TIME SUBSCHEMA
(*) TO BE CREATED
(*) RTSS I/O KEY OF RUNTIME SUBSCHEMA TO BE ATTACHED
(*) AND ITS NEW LOCATION
(*) RC O RETURN CODE
(*) < 0 WARNING
(*) = 0 OK
(*) > 0 CRITICAL ERROR
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*) ORIGINATED: 03/07/87 K. M. ROSS W315
(*)
(*)-----*)
(*) END %INCLUDE MRSCR *)

```



```

(*) %INCLUDE MSINIT. *)
(**)
  PROCEDURE MSINIT(VAR SIZE:INTEGER;
                   VAR RC:EXT_RET_CODE);SUBPROGRAM;
(**)
(*-----*)
(*)
(*) $FUNCTION: (*)
(*)   INITIALIZE THE MAS NETWORK WITH AN INITIAL MODEL SIZE. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: (*)
(*)   NAME          I/O  DESCRIPTION (*)
(*)   ===          ===  ===== (*)
(*)   SIZE          I    A NUMBER REPRESENTING HOW LARGE AN (*)
(*)                   INITIAL MODEL SIZE TO CREATE (*)
(*)   RC            O    EXTERNAL RETURN CODE (*)
(*)                   = 0  OK RETURN CODE (*)
(*)                   < 0  WARNING (*)
(*)                   > 0  CRITICAL ERROR (*)
(*)
(*) $COMMONS: (*)
(*)
(*) $ENVIRONMENT: (*)
(*)   LANGUAGE: IBM PASCAL (*)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 (*)
(*)
(*) $EXECUTION PROCEDURE: (*)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE (*)
(*)
(*) $PROCESSING DESCRIPTION: (*)
(*)
(*) $COMMENTS: (*)
(*)
(*) $CHANGE CONTROL: (*)
(*)   REVISED: (*)
(*)   CHANGE : (*)
(*)
(*)   ORIGINATED: 02/17/88      K. M. ROSS      W315 (*)
(*)
(*)-----*)
%PRINT ON
(*) END %INCLUDE MSINIT *)

```

```
(* %INCLUDE MSTART. *)
(**)
  PROCEDURE MSTART(CONST ID:INTEGER);SUBPROGRAM;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    START STATISTICS GENERATION.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME      I/O  DESCRIPTION
(*    ====      ==  =====
(*    ID         I   INDICATION OF THE STATISTICS BEING KEPT
(*                   THIS FIELD MUST CORRESPOND TO ID INPUT
(*                   TO MSTOP
(*    RC         O   EXTERNAL RETURN CODE
(*                   = 0  OK
(*                   > 0  CRITICAL ERROR
(*                   < 0  WARNING
(*
(*  $COMMONS:
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*
(*  $PROCESSING DESCRIPTION:
(*    WHEN MSTART IS CALLED, THE INTEGER EQUIVALENT VALUE OF THE
(*    MAS ROUTINE ID IS ENTERED INTO A COMMON FIELD.  ALSO, A
(*    FLAG IS SET TO ON INDICATING THAT THIS PARTICULAR MAS
(*    ROUTINE IS THE ONE CURRENTLY BEING PROCESSED.
(*
(*  $COMMENTS:
(*
(*  $CHANGE CONTROL:
(*
(*    REVISED: 07/24/86      B. A. ULMER      FRMI
(*    CH'NGE ID_FLAG FIELD OF MSTATUS TO AN INTEGER SO THAT AN APPL.
(*    USER CAN KNOW HOW MANY LEVELS HE IS NESTED
(*
(*    REVISED: 07/11/85      B. A. ULMER      FRMI
(*    ADD A NEW PARAMETER TO CNVRR FOR ERROR HANDLING AND DEBUGGING
(*    PURPOSES
```

```

(*) %INCLUDE MSTOP. *)
(**)
  PROCEDURE MSTOP(CONST ID:INTEGER);SUBPROGRAM;
(**)
  (*)-----*)
  (*)
  (*) FUNCTION
  (*) STOP STATISTICS GENERATION.
  (*)
  (*) LANGUAGE
  (*) PASCAL.
  (*)
  (*) PACKAGE
  (*) STATISTICS PACKAGE.
  (*)
  (*) ARGUMENTS
  (*) INPUT
  (*) ID - INDICATION OF TYPE OF STATISTICS BEING KEPT.
  (*) THIS FIELD MUST CORRESPOND TO ID INPUT TO
  (*) CALL TO MSTART.
  (*)
  (*) OUTPUT
  (*) NONE -
  (*)
  (*) METHOD
  (*) WHEN MSTOP IS CALLED, THE INTEGER EQUIVALENT VALUE OF THE
  (*) MAS ROUTINE ID IS ENTERED INTO A COMMON FIELD. ALSO, A
  (*) FLAG IS SET TO OFF INDICATING THAT THIS PARTICULAR MAS
  (*) ROUTINE IS NO LONGER ACTIVELY BEING PROCESSED, BUT THE ID
  (*) WILL INDICATE THAT IT WAS THE LAST ONE CALLED.
  (*)
  (*)-----*)
(**)
  (*) END %INCLUDE MSTOP. *)

```

```
(* %INCLUDE NDSCMM. *)
(**)
  PROCEDURE NDSCMM;EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      DUMMY PROGRAM DEFINES NDSREM COMMON.
(*      USED AS THE 'SEED' OF THE MAS NDS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      NETWORK PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      NONE      -
(*
(*      OUTPUT
(*      NONE      -
(*
(*      METHOD
(*      SYSTEM INCONGRUITIES FORCE NESTING OF DEF WITHIN A
(*      PROCEDURE.
(*-----*)
(**)
(* END %INCLUDE NDSCMM. *)
```

```
(* %INCLUDE NDSFCT *)
(**)
PROCEDURE NDSFCT(VAR MODSIZ:INTEGER; VAR FRESIZ:INTEGER;
VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*     COMPUTES THE AMOUNT OF USED MODEL SPACE AND THE AMOUNT OF
(*     FREESPACE IN THE ALLOCATED MEMORY BLOCKS.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ===  =====
(*     MODSIZ         0    TOTAL BYTES OF USED MODEL SPACE
(*     FRESIZ         0    NUMBER OF BYTES OF FREE SPACE.
(*     RR             0    RETURN CODE
(*                        = 0  OK RETURN CODE
(*                        > 0  CRITICAL ERROR
(*                        < 0  WARNING
(*
(* $COMMONS:
(*     PCMGT
(*     PTR          I    POINTER TO THE 1ST ALLOCATED MEMORY BLOCK.
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*     USED ONLY WITH THE MAS MEMORY MANAGER.  CAN NOT BE USED
(*     WITH THE PASCAL MEMORY MANAGER.
(*
(* $PROCESSING DESCRIPTION:
(*     EACH ALLOCATED BLOCK IS FOUND USING THE BLOCK CHAIN OF THE
(*     SPACE CONTROL BLOCK ($CB). THE FREE CHAIN IS USED TO SUM
(*     THE SIZE OF EACH FREED ENTRY. THE BLOCK SIZES OF ALL
(*     ALLOCATED BLOCKS ARE ALSO TOTALED.
(*     MODSIZ = TOTAL SPACE ALLOCATED - FREESIZE
(*
(* $COMMENTS:
(*     THE STRUCTURE OF THE MEMORY MANAGER CONTROL BLOCKS ARE
(*     DESCRIBED IN THE INCLUDE MEMBER 'PCMGT'.
(*
(* $CHANGE CONTROL:
```

CI PS560240032U
April 1990

(* CHANGED: 07/16/85 B. A. ULMER W315 *)
(* REASON: CHANGED \$PCMGT TO PCMGT FOR VAX COMPATABILITY *)
(*
(* ORIGINATED: 04/09/85 E. SHREVE W315 *)
(*
(*-----*)
(*END-----*)
(* END %INCLUDE NDSFCT *)

```
(* %INCLUDE NDSGBM. *)
(**)
  PROCEDURE NDSGBM;EXTERNAL;
(**)
(*-----*)
(*                                     *)
(*      FUNCTION                                     *)
(*      DUMMY PROCEDURE FOR COMPILE TIME INITIALIZATION OF NDS *)
(*      GLOBAL AREA.  CONTAINS NDS GLOBAL VARIABLE. *)
(*                                     *)
(*      LANGUAGE                                     *)
(*      PASCAL. *)
(*                                     *)
(*      PACKAGE                                     *)
(*      NETWORK PACKAGE. *)
(*                                     *)
(*      ARGUMENTS                                     *)
(*      INPUT *)
(*      NONE      - *)
(*                                     *)
(*      OUTPUT *)
(*      NONE      - *)
(*                                     *)
(*      COMMENT                                     *)
(*      DEFINED WITHIN THIS PROCEDURE ARE THE LIST OF ALL NETWORKS *)
(*      AND THE LIST OF ALL LISTS. *)
(*                                     *)
(*-----*)
(**)
(* END %INCLUDE NDSGBM. *)
```

```
(* %INCLUDE NDSRML *)
(**)
  PROCEDURE NDSRML;EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   RELEASE ALL MEMORY BLOCKS ALLOCATED TO THE WORKING FORM.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ==  =====
(*   NONE
(*
(* $COMMONS:
(*   $PCMGR
(*   PTR          I   POINTER TO THE FIRST ALLOCATED BLOCK.
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381 - MAS PACKAGE USING
(*   THE MODEL ACCESS MEMORY MANAGER.
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*   THIS ROUTINE CAN ONLY BE USED WITH THE MAS MEMORY MANAGER.
(*   IF THE PASCAL MEMORY MANAGER IS USED, THE ROUTINE DISPNDM
(*   MUST BE SUBSTITUTED FOR NDSRML.
(*
(* $PROCESSING DESCRIPTION:
(*   BEGINNING WITH THE POINTER IN $PCMGR, EACH MEMORY BLOCK
(*   ALLOCATED TO THE WORKING FORM IS LOCATED AND FREED.
(*
(* $COMMENTS:
(*   THE 1ST WORD OF EACH MEMORY AREA CONTAINS THE POINTER THAT
(*   CHAINS ALL WORKING FORM MEMORY AREAS.
(*
(* $CHANGE CONTROL:
(*   REVISED: 07/11/85          B. A. ULMER          W315
(*   CHANGED $PCMGT TO PCMGT FOR VAX COMPATABILITY
(*
(*   ORIGINATED: 04/05/85      E.D.  SHREVE          W315
(*-----*)
```



```
(*-----*)
(*  DATA STRUCTURES/MAJOR VARIABLES:  *)
(*      THE INCLUDE MEMBER '$PCMGR' DESCRIBES THE STRUCTURE OF THE  *)
(*      CONTROL BLOCKS THAT CONTROL THE MEMORY AREAS AND LINKS THEM *)
(*      TOGETHER.  *)
(*-----*)
(*  *)
(*END-----*)
(* END %INCLUDE NDSRML *)
```

```
(* %INCLUDE NEWCRB *)
(**)
  PROCEDURE NEWCRB(VAR CRB:CRBPNTN; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*
(*   AUTHOR:  B. A. ULMER           FRMI   CREATED: 85/02/08  CC??*)
(*   VERSION: XXXX                   REVISED: YY/MM/DD  CC  *)
(*
(*   FUNCTION:
(*       CREATE A CRB
(*
(*   ENVIRONMENT:
(*       IBM PASCAL LANGUAGE
(*       IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(*   EXECUTION PROCEDURE:
(*       HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(*   DESCRIPTION OF ARGUMENTS:
(*       NAME      I/O  DESCRIPTION
(*       CRB       I/O  CONSTITUENT READ BLOCK ADDRESS
(*       RR        0    ERROR CONDITION RETURN CODE
(*                   = 0  OK RETURN CODE
(*                   = 1  YOU BLEW IT
(*                   = 2  THE ROUTINE BLEW IT
(*
(*   COMMONS:
(*       COM1
(*       VAR1      I    VAR1 NAME MUST BE FILLED, CHARACTER DATA
(*                   MUST BE PROVIDED
(*       VAR2      I    VAR2 MUST BE SPECIFIED
(*       COM2
(*       VAR3      I    CHARACTER DATA MUST BE SPECIFIED
(*
(*   PROCESSING DESCRIPTION:
(*       DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*       FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(*   COMMENTS:
(*       TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND
(*       THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
(*   CHANGE CONTROL:
(*       YY/MM/DD  CCZZ  I. M. THECHANGER
(*       DESCRIPTION OF LATEST CHANGE MADE.
```

CI PS560240032U
April 1990

```
(*      YY/MM/DD  CCYY  I. M. THEPROGRAMMER      *)
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE      *)
(*      NARRATION ON THE NEXT LINE.      *)
(*      YY/MM/DD  CCXX  I. M. APERSON      *)
(*      DESCRIPTION OF FIRST CHANGE MADE.      *)
(*      -----*)
(**)
(* END %INCLUDE NEWCRB *)
```

```
(* %INCLUDE NEWEMM. *)
(**)
  PROCEDURE NEWEMM(VAR KEYE:ENTKEY;CONST FORM:ENTITIES;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A NEW NDS OBJECT.  FORM DETERMINES WHAT IS CREATED.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      ENTITY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      FORM      - THE FORM OF THE ENTITY TO CREATE.
(*
(*      OUTPUT
(*      KEYE      - THE POINTER TO THE CREATED ENTITY.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL:
(*      CHANGED: 12/10/84 J. JOHNSON - TO CALL 'MASNEW'.
(*-----*)
(**)
(* END %INCLUDE NEWEMM. *)
```

```

(*) %INCLUDE NEWIIM *)
(**)
  PROCEDURE NEWIIM(CONST ROOT:ENTKEY;VAR KEYE:ENTKEY;
    VAR ENTDEF:ENTBLOCK;VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*)      AUTHOR:  UNKNOWN          CADD      CREATED: YY/MM/DD CC  *)
(*)      VERSION: MAS VER 2              REVISED: 84/10/11 CC  *)
(*)
(*)  $FUNCTION:
(*)      CREATE A NEW ENTITY AND COPY THE APPLICATION ENTDATA INTO
(*)      IT.  CALLING PROCEDURE MUST CONNECT ENTITY TO PROPER POINT
(*)      IN NDS.
(*)
(*)  $ENVIRONMENT:
(*)      IBM PASCAL LANGUAGE
(*)      IBM 30XX, 43XX, DEC VAX 11/780
(*)
(*)  $DESCRIPTION OF ARGUMENTS:
(*)      NAME      I/O  DESCRIPTION
(*)      ROOT      I    THE NDS INTERNAL ROOT TO BE THE OWNER OF
(*)                  THE ENTITY.
(*)      ENTDEF    I    CONTAINS THE DATA TO BE COPIED INTO THE
(*)                  NEW ENTITY.
(*)      KEYE      O    THE KEY OF THE NEW ENTITY.
(*)      RR        O    ERROR CONDITION RETURN CODE.
(*)                  = 0  NORMAL RETURN CODE.
(*)
(*)  $COMMONS:
(*)
(*)  $PROCESSING DESCRIPTION:
(*)      ALLOCATES A NEW T_ENTITY AND CREATES EMPTY USER AND CNSTS
(*)      LISTS AND POINTS TO THEM. IT CREATES THE ADB.
(*)
(*)  $COMMENTS:
(*)
(*)  $CHANGE CONTROL:
(*)      04/26/85          E. D. SHREVE          W315
(*)                  TO INITIALIZE THE CRBEXIT AND MAPROB FIELDS
(*)
(*)      84/10/11  MAS VER 2  D. J. KERCHNER
(*)      UPDATED DOCUMENTATION.
(*)      84/10/04  MAS VER 2  E. D. SHREVE
(*)      CHANGED DECLARATION OF ENTDEF TO VAR.
(*)
(**)
(*) END %INCLUDE NEWIIM *)

```

```
(* %INCLUDE NEWLSM. *)
(**)
  PROCEDURE NEWLSM(CONST SIZE:LISTSIZE;VAR POSITION:LISTPSTN;
    VAR LISTREF:LISTPNTR; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      LISTREF IS INITIALIZED AND ALLOCATED ENOUGH SPACE TO HOLD
(*      SIZE ENTITIES.  IF ALREADY INITIALIZED, LISTREF IS DELETED
(*      PRIOR TO ALLOCATION OF SPACE.  IF SIZE IS ZERO, NO SPACE
(*      IS ALLOCATED.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      SIZE      - NUMBER OF ENTITIES TO BE ALLOCATED.
(*
(*      OUTPUT
(*      POSITION   - POSITION OF LIST.
(*      LISTREF   - POINTER TO A SYSTEM LIST WITH SIZE ENTITIES
(*                  ALLOCATED TO IT.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL
(*      CHANGED: 12/10/84  J. JOHNSON - TO CALL 'MASNEW'.
(*-----*)
(**)
(* END %INCLUDE NEWLSM. *)
```

```
(* %INCLUDE NEWNDM. *)
(**)
  PROCEDURE NEWNDM(VAR NDSREM:NDS;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A NEW EMPTY MODEL IN MEMORY.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      NETWORK PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      NONE      -
(*
(*      OUTPUT
(*      NDSREM    - CONNECTED TO THE NEW NDS.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      CHANGE CONTROL:
(*      EDS - MAS VERSION 2 - 9/17/84  REMOVE 'MARK' FUNCTION.
(*
(*-----*)
(**)
(* END %INCLUDE NEWNDM. *)
```

```
(* %INCLUDE NEWNM. *)
(**)
  PROCEDURE NEWNM(VAR KEYL:LISTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      $FUNCTION
(*      CREATE AN EMPTY APPLICATION LIST.
(*
(*      $DESCRIPTION OF ARGUMENTS
(*      NAME          I/O      DESCRIPTION
(*      ====          ==      =====
(*      KEYL          0        KEY OF THE CREATED APPL LIST
(*      RR            0        RETURN CODE
(*                               =0  GOOD RETURN
(*                               >0  CRITICAL ERROR
(*                               <0  WARNING
(*
(*      $COMMONS
(*      NDSGVM
(*      STACK_OF_LISTS      I    USED TO FIND LIST_OF_LISTS TO ADD
(*                               THE NEW LIST KSY.
(*
(*      $ENVIRONMENT:
(*      LANGUAGE:  IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*      $EXECUTION PROCEDURE:
(*      INTERNAL PROCEDURE OF THE MODEL ACCESS SOFTWARE
(*
(*      $PROCESSING DESCRIPTION:
(*      CREATES A NEW APPLICATION LIST ELEMENT AND ATTACHES IT
(*      TO THE LIST_OF_LISTS.  IT CREATES A NEW SYSTEM LIST THAT
(*      IS EMPTY AND ATTACHES IT TO THE APPLICATION LIST ELEMENT.
(*      THE FIELDS OF THE ELEMENTS ARE INITIALIZED.
(*
(*      $CHANGE CONTROL:
(*      REVISED: 04/23/85      E.D. SHREVE      W315
(*      CHANGED TO INITIALIZE THE NEW 'DELTFLG' FIELD.
(*
(*      ORIGINATED:  ORIGINAL NDS PACKAGE
(*-----*)
(**)
(*END %INCLUDE NEWNM. *)
```



```

(*) %INCLUDE NEWNMM. *)
(**)
  PROCEDURE NEWNMM(VAR KEYL:LISTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*) $FUNCTION
(*)   CREATE AN EMPTY APPLICATION LIST WITHOUT ADDING IT TO THE
(*)   LIST OF LISTS
(*)
(*) $DESCRIPTION OF ARGUMENTS
(*)   NAME          I/O      DESCRIPTION
(*)   =====
(*)   KEYL          0        KEY OF THE CREATED APPL LIST
(*)   RR            0        RETURN CODE
(*)                       =0   GOOD RETURN
(*)                       >0   CRITICAL ERROR
(*)                       <0   WARNING
(*)
(*) $COMMONS
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE:  IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE OF THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CREATES A NEW APPLICATION LIST ELEMENT AND ATTACHES IT
(*)   TO THE LIST_OF_LISTS.  IT CREATES A NEW SYSTEM LIST THAT
(*)   IS EMPTY AND ATTACHES IT TO THE APPLICATION LIST ELEMENT.
(*)   THE FIELDS OF THE ELEMENTS ARE INITIALIZED.
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 04/23/85      E.D. SHREVE      W315
(*)   CHANGED TO INITIALIZE THE NEW 'DELTF LG' FIELD.
(*)
(*)   ORIGINATED:  ORIGINAL NDS PACKAGE
(*)-----*)
(**)
(*END %INCLUDE NEWNMM. *)

```

```
(* %INCLUDE NEWNODE *)
(**)
  PROCEDURE NEWNODE(CONST NDSREM:NDS;VAR KEYE:ENTKEY;
    VAR ENTDEF:ENTBLOCK;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      AUTHOR:   UNKNOWN                      CADD      CREATED: YY/MM/DD CC
(*      VERSION: MAS VER 2                      REVISED: 84/10/11 CC
(*
(*      FUNCTION:
(*      CREATE A NEW ENTITY IN THE NDS AND COPY THE APPLICATION
(*      ENTDATA INTO IT.
(*
(*      ENVIRONMENT:
(*      IBM PASCAL LANGUAGE
(*      IBM 30XX, 43XX, DEC VAX 11/780
(*
(*      DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      NDSREM    I    THE NDS TO BE THE OWNER OF THE ENTITY.
(*      ENTDEF    I    CONTAINS THE DATA TO BE COPIED INTO THE
(*                      NEW ENTITY.
(*      KEYE      0    THE KEY OF THE NEW ENTITY.
(*      RR        0    ERROR CONDITION RETURN CODE.
(*                      = 0  NORMAL RETURN CODE.
(*
(*      COMMONS:
(*
(*      PROCESSING DESCRIPTION:
(*
(*      COMMENTS:
(*
(*      CHANGE CONTROL:
(*      84/10/11 MAS VER 2  D. J. KERCHNER
(*      UPDATED DOCUMENTATION.
(*      84/10/04 MAS VER 2  E. D. SHREVE
(*      CHANGED DECLARATION FOR ENTDEF TO VAR.
(*-----*)
(**)
(* END %INCLUDE NEWNODE *)
```

```
(* %INCLUDE NEWNSI. *)
(**)
  PROCEDURE NEWNSI(VAR ROOT:ENTKEY;VAR KEYE:ENTKEY;VAR RR:RET_REC);
    EXTERNAL;
(**)
(*-----*)
(*
(* FUNCTION
(*   CREATE AN EMPTY SCHEMA INSTANCE COLLECTOR ATTACHED TO THE
(*   SCHEMA ROOT.
(*
(* LANGUAGE
(*   PASCAL.
(*
(* PACKAGE
(*   SCHEMA PACKAGE.
(*
(* ARGUMENTS
(*   INPUT
(*     ROOT      - THE INTERNAL ROOT TO WHICH THE CREATED
(*                INSTANCE COLLECTOR WILL BE ATTACHED.
(*
(*   OUTPUT
(*     KEYE      - KEY OF THE CREATED INSTANCE COLLECTOR ENTITY.*
(*     RR        - THE FUNCTION RETURN RECORD.
(*
(* METHOD
(*   THIS PROGRAM IS CALLED FOR NO OTHER REASON THAN TO AVOID
(*   PASCAL TYPE CHECKING BY USING A DIFFERENT DEFINITION OF
(*   ENTBLOCK.
(*-----*)
(**)
(* END %INCLUDE NEWNSI. *)
```

```
(* %INCLUDE NEWNSR. *)
(**)
PROCEDURE NEWNSR(VAR ROOT:ENTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE A NEW NULL SCHEMA ROOT AND ATTACH IT TO THE NDS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      SCHEMA PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      ROOT      - THE INTERNAL ROOT TO WHICH THE CREATED
(*                  SCHEMA ROOT WILL BE ATTACHED.
(*
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*      METHOD
(*      THIS PROGRAM IS CALLED FOR NO OTHER REASON THAN TO AVOID
(*      PASCAL TYPE CHECKING BY USING A DIFFERENT DEFINITION OF
(*      ENTBLOCK.
(*-----*)
(**)
(* END %INCLUDE NEWNSR. *)
```

```
(* %INCLUDE NEWSADB *)
(**)
  PROCEDURE NEWSADB(CONST SIZE:ENTSIZE;VAR ENTBPNTR:ENTPNTR;
    VAR RR:RET_REC); EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC  *)
(*   VERSION:  MAS VER 1         REVISED: 12/10/84          *)
(*
(*   FUNCTION:
(*     ALLOCATE SPACE FOR DATA TO A SYSTEM UDB.
(*
(*   ENVIRONMENT:
(*     IBM PASCAL LANGUAGE
(*     IBM 30XX, 43XX, DEC VAX 11/780
(*
(*   DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     SIZE      I    SIZE OF ENTDATA TO BE COPIED.
(*     ENTBPNTR  0    POINTER TO ENTBLOCK CREATED.
(*     RR        0    ERROR CONDITION RETURN CODE.
(*                   = 0  NORMAL RETURN CODE.
(*
(*   COMMONS:
(*
(*   PROCESSING DESCRIPTION:
(*     NEWSADB USES THE PASCAL/VS COMPILER SUPPORT ROUTINE AMPXNEW.
(*
(*   COMMENTS:
(*
(*   CHANGE CONTROL:
(*     84/10/11  MAS VER 2  D. J. KERCHNER
(*     UPDATED DOCUMENTATION.
(*     84/12/10  MAS VER 2  J. JOHNSON
(*     TO CALL MASNEW.
(*-----*)
(**)
(* END %INCLUDE NEWSADB *)
```

```
(* %INCLUDE NEWSCHI. *)
(**)
  PROCEDURE NEWSCHI(CONST ROOT:ENTKEY;VAR KEYE:ENTKEY;
    VAR ENTDEF:ENTBLOCK;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE AN EMPTY SCHEMA INSTANCE COLLECTOR ENTITY ATTACHED
(*      TO THE SCHEMA ROOT.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      SCHEMA PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      ROOT      - THE INTERNAL ROOT TO WHICH THE CREATED
(*                  INSTANCE COLLECTOR WILL BE ATTACHED.
(*
(*      OUTPUT
(*      KEYE      - KEY OF THE INITIALIZED ENTITY.
(*      ENTDEF    - WORK AREA TO PASS TO NEWIIM.
(*      SCH_PTR   - POINTER TO THE CREATED INSTANCE COLLECTOR
(*                  ENTITY.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE NEWSCHI. *)
```

```
(* %INCLUDE NEWSCHR. *)
(**)
  PROCEDURE NEWSCHR(VAR ROOT:ENTKEY;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CREATE AN EMPTY ROOT COLLECTOR ENTITY ATTACHED TO THE NDS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      SCHEMA PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      ROOT      - THE INTERNAL ROOT TO WHICH THE CREATED
(*                  SCHEMA_ROOT WILL BE ATTACHED.
(*
(*      OUTPUT
(*      ENTDEF    - WORK AREA TO BE PASSED TO NEWIIM.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE NEWSCHR. *)
```

April 1990

```

(*) %INCLUDE NODECNM. *)
(**)
  PROCEDURE NODECNM(CONST KEYE:ENTKEY;VAR KEYLOUT:LISTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*)   CREATE A LIST WHICH CONTAINS A COPY OF THE ENTITY'S *)
(*)   CONSTITUENT LIST. *)
(*) *)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*)   NAME          I/O  DESCRIPTION *)
(*)   ====          ===  ===== *)
(*)   KEYE           I    KEY OF THE ENTITY. *)
(*)   KEYLOUT        0    LIST OF THE ENTITY'S CONSTITUENT ENTITIES*)
(*)   RR             0    THE FUNCTION RETURN RECORD. *)
(*)                   = 0  OK RETURN CODE *)
(*)                   = 1  YOU BLEW IT *)
(*)                   = 2  THE ROUTINE BLEW IT *)
(*)                   = ?  ERRORS FROM INTERNALLY CALLED *)
(*)                   FUNCTIONS *)
(*) *)
(*) $COMMONS: *)
(*) *)
(*) $ENVIRONMENT: *)
(*)   LANGUAGE: IBM PASCAL *)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*) *)
(*) $EXECUTION PROCEDURE: *)
(*)   MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*)   OR *)
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*) *)
(*) $PROCESSING DESCRIPTION: *)
(*) *)
(*) $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) *)
(*)   REVISED: 06/28/85 CCXX      B. A. ULMER      FRMI *)
(*)   CHANGE THE RETURN CODE FROM (END_OF_LIST TO NO_LIST_CREATED) *)
(*) *)

```



```
(* %INCLUDE NODECNN. *)
(**)
  PROCEDURE NODECNN(CONST KEYE:ENTKEY;VAR KEYLOUT:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*  $FUNCTION: *)
(*    CREATE A LIST WHICH CONTAINS A COPY OF THE ENTITY'S *)
(*    CONSTITUENT LIST WITHOUT ADDING AN ENTRY IN THE LIST OF LISTS*)
(*-----*)
(*  $DESCRIPTION OF ARGUMENTS: *)
(*    NAME          I/O  DESCRIPTION *)
(*    ====          ===  =====*)
(*    KEYE           I    KEY OF THE ENTITY. *)
(*    KEYLOUT        O    LIST OF THE ENTITY'S CONSTITUENT ENTITIES*)
(*    RR             O    THE FUNCTION RETURN RECORD. *)
(*                        = 0  OK RETURN CODE *)
(*                        = 1  YOU BLEW IT *)
(*                        = 2  THE ROUTINE BLEW IT *)
(*                        = ?  ERRORS FROM INTERNALLY CALLED *)
(*                        FUNCTIONS *)
(*-----*)
(*  $COMMONS: *)
(*-----*)
(*  $ENVIRONMENT: *)
(*    LANGUAGE: IBM PASCAL *)
(*    HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*-----*)
(*  $EXECUTION PROCEDURE: *)
(*    MODEL ACCESS SOFTWARE INTERFACE ROUTINE *)
(*    OR *)
(*    INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*-----*)
(*  $PROCESSING DESCRIPTION: *)
(*-----*)
(*  $COMMENTS: *)
(*-----*)
(*  $CHANGE CONTROL: *)
(*-----*)
(*    ORIGINATED: 03/07/87          K. M. ROSS          DBMA *)
(*-----*)
(*  DATA STRUCTURES/MAJOR VARIABLES: *)
(*-----*)
(*END-----*)
(**)
(* END %INCLUDE NODECNN. *)
```

```

(* INCLUDE NODEUHM. *)
(**)
  PROCEDURE NODEUHM(CONST FEYE:ENTFEY;VAR FEYLOUT:LISTFEY;
    VAR RR:RET REC);EXTERNAL;
(**)
(*
(*
(*
(* $FUNCTION:
(*   CREATE A LIST WHICH CONTAINS A COPY OF THE ENTITY'S
(*   USER LIST.
(*
(*
(* $DESCRIPTION OF ARGUMENTS:
(*
(*   NAME          I/O  DESCRIPTION
(*   ----          -
(*   FEYE          I    KEY OF THE ENTITY.
(*   FEYLOUT       O    LIST OF THE ENTITY'S USER ENTITIES
(*   RR            O    THE FUNCTION RETURN RECORD.
(*                      - 0  OF RETURN CODE
(*                      - 1  YOU BLEW IT
(*                      - 2  THE ROUTINE BLEW IT
(*                      - 3  ERROR FROM INTERNALLY CALLED
(*                      FUNCTIONS
(*
(*
(* $COMMON:
(*
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*
(* $EXECUTION PROCEDURE:
(*   MODEL ACCESS SOFTWARE INTERFACE ROUTINE
(*   OR
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(*
(* $PROCESSING DESCRIPTION:
(*
(*
(* $COMMENTS:
(*
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: 06/28/85 CCXX      W. A. ULMER      FRM1
(*   CHANGE THE RETURN CODE FROM (END OF LIST TO NO LIST CREATED)
(*
(*

```

```
(* %INCLUDE OCOUNT *)
(**)
PROCEDURE OCOUNT(VAR SIZE:INTEGER);EXTERNAL;
(**)
(*-----*)
(*
(* AUTHOR:  B. A. ULMER          FRMI    CREATED: 86/03/13  CC??*)
(* VERSION: XXXX                REVISED: YY/MM/DD  CC  *)
(*
(* FUNCTION:
(*   COUNT THE NUMBER OF TIMES THE OVERFLOW BUFFER HAS BEEN USED *)
(*
(* ENVIRONMENT:
(*   IBM PASCAL LANGUAGE
(*   IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(* EXECUTION PROCEDURE:
(*   HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(* DESCRIPTION OF ARGUMENTS:
(*   NAME      I/O  DESCRIPTION
(*   SIZE      I/O  SIZE TO BE STORED IN THE REQUESTED SIZE ARRAY
(*                   IN THE MSTATUS COMMON
(*
(* COMMONS:
(*   MSTATUS
(*
(* PROCESSING DESCRIPTION:
(*   DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*   FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(* COMMENTS:
(*   TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND
(*   THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
(* CHANGE CONTROL:
(*   YY/MM/DD  CCZZ  I. M. THECHANGER
(*   DESCRIPTION OF LATEST CHANGE MADE.
(*   YY/MM/DD  CCYY  I. M. THEPROGRAMMER
(*   DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE
(*   NARRATION ON THE NEXT LINE.
(*   YY/MM/DD  CCXX  I. M. APERSON
(*   DESCRIPTION OF FIRST CHANGE MADE.
(*-----*)
(**)
(* END %INCLUDE OCOUNT *)
```

```

(*) %INCLUDE ORDRLST. *)
(**)
  PROCEDURE ORDRLST(VAR IN_LIST:LISTPNTR; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   GIVEN AN APPLICATION LIST OF ENTITIES REORDER THEN SO THAT
(*)   THEY ARE IN USER TO CONSTITUENT ORDER
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   IN_LIST        I    SYSTEM LIST THAT IS TO BE REORDERED
(*)   RC              0    EXTERNAL RETURN CODE
(*)                      = 0  OK
(*)                      > 0  CRITICAL ERROR
(*)                      < 0  WARNING
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CREATE A COPY OF IN_LIST IN SRT_LST.
(*)   REPEAT FOR EACH ENTITY OF SRT_LST:
(*)     GET ALL USERS OF I-TH ENTITY OF SRT_LST.
(*)     IF A USER OF SRT_LST(I) APPEARS AT SRT_LST(J) AND I<J
(*)     THEN
(*)       SWAP SRT_LST(I) AND SRT_LST(J).
(*)     ELSE
(*)       GET NEXT SRT_LST(I)
(*)   UNTIL END OF LIST IN SRT_LST.
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)

```

```
(* %INCLUDE ORDRLSTI. *)
(**)
PROCEDURE ORDRLSTI(VAR IN_LIST:LISTPNTR; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(*   GIVEN AN APPLICATION LIST OF ENTITIES REORDER THEN SO THAT
(*   THEY ARE IN INCLUSIVE USER TO CONSTITUEN ORDER
(*
(* $DESCRIPTION OF ARGUMENTS:
(*   NAME          I/O  DESCRIPTION
(*   ====          ===  =====
(*   IN_LIST       I    SYSTEM LIST THAT IS TO BE REORDERED
(*   RC            O    EXTERNAL RETURN CODE
(*                       = 0  OK
(*                       > 0  CRITICAL ERROR
(*                       < 0  WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*   LANGUAGE: IBM PASCAL
(*   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(*
(*   REPEAT FOR EACH ENTITY OF SRT_LST:
(*
(*     CREATE THE LIST OF INCLUSIVE USERS
(*
(*     ADD INCLUSIVE USERS NOT ALREADY PROCESSED TO THE OUTPUT
(*     LIST
(*
(*     WITH THE ORDERED INCLUSIVE USER LIST REMOVE MEMBERS OF THE
(*     INPUT LIST AND ADD TO THE OUTPUT LIST
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*
(*   REVISED: MM/DD/YY          I. M. CHANGER          FRMI
(*)
```

```
(*      COMMENTS AS NECESSARY                                *)
(**                                           *)
(*      ORIGINATED: 10/14/86          K. M. ROSS          DBMA      *)
(**                                           *)
(*      CHANGED: K. M. ROSS          10/24/86            *)
(*      REASON : APPLICATIONS ABENDING SOC4                *)
(*      CHANGE : INITIALIZE POINTERS                        *)
(**                                           *)
(*      CHANGED: K. M. ROSS          10/24/86            *)
(*      REASON : SLOW RESPONCE WITH LARGE LISTS            *)
(*      CHANGE : OPTIMIZE ALGORITHM,CREATE USERS LISTS FOR EACH ELEMENT *)
(**                                           *)
(*-----*)
(**)
(* END %INCLUDE ORDLSTI. *)
(**)
```

```
(* %INCLUDE OSTART *)
(**)
  PROCEDURE OSTART; EXTERNAL;
(**)
(*-----*)
(*
(*      AUTHOR:  B. A. ULMER          FRMI    CREATED: 86/03/13  CC??*)
(*      VERSION: XXXX                REVISED: YY/MM/DD  CC  *)
(*
(*      FUNCTION:
(*      INITIALIZE THE INFORMATION DEALING WITH THE OVERFLOW BUFFER
(*      IN THE MSTATUS COMMON
(*
(*      ENVIRONMENT:
(*      IBM PASCAL LANGUAGE
(*      IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(*      EXECUTION PROCEDURE:
(*      HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(*      DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*
(*      COMMONS:
(*      MSTATUS
(*
(*      PROCESSING DESCRIPTION:
(*      DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(*      FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(*      COMMENTS:
(*      TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND
(*      THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*
(*      CHANGE CONTROL:
(*      YY/MM/DD  CCZZ  I. M. THECHANGER
(*      DESCRIPTION OF LATEST CHANGE MADE.
(*      YY/MM/DD  CCYY  I. M. THEPROGRAMMER
(*      DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE
(*      NARRATION ON THE NEXT LINE.
(*      YY/MM/DD  CCXX  I. M. APERSON
(*      DESCRIPTION OF FIRST CHANGE MADE.
(*-----*)
(**)
(* END %INCLUDE OSTART *)
```

```
(* %INCLUDE PASASM. *)
(**)
PROCEDURE PASASM(CONST KEYP:ENTKEY; VAR BLOCK:ENTBLOCK; VAR
DATAREC:BLKDATA;
  VAR RC:EXT_RET_CODE; CONST NAME:ROUTINE);FORTRAN;
(**)
(*-----*)
(*
(*      AUTHOR:  D. KERCHNER      PDDI      CREATED:  84/09/11      *)
(*      VERSION: MAS2              REVISED: YY/MM/DD      *)
(*
(*      FUNCTION:
(*          THIS ROUTINE SERVES AS A LINK ROUTINE BETWEEN THE MAS
(*          INTERFACE PACKAGE AND THE USER'S APPLICATION DEFINED
(*          PROCEDURE MAKING IT FORTRAN CALLABLE
(*
(*      ENVIRONMENT:
(*          IBM ASSEMBLER LANGUAGE
(*          IBM 4341/3083 VAX 11/780 SYSTEMS
(*
(*      EXECUTION PROCEDURE:
(*          THIS ROUTINE IS INVOKED BY A CALL FROM A MAS INTERFACE
(*          ROUTINE SUCH AS MALXEQ OR MAEXEQ, IN ORDER TO INVOKE A
(*          USER DEFINED PROCEDURE WHICH IS IN THE USER'S MODULE
(*
(*      DESCRIPTION OF ARGUMENTS:
(*          NAME      TYPE      I/O  DESCRIPTION
(*          KEYP              I      ENTITY KEY
(*          ENTBLOCK          I      APPLICATION DEFINED BLOCK
(*          DATAREC           I/O    USER PASSED DATA (I/O)
(*          RC                0      ERROR CONDITION RETURN CODE (PASSED
(*                               ONLY, NOT PASSED
(*          ROUTINE          I      NAME OF THE USER DEFINED PROCEDURE
(*
(*      COMMONS:
(*
(*      PROCESSING DESCRIPTION:
(*          PASAM RECEIVES CALL FROM MAS PASCAL ROUTINE.  THIS
(*          ASSEMBLER CSECT THEN BRANCHES TO THE USER DEFINED ROUTINE
(*          BY PASSING THE ADDRESS OF THAT ROUTINE IN A BRANCH REGISTER)
(*          INSTRUCTION.  WHEN THE USER ROUTINE COMPLETES PROCESSING,
(*          CONTROL IS RETURNED TO THE MAS PASCAL ROUTINE VIA THE
(*          USER DEFINED ROUTINE.
(*
(*      COMMENTS:
(*          VARIABLES ARE NOT ACCESSED, BUT ARE PASSED THROUGH TO THE
```



```
(*          USER DEFINED ROUTINE.          *)
(*)                                          *)
(*)  DATA STRUCTURES/MAJOR VARIABLES:      *)
(*)                                          *)
(*)  REGISTER USAGE:                        *)
(*)      R1  - PARAMETER LIST               *)
(*)      R15 - BRANCHING REGISTER           *)
(*)                                          *)
(*)  CHANGE CONTROL:                        *)
(*)                                          *)
(*END %INCLUDE PASASM                      *)
```

```
(* %INCLUDE RDLSM. *)
(**)
  PROCEDURE RDLSM(VAR POSITION:LISTPSTN; CONST LISTREF:LISTPNTR;
    VAR KEYE:ENTKEY; VAR EOL:BOOLEAN; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* FUNCTION
(*   READ A SYSTEM LIST AS A FIRST IN FIRST OUT ORDER.
(*
(* LANGUAGE
(*   PASCAL.
(*
(* PACKAGE
(*   LIST PACKAGE.
(*
(* ARGUMENTS
(*   INPUT
(*     POSITION - INDICATING NEXT ENTITY IN LISTREF TO BE READ.
(*     LISTREF - LIST TO BE READ.
(*
(*   OUTPUT
(*     POSITION - UPDATED TO NEXT ENTITY.
(*     KEYE    - ENTITY READ FROM LIST.
(*     EOL     - TRUE IF ENTITY WAS READ ELSE FALSE.
(*     RR      - THE FUNCTION RETURN RECORD.
(*
(* METHOD
(*   IF THERE IS AN ENTITY AT INDICATED POSITION THEN PLACE
(*     NEXT ENTITY INDICATED BY POSITION IN KEYE, ADJUST
(*     POSITION TO INDICATE NEXT ENTITY, RETURN EOL SET TO
(*     FALSE,
(*   ELSE
(*     RETURN EOL SET TO TRUE.
(*-----*)
(**)
(* END %INCLUDE RDLSM. *)
```

```
(* %INCLUDE RDRLSM. *)
(**)
  PROCEDURE RDRLSM(CONST POSITION:LISTPSTN;CONST LISTREF:LISTPNTR;
    VAR KEYE:ENTKEY;VAR EOL:BOOLEAN;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      READ THE LAST ENTITY KEY FROM LISTREF.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      POSITION  - RELATIVE POSITION IN LISTREF OF ENTITY
(*                TO BE READ.
(*      LISTREF  - LIST WHOSE POSITION-TH ENTITY IS TO BE READ.
(*
(*      OUTPUT
(*      KEYE     - KEY OF POSITION-TH ENTITY IN LISTREF.
(*      EOL      - TRUE IF NO POSITION-TH ENTITY IN LISTREF.
(*      RR       - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE RDRLSM. *)
```

```
(* %INCLUDE RDTLSM. *)
(**)
  PROCEDURE RDTLSM(CONST LISTREF:LISTPNTR;VAR KEYE:ENTKEY;
    VAR EMPTY:BOOLEAN;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      READ THE LAST ENTITY KEY FROM LISTREF.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LISTREF    - LIST WHOSE LAST ENTITY IS TO BE READ.
(*
(*      OUTPUT
(*      KEYE       - RETURNS LAST ENTITY IN LISTREF.
(*      EMPTY     - TRUE IF NO ENTITIES IN LIST, ELSE FALSE.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE RDTLSM. *)
```

```
(* %INCLUDE REVAADB. *)
(**)
  PROCEDURE REVAADB(CONST ENTBPNTR:ENTPNTR;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*      *)
(*      FUNCTION      *)
(*      ASSIGN THE VALUE OF A SYSTEM UDB TO AN APPLICATION ENTBLOCK.*)
(*      *)
(*      LANGUAGE      *)
(*      PASCAL.      *)
(*      *)
(*      PACKAGE      *)
(*      UDB PACKAGE.  *)
(*      *)
(*      ARGUMENTS      *)
(*      INPUT      *)
(*      ENTBPNTR  -  POINTER TO ENTBLOCK CREATED.  *)
(*      *)
(*      OUTPUT      *)
(*      ENTDEF    -  THE ENTBLOCK WITH THE VALUE OF SYSUDB *)
(*      ASSIGNED TO IT.  *)
(*      RR        -  THE FUNCTION RETURN RECORD.  *)
(*      *)
(*      METHOD      *)
(*      REVAADB USES SYSTEM ROUTINE AMPXMOVE TO MOVE DATA IN *)
(*      MEMORY.  THE NUMBER OF BYTES TO MOVE MUST BE SPECIFIED. *)
(*      *)
(*-----*)
(**)
(* END %INCLUDE REVAADB. *)
```

```
(* %INCLUDE REVNODM *)
(**)
  PROCEDURE REVNODM(VAR KEYE:ENTKEY;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC  *)
(*   VERSION: MAS VER 2           REVISED: 84/10/11 CC  *)
(*
(*   FUNCTION:
(*     REVISE AN ENTITY'S USER DATA BLOCK.
(*
(*   ENVIRONMENT:
(*     IBM PASCAL LANGUAGE
(*     IBM 30XX, 43XX, DEC VAX 11/780
(*
(*   DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     KEYE      I    KEY OF ENTITY TO BE REVISED.
(*     ENTDEF    I    NEW DATA FOR ENTITY TO BE REVISED.
(*     RR        0    ERROR CONDITION RETURN CODE.
(*                   = 0  NORMAL RETURN CODE.
(*
(*   COMMONS:
(*
(*   PROCESSING DESCRIPTION:
(*
(*   COMMENTS:
(*
(*   CHANGE CONTROL:
(*     84/10/11 MAS VER 2  D. J. KERCHNER
(*     UPDATED DOCUMENTATION.
(*     84/10/04 MAS VER 2  E. D. SHREVE
(*     CHANGED DECLARATION ON KEYE AND ENTDEF TO VAR.
(*-----*)
(**)
(* END %INCLUDE REVNODM *)
```

```
(* %INCLUDE REVRLSM. *)
(**)
  PROCEDURE REVRLSM(CONST POSITION:LISTPSTN;CONST KEYE:ENTKEY;
    CONST LISTREF:LISTPNTR;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      CHANGE AN ENTITY IN A SYSTEM LIST.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      POSITION  - THE RELATIVE POSITION OF THE ENTITY IN
(*                THE LIST.
(*      KEYE     - THE NEW ENTITY KEY.
(*      LISTREF  - A POINTER TO A SYSTEM LIST.
(*
(*      OUTPUT
(*      RR       - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE REVRLSM. *)
```

```
(* %INCLUDE REVSADB *)
(**)
  PROCEDURE REVSADB(VAR ENTDEF:ENTBLOCK;VAR ENTBPNTNTR:ENTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      AUTHOR:   UNKNOWN          CADD      CREATED: YY/MM/DD CC  *)
(*      VERSION:  MAS VER 2              REVISED: 84/10/11 CC  *)
(*                                          REVISED: 84/12/10  *)
(*
(*      FUNCTION:
(*      REPLACE THE VALUE OF A SYSTEM ENTBLOCK WITH THE VALUE OF
(*      ENTDEF.
(*
(*      ENVIRONMENT:
(*      IBM PASCAL LANGUAGE
(*      IBM 30XX, 43XX, DEC VAX 11/780
(*
(*      DESCRIPTION OF ARGUMENTS:
(*      NAME      I/O  DESCRIPTION
(*      ENTDEF     I   THE APPLICATION ENTBLOCK VALUE TO ASSIGN
(*                   TO A SYSTEM ENTBLOCK.
(*      ENTBPNTNTR O   POINTER TO THE SYSTEM ENTBLOCK TO BE REVISED.
(*      RR         0   ERROR CONDITION RETURN CODE.
(*                   = 0  NORMAL RETURN CODE.
(*
(*      COMMONS:
(*
(*      PROCESSING DESCRIPTION:
(*      REVSADB USES SYSTEM ROUTINE AMPXMOVE TO MOVE DATA IN
(*      MEMORY.  THE NUMBER OF BYTES TO MOVE MUST BE SPECIFIED.
(*
(*      COMMENTS:
(*
(*      CHANGE CONTROL:
(*      84/10/11  MAS VER 2  D. J. KERCHNER
(*                   UPDATED DOCUMENTATION.
(*      84/10/04  MAS VER 2  E. D. SHREVE
(*                   CHANGED ENTDEF FROM CONST TO VAR.
(*      84/12/10  MAS VER 2  J. JOHNSON
(*                   TO CALL MASDSP.
(*-----*)
(**)
(* END %INCLUDE REVSADB *)
```



```
(* %INCLUDE RLSNM. *)
(**)
  PROCEDURE RLSNM(VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*                                           *)
(*  FUNCTION                                           *)
(*    RELEASE ALL THE LISTS ON THE CURRENT LIST OF LISTS. *)
(*                                           *)
(*  LANGUAGE                                           *)
(*    PASCAL. *)
(*                                           *)
(*  PACKAGE                                           *)
(*    LIST PACKAGE. *)
(*                                           *)
(*  ARGUMENTS                                           *)
(*    INPUT                                           *)
(*      NONE - *)
(*                                           *)
(*    OUTPUT                                           *)
(*      RR - THE FUNCTION RETURN RECORD. *)
(*-----*)
(**)
(* END %INCLUDE RLSNM. *)
```

```
(* %INCLUDE RSTLSM. *)
(**)
  PROCEDURE RSTLSM(VAR POSITION:LISTPSTN;CONST LISTREF:LISTPNTR;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      RESETS POSITION TO INDICATE THE BEGINNING OF A LIST.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LISTREF    - POINTER TO A LIST.
(*
(*      OUTPUT
(*      POSITION    - RESET TO INDICATE BEGINNING OF LIST.
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE RSTLSM. *)
```

```
(* %INCLUDE RSTSFLG *)
(**)
  PROCEDURE RSTSFLG(CONST LISTP:LISTPNTR;
    CONST SETTING:BOOLEAN;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*    RESET THE REQUESTED POSITION IN THE INTERNAL MAS PROCESS
(*    FLAG (MAPROB) IN THE IIT TO THE REQUESTED BOOLEAN VALUE.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*    NAME      I/O      DESCRIPTION
(*    ====      ===      =====
(*    LISTP      I      THE LIST OF ENTITIES THAT ARE TO HAVE A BYTE
(*                      IN THE SYSUSE FLAG SET.
(*    SETTING     I      BOOLEAN VALUE THE SYSUSE(FLG_POS) BYTE IS TO
(*                      BE SET TO. (IE: TRUE OR FALSE)
(*    RR          0      FUNCTION RETURN CODE
(*                      = 0  GOOD RETURN
(*                      > 0  CRITICAL ERROR
(*                      < 0  WARNING
(*
(*  $COMMONS
(*    NONE
(*
(*  $ENVIRONMENT:
(*    LANGUAGE: IBM PASCAL
(*    HARDWARE SYSTEM: IBM 360,370,43XX
(*
(*  $EXECUTION PROCEDURE:
(*    INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(*  $PROCESSING DESCRIPTION:
(*    FOR EACH ENTITY ON THE LIST OF ENTITIES, THE MAPROB
(*    BYTE IS SET TO THE INPUT SETTING.
(*
(*  $COMMENTS
(*    USES THE MAPROB FLAG IN THE T_ENTITY.
(*
(*  $CHANGE CONTROL:
(*    REVISED:  04/26/85      E.D. SHREVE      W315
(*              TO SET THE MAPROB BYTE IN THE T_ENTITY INSTEAD
(*              OF THE SYSUSE OF THE ADB. FOR INTERNAL MAS.
(*
(*    ORIGINATED: 07/10/84      C. J. SAMPLE      W315
(*-----*)
(**)
(* END %INCLUDE RSTSFLG *)
```

```
(* %INCLUDE RVRLSM.*)
(**)
  PROCEDURE PVRLSM(VAR KEYIN:LISTPNTR; VAR KEYOUT:LISTPNTR;
                  VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*  $FUNCTION:
(*      CREATE AN OUTPUT LIST THAT CONTAINS THE ENTITIES ON THE
(*      INPUT LIST IN REVERSE ORDER.
(*
(*  $DESCRIPTION OF ARGUMENTS:
(*      NAME          I/O  DESCRIPTION
(*      ===          ==  =====
(*      KEYIN         I    LIST TO COPY FROM
(*      KEYOUT        O    NEW LSIT WITH ENTITY'S REVERSED
(*      RR            O    EXTERNAL RETURN CODE
(*                      = 0  OK RETURN CODE
(*
(*  $COMMONS:
(*      NONE
(*
(*  $ENVIRONMENT:
(*      LANGUAGE: IBM PASCAL
(*      HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(*  $EXECUTION PROCEDURE:
(*      INTERNAL MODEL ACCESS SOFTWARE ROUTINE
(*
(*  $PROCESSING DESCRIPTION:
(*      IF THE INPUT LIST IS NOT EMPTY, A NEW OUTPUT LIST IS
(*      CREATED. THEN THE ENTITIES ARE MOVED INTO THE NEW LIST
(*      IN REVERSE ORDER.
(*
(*  $COMMENTS:
(*      NONE
(*
(*  $CHANGE CONTROL:
(*
(*      ORIGINATED: 04/11/86 MAS2  E. D. SHREVE          W315
(*
(*-----*)
(*END-----*)
(* END %INCLUDE RVRLSM. *)
```

```
(* %INCLUDE SETRULS. *)
(**)
PROCEDURE SETRULS(CONST USER:ENTKEY; CONST CNST:ENTKEY; CONST
  DEL_LIST:LISTPNTR; VAR RULE:T_RULE; VAR MIN_CNST:LISTPSTN;
  VAR RR:RET_REC);EXTERNAL;
%PAGE
(**)
(*-----*)
(* *)
(* $FUNCTION: *)
(* SET DELETE FLAGS ACCORDING TO USER'S DEPENDENCE & STRENGTH *)
(* RULES. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME I/O DESCRIPTION *)
(* === === *)
(* USER I USER WHOSE RULES ARE TO BE FOUND BASED ON*)
(* THE RELATIONSHIP WITH CNST *)
(* CNST I CNST WHOSE RULES ARE TO BE FOUND BASED ON*)
(* THE RELATIONSHIP WITH USER *)
(* DEL_LST I LIST OF KEYS THAT ARE ELIGIBLE FOR *)
(* DELETION *)
(* RULE O INDICATES WHICH DELETE AND COMPRESS RULES*)
(* ARE VALID FOR THIS RELATIONSHIP *)
(* MIN_CNST O MINIMUM NUMBER OF CONSTITUENTS FOR USER *)
(* RR O THE FUNCTION RETURN CODE. *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* IF THE USER IS IN THE DEL_LST THEN EXIT *)
(* ELSE *)
(* THE RULES OF THE CONNECTION ARE FOUND AND THE RULE SET IS *)
(* FILLED APPROPRIATELY *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* *)
```

(* REVISED: 06/17/86 B. A. ULMER FRMI *)
(* ADD NEW PARAMETERS TO SETRULS AND CHANGE PROCESSING TO HANDLE *)
(* THE NEW DELETE RULES - MAJOR REWRITE *)
(* *)
(* REVISED: 09/ /85 B. A. ULMER FRMI *)
(* ADD NEW PARAMETERS TO FNDURUL TO HANDLE TWO NEW DELETE RULES *)
(* *)

```

(*) %INCLUDE SORTDLST. *)
(**)
PROCEDURE SORTDLST(CONST DEL_LST:LISTKEY;VAR SRT_LST:LISTPNTR;
VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) $FUNCTION: *)
(*) GIVEN AN APPLICATION LIST OF ENTITIES TO BE DELETED, *)
(*) DEL_LST RETURNS A SYSTEM LIST SORTED IN USER-CONSTITUENT *)
(*) ORDER IN SRT_LST. *)
(*)
(*) $DESCRIPTION OF ARGUMENTS: *)
(*) NAME I/O DESCRIPTION *)
(*) === === *)
(*) DEL_LST I APPLICATION LIST CONTAINING THE LISTKEY *)
(*) OF TH E ENTITIES TO BE DELETED *)
(*) SRT_LST O POINTER TO A SYSTEM LIST CONTAINING THE *)
(*) ENTITIES OF THE DEL_LST SORTED IN USER- *)
(*) CONSTITUENT ORDER *)
(*) RC O EXTERNAL RETURN CODE *)
(*) = 0 OK *)
(*) > 0 CRITICAL ERROR *)
(*) < 0 WARNING *)
(*)
(*) $COMMONS: *)
(*)
(*) $ENVIRONMENT: *)
(*) LANGUAGE: IBM PASCAL *)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(*)
(*) $EXECUTION PROCEDURE: *)
(*) INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE *)
(*)
(*) $PROCESSING DESCRIPTION: *)
(*) SET MAPROB FLAG ON FOR ALL ENTITIES IN THE DEL-LST *)
(*) THAT ARE NOT 'MARKED FOR DELETE' *)
(*) REPEAT FOR EACH ENTITY IN DEL_LST *)
(*) IF NOT PROCESSED (MAPROB = FALSE) *)
(*) CALL SRTBYUSR TO PUT ALL USER ENTITIES ON THE *)
(*) SRT_LST BEFORE ADDING THE ENTITY. *)
(*) RESET MAPROB AND MAPROB2 *)
(*)
(*) $COMMENTS: *)
(*) USES NDS PROCEDURES RSTLSM, RDLISM, NEWLSM AND SRTBYUSR *)
(*)

```

```
(* $CHANGE CONTROL: *)
(*)
(*) REVISD: 12/17/85 B. A. ULMER FRMI *)
(*) CHANGE SRTBYUSR TO SRTBYCNT - SORT LIST NOW IN CONSTITUENT TO *)
(*) USER ORDER *)
(*)
(*) REVISD: 12/03/85 E. D. SHREVE FRMI *)
(*) REWRITTEN TO REPLACE THE COMPARE SORT WITH A SUBROUTINE THAT *)
(*) USES THE SYSTEM FLAGS (MAPROB AND MAPROB2) FOR SORTING *)
(*)
(*) REVISD: 07/01/85 B. A. ULMER FRMI *)
(*) ELIMINATE THE LEAVE FUNCTION TO IMPROVE COMPATABILITY WITH VAX *)
(*)
(*) REVISD: 04/10/85 B. A. ULMER FRMI *)
(*) DO NOT PROCESS THE ALREADY "MARKED FOR DELETE" ENTITIES *)
(*)
(*) ORIGINATED: 06/19/84 R. A. MCCLUSKEY FRMI *)
(*)
(*)-----*)
%PAGE *)
(*)-----*)
(*) DATA STRUCTURES/MAJOR VARIABLES: *)
(*)-----*)
(*)
(**)
(* END %INCLUDE SORTDLST. *)
(**)
```



```
(* %INCLUDE SORTLSM. *)
(**)
  PROCEDURE SORTLSM(VAR LISTREF:LISTPNTR; CONST PROCNAME:ROUTINE;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      SORT A SYSTEM LIST.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      LIST PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      LISTREF    - LIST TO BE SORTED.
(*
(*      OUTPUT
(*      RR          - THE FUNCTION RETURN RECORD.
(*
(*      METHOD
(*      THE SYSTEM LIST LISTREF IS SORTED IN APPLICATION DEFINED
(*      ORDER. THE ORDER IS DETERMINED BY A USER DEFINED FUNCTION;
(*      ORDER. ORDER RETURNS FALSE IF TWO ENTITIES ARE IN ORDER ELSE*
(*      IT RETURNS TRUE. IF LISTREF HAS LESS THAN TWELVE ENTITIES,
(*      THE BUBBLE SORT ALGORITHM IS USED. IF LISTREF CONTAINS MORE
(*      THAN ELEVEN ENTITIES A SLIGHT VARIATION OF QUICK SORT IS
(*      USED WHEN THE SUBLISTS CREATED BY STANDARD QUICK SORT
(*      CONTAIN LESS THAN TWELVE ENTITIES, SORTLSM REVERTS BACK TO
(*      BUBBLE SORT. IN GENERAL SORTLSM IS FASTER THAN EITHER
(*      BUBBLE SORT OR QUICK SORT.
(*-----*)
(**)
(* END %INCLUDE SORTLSM. *)
```

```
(* %INCLUDE SRTBYCNT. *)
(**)
PROCEDURE SRTBYCNT(VAR KEY1:ENTKEY;VAR SRT_LST:LISTPNTR;
VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* $FUNCTION:
(* THIS IS A RECURSIVE ROUTINE THAT PLACES THE CNST ENTITIES
(* OF KEY1, THAT ARE ON THE DELETE LIST, INTO THE SRT_LST
(* BEFORE KEY1 IS ADDED.
(*
(* $DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* ===
(* KEY1 I THE ENTITY THAT WILL BE PLACED ON THE
(* OUTPUT LIST ALONG WITH ITS CONSTITUENTS
(* SRT_LST O POINTER TO A SYSTEM LIST CONTAINING THE
(* ENTITIES OF THE DEL_LST SORTED IN
(* CONSTITUENT-USER ORDER
(* RC O EXTERNAL RETURN CODE
(* = 0 OK
(* > 0 CRITICAL ERROR
(* < 0 WARNING
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(* LANGUAGE: IBM PASCAL
(* HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(* INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*
(* $PROCESSING DESCRIPTION:
(* (ALL ENTITIES THAT WERE ON THE ORIGINAL DELETE LIST HAVE BEEN
(* FLAGGED IN THE ENTITY BLOCK (MAPROB = TRUE))
(* EACH CNST OF KEY1 IS PROCESSED:
(* IF NOT PROCESSED (MAPROB2 = FALSE) AND IN THE DELETE
(* LIST (MAPROB = TRUE) THEN
(* CALL SRTBYCNT TO PUT THE CNST ENTITY ON SRT_LST
(* ADD KEY1 TO THE SRT_LST.
(* SET KEY1 (MAPROB2 = TRUE)
(*
(* $COMMENTS:
```

```
(*      USES NDS PROCEDURES RSTLSM. RDLSTM,  AND SRTBYCNT      *)
(*)                                                            *)
(*)  $CHANGE CONTROL:                                          *)
(*)                                                            *)
(*)  CREATED: 12/17/85      B. A. ULMER      FRMI      *)
(*)    THIS ROUTINE IS USED BY SORTDLST FOR SORTING.  IT REPLACES *)
(*)    THE COMPARE SORT IN THE OLD SORTDLST WHICH WAS INEFFICIENT. *)
(*)                                                            *)
(*)-----*)
%PAGE                                                            *)
(*)-----*)
(*)  DATA STRUCTURES/MAJOR VARIABLES:                        *)
(*)    THESE ARE DESCRIBED IN THE NDSACL INCLUDE MEMBER.      *)
(*)-----*)
(*)                                                            *)
(**)
(*) END %INCLUDE SRTBYCNT. *)
(**)
```

```
(* %INCLUDE UPDCRBE *)
(**)
PROCEDURE UPDCRBE(CONST CRB:CRBPNTN; CONST EKEY:ENTKEY;
VAR POS:LISTPSTN; VAR DIR:LISTDIR; VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(* AUTHOR: B. A. ULMER FRMI CREATED: 85/02/08 CC??*)
(* VERSION: XXXX REVISED: YY/MM/DD CC *)
(*
(* FUNCTION:
(* UPDATE AN ENTRY IN THE CRB
(*
(* ENVIRONMENT:
(* IBM PASCAL LANGUAGE
(* IBM 30XX, 43XX DEPENDENT CODE, OR OTHER APPROPRIATE H/W.
(*
(* EXECUTION PROCEDURE:
(* HOW IS THIS ROUTINE/MODULE TO BE EXECUTED.
(*
(* DESCRIPTION OF ARGUMENTS:
(* NAME I/O DESCRIPTION
(* CRB I/O CONSTITUENT READ BLOCK ADDRESS
(* EKEY I ENTITY KEY OF ENTRY TO UPDATE
(* POS I NEW LIST POSITION SETTING
(* DIR I NEW DIRECTION OF LIST (FORWARD OR REVERSE)
(* RR 0 ERROR CONDITION RETURN CODE
(* = 0 OK RETURN CODE
(* = 1 YOU BLEW IT
(* = 2 THE ROUTINE BLEW IT
(*
(* COMMONS:
(* COM1
(* VAR1 I VAR1 NAME MUST BE FILLED, CHARACTER DATA
(* MUST BE PROVIDED
(* VAR2 I VAR2 MUST BE SPECIFIED
(* COM2
(* VAR3 I CHARACTER DATA MUST BE SPECIFIED
(*
(* PROCESSING DESCRIPTION:
(* DETAILED DESCRIPTION OF HOW THIS ROUTINE WORKS, WHICH
(* FILES NEED TO BE OPENED/CLOSED, FILES USED, ETC.
(*
(* COMMENTS:
(* TEXT OF ANY FURTHER COMMENTS WHICH MIGHT HELP TO UNDERSTAND*)
(* THE FUNCTION/EXECUTION OF THIS ROUTINE.
(*)
```

```
(*  CHANGE CONTROL: *)
(*  YY/MM/DD  CCZZ  I. M. THECHANGER *)
(*  DESCRIPTION OF LATEST CHANGE MADE. *)
(*  YY/MM/DD  CCYY  I. M. THEPROGRAMMER *)
(*  DESCRIPTION OF CHANGE MADE.  IF LENGTHY, CONTINUE THE *)
(*  NARRATION ON THE NEXT LINE. *)
(*  YY/MM/DD  CCXX  I. M. APERSON *)
(*  DESCRIPTION OF FIRST CHANGE MADE. *)
(*  ----- *)
(**)
(* END %INCLUDE UPDCRBE *)
```

```

(*) %INCLUDE VERAPN. *)
(**)
  PROCEDURE VERAPN(CONST KEY1:ANYKEY;CONST KEY2:ANYKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*)-----*)
(*)
(*) FUNCTION *)
(*)   VERIFY LEGALITY OF APPENDING AN ENTITY OR LIST OF ENTITIES *)
(*)   (KEY2) TO AN ENTITY OR LIST OF ENTITIES (KEY1). *)
(*) *)
(*) LANGUAGE *)
(*)   PASCAL. *,
(*) *)
(*) PACKAGE *)
(*)   VERIFY PACKAGE. *)
(*) *)
(*) ARGUMENTS *)
(*)   INPUT *)
(*)     KEY1      - KEY OF APPLICATION LIST TO WHICH ENTITIES *)
(*)                ARE TO BE APPENDED. IF ENTITY KEY, THEN *)
(*)                ADD TO CONSTITUENT LIST. *)
(*)     KEY2      - KEY OF APPLICATION LIST OF ENTITIES TO *)
(*)                APPEND. IF ENTITY KEY, THEN ADD ENTITY *)
(*)                TO LIST. *)
(*) *)
(*)   OUTPUT *)
(*)     RR      - THE FUNCTION RETURN RECORD. *)
(*) *)
(*)-----*)
(**)
(*) END %INCLUDE VERAPN. *)

```

```
(* %INCLUDE VERCN. *)
(**)
  PROCEDURE VERCN(CONST KEYLU:LISTKEY;CONST KEYLC:LISTKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      VERIFY LEGALITY OF CONNECTING EACH ENTITY ON A LIST OF
(*      USERS TO EACH ENTITY ON A LIST OF CONSTITUENTS.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      VERIFY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYLU      - KEY OF LIST OF USERS.
(*      KEYLC      - KEY OF LIST OF CONSTITUENTS.
(*
(*      OUTPUT
(*      RR         - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE VERCN. *)
```

```
(* %INCLUDE VERCR *)
(**)
  PROCEDURE VERCR(VAR ENTDEF:ENTBLOCK;CONST KEYE:ANYKEY;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*   AUTHOR:   UNKNOWN           CADD   CREATED: YY/MM/DD CC
(*   VERSION:  MAS VER 2         REVISED: 84/10/11 CC
(*
(*   FUNCTION:
(*     VERIFY LEGALITY OF CREATING AN ENTITY WITH THE USER
(*     SUPPLIED ENTITY DATA BLOCK AND LIST OF CONSTITUENTS.
(*
(*   ENVIRONMENT:
(*     IBM PASCAL LANGUAGE
(*     IBM 30XX, 43XX, DEC VAX 11/780
(*
(*   DESCRIPTION OF ARGUMENTS:
(*     NAME      I/O  DESCRIPTION
(*     ENTDEF    I    USER SUPPLIED DATA FOR ENTITY BLOCK.
(*     KEYE      I    KEY OF ENTITY OR APPLICATIONS LIST OF
(*                     ENTITIES TO BE CONSTITUENTS OF THIS ENTITY.
(*     RR        0    ERROR CONDITION RETURN CODE.
(*                     = 0  NORMAL RETURN CODE.
(*
(*   COMMONS:
(*
(*   PROCESSING DESCRIPTION:
(*
(*   COMMENTS:
(*
(*   CHANGE CONTROL:
(*     84/10/11  MAS VER 2  D. J. KERCHNER
(*               UPDATED DOCUMENTATION.
(*     84/10/04  MAS VER 2  E. D. SHREVE
(*               CHANGED ENTDEF FROM CONST TO VAR.
(*-----*)
(**)
(* END %INCLUDE VERCR *)
```



```
(* %INCLUDE VERDEL. *)
(**)
  PROCEDURE VERDEL(CONST KEYE:ANYKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      VERIFY LEGALITY OF DELETING AN ENTITY.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      VERIFY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF ENTITY TO BE DELETED FROM NETWORK.
(*
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE VERDEL. *)
```

```
(* %INCLUDE VERGT. *)
(**)
  PROCEDURE VERGT(CONST KEYE:ENTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      FUNCTION
(*      VERIFY LEGALITY OF RETRIEVING AN ENTITY WITH THE USER
(*      SUPPLIED ENTITY KEY.
(*
(*      LANGUAGE
(*      PASCAL.
(*
(*      PACKAGE
(*      VERIFY PACKAGE.
(*
(*      ARGUMENTS
(*      INPUT
(*      KEYE      - KEY OF ENTITY TO BE RETRIEVED FROM NETWORK.
(*
(*      OUTPUT
(*      RR        - THE FUNCTION RETURN RECORD.
(*
(*-----*)
(**)
(* END %INCLUDE VERGT. *)
```

```
(* %INCLUDE VERUD *)
(* (VERFORM) VERIFY ROUTINE FORMALS. *)
(**)
  PROCEDURE VERUD(CONST KEYE:ENTKEY;VAR ENTDEF:ENTBLOCK;
    VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*
(*      AUTHOR:  UNKNOWN                      CADD      CREATED: YY/MM/DD CC
(*      VERSION: MAS VER 2                      REVISED: 84/10/11 CC
(*
(*      FUNCTION:
(*          VERIFY LEGALITY OF UPDATING AN ENTITY WITH THE USER
(*          SUPPLIED ENTITY KEY USING THE USER SUPPLIED ENTITY DATA
(*          BLOCK AND LIST OF CONSTITUENTS.
(*
(*      ENVIRONMENT:
(*          IBM PASCAL LANGUAGE
(*          IBM 30XX, 43XX, DEC VAX 11/780
(*
(*      DESCRIPTION OF ARGUMENTS:
(*          NAME      I/O  DESCRIPTION
(*          KEYE      I    KEY OF EXISTING ENTITY.
(*          ENTDEF    I    USER SUPPLIED DATA FOR NEW ENTITY BLOCK.
(*          KEYL      I    KEY OF LIST OF CONSTITUENTS TO BE CONNECTED
(*                      TO THIS ENTITY.
(*          RR        0    ERROR CONDITION RETURN CODE.
(*                      = 0  NORMAL RETURN CODE.
(*
(*      COMMONS:
(*
(*      PROCESSING DESCRIPTION:
(*
(*      COMMENTS:
(*
(*      CHANGE CONTROL:
(*          84/10/11 MAS VER 2  D. J. KERCHNER
(*          UPDATED DOCUMENTATION.
(*          84/10/04 MAS VER 2  E. D. SHREVE
(*          CHANGED ENTDEF FROM CONST TO VAR.
(*-----*)
(**)
(* END %INCLUDE VERUD *)
```

```

(*) %INCLUDE XIEMM. *)
(**)
  PROCEDURE XIEMM(VAR KEYE:ENTKEY;VAR RR:RET_REC);EXTERNAL;
(**)
(*-----*)
(*)
(*) $FUNCTION:
(*)   TO DELETE AN ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KEYE          I/O  KEY OF ENTITY TO BE DELETED, WILL BE
(*)                   SET TO NIL.
(*)
(*)   RR            O    THE FUNCTION RETURN CODE.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   INTERNAL PROCEDURE FOR THE MODEL ACCESS SOFTWARE
(*)
(*) $PROCESSING DESCRIPTION:
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)
(*)   REVISED: 09/13/85          L. J. BEHAN          FRMI
(*)   CHANGED TO ENSURE THE DECREMENTING OF THE READ POSITION OF A
(*)   USER ENTITY CONSTITUENT LIST
(*)
(*)   REVISED: 02/18/85          B. A. ULMER          FRMI
(*)   CHANGED THE STRUCTURE OF THE INTERNAL ITEM FOR THE
(*)   IMPLEMENTATION OF THE CRB
(*)
(*)   REVISED: 10/05/84          E. D. SHREVE          FRMI
(*)   CHANGED THE KEYE PARMS FOR XULST AND XCLST TO VAR
(*)

```

3.10.3 Name/Value Interface

3.10.3.1 Index

<u>Routine</u>	<u>Function</u>
ADBLOCA	- DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
ENUMLOCA	- DETERMINE THE LOCATION OF THE ENUMERATION TYPE.
GETDD	- RETRIEVE THE DATA DICTIONARY ENTRY FOR THE ENTITY KIND.
GETDDBN	- RETRIEVE THE DATA DICTIONARY ENTRY FOR THE ENTITY NAME.
NVCPATAV	- COPY A VALUE OF ARBITRARY SIZE.
NVCPAV	- COPY A VALUE OF ARBITRARY SIZE.
NVCRIM	- ENTER ATTRIBUTE NAMES INTO TRAVELSAL MAP.
NVDLTM	- DELETE TRAVERSAL MAP.
NVDQAN	- RETRIEVE THE VALUE OF THE REQUESTED ENTITY ATTRIBUTE.
NVDQARLO	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (NON-POINTER).
NVDQARPT	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (POINTER).
NVDQGTAV	- DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
NVDSARLO	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (NON-POINTER).
NVDSARPT	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (POINTER).
NVDSAV	- REPLACE THE VALUE OF THE REQUESTED ENTITY ATTRIBUTE.
NVDSENLO	- DETERMINE THE LOCATION OF THE ENUMERATION TYPE.
NVDSGTAV	- DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
NVEQAV	- IF VALUE EQUAL, THEN ADD ENTITY TO LIST.
NVGEAV	- IF VALUE GREATER OR EQUAL, THEN ADD ENTITY TO LIST.
NVGRAV	- IF VALUE GREATER, THEN ADD ENTITY TO LIST.
NVGTAN	- EXTRACT ATTRIBUTE NAME FROM STRING OF QUALIFIERS.
NVGTAT	- GET DATA DICTIONARY ATTRIBUTE DEFINITION.
NVGTDD	- GET DATA DICTIONARY ENTITY DEFINITION.
NVGTE	- GET ENTITY KIND, ADB SIZE, NUMBER OF CONSTITUENTS.
NVGTRS	- GET RUN-TIME SUBSCHEMA ENTITY DEFINITION.
NVLCAV	- LOCATE ATTRIBUTE VALUE USING TRAVERSAL MAP.
NVLEAV	- IF VALUE LESS OR EQUAL, THEN ADD ENTITY TO LIST.
NVLTAV	- IF VALUE LESS, THEN ADD ENTITY TO LIST.
NVNEAV	- IF VALUE NOT EQUAL, THEN ADD ENTITY TO LIST.
NVPQARLO	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (NON-POINTER).
NVPQARPT	- DETERMINE THE LOCATION OF THE REQUESTED ARRAY (POINTER).
NVPQAV	- CREATE LIST OF ENTITIES WITH THE SPECIFIED VALUE FOR THE SPECIFIED ATTRIBUTE FOR THE SPECIFIED KIND OF ENTITY.
NVRTVRS	- RETRIEVE ENTITY DEFINITIONS FROM THE FILE.
RSCPAI	- COPY ARRAY INDEX TABLE INTO THE RUN-TIME SUBSCHEMA.
RSCPAT	- COPY SIZE AND LOWER BOUND OF ARRAY INTO THE RUN-TIME SUBSCHEMA.
RSCPCI	- COPY THE POINTER INDEX TABLE INTO THE RUN-TIME SUBSCHEMA.
RSCPCT	- COPY THE KINDS OF POINTERS INTO THE RUN-TIME SUBSCHEMA.
RSCPEI	- COPY THE ENUMERATION INDEX TABLE INTO THE RUN-TIME SUBSCHEMA.
RSCPET	- COPY THE ENUMERATION VALUES INTO THE RUN-TIME SUBSCHEMA.
RSGTDD	- BUILD RUN-TIME SUBSCHEMA FROM DATA DICTIONARY ENTRY.
RSTRDD	- TRANSLATE A DATA DICTIONARY ENTRY INTO A RUN-TIME SUBSCHEMA ENTRY.

3.10.3.2 Listings

```
(* BEGIN %INCLUDE ADBLOCA *****)
(*)
FUNCTION ADBLOCA ( CONST ENTITY_POINTER    : ENTPNTR;
                  CONST OFFSET             : INTEGER)
                  : T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*)   DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ENTITY_POINTER  I    POINTER TO ADB OF ENTITY INSTANCE
(*)   OFFSET          I    OFFSET TO ATTRIBUTE VALUE
(*)   ADBLOCA         O    POINTER TO ATTRIBUTE VALUE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   POINT TO ATTRIBUTE LOCATION
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)   ORIGINATED: 06 NOVEMBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE ADBLOCA *****)
```

```

(*) BEGIN %INCLUDE ENUMLOCA *****
(*)
FUNCTION ENUMLOCA ( CONST ENTITY_POINTER    : ENTPNTR;
                   CONST INDEX              : INTEGER;
                   CONST SCHEMA             : T_SCHEMA_POINTER )
                   : T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF THE ENUMERATION TYPE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ===  =====
(*)     ENTITY_POINTER    I  POINTER TO ADB OF ENTITY INSTANCE
(*)     ENUMLOCA          O  POINTER TO ATTRIBUTE VALUE
(*)     INDEX             I  INDEX OF THE CURRENT SEGMENT
(*)     SCHEMA            I  RUN-TIME SUBSCHEMA ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST POINTER TO SELECTOR IN ADB OF ENTITY INSTANCE
(*)     PCINT TO ENUMERATION INDEX TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ENUMERATION VALUES TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ATTRIBUTE VALUE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 06 NOVEMBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE ENUMLOCA *****

```

```
(* BEGIN %INCLUDE GETDD ***** *)
(*)
PROCEDURE GETDD ( CONST KIND           : INTEGER;
                  CONST MAX_AVAIL      : INTEGER;
                  CONST ATTRIBUTE_ORDER : CHAR;
                  VAR  USER_ARRAY      : T_USER_ARRAY;
                  VAR  MAX_ACTUAL       : INTEGER;
                  VAR  RETURN_CODE     : INTEGER );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   READ THE DATA DICTIONARY INTO THE APPLICATION PROGRAM.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====          ===  =====
(*)   ATTRIBUTE_ORDER I    SPECIFICATION OF THE ORDER FOR
(*)                       ATTRIBUTES IN THE ENTITY
(*)                       DEFINITION
(*)   KIND            I    A KIND NUMBER OF ENTITY
(*)   MAX_ACTUAL       O    AN ACTUAL NUMBER OF RECORDS IN
(*)                       ENTITY DEFINITION
(*)   MAX_AVAIL        I    A NUMBER OF 80 CHARACTER RECORDS
(*)                       AVAILABLE IN CALLER TO HOLE
(*)                       ENTITY DEFINITION
(*)   USER_ARRAY       O    AN ENTITY DEFINITION
(*)   RETURN_CODE      O    RETURN CODE
(*)                       -1 = ACTUAL SIZE GREATER THAN
(*)                           SPACE AVAILABLE
(*)                       0 = SUCCESS
(*)                       1 = KIND NOT IN DATA DICTIONARY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH DATA DICTIONARY INDEX FILE
(*)   IF KIND IN DATA DICTIONARY THEN
(*)       GET ENTITY DEFINITION FROM DDFILE
(*)       FILL UP THE ARRAY OF ENTITY DEFINITIONS UP TO NUMBER
```


April 1990

```

(*) BEGIN %INCLUDE GETDDBN *****
(*)
PROCEDURE GETDDBN ( CONST ENTITY_NAME      : T_ENTITY_NAME;
                   CONST MAX_AVAIL        : INTEGER;
                   CONST ATTRIBUTE_ORDER   : CHAR;
                   VAR  USER_ARRAY         : T_USER_ARRAY;
                   VAR  MAX_ACTUAL         : INTEGER;
                   VAR  RETURN_CODE        : INTEGER );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   READ THE DATA DICTIONARY INTO THE APPLICATION PROGRAM,
(*)   GIVEN THE ENTITY NAME.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ATTRIBUTE_ORDER  I   SPECIFICATION OF THE ORDER FOR
(*)                       ATTRIBUTES IN THE ENTITY
(*)                       DEFINITION
(*)   ENTITY_NAME      I   AN ENTITY NAME
(*)   MAX_ACTUAL        O   AN ACTUAL NUMBER OF RECORDS IN
(*)                       ENTITY DEFINITION
(*)   MAX_AVAIL         I   A NUMBER OF 80 CHARACTER RECORDS
(*)                       AVAILABLE IN CALLER TO HOLE
(*)                       ENTITY DEFINITION
(*)   USER_ARRAY        O   AN ENTITY DEFINITION
(*)   RETURN_CODE       O   RETURN CODE
(*)                       -1 = ACTUAL SIZE GREATER THAN
(*)                           SPACE AVAILABLE
(*)                       0 = SUCCESS
(*)                       1 = KIND NOT IN DATA DICTIONARY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH DATA DICTIONARY INDEX FILE
(*)   IF ENTITY NAME IN DATA DICTIONARY THEN
(*)   GET ENTITY DEFINITION FROM DDFILE

```

```
(*          FILL UP THE ARRAY OF ENTITY DEFINITIONS UP TO NUMBER  *)
(*          OF RECORDS AVAILABLE IN CALLER                          *)
(*          END IF                                                  *)
(*          END LOOP                                                *)
(*          *)                                                      *)
(*  $COMMENTS:                                                      *)
(*          *)                                                      *)
(*  $CHANGE CONTROL:                                                *)
(*          ORIGINATED: 24 NOVEMBER 1987, M. H. CHOI, DBMA        *)
(*          *)                                                      *)
(*  END %INCLUDE GETDDBN *****)
```

```
(* BEGIN %INCLUDE NVCPATAV *****)
(*)
PROCEDURE NVCPATAV ( VAR OUTPUT_VALUE  : T_WORD;
                    CONST INPUT_VALUE   : T_WORD;
                    CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY A VALUE OF ARBITRARY SIZE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY ATTRIBUTE VALUE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 14 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE NVCPATAV *****)
```

```

(* BEGIN %INCLUDE NVCPAV *****)
(*)
PROCEDURE NVCPAV ( VAR  OUTPUT_VALUE  : T_VALUE;
                  CONST INPUT_VALUE   : T_VALUE;
                  CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY A VALUE OF ARBITRARY SIZE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ===  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY ATTRIBUTE VALUE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED
(*)             GLOBAL DECLARATIONS INTO 'NVITYP'.
(*)     ORIGINATED: 15 OCTOBER 1985, G. A. WHITE, FRMI
(*)
(*) END %INCLUDE NVCPAV *****)

```

```

(*) BEGIN %INCLUDE NVCRTM *****
(*)
PROCEDURE NVCRTM ( CONST NAME_STRING      : T_ATTRIBUTE_NAME;
                   VAR  NAME_ROOT         : T_NAME_POINTER;
                   VAR  TRAVERSAL_DEPTH   : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     ENTER ATTRIBUTE NAMES INTO TRAVERSAL MAP.
(*)
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     NAME_STRING    I    SCHEMA NAME FOR THE ATTRIBUTE WHICH
(*)                       IS TERMINATED BY A NULL (HEX '00').
(*)     NAME_ROOT      O    POINTER TO TRAVERSAL MAP WHICH
(*)                       CONTAINS ATTRIBUTE NAMES AND THE
(*)                       CORRESPONDING SCHEMA DEFINITIONS.
(*)     NO_OF_DIMENSION O    NUMBER OF ARRAY DIMENSION
(*)     TRAVERSAL_DEPTH O    NUMBER OF NAMES
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH ATTRIBUTE NAME STRING
(*)     OBTAIN ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING
(*)     STORE ATTRIBUTE NAME IN THE TRAVERSAL MAP
(*)     INCREMENT TRAVERSAL_DEPTH
(*)     ENDLOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 05 MAY 1986, M. H. CHOI, FRMI
(*)
(*) END %INCLUDE NVCRTM *****

```

April 1990

```

(* BEGIN %INCLUDE NVDLTM *****)
(*)
PROCEDURE NVDLTM ( VAR   NAME_ROOT      : T_NAME_POINTER;
                   VAR   TRAVERSAL_SIZE : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   DELETE TRAVERSAL MAP.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ==  =====
(*)   NAME_ROOT      I    POINTER TO TRAVERSAL MAP WHICH
(*)                   CONTAINS ATTRIBUTE NAMES AND THEIR
(*)                   CORRESPONDING SCHEMA DEFINITIONS.
(*)   TRAVERSAL_SIZE  O    NUMBER OF NAMES
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   LOOP THROUGH ATTRIBUTE NAMES
(*)   LOOP THROUGH SCHEMA DEFINITIONS FOR ATTRIBUTE NAME
(*)   DELETE ATTRIBUTE DEFINITION
(*)   INCREMENT TRAVERSAL MAP SIZE
(*)   ENDLOOP
(*)   DELETE ATTRIBUTE NAME
(*)   ENDLOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)   ORIGINATED: 05 MAY 1986, M. H. CHOI, FRMI
(*)
(*) END %INCLUDE NVDLTM *****)

```

```

(*) BEGIN %INCLUDE NVDQAN *****
(*)
PROCEDURE NVDQAN(CONST ENTITY_KEY      : ENTKEY;
                  CONST NAME_STRING    : T_ATTRIBUTE_NAME;
                  CONST DIMEN_VALUE    : T_DIMEN_VALUE;
                  VAR  ATTRIBUTE_VALUE : T_ATTRIBUTE_VALUE;
                  VAR  NVI_RETURN_CODE : EXT_RET_CODE);
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     RETRIEVE THE VALUE OF THE REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     ATTRIBUTE_VALUE  0  VALUE OF THE ATTRIBUTE
(*)     DIMEN_VALUE      I  VALUE OF ARRAY SUBSCRIPT
(*)     ENTITY_KEY       I  POINTER TO THE ENTITY INSTANCE
(*)     NAME_STRING      I  SCHEMA NAME FOR THE ATTRIBUTE (OR
(*)                          CONCATENATED SCHEMA NAME FOR THE
(*)                          ATTRIBUTE OF A CONSTITUENT) WHICH IS
(*)                          TERMINATED BY A NULL (HEX'00')
(*)     NVI_RETURN_CODE  0  EXTERNAL RETURN CODE
(*)                          = 0  SUCCESS
(*)                          > 0  CRITICAL ERROR:
(*)                              1  KIND NOT IN RUN-TIME SUBSCHEMA
(*)                              2  ATTRIBUTE NOT IN ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST ATTRIBUTE LOCATION
(*)     IF ATTRIBUTE LOCATION OBTAINED THEN
(*)         COPY ATTRIBUTE VALUE
(*)         RETURN SUCCESS
(*)     ELSE
(*)         RETURN FAILURE
(*)     ENDIF

```



```
(* $COMMENTS: *)
(*) *)
(*) $CHANGE CONTROL: *)
(*) REVISD: 26 FEBRUARY 1987, M. H. CHOI, DBMA *)
(*) HANDLE ARRAY ATTRIBUTES *)
(*) REVISD: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED *)
(*) GLOBAL DECLARATIONS INTO 'NVITYP'. *)
(*) REVISD: 21 MARCH 1986, G. A. WHITE, FRMI, *)
(*) DETECT NIL ENTITY KEY AS ERROR *)
(*) ORIGINATED: 15 OCTOBER 1985, G. A. WHITE, FRMI *)
(*) *)
(*) END %INCLUDE NVDQAN *****)
```

April 1990

```

(*) BEGIN %INCLUDE NVDQARLO *****
(*)
FUNCTION NVDQARLO ( CONST ENTITY_INSTANCE : T_INT_ITEM;
                    CONST INDEX           : INTEGER;
                    CONST SCHEMA          : T_SCHEMA_POINTER;
                    CONST DIMEN_VALUE     : T_DIMEN_VALUE;
                    CONST NO_OF_DIMENSION : INTEGER;
                    VAR  ARRAY_SIZE       : INTEGER;
                    VAR  ATTRIBUTE_VALUE  : T_ATTRIBUTE_VALUE;
                    VAR  ARRAY_TYPE       : INTEGER;
                    VAR  DIMEN_COUNT      : INTEGER;
                    VAR  NVI_RETURN_CODE  : EXT_RET_CODE )
    : T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE
(*)     OF ARRAY TYPE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ===  =====
(*)     NVDQARLO       0    POINTER TO ATTRIBUTE VALUE
(*)     ARRAY_SIZE     0    SIZE OF THE ATTRIBUTE VALUE
(*)     DIMEN_VALUE     I    VALUE OF ARRAY SUBSCRIPT
(*)     ENTITY_INSTANCE I    POINTER TO ADB OF ENTITY INSTANCE
(*)     INDEX           I    INDEX OF THE CURRENT SEGMENT
(*)     NO_OF_DIMENSION I    NUMBER OF ARRAY DIMENSIONS
(*)     SCHEMA         I    RUN-TIME SUBSCHEMA ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST POINTER TO SELECTOR IN ADB OF ENTITY INSTANCE
(*)     POINT TO ARRAY INDEX TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ARRAY LIST TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ATTRIBUTE LOCATION
(*)

```

```
(*  $COMMENTS:                                     *)
(*  *)                                              *)
(*  $CHANGE CONTROL:                               *)
(*    REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*    ORIGINATED: 02 JANUARY 1987, M. H. CHOI, DBMA *)
(*  *)                                              *)
(* END %INCLUDE NVDQARLO *****)
```

```

(* BEGIN %INCLUDE NVDQARPT *****)
PROCEDURE NVDQARPT ( CONST ENTITY      : ENTKEY;
                    VAR  VALUE_INDEX   : INTEGER;
                    VAR  ATTRIBUTE_VALUE : T_ATTRIBUTE_VALUE );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF THE ARRAY OF POINTER.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME      I/O  DESCRIPTION
(*)     ====      ==  =====
(*)     CNST_LIST      0  ENTITIES IN THE CONSTITUENT LIST
(*)     ENTITY         I  POINTER TO ADB OF ENTITY INSTANCE
(*)     NO_OF_CL       0  NUMBER OF INSTANCES IN THE
(*)                       CONSTITUENT LIST
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH NUMBER OF INSTANCES IN THE CONSTITUENT LIST
(*)     WHILE CONSTITUENT KIND <> 1100 THEN
(*)         POINT TO CONSTITUENT KEY
(*)     END LOOP
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 18 FEBRUARY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE NVDQARPT *****)

```

```
(* BEGIN %INCLUDE NVDQGTAV *****)
(*)
PROCEDURE NVDQGTAV ( CONST ENTITY_INSTANCE : T_INT_ITEM;
                     CONST NAME_STRING      : T_ATTRIBUTE_NAME;
                     CONST DIMEN_VALUE      : T_DIMEN_VALUE;
                     VAR  POSITION           : INTEGER;
                     VAR  POINTER           : T_VARIANT_POINTER;
                     VAR  ATTRIBUTE_SIZE    : INTEGER;
                     VAR  ATTRIBUTE_VALUE   : T_ATTRIBUTE_VALUE;
                     VAR  DIMEN_COUNT       : INTEGER;
                     VAR  NVI_RETURN_CODE   : EXT_RET_CODE);

    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     ATTRIBUTE_SIZE    0  SIZE OF THE ATTRIBUTE (BYTES)
(*)     DIMEN_VALUE       I  VALUE OF ARRAY SUBSCRIPT
(*)     ENTITY_INSTANCE   I  ENTITY INSTANCE NODE
(*)     NAME_STRING       I  SCHEMA NAME FOR THE ATTRIBUTE (OR
(*)                          CONCATENATED SCHEMA NAME FOR THE
(*)                          ATTRIBUTE OF A CONSTITUENT) WHICH IS
(*)                          TERMINATED BY A NULL (HEX'00')
(*)     NVI_RETURN_CODE   0  EXTERNAL RETURN CODE
(*)                          = 0 SUCCESS
(*)                          > 0 CRITICAL ERROR:
(*)                          1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)                          2 ATTRIBUTE NOT IN ENTITY
(*)     POINTER           0  POINTER TO ATTRIBUTE VALUE
(*)     POSITION           I/O LOCATION OF ATTRIBUTE NAME IN NAME
(*)                          STRING (FOR REFERRING TO ATTRIBUTE
(*)                          NAME OF A CONSTITUENT ENTITY)
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT EXTERNAL)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM SUBPROGRAM NVDQAN ORIGINALLY, WILL CALL ITSELF
(*)     RECURSIVELY FOR POINTER ATTRIBUTES
(*)
```

```
(* $PROCESSING DESCRIPTION: *)
(* REQUEST RUN-TIME SUBSCHEMA DEFINITION OF ENTITY *)
(* IF RUN-TIME SUBSCHEMA ENTITY DEFINITION OBTAINED THEN *)
(* OBTAIN ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING *)
(* IF ATTRIBUTE NAME OBTAINED THEN *)
(* SEARCH SCHEMA ENTITY DEFINITION FOR ATTRIBUTE NAME *)
(* IF ATTRIBUTE NAME FOUND THEN *)
(* POINT TO ATTRIBUTE LOCATION *)
(* OBTAIN ATTRIBUTE SIZE *)
(* RETURN SUCCESS *)
(* ELSE *)
(* RETURN FAILURE *)
(* ENDIF *)
(* ELSE *)
(* POINT TO APPLICATION DATA BLOCK *)
(* OBTAIN APPLICATION DATA BLOCK SIZE *)
(* RETURN SUCCESS *)
(* ENDIF *)
(* ELSE *)
(* RETURN FAILURE *)
(* ENDIF *)
(* $COMMENTS: *)
(* $CHANGE CONTROL: *)
(* REVISED: 26 FEBRUARY 1987, M. H. CHOI, DBMA *)
(* HANDLE ARRAY ATTRIBUTES *)
(* REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED *)
(* GLOBAL DECLARATIONS INTO 'NVITYP'. *)
(* ORIGINATED: OCTOBER 1985, G. A. WHITE, FRMI *)
(* END %INCLUDE NVDQGTAV *****)
```

```
(* BEGIN %INCLUDE NVDSARLO *****)
(*)
FUNCTION NVDSARLO ( CONST ENTITY_INSTANCE : T_INT_ITEM;
                   CONST INDEX           : INTEGER;
                   CONST SCHEMA          : T_SCHEMA_POINTER;
                   CONST DIMEN_VALUE     : T_DIMEN_VALUE;
                   CONST NO_OF_DIMENSION : INTEGER;
                   VAR  ARRAY_SIZE       : INTEGER;
                   VAR  ATTRIBUTE_VALUE  : T_ATTRIBUTE_VALUE;
                   VAR  ARRAY_TYPE       : INTEGER;
                   VAR  DIMEN_COUNT      : INTEGER;
                   VAR  NVI_RETURN_CODE  : EXT_RET_CODE )
: T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*)   DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE
(*)   OF ARRAY TYPE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O DESCRIPTION
(*)   =====
(*)   NVDSARLO       0   POINTER TO ATTRIBUTE VALUE
(*)   ARRAY_SIZE     0   SIZE OF THE ATTRIBUTE VALUE
(*)   DIMEN_VALUE     I   VALUE OF ARRAY SUBSCRIPT
(*)   ENTITY_INSTANCE I   POINTER TO ADB OF ENTITY INSTANCE
(*)   INDEX           I   INDEX OF THE CURRENT SEGMENT
(*)   NO_OF_DIMENSION I   NUMBER OF ARRAY DIMENSIONS
(*)   SCHEMA          I   RUN-TIME SUBSCHEMA ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   REQUEST POINTER TO SELECTOR IN ADB OF ENTITY INSTANCE
(*)   POINT TO ARRAY INDEX TABLE IN RUN-TIME SUBSCHEMA
(*)   POINT TO ARRAY LIST TABLE IN RUN-TIME SUBSCHEMA
(*)   POINT TO ATTRIBUTE LOCATION
(*)
```

CI PS560240032U
April 1990

```
(*  $COMMENTS:                                     *)
(*  *)                                              *)
(*  $CHANGE CONTROL:                               *)
(*    REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*    ORIGINATED: 14 DECEMBER 1987, M. H. CHOI, DBMA *)
(*  *)                                              *)
(* END %INCLUDE NVDSARLO *****)
```



```

(* BEGIN %INCLUDE NVDSARPT *****
PROCEDURE NVDSARPT ( CONST ENTITY          : ENTKEY;
                     VAR  VALUE_INDEX      : INTEGER;
                     VAR  ATTRIBUTE_VALUE  : T_ATTRIBUTE_VALUE );
    EXTERNAL;

(* *)
(* $FUNCTION: *)
(* DETERMINE THE LOCATION OF THE ARRAY OF POINTER. *)
(* *)
(* $DESCRIPTION OF ARGUMENTS: *)
(* NAME          I/O  DESCRIPTION *)
(* ===== *)
(* CNST_LIST      0    ENTITIES IN THE CONSTITUENT LIST *)
(* ENTITY         1    POINTER TO ADB OF ENTITY INSTANCE *)
(* NO_OF_CL       0    NUMBER OF INSTANCES IN THE *)
(*                  CONSTITUENT LIST *)
(* *)
(* $COMMONS: *)
(* *)
(* $ENVIRONMENT: *)
(* LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM) *)
(* HARDWARE SYSTEM: IBM 360/370/4341/4381 *)
(* *)
(* $EXECUTION PROCEDURE: *)
(* NAME/VALUE INTERFACE *)
(* CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM *)
(* *)
(* $PROCESSING DESCRIPTION: *)
(* LOOP THROUGH NUMBER OF INSTANCES IN THE CONSTITUENT LIST *)
(* WHILE CONSTITUENT KIND <> 1100 THEN *)
(*     POINT TO CONSTITUENT KEY *)
(* END LOOP *)
(* *)
(* $COMMENTS: *)
(* *)
(* $CHANGE CONTROL: *)
(* REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(* ORIGINATED: 14 DECEMBER 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVDSARPT *****

```

```

(*) BEGIN %INCLUDE NVDSAV *****
(*)
PROCEDURE NVDSAV(CONST ENTITY_KEY      : ENTKEY;
                  CONST NAME_STRING     : T_ATTRIBUTE_NAME;
                  CONST DIMEN_VALUE     : T_DIMEN_VALUE;
                  VAR  ATTRIBUTE_VALUE  : T_ATTRIBUTE_VALUE;
                  VAR  NVI_RETURN_CODE  : EXT_RET_CODE);
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     REPLACE THE VALUE OF THE REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     ATTRIBUTE_VALUE  I   VALUE OF THE ATTRIBUTE
(*)     DIMEN_VALUE     I   VALUE OF ARRAY SUBSCRIPT
(*)     ENTITY_KEY      I   POINTER TO THE ENTITY INSTANCE
(*)     NAME_STRING     I   SCHEMA NAME FOR THE ATTRIBUTE (OR
(*)                          CONCATENATED SCHEMA NAME FOR THE
(*)                          ATTRIBUTE OF A CONSTITUENT) WHICH IS
(*)                          TERMINATED BY A NULL (HEX'00')
(*)     NVI_RETURN_CODE  O   EXTERNAL RETURN CODE
(*)                          = 0  SUCCESS
(*)                          > 0  CRITICAL ERROR:
(*)                              1  KIND NOT IN RUN-TIME SUBSCHEMA
(*)                              2  ATTRIBUTE NOT IN ENTITY
(*)                              3  NIL ENTITY KEY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST ATTRIBUTE LOCATION
(*)     IF ATTRIBUTE LOCATION OBTAINED THEN
(*)         REPLACE ATTRIBUTE VALUE
(*)         RETURN SUCCESS
(*)     ELSE
(*)         RETURN FAILURE

```

```
(*      ENDIF                                          *)
(*)                                                  *)
(*) $COMMENTS:                                       *)
(*)                                                  *)
(*) $CHANGE CONTROL:                                *)
(*)      REVISED : 26 FEBRUARY 1987, M. H. CHOI, DBMA *)
(*)      HANDLE ARRAY ATTRIBUTES                    *)
(*)      ORIGINATED: 09 SEPTEMBER 1986, M. H. CHOI, DBMA *)
(*)                                                  *)
(*) END %INCLUDE NVDSAV *****)
```

```
(* BEGIN %INCLUDE NVDSSENLO *****)
(*)
FUNCTION NVDSSENLO ( CONST ENTITY_POINTER    : ENTPNTR;
                     CONST INDEX              : INTEGER;
                     CONST SCHEMA             : T_SCHEMA_POINTER;
                     CONST VALUE_NAME         : T_SCHEMA_NAME;
                     VAR  SCALAR              : T_HEX_BYTE;
                     VAR  RETURN_CODE         : EXT_RET_CODE )
    : T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*)     DETERMINE THE LOCATION OF THE ENUMERATION TYPE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ===  =====
(*)     ENTITY_POINTER    I  POINTER TO ADB OF ENTITY INSTANCE
(*)     NVDSSENLO         0  POINTER TO ATTRIBUTE VALUE
(*)     INDEX             I  INDEX OF THE CURRENT SEGMENT
(*)     SCHEMA            I  RUN-TIME SUBSCHEMA ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST POINTER TO SELECTOR IN ADB OF ENTITY INSTANCE
(*)     POINT TO ENUMERATION INDEX TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ENUMERATION VALUES TABLE IN RUN-TIME SUBSCHEMA
(*)     POINT TO ATTRIBUTE VALUE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*)     ORIGINATED: 12 DECEMBER 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE NVDSSENLO *****)
```

```

(* BEGIN %INCLUDE NVDSGTAV *****)
(*)
PROCEDURE NVDSGTAV ( CONST ENTITY_INSTANCE : T_INT_ITEM;
                     CONST NAME_STRING    : T_ATTRIBUTE_NAME;
                     CONST DIMEN_VALUE    : T_DIMEN_VALUE;
                     VAR  POSITION         : INTEGER;
                     VAR  POINTER         : T_VARIANT_POINTER;
                     VAR  ATTRIBUTE_VALUE : T_ATTRIBUTE_VALUE;
                     VAR  ATTRIBUTE_SIZE  : INTEGER;
                     VAR  DIMEN_COUNT     : INTEGER;
                     VAR  NVI_RETURN_CODE : EXT_RET_CODE);

    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   =====
(*)   ATTRIBUTE_SIZE  0   SIZE OF THE ATTRIBUTE (BYTES)
(*)   DIMEN_VALUE     I   VALUE OF ARRAY SUBSCRIPT
(*)   ENTITY_INSTANCE I   ENTITY INSTANCE NODE
(*)   NAME_STRING     I   SCHEMA NAME FOR THE ATTRIBUTE (OR
(*)                       CONCATENATED SCHEMA NAME FOR THE
(*)                       ATTRIBUTE OF A CONSTITUENT) WHICH IS
(*)                       TERMINATED BY A NULL (HEX'00')
(*)   NVI_RETURN_CODE  0   EXTERNAL RETURN CODE
(*)                       = 0 SUCCESS
(*)                       > 0 CRITICAL ERROR:
(*)                           1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)                           2 ATTRIBUTE NOT IN ENTITY
(*)   POINTER         0   POINTER TO ATTRIBUTE VALUE
(*)   POSITION         I/O  LOCATION OF ATTRIBUTE NAME IN NAME
(*)                       STRING (FOR REFERRING TO ATTRIBUTE
(*)                       NAME OF A CONSTITUENT ENTITY)
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT EXTERNAL)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM SUBPROGRAM NVDQAN ORIGINALLY, WILL CALL ITSELF
(*)   RECURSIVELY FOR POINTER ATTRIBUTES
(*)

```

```
(* $PROCESSING DESCRIPTION: *)
(* REQUEST RUN-TIME SUBSCHEMA DEFINITION OF ENTITY *)
(* IF RUN-TIME SUBSCHEMA ENTITY DEFINITION OBTAINED THEN *)
(* OBTAIN ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING *)
(* IF ATTRIBUTE NAME OBTAINED THEN *)
(* SEARCH SCHEMA ENTITY DEFINITION FOR ATTRIBUTE NAME *)
(* IF ATTRIBUTE NAME FOUND THEN *)
(* POINT TO ATTRIBUTE LOCATION *)
(* OBTAIN ATTRIBUTE SIZE *)
(* RETURN SUCCESS *)
(* ELSE *)
(* RETURN FAILURE *)
(* ENDIF *)
(* ELSE *)
(* POINT TO APPLICATION DATA BLOCK *)
(* OBTAIN APPLICATION DATA BLOCK SIZE *)
(* RETURN SUCCESS *)
(* ENDIF *)
(* ELSE *)
(* RETURN FAILURE *)
(* ENDIF *)
(* $COMMENTS: *)
(* $CHANGE CONTROL: *)
(* ORIGINATED: 12 DECEMBER 1987, M. H. CHOI, DBMA *)
(* END %INCLUDE NVDSGTAV *****)
```

```

(*) BEGIN %INCLUDE NVEQAV *****
(*)
PROCEDURE NVEQAV ( CONST SELECTED_ENTITY : ENTKEY;
                   CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                   VAR  DATAREC           : T_DATAREC;
                   VAR  NVI_RETURN_CODE   : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT
(*)     EQUAL SPECIFIED VALUE THEN ADD ENTITY TO LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ===  =====
(*)     SELECTED_ENTITY   I   ENTITY INSTANCE KEY
(*)
(*)     ATTRIBUTE_DATA_BLOCK I   SUPPLIED BY MAKXEQ BUT NOT USED
(*)
(*)     DATAREC           I/O  A RECORD STRUCTURE PASSED THROUGH
(*)                           MAEXEQ WHICH CONTAINS THE PARAMETERS
(*)                           FROM NVPQAV : NAME_ROOT
(*)                           LIST_ROOT
(*)                           ATTRIBUTE_VALUE
(*)
(*)     NVI_RETURN_CODE   O   EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*)     IF ATTRIBUTE VALUE FOUND THEN
(*)         IF ATTRIBUTE VALUE EQUAL SPECIFIED VALUE THEN
(*)             ADD SELECTED ENTITY TO LIST
(*)         ENDIF
(*)     ENDIF
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVEQAV *****)


```

(*) BEGIN %INCLUDE NVGEAV *****
(*)
PROCEDURE NVGEAV ( CONST SELECTED_ENTITY : ENTKEY;
                   CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                   VAR  DATAREC             : T_DATAREC;
                   VAR  NVI_RETURN_CODE     : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT IS
(*) GREATER THAN OR EQUAL TO SPECIFIED VALUE THEN ADD ENTITY
(*) TO LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) =====
(*) SELECTED_ENTITY I ENTITY INSTANCE KEY
(*)
(*) ATTRIBUTE_DATA_BLOCK I SUPPLIED BY MAEXEQ BUT NOT USED
(*)
(*) DATAREC I/O A RECORD STRUCTURE PASSED THROUGH
(*) MAEXEQ WHICH CONTAINS THE PARAMETERS
(*) FROM NVPQAV : NAME_ROOT
(*) LIST_ROOT
(*) ATTRIBUTE_VALUE
(*)
(*) NVI_RETURN_CODE O EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) NAME/VALUE INTERFACE
(*) CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*) IF ATTRIBUTE VALUE FOUND THEN
(*) IF ATTRIBUTE VALUE GREATER THAN OR EQUAL
(*) SPECIFIED VALUE THEN
(*) ADD SELECTED ENTITY TO LIST
(*) ENDIF
(*) ENDIF
(*)

```

CI PS560240032U
April 1990

```
(*  $COMMENTS:                                     *)
(*  $CHANGE CONTROL:                               *)
(*  ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA      *)
(*  *)                                              *)
(* END %INCLUDE NVGEAV *****)
```

```

(*) BEGIN %INCLUDE NVGRAV *****
(*)
PROCEDURE NVGRAV ( CONST SELECTED_ENTITY : ENTKEY;
                  CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                  VAR  DATAREC             : T_DATAREC;
                  VAR  NVI_RETURN_CODE    : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT IS
(*)     GREATER THAN SPECIFIED VALUE THEN ADD ENTITY TO LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME             I/O  DESCRIPTION
(*)     ====            ===  =====
(*)     SELECTED_ENTITY   I    ENTITY INSTANCE KEY
(*)
(*)     ATTRIBUTE_DATA_BLOCK I    SUPPLIED BY MAKXEQ BUT NOT USED
(*)
(*)     DATAREC           I/O  A RECORD STRUCTURE PASSED THROUGH
(*)                          MAEXEQ WHICH CONTAINS THE PARAMETERS
(*)                          FROM NVPQAV : NAME_ROOT
(*)                                  LIST_ROOT
(*)                                  ATTRIBUTE_VALUE
(*)
(*)     NVI_RETURN_CODE   0    EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*)     IF ATTRIBUTE VALUE FOUND THEN
(*)         IF ATTRIBUTE VALUE GREATER THAN SPECIFIED VALUE THEN
(*)             ADD SELECTED ENTITY TO LIST
(*)         ENDIF
(*)     ENDIF
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVGRAV *****)

```

(*) BEGIN %INCLUDE NVGTAN *****
(*)
PROCEDURE NVGTAN(CONST NAME_STRING : T_ATTRIBUTE_NAME;
                  VAR   NO_OF_DIMENSION : INTEGER;
                  VAR   POSITION       : INTEGER;
                  VAR   NAME          : T_SCHEMA_NAME);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     EXTRACT ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING, STARTING
(*)     AT STRING POSITION.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     NAME          O    CURRENT SEGMENT OF ATTRIBUTE NAME
(*)     NAME_STRING   I    FULLY QUALIFIED ATTRIBUTE NAME
(*)     NO_OF_DIMENSION I/O  NUMBER OF ARRAY DIMENSIONS
(*)     POSITION       I/O  POSITION OF THE CURRENT SEGMENT
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT EXTERNAL)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     COPY ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING
(*)     UPDATE POSITION IN STRING
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     REVISED: 31 MARCH 1987, M. H. CHOI, DBMA
(*)             ADDED A NO_OF_DIMENSION PARAMETER TO HANDLE ARRAY
(*)     REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED
(*)             GLOBAL DECLARATIONS INTO 'NVITYP'.
(*)     ORIGINATED: 4 NOVEMBER 1985, G. A. WHITE, FRMI
(*)
(*) END %INCLUDE NVGTAN *****

```

April 1990

```

(* BEGIN %INCLUDE NVGTAT *****)
(*)
PROCEDURE NVGTAT ( CONST KIND          : INTEGER;
                   CONST ATTRIBUTE_NAME : T_ATTRIBUTE_NAME;
                   VAR  DATA_TYPE      : T_DATA_TYPE;
                   VAR  SIZE            : INTEGER;
                   VAR  RETURN_CODE     : INTEGER );
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   OBTAIN THE DATA TYPE AND THE ATTRIBUTE SIZE FOR THE
(*)   REQUESTED ENTITY ATTRIBUTE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   KIND          I    KIND NUMBER
(*)   ATTRIBUTE_NAME I    SCHEMA NAME FOR THE ATTRIBUTE (OR
(*)                       CONCATENATED SCHEMA NAME FOR THE
(*)                       ATTRIBUTE OF A CONSTITUENT) WHICH IS
(*)                       TERMINATED BY A NULL (HEX'00')
(*)   DATA_TYPE     O    ENTITY ATTRIBUTE DATA TYPE
(*)   SIZE           O    SIZE OF ENTITY ATTRIBUTE
(*)   RETURN_CODE    O    EXTERNAL RETURN CODE
(*)                       = 0 SUCCESS
(*)                       > 0 CRITICAL ERROR:
(*)                           1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)                           2 ATTRIBUTE NOT IN ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT EXTERNAL)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   REQUEST RUN-TIME SUBSCHEMA DEFINITION OF ENTITY
(*)   IF RUN-TIME SUBSCHEMA ENTITY DEFINITION OBTAINED THEN
(*)   OBTAIN ATTRIBUTE NAME FROM ATTRIBUTE NAME STRING
(*)   IF ATTRIBUTE NAME OBTAINED THEN
(*)   SEARCH SCHEMA ENTITY DEFINITION FOR ATTRIBUTE NAME
(*)   IF ATTRIBUTE NAME FOUND THEN
(*)   OBTAIN DATA TYPE

```

```
(*          OBTAIN ATTRIBUTE SIZE                      *)
(*          END IF                                      *)
(*          END IF                                      *)
(*          END IF                                      *)
(*          $COMMENTS:                                  *)
(*          $CHANGE CONTROL:                            *)
(*          ORIGINATED: 13 NOVEMBER 1987, M. H. CHOI, DBMA *)
(*          END %INCLUDE NVGTAT *****)
```

```

(*) BEGIN %INCLUDE NVGTDD *****
(*)
PROCEDURE NVGTDD ( CONST KIND_OF_ENTITY : ORD_KIND;
                   VAR  SCHEMA_POINTER  : T_SCHEMA_POINTER;
                   VAR  NVI_RETURN_CODE : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     GET DATA DICTIONARY ENTITY DEFINITION
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     KIND_OF_ENTITY I    KIND VALUE OF THE ENTITY FOR WHICH THE
(*)                       RUN-TIME SUBSCHEMA IS TO BE OBTAINED
(*)     NVI_RETURN_CODE O    RETURN CODE
(*)                       = 0 SUCCESS
(*)                       > 0 CRITICAL ERROR:
(*)                           1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)     SCHEMA_POINTER  O    POINTER TO THE RUN-TIME SUBSCHEMA ENTITY
(*)                       DEFINITION AFTER IT IS STORED INTO THE
(*)                       WORKING FORM.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)     CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST POINTER TO RUN_TIME SUBSCHEMA
(*)     IF RUN-TIME SUBSCHEMA NOT IN WORKING FORM THEN
(*)         REQUEST DATA DICTIONARY FROM SCHEMA MODEL
(*)         IF DATA DICTIONARY OBTAINED THEN
(*)             STORE DATA DICTIONARY IN WORKING FORM
(*)             REQUEST POINTER TO RUN_TIME SUBSCHEMA
(*)         ELSE
(*)             RETURN FAILURE
(*)         ENDIF
(*)     ENDIF
(*)

```


CI PS560240032U
April 1990

```
(*  $COMMENTS: *)
(*  *)
(*  $CHANGE CONTROL: *)
(*  ORIGINATED 29 APRIL 1987, M. H. CHOI, DBMA *)
(*  *)
(* END %INCLUDE NVGTDD *****)
```

April 1990

```

(* BEGIN %INCLUDE NVGTED ***** )
(*)
PROCEDURE NVGTED ( CONST ENTITY_NAME      : T_ATTRIBUTE_NAME;
                   VAR  ENTITY_KIND       : ORD_KIND;
                   VAR  ADB_SIZE          : INTEGER;
                   VAR  CL_LENGTH         : INTEGER;
                   VAR  RETURN_CODE       : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   OBTAIN THE ENTITY KIND, ADB SIZE AND NUMBER OF CONSTITUENTS
(*)   FOR THE REQUESTED ENTITY NAME.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   ENTITY_NAME    I    AN ENTITY NAME
(*)   ENTITY_KIND    0    KIND NUMBER
(*)   ADB_SIZE       0    TOTAL SIZE OF THE ADB
(*)   CL_LENGTH      0    NUMBER OF CONSTITUENTS
(*)   RETURN_CODE    0    EXTERNAL RETURN CODE
(*)                      = 0 SUCCESS
(*)                      > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT EXTERNAL)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   OBTAIN ENTITY NAME FROM ENTITY NAME STRING
(*)   REQUEST DATA DICTIONARY GIVEN ENTITY NAME
(*)   IF DATA DICTIONARY ENTITY DEFINITION OBTAINED THEN
(*)       OBTAIN ENTITY KIND
(*)       LOOP THROUGH THE ENTITY DEFINITION
(*)           IF ATTRIBUTE IN ADB THEN
(*)               OBTAIN THE LARGEST PHYSICAL SCHEMA ORDER
(*)           ELSE
(*)               OBTAIN THE LARGEST CONSTITUENT LIST POSITION
(*)           END IF
(*)       END LOOP
(*)       CALCULATE THE TOTAL ADB SIZE

```

```
(*          END IF                                *)
(*)                                              *)
(*) $COMMENTS:                                    *)
(*)                                              *)
(*) $CHANGE CONTROL:                             *)
(*)      ORIGINATED: 25 NOVEMBER 1987, M. H. CHOI, DBMA *)
(*)                                              *)
(*) END %INCLUDE NVGTED *****)
```

```

(*) BEGIN %INCLUDE NVGTRS *****
(*)
PROCEDURE NVGTRS ( CONST KIND_OF_ENTITY : ORD_KIND;
                   VAR  SCHEMA_POINTER  : T_SCHEMA_POINTER;
                   VAR  NVI_RETURN_CODE : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     GET RUN-TIME SUBSCHEMA ENTITY DEFINITION
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     KIND_OF_ENTITY I    KIND VALUE OF THE ENTITY FOR WHICH THE
(*)                       RUN-TIME SUBSCHEMA IS TO BE OBTAINED
(*)     NVI_RETURN_CODE O    RETURN CODE
(*)                       = 0 SUCCESS
(*)                       > 0 CRITICAL ERROR:
(*)                           1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)     SCHEMA_POINTER  O    POINTER TO THE RUN-TIME SUBSCHEMA ENTITY*
(*)                       DEFINITION AFTER IT IS STORED INTO THE
(*)                       WORKING FORM.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)     CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)     REQUEST POINTER TO RUN_TIME SUBSCHEMA
(*)     IF RUN-TIME SUBSCHEMA NOT IN WORKING FORM THEN
(*)         REQUEST RUN-TIME SUBSCHEMA FROM SCHEMA MODEL
(*)         IF RUN-TIME SUBSCHEMA OBTAINED THEN
(*)             STORE RUN-TIME SUBSCHEMA IN WORKING FORM
(*)             REQUEST POINTER TO RUN_TIME SUBSCHEMA
(*)         ELSE
(*)             RETURN FAILURE
(*)         ENDIF
(*)     ENDIF
(*)

```

```
(*  $COMMENTS: *)
(*  *)
(*  $CHANGE CONTROL: *)
(*    REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED *)
(*    GLOBAL DECLARATIONS INTO 'NVITYP'. *)
(*    ORIGINATED 23 JANUARY 1986, G. A. WHITE, FRMI *)
(*  *)
(* END %INCLUDE NVGTRS *****)
```

```

(*) BEGIN %INCLUDE NVLCAV *****
(*)
FUNCTION NVLCAV ( CONST SELECTED_ENTITY : ENTKEY;
                  VAR  DATAREC          : T_DATAREC;
                  VAR  POINTER          : T_VARIANT_POINTER;
                  VAR  ATTRIBUTE_SIZE   : INTEGER;
                  VAR  DATA_TYPE       : INTEGER;
                  VAR  NVI_RETURN_CODE  : EXT_RET_CODE)
: BOOLEAN;
  EXTERNAL;

(*)
(*) $FUNCTION:
(*)   LOCATE ATTRIBUTE VALUE USING TRAVERSAL MAP.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   SELECTED_ENTITY  I   ENTITY INSTANCE KEY
(*)
(*)   NAME_ROOT       I   POINTER TO TRAVERSAL MAP WHICH
(*)                       CONTAINS ATTRIBUTE NAMES AND THE
(*)                       CORRESPONDING SCHEMA DEFINITIONS
(*)
(*)   POINTER         O   POINTER TO ATTRIBUTE VALUE
(*)
(*)   ATTRIBUTE_SIZE   O   SIZE OF THE ATTRIBUTE (BYTES)
(*)
(*)   NVI_RETURN_CODE  O   EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   START WITH SPECIFIED KIND
(*)   LOOP THROUGH ATTRIBUTE NAMES IN TRAVERSAL MAP
(*)   IF ATTRIBUTE DEFINITION EXISTS FOR NAME/KIND THEN
(*)       CASE ATTRIBUTE DATA TYPE OF
(*)           IN_CL      : OBTAIN CONSTITUENT
(*)                       USE CONSTITUENT KIND
(*)                       OBTAIN DEFINITION OF CONSTITUENT
(*)

```

```
(*          ATTRIBUTE                                *)
(*      IN_STRUCTURE : OBTAIN DEFINITION OF STRUCTURE ELEMENT *)
(*      OTHERWISE    : TERMINATE TRAVERSAL                *)
(*          DETERMINE LOCATION OF ATTRIBUTE VALUE        *)
(*          ATTRIBUTE VALUE FOUND                        *)
(*      END CASE                                          *)
(*      ELSE                                              *)
(*          TERMINATE TRAVERSAL                          *)
(*          ATTRIBUTE VALUE NOT FOUND                    *)
(*      ENDIF                                            *)
(*      ENDLOOP                                          *)
(*  $COMMENTS:                                          *)
(*  $CHANGE CONTROL:                                    *)
(*      REVISED: 31 MARCH 1987, M. H. CHOI, DBMA        *)
(*          ADDED A DIMENSION VALUE PARAMETER TO HANDLE ARRAY *)
(*          OF POINTER.                                  *)
(*      ORIGINATED: 12 MAY 1986, G. A. WHITE, FRMI      *)
(*  END %INCLUDE NVLCAV *****)
```

```

(*) BEGIN %INCLUDE NVLEAV *****
(*)
PROCEDURE NVLEAV ( CONST SELECTED_ENTITY : ENTKEY;
                   CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                   VAR  DATAREC           : T_DATAREC;
                   VAR  NVI_RETURN_CODE   : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT IS
(*)     LESS THAN OR EQUAL TO SPECIFIED VALUE THEN ADD ENTITY TO
(*)     LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ===  =====
(*)     SELECTED_ENTITY   I   ENTITY INSTANCE KEY
(*)
(*)     ATTRIBUTE_DATA_BLOCK I   SUPPLIED BY MAEXEQ BUT NOT USED
(*)
(*)     DATAREC           I/O  A RECORD STRUCTURE PASSED THROUGH
(*)                           MAEXEQ WHICH CONTAINS THE PARAMETERS
(*)                           FROM NVPQAV : NAME_ROOT
(*)                           LIST_ROOT
(*)                           ATTRIBUTE_VALUE
(*)
(*)     NVI_RETURN_CODE   O   EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*)     IF ATTRIBUTE VALUE FOUND THEN
(*)         IF ATTRIBUTE VALUE LESS THAN OR EQUAL
(*)         SPECIFIED VALUE THEN
(*)             ADD SELECTED ENTITY TO LIST
(*)         ENDIF
(*)     ENDIF
(*)

```



```
(*  $COMMENTS:                                     *)
(*  $CHANGE CONTROL:                               *)
(*  ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA    *)
(*  END %INCLUDE NVLEAV *****)
```

```
(* BEGIN %INCLUDE NVLTAV *****~******)
(*)
PROCEDURE NVLTAV ( CONST SELECTED_ENTITY : ENTKEY;
                   CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                   VAR  DATAREC           : T_DATAREC;
                   VAR  NVI_RETURN_CODE   : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*) IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT IS
(*) LESS THAN SPECIFIED VALUE THEN ADD ENTITY TO LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) =====
(*) SELECTED_ENTITY I ENTITY INSTANCE KEY
(*)
(*) ATTRIBUTE_DATA_BLOCK I SUPPLIED BY MAKXEQ BUT NOT USED
(*)
(*) DATAREC I/O A RECORD STRUCTURE PASSED THROUGH
(*) MAEXEQ WHICH CONTAINS THE PARAMETERS
(*) FROM NVPQAV : NAME_ROOT
(*) LIST_ROOT
(*) ATTRIBUTE_VALUE
(*)
(*) NVI_RETURN_CODE O EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) NAME/VALUE INTERFACE
(*) CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*) IF ATTRIBUTE VALUE FOUND THEN
(*) IF ATTRIBUTE VALUE LESS THAN SPECIFIED VALUE THEN
(*) ADD SELECTED ENTITY TO LIST
(*) ENDIF
(*) ENDIF
(*)
(*) $COMMENTS:
(*)
```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVLTAV *****)

April 1990

```

(*) BEGIN %INCLUDE NVNEAV *****
(*)
PROCEDURE NVNEAV ( CONST SELECTED_ENTITY : ENTKEY;
                   CONST ATTRIBUTE_DATA_BLOCK : ENTBLOCK;
                   VAR  DATAREC           : T_DATAREC;
                   VAR  NVI_RETURN_CODE   : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     IF ATTRIBUTE VALUE FOR ENTITY INSTANCE OR CONSTITUENT
(*)     IS NOT EQUAL TO SPECIFIED VALUE THEN ADD ENTITY TO LIST.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====           ==  =====
(*)     SELECTED_ENTITY   I   ENTITY INSTANCE KEY
(*)
(*)     ATTRIBUTE_DATA_BLOCK  I   SUPPLIED BY MAKXEQ BUT NOT USED
(*)
(*)     DATAREC           I/O  A RECORD STRUCTURE PASSED THROUGH
(*)                          MAEXEQ WHICH CONTAINS THE PARAMETERS
(*)                          FROM NVPQAV : NAME_ROOT
(*)                                  LIST_ROOT
(*)                                  ATTRIBUTE_VALUE
(*)
(*)     NVI_RETURN_CODE    O   EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED VIA MAEXEQ FROM NVPQAV FOR EACH INSTANCE.
(*)
(*) $PROCESSING DESCRIPTION:
(*)     IF ATTRIBUTE VALUE FOUND THEN
(*)         IF ATTRIBUTE VALUE NOT EQUAL SPECIFIED VALUE THEN
(*)             ADD SELECTED ENTITY TO LIST
(*)         ENDIF
(*)     ENDIF
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 15 JULY 1987, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVNEAV *****)

```

(* BEGIN %INCLUDE NVPQARLO *****)
(*)
FUNCTION NVPQARLO ( CONST ENTITY_INSTANCE : T_INT_ITEM;
                    CONST INDEX           : INTEGER;
                    CONST SCHEMA          : T_SCHEMA_POINTER;
                    CONST DIMEN_VALUE     : T_DIMEN_VALUE;
                    CONST NO_OF_DIMENSION : INTEGER;
                    VAR  ATTRIBUTE_SIZE   : INTEGER;
                    VAR  ATTRIBUTE_VALUE   : T_ATTRIBUTE_VALUE;
                    VAR  ARRAY_TYPE        : INTEGER;
                    VAR  DIMEN_COUNT       : INTEGER;
                    VAR  NVI_RETURN_CODE   : EXT_RET_CODE )
: T_VARIANT_POINTER;
EXTERNAL;

(*)
(*) $FUNCTION:
(*) DETERMINE THE LOCATION OF THE REQUESTED ENTITY ATTRIBUTE
(*) OF ARRAY TYPE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*) NAME I/O DESCRIPTION
(*)
(*) =====
(*) NVPQARLO 0 POINTER TO ATTRIBUTE VALUE
(*) ARRAY_SIZE 0 SIZE OF THE ATTRIBUTE VALUE
(*) DIMEN_VALUE I VALUE OF ARRAY SUBSCRIPT
(*) ENTITY_INSTANCE I POINTER TO ADB OF ENTITY INSTANCE
(*) INDEX I INDEX OF THE CURRENT SEGMENT
(*) NO_OF_DIMENSION I NUMBER OF ARRAY DIMENSIONS
(*) SCHEMA I RUN-TIME SUBSCHEMA ENTITY
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*) LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*) HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*) NAME/VALUE INTERFACE
(*) CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*) REQUEST POINTER TO SELECTOR IN ADB OF ENTITY INSTANCE
(*) POINT TO ARRAY INDEX TABLE IN RUN-TIME SUBSCHEMA
(*) POINT TO ARRAY LIST TABLE IN RUN-TIME SUBSCHEMA
(*) POINT TO ATTRIBUTE LOCATION
(*)

```

CI PS560240032U
April 1990

```
(* $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*) REVISD: (DATE, NAME, GROUP, REASON/DESCRIPTION) *)
(*) ORIGINATED: 10 MARCH 1988, M. H. CHOI, DBMA *)
(*)
(*) END %INCLUDE NVPQARLO *****)
```

```

(* BEGIN %INCLUDE NVPQARPT *****
PROCEDURE NVPQARPT ( CONST ENTITY      : ENTKEY;
                     VAR  VALUE_INDEX  : INTEGER;
                     VAR  ARRAY_SIZE   : INTEGER;
                     VAR  ATTRIBUTE_VALUE : T_ATTRIBUTE_VALUE );
    EXTERNAL;

(*
(* $FUNCTION:
(*     DETERMINE THE LOCATION OF THE ARRAY OF POINTER.
(*
(* $DESCRIPTION OF ARGUMENTS:
(*     NAME          I/O  DESCRIPTION
(*     ====          ==  =====
(*     CNST_LIST     O    ENTITIES IN THE CONSTITUENT LIST
(*     ENTITY        I    POINTER TO ADB OF ENTITY INSTANCE
(*     NO_OF_CL      O    NUMBER OF INSTANCES IN THE
(*                          CONSTITUENT LIST
(*
(* $COMMONS:
(*
(* $ENVIRONMENT:
(*     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*
(* $EXECUTION PROCEDURE:
(*     NAME/VALUE INTERFACE
(*     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*
(* $PROCESSING DESCRIPTION:
(*     LOOP THROUGH NUMBER OF INSTANCES IN THE CONSTITUENT LIST
(*         WHILE CONSTITUENT KIND <> 1100 THEN
(*             POINT TO CONSTITUENT KEY
(*         END LOOP
(*
(* $COMMENTS:
(*
(* $CHANGE CONTROL:
(*     REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)
(*     ORIGINATED: 10 MARCH 1988, M. H. CHOI, DBMA
(*
(* END %INCLUDE NVPQARPT *****

```



```

(*) BEGIN %INCLUDE NVPQAV *****
(*)
PROCEDURE NVPQAV ( CONST APPLICATION_LIST : LISTKEY;
                  CONST NAME_STRING      : T_ATTRIBUTE_NAME;
                  CONST ATTRIBUTE_VALUE   : T_ATTRIBUTE_VALUE;
                  CONST DIMEN_VALUE      : T_DIMEN_VALUE;
                  CONST SIGN              : INTEGER;
                  VAR  ENTITY_LIST       : LISTKEY;
                  VAR  NVI_RETURN_CODE   : EXT_RET_CODE );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   CREATE LIST OF ENTITIES WITH THE SPECIFIED VALUE FOR THE
(*)   SPECIFIED ATTRIBUTE FOR THE SPECIFIED KIND OF ENTITY.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ==  =====
(*)   APPLICATION_LIST  I  LIST OF ENTITY KEYS
(*)
(*)   ATTRIBUTE_VALUE   I  VALUE OF THE ATTRIBUTE
(*)
(*)   NAME_STRING      I  SCHEMA NAME FOR THE ATTRIBUTE WHICH
(*)                       IS TERMINATED BY A NULL (HEX '00')
(*)
(*)   ENTITY_KEY       O  ENTITY INSTANCE
(*)
(*)   NVI_RETURN_CODE   O  EXTERNAL RETURN CODE
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CREATE TRAVERSAL MAP
(*)   LOOP THROUGH ALL ENTITY INSTANCES IN APPLICATION LIST
(*)   MAKE A LIST OF INSTANCES WITH SPECIFIED ATTRIBUTE VALUE
(*)   ENDLOOP
(*)   DELETE TRAVERSAL MAP
(*)   RETURN LIST OF INSTANCES WITH THE SPECIFIED ATTRIBUTE VALUE
(*)

```

```
(* $COMMENTS: *)
(*)
(*) $CHANGE CONTROL: *)
(*)   REVISED: 27 JULY 1987, M. H. CHOI, DBMA *)
(*)       ADDED A SIGN PARAMETER TO ALLOW A CHOICE OF THE *)
(*)       RELATIONAL OPERATOR. *)
(*)   REVISED: 31 MARCH 1987, M. H. CHOI, DBMA *)
(*)       ADDED A DIMENSION VALUE PARAMETER TO HANDLE ARRAY *)
(*)       OF POINTERS. *)
(*)   ORIGINATED: 05 MAY 1986, M. H. CHOI, FRMI *)
(*)
(*) END %INCLUDE NVPQAV *****)
```

```

(*) BEGIN %INCLUDE NVRTVRS *****
(*)
PROCEDURE NVRTVRS ( CONST KIND          : INTEGER;
                    VAR  RUNTIME         : T_RUN_TIME;
                    VAR  RUNTIME_SIZE    : INTEGER;
                    VAR  RETURN_CODE     : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)     RETRIEVE ENTITY DEFINITIONS FROM THE FILE
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ===          ===  =====
(*)     KIND          I
(*)     RUNTIME        0   RUN-TIME SUBSCHEMA WHICH CONTAINS
(*)                       THE ENTITY DEFINITION, ALONG WITH
(*)                       ANY ENUMERATION VALUES AND ANY ARRAY
(*)                       INFORMATIONS, IN A COMPACTED FORM.
(*)     RUNTIME_SIZE   0   THE NUMBER OF BYTES ACTUALLY REQUIRED
(*)                       FOR THE COMPACTED RUN-TIME SUBSCHEMA.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     NAME/VALUE INTERFACE
(*)     CALLED FROM EITHER PASCAL OR FORTRAN APPLICATION PROGRAM
(*)
(*) $PROCESSING DESCRIPTION:
(*)     LOOP THROUGH INXFILE
(*)         IF KIND FOUND IN INXFILE THEN
(*)             LOOP THROUGH DATAFILE
(*)                 IF KIND FOUND IN DATAFILE THEN
(*)                     STORE ENTITY DEFINITION IN TEMPORARY WORK AREA
(*)                 END IF
(*)             END IF
(*)         END LOOP
(*)     STORE ENTITY DEFINITION INTO RUN-TIME SUBSCHEMA
(*)     STORE SIZE OF ENTITY DEFINITION INTO RUN-TIME SUBSCHEMA
(*)
(*) $COMMENTS:
(*)

```

CI PS560240032U
April 1990

(* \$CHANGE CONTROL: *)
(* ORIGINATED: 21 OCTOBER 1986, M. H. CHOI, DBMA *)
(* *)
(* END %INCLUDE NVRTVRS *****)

```

(*) BEGIN %INCLUDE RSCPAI *****
(*)
PROCEDURE RSCPAI ( VAR   OUTPUT_VALUE   : T_DATA_VALUE;
                   CONST INPUT_VALUE    : T_ARRAY_INDEX;
                   CONST SIZE_OF_VALUE  : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   COPY THE ARRAY INDEX TABLE INFORMATION INTO THE RUN-TIME
(*)   SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ==  =====
(*)   INPUT_VALUE     I    INPUT VALUE OF ARBITRARY SIZE
(*)   OUTPUT_VALUE    O    OUTPUT VALUE OF ARBITRARY SIZE
(*)   SIZE_OF_VALUE   I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CALL MACHINE DEPENDENT ROUTINE TO COPY ARRAY TABLE INFO
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPAI *****

```

```

(*) BEGIN %INCLUDE RSCPAT *****
(*)
PROCEDURE RSCPAT ( VAR   OUTPUT_VALUE : T_DATA_VALUE;
                  CONST INPUT_VALUE  : T_ARRAY_LIST;
                  CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;
(*)
(*) $FUNCTION:
(*)     COPY THE SIZE AND THE LOWER BOUND OF THE ARRAY INTO THE
(*)     RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O   DESCRIPTION
(*)     ====           ===   =====
(*)     INPUT_VALUE     I     INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE    O     OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE   I     SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY THE SIZE AND THE
(*)     LOWER BOUND OF THE ARRAY
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPAT *****

```

```
(* BEGIN %INCLUDE RSCPCI *****)
(*)
PROCEDURE RSCPCI ( VAR   OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE   : T_CL_INDEX;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE POINTER INDEX TABLE INFORMATION INTO THE
(*)     RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME           I/O  DESCRIPTION
(*)     ====          ===  =====
(*)     INPUT_VALUE     I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE    O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE   I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY CL INDEX INFO
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 29 JANUARY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPCI *****)
```

```
(* BEGIN %INCLUDE RSCPCT *****)
(*)
PROCEDURE RSCPCT ( VAR  OUTPUT_VALUE  : T_DATA_VALUE;
                   CONST INPUT_VALUE   : T_CL_KINDS;
                   CONST SIZE_OF_VALUE : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE KINDS OF POINTERS INTO THE RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME          I/O  DESCRIPTION
(*)     ====          ==  =====
(*)     INPUT_VALUE    I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE   O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE  I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY KINDS OF POINTER
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 29 JANUARY 1987, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPCT *****)
```



```
(* BEGIN %INCLUDE RSCPEI *****)
(*)
PROCEDURE RSCPEI ( VAR   OUTPUT_VALUE  : T_DATA_VALUE;
                  CONST INPUT_VALUE    : T_ENUM_INDEX;
                  CONST SIZE_OF_VALUE  : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)     COPY THE ENUMERATION INDEX TABLE INFORMATION INTO THE
(*)     RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)     NAME                I/O  DESCRIPTION
(*)     ====                ==  =====
(*)     INPUT_VALUE          I    INPUT VALUE OF ARBITRARY SIZE
(*)     OUTPUT_VALUE         O    OUTPUT VALUE OF ARBITRARY SIZE
(*)     SIZE_OF_VALUE        I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)     LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)     HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)     RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)     CALL MACHINE DEPENDENT ROUTINE TO COPY ENUMERATION INFO
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)     ORIGINATED: 01 OCTOBER 1986, M. H. CHOI, DBMA
(*)
(*) END %INCLUDE RSCPEI *****)
```

```
(* BEGIN %INCLUDE RSCPET *****)
(*)
PROCEDURE RSCPET ( VAR   OUTPUT_VALUE   : T_DATA_VALUE;
                   CONST INPUT_VALUE    : T_ENUMERATION;
                   CONST SIZE_OF_VALUE  : INTEGER);
    EXTERNAL;

(*)
(*) $FUNCTION:
(*)   COPY THE ENUMERATION VALUES INTO THE RUN-TIME SUBSCHEMA.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ===  =====
(*)   INPUT_VALUE     I    INPUT VALUE OF ARBITRARY SIZE
(*)   OUTPUT_VALUE     O    OUTPUT VALUE OF ARBITRARY SIZE
(*)   SIZE_OF_VALUE    I    SIZE OF VALUE TO BE COPIED
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   RUN-TIME SUBSCHEMA
(*)
(*) $PROCESSING DESCRIPTION:
(*)   CALL MACHINE DEPENDENT ROUTINE TO COPY ENUMERATION TABLE
(*)
(*) $COMMENTS:
(*)
(*) $CHANGE CONTROL:
(*)   REVISED: 16 MAY 1986, GEORGE A. WHITE, FRMI, REORGANIZED
(*)           GLOBAL DECLARATIONS INTO 'NVITYP'.
(*)   ORIGINATED: 15 OCTOBER 1985, G. A. WHITE, FRMI
(*)
(*) END %INCLUDE RSCPET *****)
```

```

(*) BEGIN %INCLUDE RSGTDD *****
(*)
PROCEDURE RSGTDD ( CONST KIND_OF_ENTITY : INTEGER;
                   VAR  RUN_TIME        : T_RUN_TIME;
                   VAR  RUN_TIME_SIZE   : INTEGER;
                   VAR  RTS_RETURN_CODE : INTEGER );
    SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   BUILD RUN-TIME SUBSCHEMA FROM DATA DICTIONARY ENTRY
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME          I/O  DESCRIPTION
(*)   ====          ===  =====
(*)   KIND_OF_ENTITY I    KIND VALUE OF THE ENTITY FOR WHICH THE
(*)                       RUN-TIME SUBSCHEMA WILL BE BUILT.
(*)   RTS_RETURN_CODE O   RETURN CODE
(*)                       = 0  SUCCESS
(*)                       > 0  CRITICAL ERROR:
(*)                           1 KIND NOT IN RUN-TIME SUBSCHEMA
(*)   RUN_TIME        O   RUN-TIME SUBSCHEMA WHICH CONTAINS THE
(*)                       ENTITY DEFINITION, ALONG WITH ANY
(*)                       ENUMERATION VALUES, CONSTITUENT LIST
(*)                       KINDS, AND ARRAY INFORMATION, IN A
(*)                       COMPACTED FORM.
(*)   RUN_TIME_SIZE   O   THE NUMBER OF BYTES ACTUALLY REQUIRED
(*)                       FOR THE COMPACTED RUN-TIME SUBSCHEMA.
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   RUN-TIME SUBSCHEMA (TEMPORARY ROUTINE)
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   IF KIND NUMBER EXIST IN DATA DICTIONARY THEN
(*)       TRANSLATE DATA DICTIONARY ENTRY INTO ENTITY ATTRIBUTES,
(*)       ENUMERATION VALUES, CONSTITUENT LIST KINDS AND ARRAY
(*)       INFORMATIONS (LOWER BOUND AND SIZE OF EACH DIMENSIONS)
(*)   IF THERE WERE ANY ENUMERATION ATTRIBUTES THEN
(*)       CALCULATE THE STARTING POSITION OF THE ENUMERATION
(*)       INDEX TABLE AND STORE INTO RUN-TIME SUBSCHEMA
(*)       DETERMINE THE ACTUAL SIZE OF THE ENUMERATION INDEX
(*)       TABLE

```

```
(*          COPY ENUMERATION INDEX TABLE INFORMATION INTO          *)
(*          RUN-TIME SUBSCHEMA                                     *)
(*          CALCULATE THE STARTING POSITION OF THE ENUMERATION      *)
(*          VALUE TABLE AND STORE INTO RUN-TIME SUBSCHEMA       *)
(*          DETERMINE THE ACTUAL SIZE OF THE ENUMERATION VALUE    *)
(*          TABLE                                               *)
(*          COPY THE ENUMERATION VALUES INTO THE RUN-TIME       *)
(*          SUBSCHEMA                                           *)
(*        ENDIF                                               *)
(*        IF THERE WERE ANY ARRAY ATTRIBUTES THEN               *)
(*          CALCULATE THE STARTING POSITION OF THE ARRAY INDEX     *)
(*          TABLE AND STORE INTO RUN-TIME SUBSCHEMA            *)
(*          DETERMINE THE ACTUAL SIZE OF THE ARRAY INDEX TABLE  *)
(*          COPY ARRAY TABLE INDEX INFORMATION INTO RUN-TIME    *)
(*          SUBSCHEMA                                           *)
(*          CALCULATE THE STARTING POSITION OF THE ARRAY LIST     *)
(*          TABLE AND STORE INTO RUN-TIME SUBSCHEMA            *)
(*          DETERMINE THE ACTUAL SIZE OF THE ARRAY LIST TABLE  *)
(*          COPY ARRAY LIST INFORMATION INTO RUN-TIME SUBSCHEMA  *)
(*        ENDIF                                               *)
(*        IF THERE WERE ANY POINTER ATTRIBUTES THEN             *)
(*          CALCULATE THE STARTING POSITION OF THE CL INDEX TABLE *)
(*          AND STORE INTO RUN-TIME SUBSCHEMA                    *)
(*          DETERMINE THE ACTUAL SIZE OF THE CL INDEX TABLE    *)
(*          COPY CL TABLE INDEX INFORMATION INTO RUN-TIME      *)
(*          SUBSCHEMA                                           *)
(*          CALCULATE THE STARTING POSITION OF THE CL LIST TABLE *)
(*          AND STORE INTO RUN-TIME SUBSCHEMA                    *)
(*          DETERMINE THE ACTUAL SIZE OF THE CL LIST TABLE    *)
(*          COPY ELIG. KINDS INFORMATION INTO RUN-TIME SUBSCHEMA *)
(*        ENDIF                                               *)
(*          CALCULATE THE SIZE OF THE RUN-TIME SUBSCHEMA        *)
(*        ELSE                                               *)
(*          RETURN FAILURE                                       *)
(*        ENDIF                                               *)
(*      $COMMENTS:                                           *)
(*      $CHANGE CONTROL:                                       *)
(*        ORIGINATED: 29 APRIL 1987, M. H. CHOI, DBMA          *)
(*      END %INCLUDE RSGTDD *****                          *)
```

```

(* BEGIN %INCLUDE RSTRDD *****)
(*)
PROCEDURE RSTRDD ( CONST KIND_OF_ENTITY   : INTEGER;
                   VAR  ENTITY            : T_SCHEMA;
                   VAR  ENUM              : T_ENUM_COMPACTOR;
                   VAR  ENUM_INDEX        : T_ENUM_INX_COMPACTOR;
                   VAR  ARRAY_LIST        : T_ARRAY_LIST_COMPACTOR;
                   VAR  ARRAY_INDEX       : T_ARRAY_INX_COMPACTOR;
                   VAR  CL_INDEX          : T_CL_INX_COMPACTOR;
                   VAR  CL_LIST           : T_CL_KINDS_COMPACTOR;
                   VAR  RTS_RETURN_CODE   : INTEGER );
  SUBPROGRAM;

(*)
(*) $FUNCTION:
(*)   TRANSLATE A DATA DICTIONARY ENTRY INTO A RUN-TIME SUBSCHEMA
(*)   ENTITY, ENUMERATION TABLE, ARRAY INFO TABLE, AND CL TABLE.
(*)
(*) $DESCRIPTION OF ARGUMENTS:
(*)   NAME           I/O  DESCRIPTION
(*)   ====           ==  =====
(*)   ARRAY_INDEX    0    RUN-TIME SUBSCHEMA ARRAY TABLE INDEX
(*)                   INFORMATION.
(*)   ARRAY_LIST     0    RUN-TIME SUBSCHEMA ARRAY TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM           0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   AND COMPACTION INFORMATION.
(*)   ENUM_INDEX     0    RUN-TIME SUBSCHEMA ENUMERATION TABLE
(*)                   INDEX INFORMATION.
(*)   ENTITY         0    RUN-TIME SUBSCHEMA ENTITY DEFINITION.
(*)   KIND_OF_ENTITY I    KIND VALUE OF THE ENTITY FOR WHICH
(*)                   THE TRANSLATION WILL BE PERFORMED.
(*)   RTS_RETURN_CODE 0    RETURN CODE
(*)                   = 0 SUCCESS
(*)                   > 0 CRITICAL ERROR:
(*)
(*) $COMMONS:
(*)
(*) $ENVIRONMENT:
(*)   LANGUAGE: IBM PASCAL (SEGMENT SUBPROGRAM)
(*)   HARDWARE SYSTEM: IBM 360/370/4341/4381
(*)
(*) $EXECUTION PROCEDURE:
(*)   NAME/VALUE INTERFACE
(*)   CALLED FROM THE NAME/VALUE INTERFACE
(*)
(*) $PROCESSING DESCRIPTION:
(*)   OBTAIN ENTITY NAME AND KIND FROM DATA DICTIONARY
(*)   STORE ENTITY NAME AND KIND INTO RUN-TIME SUBSCHEMA

```

```

(*)      LOOP THROUGH DATA DICTIONARY ENTRIES                      *)
(*)      OBTAIN ATTRIBUTE ENTRY FROM DATA DICTIONARY                *)
(*)      CASE DATA TYPE OF                                          *)
(*)          INTEGER, REAL, STRING, LOGICAL                          *)
(*)          : APPLICATION_DATA_BLOCK_ATTRIBUTE, PROCEDURE (1) *)
(*)          POINTER : CONSTITUENT_LIST_ATTRIBUTE, PROCEDURE (2) *)
(*)          ENUMERATION: ENUMERATION_ATTRIBUTE, PROCEDURE (3) *)
(*)      ENDCASE                                                    *)
(*)      ENDLLOOP                                                  *)
%PAGE
(*)      PROCEDURE (1) : APPLICATION_DATA_BLOCK_ATTRIBUTE          *)
(*)      STORE ATTRIBUTE DEFINITION FOR TYPE IN DATA DICTIONARY ENTRY*)
(*)                                                                *)
(*)      PROCEDURE (2) : CONSTITUENT_LIST_ATTRIBUTE                *)
(*)      OBTAIN CONSTITUENT LIST POSITION FROM SCHEMA MODEL          *)
(*)      STORE ATTRIBUTE DEFINITION FOR TYPE IN SCHEMA MODEL ENTRY *)
(*)                                                                *)
(*)      PROCEDURE (3) : ENUMERATION_ATTRIBUTE                     *)
(*)      STORE ATTRIBUTE DEFINITION FOR ENUMERATION TYPE            *)
(*)      OBTAIN NUMBER OF ENUMERATION VALUES FROM SCHEMA MODEL     *)
(*)      STORE NUMBER OF ENUMERATION VALUE IN ENUMERATION INDEX TABLE*)
(*)      STORE ENUMERATION VALUE TABLE INDEX POSITION IN ENUMERATION *)
(*)      INDEX TABLE                                              *)
(*)      LOOP THROUGH ENUMERATION VALUES                          *)
(*)      OBTAIN ENUMERATION VALUE FROM SCHEMA MODEL                *)
(*)      STORE ENUMERATION VALUE IN ENUMERATION VALUE TABLE       *)
(*)      END LOOP                                                  *)
(*)                                                                *)
(*)      PROCEDURE (4) : ARRAY_ATTRIBUTE                            *)
(*)      DETERMINE THE NUMBER OF ARRAY DIMENSIONS                  *)
(*)      STORE ARRAY INFORMATION INTO RUN-TIME SUBSCHEMA            *)
(*)      STORE TABLE INDEX POSITION FOR ARRAY LIST TABLE AND THE  *)
(*)      NUMBER OF DIMENSIONS INTO ARRAY INDEX TABLE              *)
(*)      CALCULATE TOTAL SIZE OF THE ARRAY AND STORE INTO ARRAY    *)
(*)      INDEX TABLE                                              *)
(*)      FOR THE NUMBER OF ARRAY DIMENSIONS                        *)
(*)      CALCULATE THE SIZE OF EACH ARRAY                          *)
(*)      STORE SIZE AND LOW-BOUND INTO ARRAY LIST TABLE           *)
(*)      END LOOP                                                  *)
(*)                                                                *)
(*)      $COMMENTS:                                                *)
(*)                                                                *)
(*)      $CHANGE CONTROL:                                          *)
(*)      REVISED: (DATE, NAME, GROUP, REASON/DESCRIPTION)          *)
(*)      ORIGINATED: 29 APRIL 1987, M. H. CHOI, DBMA                *)
(*)                                                                *)
(*)      END %INCLUDE RSTRDD *****

```