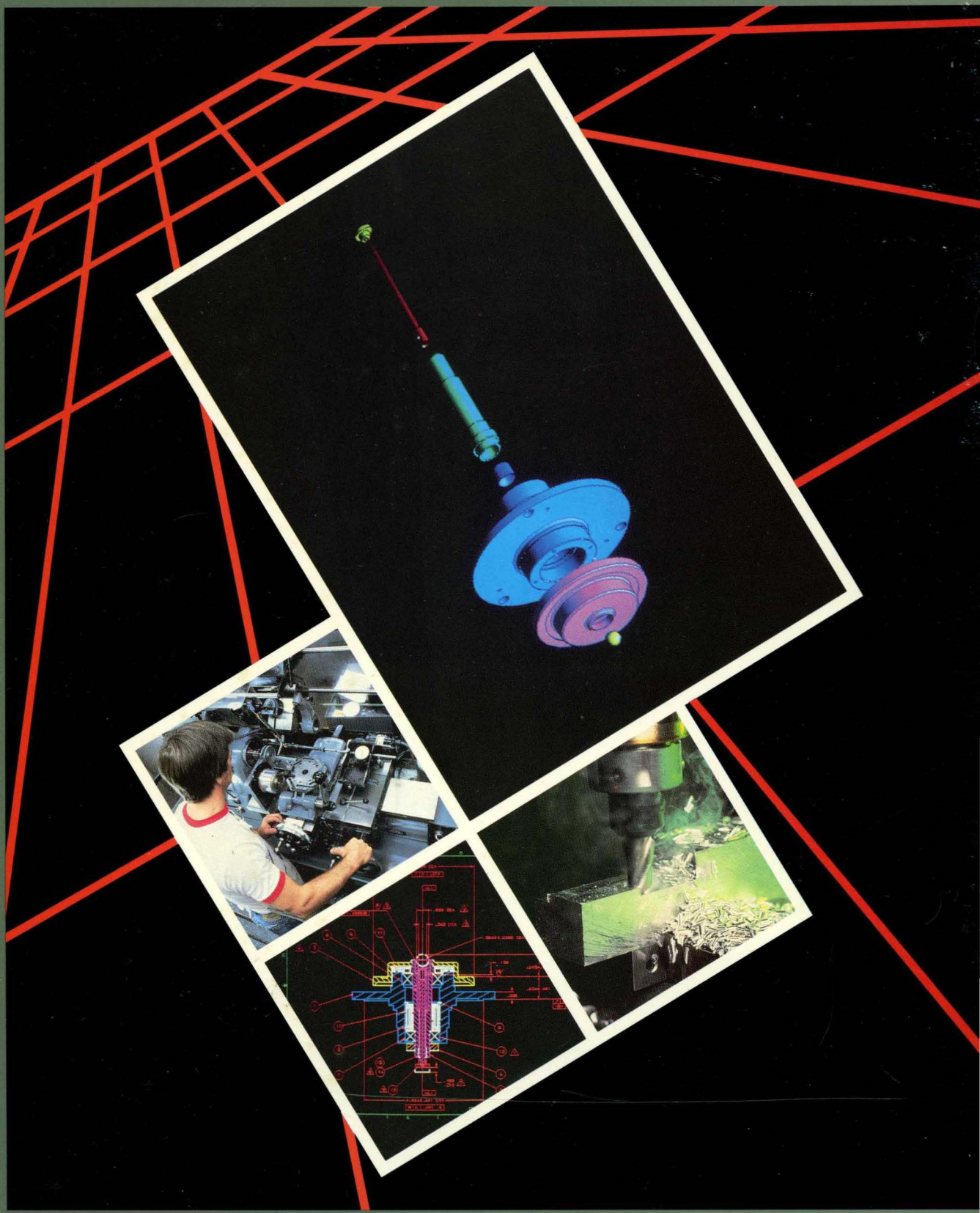
ICEM Engineering Data Library

Customization Guide for NOS

CONTROL DATA



60000168

ICEM Engineering Data Library

Customization Guide for NOS

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

Publication Number 60000168

Manual History

Revision A printed in January 1987 describes customization techniques for EDL version 1.2.5 running under the Network Operating System (NOS) version 2.5.1 at PSR level 664.

Contents

Figures

About This Manual

CDC® ICEM Engineering Data Library (EDL) is an application designed to provide a user-friendly interface to Control Data's CAD/CAM products and to manage the engineering data produced by these products. The EDL system runs under Control Data's Network Operating System (NOS).

EDL interfaces to the following application packages:

- ICEM DDN 1.62
- ICEM Solid Modeler 1.13
- PATRAN 1.5
- UNISTRUCT II
- ICEM Schematics
- XEDIT Text Editor
- Full Screen Editor (FSE)

The software product IMF is included with EDL.

Other software products are required to use EDL:

- Network Operating System (NOS)
  - FORTRAN 5 is required to perform some kinds of customization operations.

This manual describes how to customize EDL. Information about installing EDL is provided with your release tapes.

Audience

This manual is intended for system and database administrators whose duties include the maintenance and modification of the EDL database. You should have a thorough understanding of EDL, NOS, FORTRAN, and QUERY UPDATE before attempting to customize your EDL database.

Organization

The organization of this manual is as follows:

Chapter 1	Provides an introduction to the EDL file structure and the duties of the System Administrator.
Chapter 2	Describes customization of the Message and Task Database using the interactive MENUMOD utility.
Chapter 3	Describes more complex customization and manipulation of the EDL databases involving the use of OVCAPS, Information Base (IB) routines, and other EDL subroutines.
Chapter 4	Describes adding an application to EDL.
Appendix A	Contains complete database schemata for the Message and Task Database and the Engineering Data Database.
Appendix B	Provides the pseudo QU/DML format for the IB routines used to manipulate data records in EDL databases.
Appendix C	Lists descriptions and parameter information for standard EDL subprograms used in database customization.

Conventions

The word "system" when used in this manual refers to the ICEM EDL software system. When the Control Data Network Operating System is referred to, it is called either NOS or the operating system.

All text that the system displays is shown in uppercase letters and highlighted with a special typeface, as shown below:

```
SYSTEM ADMINSTRATOR TASKS

1. EXIT                                    E,EXIT
2. UPDATE THE MESSAGE AND TASK DATABASE    MENUMGMT
3. UPDATE THE DATABASE WITH BATCH INPUT    QUBATCH
4. INTERACTIVE QUERY UPDATE                QU
5. INTERACTIVE MENU MODIFICATION           MENUMOD
6. DISPLAY THE CURRENT EDL EXECUTION STACK STACK
7. DISPLAY THE CURRENT EDL VARIABLES       DISVAR
```

Related Publications

The following manuals contain information about ICEM Engineering Data Library (EDL), the NOS Operating System, and related applications.

EDL Manuals	Publication Number
EDL DBA Manual for NOS	60458880
EDL Instant for NOS	60000166
EDL Reference Manual for NOS	60459740
EDL User's Guide for NOS	60000167

Operating System Manuals	Publication Number
NOS Full Screen Editor User's Guide	60460420
NOS Version 2 Information Management Facility Version 2 Reference Manual	60484600
NOS Version 2 Reference Set, Volume 1 Introduction to Interactive Usage	60459660
NOS Version 2 Reference Set, Volume 3 System Commands	60459680
NOS Version 2 Reference Set, Volume 4 Program Interface	60459690
Query Update Version 3 Reference Manual	60498300
XEDIT Version 3 Reference Manual	60455730

ICEM Applications Manuals	Publication Number
CYBERNET UNISTRUCT II Reference Manual	76079600
ICEM Advanced Design for NOS	60461430
ICEM DDN Instant for NOS	60457140
ICEM Design/Drafting Basic Construction for NOS	60461420
ICEM Design/Drafting Data Management for NOS	60461410
ICEM Design/Drafting Drafting Functions for NOS	60461440
ICEM Design/Drafting GRAPL Programming Language for NOS	60461460
ICEM Design/Drafting Introduction and System Controls for NOS	60457130
ICEM Design/Drafting User's Guide for NOS	60456940
ICEM GPL for NOS	60462520
ICEM Numerical Control for NOS	60461450
ICEM Schematics Reference Manual	60456540

ICEM Applications Manuals (Cont)	Publication Number
IGES Translator for NOS	60463050
PATRAN User's Guide, Volume 1	60459330
PATRAN User's Guide, Volume 2	60459340
UNIPLOT Version 3 User's Guide/Reference Manual	60454730
UNISTUCT II User's Guide	60457550

Required Equipment

You can use any alphanumeric terminal for the EDL customization procedures described in this manual. You need extended terminal capabilities only if you access an application that requires them. For example, the ICEM Solid Modeler application requires a graphics capability.

Ordering Manuals

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

Submitting Comments

The last page of this manual is a comment sheet. Please use it to give us your opinion of the manual's usability, to suggest specific improvements, and to report technical or typographical errors. If the comment sheet has already been used, you can mail your comments to:

Control Data Corporation
Technology and Publications Division ARH219
4201 Lexington Avenue North
St. Paul, Minnesota 55126-6198

Please indicate whether you would like a written response.

System Administration Overview 1

System Administration Overview 1

This chapter provides an overview of the EDL file structure. It introduces the duties of the system administrator and concepts of EDL customization.

Customizing EDL

The ICEM Engineering Data Library is a flexible system that can be modified in many different ways. To successfully customize EDL you must be thoroughly familiar with EDL, NOS 2, FORTRAN, IMF, CYBER Control Language (CCL), and Query Update. You are responsible for ensuring that your customizations are well designed and tested. Customizations that work incorrectly (or fail to consider all potential impacts) can seriously damage your EDL databases or your applications data.

NOTE

Control Data cannot guarantee that the customizations you make to one version of EDL will automatically operate on subsequent versions of EDL or ICEM applications. We consider the impact of changes to customizations and provide conversion procedures to upgrade data maintained by standard code. However, normal enhancement, bug fixes, and product evolution may result in changes to the database structure and the function of code supplied by Control Data. These changes mean that you should re-adapt and retest your site-specific code, transaction files, Query Update directives, and CCL procedures at every EDL release.

There are several ways to customize EDL to fit your site. Each of the following customization techniques is described in this manual.

- Changing the text displayed by prompts, messages, and menus
- Reorganizing the EDL task menu structure
- Adding new applications, file types, and data types
- Adding or changing engineering categories and their standard attributes
- Creating new reports or modifying standard ones
- Creating new FORTRAN modules to perform site-specific functions
- Creating new procedures to be invoked by EDL

To change the tasks performed by EDL, you must first change the databases as required and then create CCL procedure files and/or a new version of the EDL program.

- Changes to the Message and Task Database (MDB) affect the verbage, structure, content, and operation of the user interface.
- Changes to the Engineering Data Database (DDB) define site-specific notions of data and allow you to integrate new applications.

The EDL system consists of the following four basic files that you can modify to fit your needs.

File Name	Description
E125PRC	Standard EDL procedure file
E125ABS	EDL absolute program
E125MDB	Message and Task Database
E125DDB	Engineering Data Database

CAUTION

To prevent serious damage to your EDL database, you should first make all changes to a *copy* of the working database.

1. Copy your EDL directory structure into a working area.
2. Make your changes to this copy and test them out.
3. After testing your changes, implement them on your current databases.

System Administrator Tasks Menu

The System Administrator Tasks menu lists the tasks used for EDL customization.

```
SYSTEM ADMINSTRATOR TASKS

 1. EXIT                                        E,EXIT
 2. UPDATE THE MESSAGE AND TASK DATABASE        MENUMGMT
 3. UPDATE THE DATABASE WITH BATCH QU INPUT     QUBATCH
 4. INTERACTIVE QUERY UPDATE                    QU
 5. INTERACTIVE MENU MODIFICATION               MENUMOD
 6. DISPLAY THE CURRENT EDL EXECUTION STACK     STACK
 7. DISPLAY THE CURRENT EDL VARIABLES           DISVAR
```

The following table summarizes the functions of these system administrator tasks:

Task Name	Description
EXIT	Terminates EDL processing of the current task and returns control to the previous task.
UPDATE THE MESSAGE AND TASK DATABASE	Lets you customize tasks and messages in batch mode. Examples of this type of customization are included in chapter 3.
UPDATE THE DATABASE WITH BATCH QU INPUT	Allows you to update the application database in batch mode. This procedure is discussed in chapter 3.
INTERACTIVE QUERY UPDATE	Allows you to update the application database interactively using Query Update.
INTERACTIVE MENU MODIFICATION	Accesses the interactive MENUMOD utility for customizing the Menu and Task database. This utility is described in chapter 2.
DISPLAY THE CURRENT EDL EXECUTION STACK	Displays the current tasks in the execution stack. Figure 1-1 shows a sample execution stack.
DISPLAY THE CURRENT EDL VARIABLES	Displays variables created by PUTVAR and accessible by GETVAR. The PUTVAR subroutine is described in chapter 3. Figure 1-2 shows a sample display produced in response to this selection.

TASK	TASK NAME	SEQUENCE	TYPE	NAME
LAST EDL	LAST EDL	10	TASK MENU	LASTEDL
USER	USER	10	TASK MENU	USER
SYSADMIN	SYSADMIN	10	TASK MENU	SYSADMIN

Figure 1-1. Sample EDL Execution Stack

```
NAME        VALUE
USR         EDLID
HOST
AUN
MDB         E125MDB
MUN         E125PRC
DDB         E125DDB
DUN
EDITOR      FSE
DDNVER      1.62
```

Figure 1-2. Sample Current EDL Variable Display

Figure 1-3 shows the hierarchy of tasks you can select from the System Administrator Tasks menu.

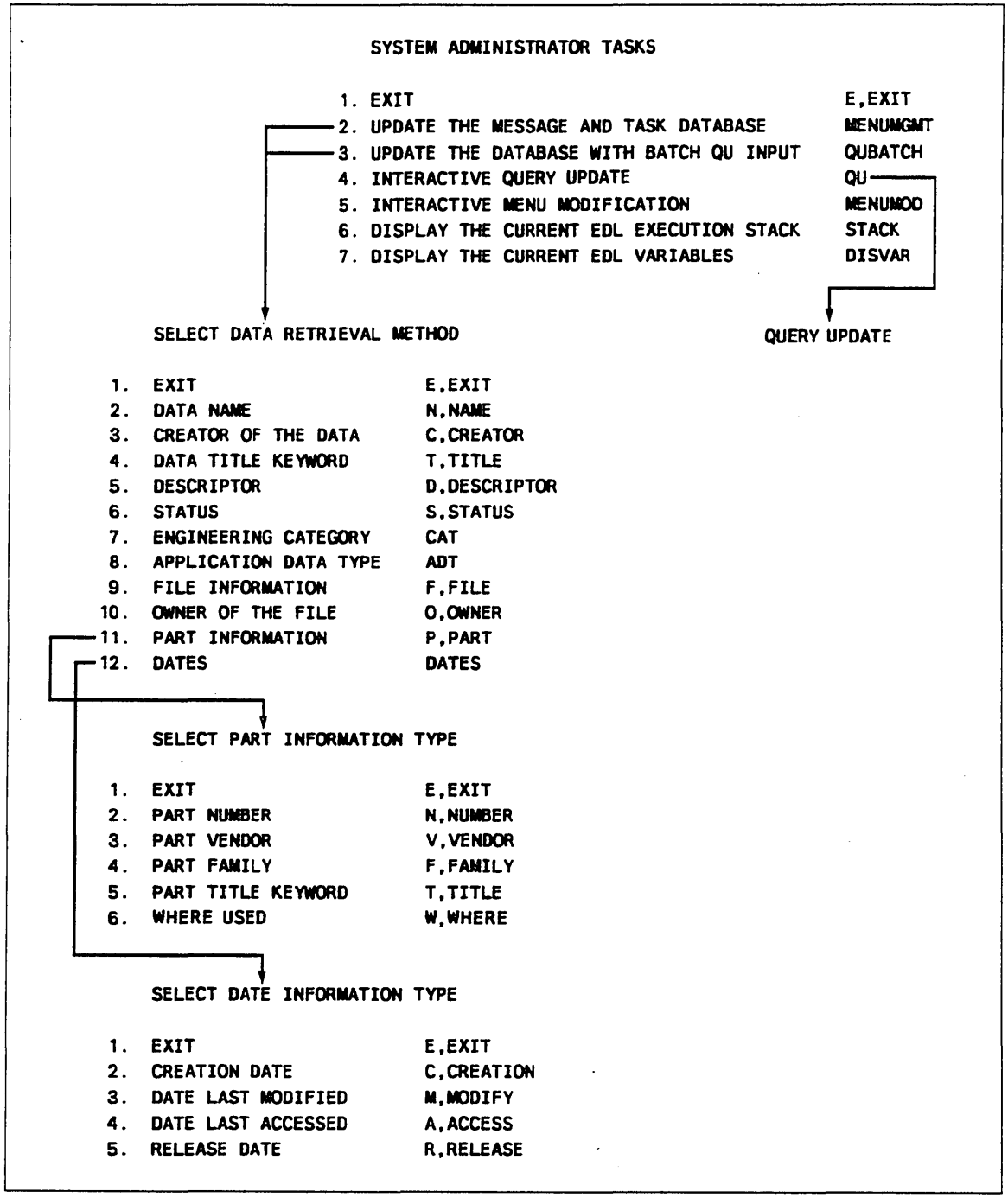


Figure 1-3. System Administrator Tasks and Secondary Menus

Concurrent Database Operations

Concurrent access allows multiple users to access a database at the same time. To provide optimum system performance, the EDL databases handle concurrent access differently:

- The Engineering Data Database permits concurrent access by all users in both READ and WRITE modes.
- The Message and Task Database permits concurrent access only in READ mode. Any operations that require changes to the MDB – add, change, or delete – cannot be performed while any EDL user is active. Conversely, no normal EDL usage is possible until a change operation is completed.

Figure 1-3 shows the hierarchy of tasks you can select from the System Administrator Tasks menu.

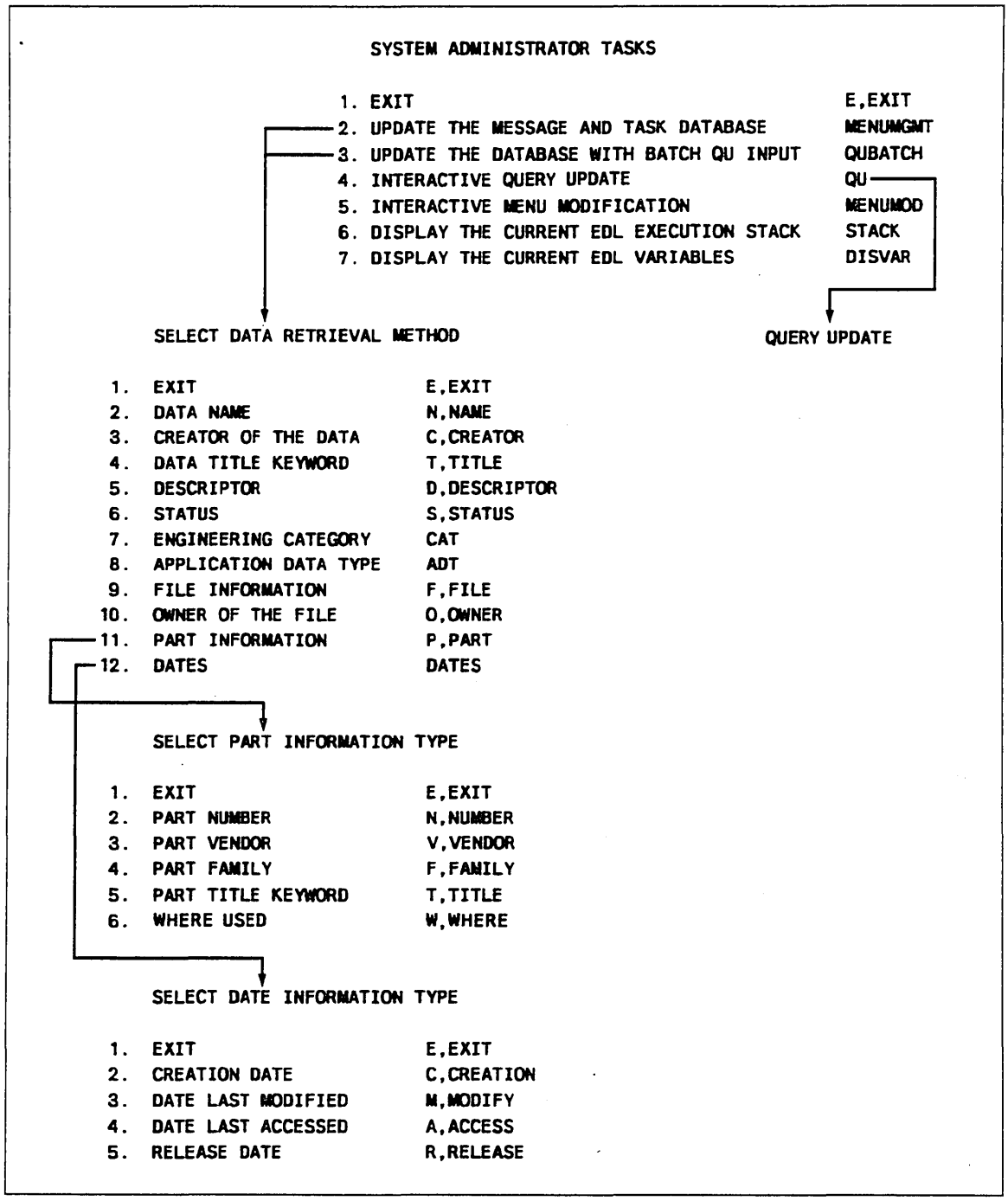


Figure 1-3. System Administrator Tasks and Secondary Menus

Concurrent Database Operations

Concurrent access allows multiple users to access a database at the same time. To provide optimum system performance, the EDL databases handle concurrent access differently:

- The Engineering Data Database permits concurrent access by all users in both READ and WRITE modes.
- The Message and Task Database permits concurrent access only in READ mode. Any operations that require changes to the MDB – add, change, or delete – cannot be performed while any EDL user is active. Conversely, no normal EDL usage is possible until a change operation is completed.

Setting up the EDL Procedure

The procedure EDL in E125PRC is the base procedure for all EDL operations. Its basic purpose is to attach the EDL program file, execute it, execute the procedure calls written on EEEDL2, and loop until the user chooses to quit EDL.

The procedure header has several parameters, which are all passed to the E125ABS execution statement.

```
.PROC,EDL,I=INPUT/INPUT,IT=0/IT,OT=0/OT,HOST=,AUN=.
```

I — Alternate Input File: Specifies an alternate file for input. Default file name: INPUT.

IT — Input Trace File Name: Specifies the trace file that records all input entered by the user. Default file name: IT. If not specified or IT=0, no input trace is created.

OT — Output Trace File Name: Specifies the trace file for all EDL output displayed at the terminal and all input entered by the user. Default file name: OT. If not specified or OT=0, no output trace is created.

HOST — Host Identification Code: Specifies the mainframe where this version of EDL resides. Default is blank.

AUN — Alternate User Name: Specifies the user name for the EDL procedure file E125PRC and the absolute E125ABS. If not specified, EDL assumes these files are located under the user's own account. It is good practice to edit the procedure header to include a default value here so that users do not need to include the AUN parameter on their BEGIN statements.

You can also include the following parameters on the E125ABS statement:

L — Alternate Output File: Specifies an alternate file for output. Default file name: OUTPUT.

DUN — Data Database: Specifies the user name for the Engineering Data Database. If not specified, EDL assumes the value specified by the AUN parameter.

MUN — Menu Database: Specifies the user name for the Message and Task Database. If not specified, EDL uses the value specified by the AUN parameter.

ECHO — If specified, user input is included in the output file. This parameter is generally used for debugging purposes when the output file is renamed.

Creating Alternate Procedures

You can write your own procedures to be invoked by EDL. It is suggested that you put them on a separate procedure file other than E125PRC. To allow the user to invoke them, you need to set up EDL tasks and task processes as explained later.

Adding a Directory to E125PRC

The system can find procedures from a proc file faster if the file has a random access directory. We recommend that you use LIBEDIT to build a directory on E125PRC after you edit it for any reason. The following example illustrates this procedure. Refer to volume 3 of the NOS Version 2 Reference Manual for additional information.

```
ATTACH,E125PRC/M=W.
GTR,E125PRC,TEMP.PROC/*
FSE,TEMP.
  :
(enter full screen editor commands)
(exit the editor)
  :
LIBEDIT,P=TEMP,N=NEW.
*BUILD DIR
REWIND,*.
COPYEI,NEW,E125PRC,V.
```

Upgrading from EDL 1.2.3 to EDL 1.2.5

The following steps outline the procedure you should follow to apply your customization of EDL 1.2.3 to version EDL 1.2.5. Subsequent chapters of this manual provide additional detail about these database customization techiniques.

1. Install the default EDL 1.2.5 database according to the installation instructions provided with your release tapes.

2. Rerun all Message and Task Database transactions using the MENUMGMT task. The structure of the Message and Task Database has not changed. The TITYP field is used to control whether a task is allowed to run on a subordinate host in a network. Set this field to MASTER on any site-defined administrative tasks that can only run on the master machine.

3. Edit all QU directive files as needed. A few records in the Engineering Database have changed, as documented in the database schema definitions listed in appendix A. Rerun the QU directives files using the QUBATCH task.

4. Run the EDL database conversion procedure CONV123.

5. Edit the source programs for any site-defined OVCAPS. You must replace the COMMON block definitions of all DDB records with the new definitions found on the EDLCOM file. Several COMMON blocks have changed format and names.

6. Use the LOADEDL procedure to recompile and load the OVCAPS.

7. Test everything.

Customizing the Plotting Interface

You must modify the plotting interface supplied on the EDL release tape in order for it to work correctly at your site.

The option menu named PLOTN1 shows users which plotters are available at your site. You can use the interactive MENUMOD utility described in chapter 2, or the batch transaction method described in chapter 3 to update this menu. The OVVAL field should contain a site-defined destination code that is eventually passed to procedure PLOTN as the DEST parameter when a user chooses a plot destination.

Procedure PLOTN in E125PRC is designed to convert a neutral picture file (NPFILE) to a plotter-specific representation and route it to the plotter. You must edit this procedure to execute UNIPOST with the correct directives for the specific plotters at your site, and to route the plot file to the correct plotter based on the value of the DEST parameter. Refer to the UNIPLOT manual for details about the appropriate directives for your plotters.

Customizing the Message and Task Database 2

Customizing the Message and Task Database 2

The overall control and user dialog in EDL is defined by the Message and Task Database, also called the Menu Database (MDB). The records in this database contain the definitions of all messages, prompts, menus, and tasks used by EDL.[1]

Record Name	Description
MH	Message help records specify the help text for a message.
MI	Message information records are the header record for all prompts, menus, and error messages.
OK	Option keyword records specify keywords that may be used to select an option menu line.
OM	Option menu records contain the text displayed on option menu lines.
OV	Option value records specify the value returned to the program when a user selects an option from an option menu.
TC	Task command records specify the commands used to invoke tasks.
TI	Task information records serve as headers for EDL tasks.
TM	Task menu records contain the text displayed on task menus.
TP	Task process records specify the processes executed sequentially when a task is invoked.
TV	Task parameter value records specify the parameters passed to CCL procedures and overlay capsules (OVCAPS) when they are executed as EDL task processes. There are six types of parameters, differentiated by the TVTYP field of the TV record:

CONSTANT	Passes a constant to the process.
PROMPT	Prompts the user for the value of the parameter to be passed to the process.
NULL	Passes the process a null value. (Simulates a carriage return.)
VARIABLE	Passes the value of an EDL global variable to the process. An EDL global variable must be previously set by a subprogram that uses the PUTVAR subroutine.

1. Refer to record schemata in appendix A for a complete description of EDL record types.

CONFIG	Passes a parameter value based on the contents of the application configuration (AC) records, application information (AI) status (must be ACTIVE), and the user's current terminal configuration.
TRANSFER	Passes the set of all variables required for a transfer.

There are two ways to modify the Message and Task Database. Which method you choose depends somewhat on the complexity of the changes you want to implement.

- You can use the interactive MENUMOD utility described in this chapter for simple modification of menus, prompts, and messages.
- More complex customization may require the preparation of a batch transaction data file, OVCAP, and procedure files; this method is described in chapter 3.

Using Query Update, you can invoke several report tasks to produce listings of the contents of the MDB.

Using MENUMOD

MENUMOD is an interactive utility for modifying the EDL Message and Task Database. You invoke it by selecting 5. INTERACTIVE MENU MODIFICATION from the System Administrator Tasks menu or by entering the MENUMOD command from any task menu. This produces the Interactive Menu Modification menu shown below.

```
      INTERACTIVE MENU MODIFICATION
  1. EXIT                          E,EXIT
  2. MODIFY OR ADD A TASK          TASKMOD
  3. MODIFY OR ADD A TASK MENU     TMENUMOD
  4. MODIFY OR ADD AN OPTION MENU  OMENUMOD
  5. MODIFY A MESSAGE              MESSAGEMOD
  6. BATCH MENU MODIFICATION       MASSMOD
ENTER TASK
?_____________
```

The tasks selectable from this menu let you add, change, or delete tasks, task menus, option menus, error messages, prompts, and informative messages. A local file called MASSMOD records the final image of any changes you make using MENUMOD.

CAUTION

In order to use MENUMOD, your database must be opened exclusively. To avoid damaging your current system, you should first make all changes to a *copy* of the working database.

The following steps outline the general procedure for using MENUMOD.

1. Select one of the MENUMOD tasks.
2. Enter the name of the structure you want to change. In response, the system writes the current information of the structure to a file and invokes your editor to display this file for modification.

3. Use your cursor keys and text editor commands to make your changes to the file. Modifications made using MENUMOD do not require a strict fixed-column format; however, you must separate fields by at least two spaces. You can create new tasks, menus, or messages by renaming an existing structure and modifying it accordingly.

4. If you elect to save your modified information, MENUMOD changes the EDL Message and Task Database, replacing the previous information. The system automatically appends images of your changes to the file MASSMOD.

The remaining sections in this chapter provide additional information about each of the MENUMOD tasks and examples of their use. The descriptions of the MENUMOD screen lines reference corresponding record fields (TITNA, TIDSC, and so on). Refer to the database schemata in appendix A for a complete description the EDL record formats.

Modifying a Task

When you select 2. MODIFY OR ADD A TASK from the Interactive Menu Modification menu or enter the TASKMOD command, MENUMOD extracts a task in the following format:

```
taskname
DESCRIPTION:  task description
SECURITY CATEGORY:  category
TYPE:  MASTER or blank
COMMANDS:  command1   command2
PROCESS:  process type   process name
PARAMETER:  parameter name   parameter type   parameter value
```

Line	Description
1	10-character task name (TITNA).
2	70-character task description (TIDSC). The labels on lines 2 through 5 are only for ease in task modification. If the line contains a colon (:), the system ignores everything up to that colon and interprets the remaining text (excluding leading blanks) as your input.
3	10-character task security code (TISEC). If TISEC is blank, this task is available to all users. If this field is not blank, users can only execute this task if they are members of a group with the specified security code.
4	10-character task type (TITYP). This field reads MASTER if EDL is running in a network and if the task is one that should only be run on the master host; otherwise, the field is blank. Even if blank, MENUMOD reserves line 4 of the task image for the task type.
5	10-character task commands (TCCMD) used to invoke the procedure. Each command must be separated by at least two spaces.
6+	Subsequent lines list task processes (TP records) such as OVCAPS, CCL PROCS, task menus or tasks, or task parameter values (TV records). These two records are differentiated by the first two letters of the line (excluding leading blanks). Task processes require a 10-character process type (TPTYP) and 10-character process name (TPNAM); optionally, they can also include a 10-character file name (TPFNA) and 10-character user name (TPFUN).

For example, if you chose to modify the task for retrieving ICEM DDN data (RETDDN), MENUMOD would extract and display the image in figure 2-1.

```
RETDDN
DESCRIPTION:  RETRIEVE ICEMDDN DATA
SECURITY CATEGORY:
TYPE (MASTER OR BLANK)
COMMANDS:   RETDDN
PROCESS:  OVCAP       XRETREV
PARAMETER:  ADT         CONSTANT    DRAWING
PARAMETER:  ADT         CONSTANT    GLOBAL DRAWING
PARAMETER:  SELECT      CONSTANT    LOCAL
PARAMETER:  INTENT      CONSTANT    W
PROCESS:  OVCAP       XGETAPN
```

Figure 2-1. Using MENUMOD to Extract a Task

CAUTION

If your modification of a standard EDL task significantly changes its function, you might encounter compatibilty problems on subsequent releases of EDL or ICEM applications. Instead, we recommend that you create new tasks to satisfy your requirements; refer to "Adding a New Task" in the examples at this end of this chapter.

Modifying a Task Menu

When you select 3. MODIFY OR ADD A TASK MENU from the Interactive Menu Modification menu or enter the TMENUMOD command, MENUMOD extracts a task menu in the following format:

```
task menu name
task menu header
task menu text   task name
```

Because the format of a task menu is simpler than that of a task, there are no labels.

Line	Description
1	10-character task menu name (TMMNA).
2	70-character task menu header (MITTL).
3+	Each subsequent line contains two fields: • 40-character menu text (TMTXT). • 10-character task name (TMTNA) separated from the menu text by at least two spaces.

NOTE

Be sure to use the actual task name here, not the name of the command that calls the task.

For example, if you chose to modify the USER task menu, MENUMOD would extract and display the image shown in figure 2-2.

```
USER
          USER TASKS
          EXIT                                EXIT-TASK
          ICEM APPLICATIONS                   ICEM
          RETRIEVE ENGINEERING DATA           RETRIEVE
          TRANSFER ENGINEERING DATA           TRANSFER
          RELEASE ENGINEERING DATA            RELEASE
          FILE MANAGEMENT                     FIMGMT
          UPDATE EDL FOR ENGINEERING DATA     UPDATE
          USER PROFILE                        USERINFO
          REPORTS                             REPORTS
          JOB QUEUE CONTROL                   QUEUE
          PART STRUCTURE MANAGEMENT           STRUCTURE
```

Figure 2-2. Using MENUMOD to Extract a Task Menu

Modifying an Option Menu

When you select 4. MODIFY OR ADD AN OPTION MENU from the Interactive Menu Modification menu or enter the OMENUMOD command, MENUMOD extracts an option menu in the following format:

```
option menu name
option menu header
option menu text   option keyword   option value
```

The option menu extraction format is much like the task menu format.

Line	Description
1	10-character option menu name (OMMNA).
2	70-character option menu header (MITTL).
3+	Each subsequent line contains menu detail consisting of three different fields: • 40-character option menu text (OMTXT). • 10-character option keyword(s) (OKKEY) separated from the menu text by at least two spaces. If there are multiple keywords, separate them by commas, just as they appear on the menu display. • 40-character option value (OVVAL) separated from the option keyword(s) by at least two spaces.

For example, if you chose to modify the option menu for retrieval methods (EXTRAC), MENUMOD would extract and display the image shown in figure 2-3.

```
EXTRAC
          SELECT DATA RETRIEVAL METHOD
 EXIT                                  E,EXIT     EXIT
 DATA NAME                             N,NAME     NAME
 CREATOR OF THE DATA                   C,CREATOR     CRE
 DATA TITLE KEYWORD                    T,TITLE     TITLE
 DESCRIPTOR                            D,DESCRIPTOR     DESC
 STATUS                                S,STATUS     STA
 ENGINEERING CATEGORY                  CAT     EDT
 APPLICATION DATA TYPE                 ADT     ADT
 FILE INFORMATION                      F,FILE     FILE
 OWNER OF THE FILE                     O,OWNER     OWNER
 PART INFORMATION                      P,PART     PART
 DATES                                 DATES     DATE
```

Figure 2-3. Using MENUMOD to Extract an Option Menu

Modifying a Message

When you select 5. MODIFY A MESSAGE from the Interactive Menu Modification menu or enter the MESSAGEMOD command, MENUMOD extracts a message in the following format:

```
message name   message type
message text
message help
```

Line	Description
1	10-character message name (MIMNA) followed by the 10-character message type (MITYP). The message type can be PROMPT, ERROR, or MESSAGE.
2	70-character message text (MITTL).
3+	Subsequent 70-character lines of text (MHTXT) specify HELP text for the message.

For example, if you chose to modify the EXTNAM1 prompt, MENUMOD would extract and display the image shown in figure 2-4.

```
EXTNAM1    PROMPT
ENTER THE DATA NAME OR CR TO RETURN
EDL WILL GENERATE A LIST OF ALL DATA NAMES WHICH BEGIN WITH THE
CHARACTERS YOU ENTER.  FOR EXAMPLE, IF YOU ENTER "ENGINE", EDL WILL
INCLUDE DATA NAMES "ENGINE-HOUSING" AND "ENGINEER".
A CARRIAGE RETURN WILL RETURN TO THE RETRIEVAL METHODS MENU.
```

Figure 2-4. Using MENUMOD to Extract a Message

Using MASSMOD for Batch Modifications

MENUMOD uses a local file named MASSMOD to record the images of your interactive Menu Database modifications. A one-line header ****type identifies the image as a TASK, OMENU, TMENU, or MESSAGE modification. You can save MASSMOD in EDL under the application data type MDB Images. This allows you to re-apply your customizations of the Menu Database when you upgrade to a new version of EDL, or allows you to apply the same changes to other copies of the Menu Database running on your system.

The MASSMOD command functions in the same manner as the MENUMGMT command. When you select 6. BATCH MENU MODIFICATION from the Interactive Menu Modification menu or enter the MASSMOD command, you are asked to select the file of images from the standard retrieval list. As MASSMOD executes, it displays the header lines on your terminal.

The following steps describe the procedures involved in batch modification using MASSMOD.

1. EDL must know about the image file. Use the ADDINFO command to update the EDL database using the application data type: EDL MDB IMAGES.
2. Enter the MASSMOD command.
3. Select the image data from the standard retrieval list. EDL displays the ****type header for each image processed. When complete, EDL returns to the previous task menu.

Keeping Track of Your Changes

After modifying the Message and Task Database you might want to generate a new EDLLIST file to reflect your changes. The following example shows how to add the EDLLIST generator to your EDL system.

1. The source for the EDLLIST generator is supplied to you in the file MOUT. Compile this file using the following commands:

```
GET,MOUT.
FTN5,I=MOUT,B=MOUTB,L=MOUTL.
```

   The compiled routine is put on the file MOUTB.

2. Create and save the following OVCAP:

```
OVCAP.
SUBROUTINE XMOUT
CALL MOUT
RETURN
END
```

   For this example, the file is saved on OVMOUT.

3. Put the MOUT routine (on file MOUTB) into a library using the following command:

```
LIBGEN,F=MOUTB,P=LIBR
```

4. Enter the following statement to create a new EDL program:

```
BEGIN,LOADEDL,E125PRC,OVMOUT,LIBR
```

   This generates the new file E125ABS.

5. Create a task to generate the EDLLIST file using TASKMOD. This example uses the existing task called STACK.

   The original task looks like this:

```
STACK
DESCRIPTION:  DISPLAY THE EDL EXECUTION STACK
SECURITY CATEGORY:
TYPE (MASTER OR BLANK):
COMMANDS:   STACK
PROCESS:  OVCAP       XDISSTK
```

   Use your editor to modify the task as follows:

```
EDLLIST
DESCRIPTION:  LIST THE EDL MENU DATABASE
SECURITY CATEGORY: SYSADMIN
TYPE (MASTER OR BLANK):
COMMANDS:   EDLLIST
PROCESS:  OVCAP       XMOUT
```

6. When you execute the EDLLIST task it generates a local file called EDLLIS. The file is called EDLLIS to avoid confusion with the EDLLIST file supplied with the EDL release.

Examples

The following examples illustrate the use of MENUMOD tasks in database customization.

Changing a Prompt

You might want to change the wording of an EDL prompt to satisfy site-specific needs. For example, if your site uses the term Engineering Change Notice (or ECN) rather than Engineering Change Order (ECO) as used in EDL Part Structure, you would take the following steps to change it:

1. Select 5. MODIFY A MESSAGE from the Interactive Menu Modification menu, or enter the MESSAGEMOD command from any task menu.

2. When prompted for the name of the message to modify, enter the message name PSADD3. In response, the system invokes your editor and displays the following file:

```
PSADD4    PROMPT
ENTER THE ECO FOR THE REVISION OR CR TO RETURN
ENTER THE ENGINEERING CHANGE ORDER IDENTIFIER WHICH CAUSES THIS
PART REVISION TO BE CREATED.  THIS FIELD IS REQUIRED.
```

3. Use your editor to change ECO to ECN, and the phrase ENGINEERING CHANGE ORDER IDENTIFIER to ENGINEERING CHANGE NUMBER. The resulting file looks like this:

```
PSADD4   PROMPT
ENTER THE ECN FOR THE REVISION OR CR TO RETURN
ENTER THE ENGINEERING CHANGE NUMBER WHICH CAUSES THIS
PART REVISION TO BE CREATED.  THIS FIELD IS REQUIRED.
```

4. After you exit your editor, the system asks:

```
DO YOU WISH TO CONTINUE WITH THIS MESSAGE MODIFICATION?
```

   Your changes are added to the Menu Database only if you enter YES.

Removing a Prompt from a Task

If you are not running EDL in a network, you might not want to be prompted for a host name every time you update EDL with new data. The following example shows you how to remove that prompt.

1. Select 2. MODIFY OR ADD A TASK from the Interactive Menu Modification menu, or enter the TASKMOD command from any task menu.

2. When prompted for the name of the task to modify, enter the task name ADDINFO. In response, the system invokes your editor and displays the following file:

```
ADDINFO
DESCRIPTION:  ADD EDL INFORMATION FOR ENGINEERING DATA
SECURITY CATEGORY:
TYPE (MASTER OR BLANK):
COMMANDS:   ADDINFO
PROCESS:  OVCAP       XUPDADD
```

3. Use your editor to add a task parameter with the same name as the prompt you want to remove, and a parameter type of NULL. The resulting file looks like this:

```
ADDINFO
DESCRIPTION:  ADD EDL INFORMATION FOR ENGINEERING DATA
SECURITY CATEGORY:
TYPE (MASTER OR BLANK):
COMMANDS:   ADDINFO
PROCESS:  OVCAP       XUPDADD
PARAMETER:  UPADD3      NULL
```

4. After you exit your editor, the system displays the following prompt:

```
DO YOU WISH TO CONTINUE WITH THIS TASK MODIFICATION?
```

   Your changes are added to the Menu Database only if you enter YES.

Adding a New Task

The TASKMOD task is used to modify or add a task. By modifying TASKMOD, you can create a task that allows a user to look at the structure of a task, but not to change it. The following steps describe how to implement this change.

1. Select 2. MODIFY OR ADD A TASK from the Interactive Menu Modification menu, or enter the TASKMOD command from any task menu.

2. When prompted for the name of the task to modify, enter the task name TASKMOD. In response, the system invokes your editor and displays the following file:

```
TASKMOD
DESCRIPTION:  MODIFY OR ADD AN EDL TASK
SECURITY CATEGORY:  SYSADMIN
TYPE (MASTER OR BLANK)
COMMANDS:   TASKMOD
PROCESS:  OVCAP       XDISTSK
PROCESS:  CCL PROC    EDIT
PARAMETER:  EDITOR      VARIABLE    EDITOR
PARAMETER:  LFN         VARIABLE    EDITF
PROCESS:  OVCAP       XADDTSK
```

3. Use your editor to make the following changes:

   a. Change the name and command from TASKMOD to TASKLOOK. By changing the task name, you create a new task; the old task (TASKMOD) remains unchanged.

b. Remove the last OVCAP (ADDTSK), which is the one that actually performs the update of the Menu Database.

The resulting file looks like this:

```
TASKLOOK
DESCRIPTION:  LOOK AT AN EDL TASK
SECURITY CATEGORY:  SYSADMIN
TYPE (MASTER OR BLANK)
COMMANDS:   TASKLOOK
PROCESS:  OVCAP       XDISTSK
PROCESS:  CCL PROC    EDIT
PARAMETER:  EDITOR      VARIABLE    EDITOR
PARAMETER:  LFN         VARIABLE    EDITF
```

4. After you exit your editor, EDL asks you:

```
DO YOU WISH TO CONTINUE WITH THIS TASK MODIFICATION?
```

Your changes are added to the Menu Database only if you enter YES.

Adding a Task with an OVCAP

A slightly more complex task might involve adding a user defined OVCAP.[2] For example, to add a task that would list all EDL IDs and their names sorted by last name, you would take the following steps.

NOTE

Before beginning this example, make sure that you first back up the current E125ABS.

1. Write the OVCAP and include the UI COMMON block from file EDLCOM, as shown in the following example.

```
      OVCAP.
      SUBROUTINE LISTID
C
C     THIS ROUTINE LISTS ALL EDL USER ID'S SORTED BY THE USER'S
C     LAST NAME
C
      COMMON / UI      / UIUSR  ,UIPWD  ,UISTA  ,UIFIN  ,UIMIN
     *,UILNA  ,UISTR  ,UICTY  ,UIPHO  ,UITTL  ,UIDPT  ,UICMD
     *,UIDELD ,UIDELS ,UIEDT
      CHARACTER UIUSR  *10,UIPWD  *10,UISTA  *10,UIFIN  *10
     *,UIMIN  *10,UILNA  *10,UISTR  *70,UICTY  *70,UIPHO  *20
     *,UITTL  *40,UIDPT  *20,UICMD  *10,UIDELD *1,UIDELS *1,UIEDT  *10
      LOGICAL OK
      CALL IBFUI1(OK)
100     IF(OK)THEN
           PRINT*,UIUSR,UILNA,UIFIN,UIMIN
           CALL IBNUI1(OK)
```

2. OVCAPS are further explained in chapter 3 of this manual.

```
        GO TO 100
      ENDIF
    RETURN
    END
```

2. Run the procedure LOADEDL to insert your routine into the EDL program. For example, if you placed the OVCAP on file LISTID, you would type:

   ```
   BEGIN,LOADEDL,E125PRC,LISTID
   ```

   This procedure is described in greater detail in the example "Adding a Task to EDL" in chapter 3.

3. Select 2. MODIFY OR ADD A TASK from the Interactive Menu Modification menu, or enter the TASKMOD command from any task menu.

4. When prompted for the name of the task to modify, enter an existing task name; for example, RETDDN. In response, the system invokes your editor and displays the following file:

   ```
   RETDDN
   DESCRIPTION:  RETRIEVE ICEMDDN DATA
   SECURITY CATEGORY:
   TYPE (MASTER OR BLANK)
   COMMANDS:   RETDDN
   PROCESS:  OVCAP       XRETREV
   PARAMETER:  ADT         CONSTANT    DRAWING
   PARAMETER:  ADT         CONSTANT    GLOBAL DRAWING
   PARAMETER:  SELECT      CONSTANT    LOCAL
   PARAMETER:  INTENT      CONSTANT    W
   PROCESS:  OVCAP       XGETAPN
   ```

5. Use your editor to change the task name and remove all but the first OVCAP, which you rename LISTID. You would probably also want to set the security category to at least ADMIN. The resulting file looks like this:

   ```
   LISTID
   DESCRIPTION:  LIST EDL ID'S
   SECURITY CATEGORY:  ADMIN
   TYPE (MASTER OR BLANK)
   COMMANDS:   LISTID
   PROCESS:  OVCAP       LISTID
   ```

6. After you exit your editor, EDL asks you:

   ```
   DO YOU WISH TO CONTINUE WITH THIS MESSAGE MODIFICATION?
   ```

   Your changes are added to the Menu Database only if you enter YES.

Manipulating the Engineering Data Database 3

Manipulating the Engineering Data Database 3

This chapter discusses customizations to EDL that involve the Engineering Data Database (DDB). This database contains all the information about EDL users, applications, files, and data. In addition to the changes made to the Engineering Data Database in the course of normal EDL operation, you can use Query Update to update certain DDB records. You can also write customized EDL code to read and manipulate the data contained in these records.

Customizing EDL Using Engineering Data Records

This section describes the DDB record types that you can update with Query Update to customize EDL. Other record types in the Engineering Data Database are updated by the system when users run EDL; refer to "Records Updated by EDL" later in this chapter for a description of these records. Refer to appendix A for a complete description of EDL record types.

Record Name	Description
AC	Application configuration records define parameters to be passed to an application depending on the terminal's configuration.
AI	Application information records define the application systems under EDL and indicates which version is active.
AT	Application data type records describe the types of data managed by EDL.
CL	Communication link records specify RHF connections.
EA	Engineering attribute records specify the standard attributes your users will be prompted for when updating EDL information for engineering data.
ET	Engineering category records define the engineering categories that you establish at your site. Engineering categories provide a way to describe and separate your engineering data based on how it is used.
FT	File type records describe the types of files managed by EDL.
HI	Host information records specify the host family codes of computer systems that contain data you want to record in EDL.
RT	Release transfer records specify allowable transfers during the engineering data release process.
TT	Transfer and translation task records specify allowable data transfers and the tasks used to perform the transfer.
UM	Units of measure records define valid codes or abbreviations for the units of measure field in PS records.

Ordering Your Changes

The definition of the database imposes constraints on the records and requires that you make your changes in a definite order. Appendix A contains the database schemata for EDL; these schema diagrams specify the record interdependencies that control the order of changes. You must be aware of these database constraints and record interdependencies before making changes that involve the Engineering Data Database. The order for adding entries is as follows:

AI
AC
ET
EA
FT
AT
TT
RT
HI
CL
UM

Reverse the order for this list when deleting entries.

Setting Engineering Categories and Standard Attributes

Before data can be added to EDL, it must have an engineering category code. This code is stored in the ETEDT field of the engineering category (ET) record. These 20-character engineering category codes are completely site-defined. Their purpose is to enhance the description and retrieval of engineering data. EDL and its application programs function the same regardless of the category assigned to the data.

You can also set up data descriptors to aid in describing and retrieving the data. A descriptor is a pair of character strings associated to the data, a 20-character attribute name and a 40-character attribute value. For example, a user may ask for attribute name = PROJECT CODE and attribute value = T110 to retrieve all permitted data for the T110 project.

You can associate one or more standard attribute names to an engineering category. The system can then prompt your users to enter values for each of these attributes when they update EDL data descriptors. Setting up standard attributes does not restrict users to just those standards. In special cases, users may want to enter a descriptor with a nonstandard attribute name. In general, however, users should be encouraged to use the standard attribute prompting feature and to take some care to use consistent attribute values. Attributes are a means of keeping track of data; your operational policies are the only means of keeping track of attributes.

The following example shows how you can set up engineering categories and standard attributes by using Query Update to update the ET and EA records in the database.

```
STORE SETTING ETEDT
$ $
$EDL SYSTEM$
$PREPRODUCTION$
$PRODUCTION$
$TOOLING$
*END
```

```
STORE SETTING EAEDT EAATR
$PREPRODUCTION$ $PROJECT CODE$
$PRODUCTION$    $PRODUCT LINE$
$TOOLING$       $TOOL TYPE$
*END
```

After these records have been added, the system automatically prompts the user for the corresponding attributes (ATR) of the specified engineering category. For example, if the user specified the engineering category PRODUCTION, EDL prompts for a descriptor value for the attribute name PRODUCT LINE. These trivial categories and attributes are installed as part of standard EDL and should be adapted to fit the environment at your site.

Records Updated by EDL

The DDB records described in this section are updated by the system in the course of normal EDL usage. Although you cannot directly update these records, you can write customized code to read and manipulate the data they contain. You can also use Query Update to invoke several report tasks that produce listings of the contents of these records.

Record Name	Description
DD	Data descriptor records establish attribute/value pairs for EDL data.
DF	Default file records specify files that must automatically attach when a user enters the specified application.
DI	Engineering data information records establish data managed by EDL.
DR	Data required records relate data sets that must be available to complete the current data.
DS	Data source records relate data sets from which the current data was derived.
FD	Family definition records relate family codes and engineering data.
FI	File information records establish files to be managed by EDL.
FM	Family information records associate family codes to part numbers.
FP	File permit records are produced by Group Permit and User Permit records.
GI	Group information records define groups of EDL users.
GM	Group member records establish members for previously defined EDL groups.
GP	Group permit records establish file permits for groups.

Record Name	Description
GS	Group security authorization records specify task category authorization for group members.
HI	Host information records.
ME	Message records define messages sent by a user.
ML	Message line records define user message text.
MN	Message instance records define users receiving messages and indicate whether the message has been read.
PD	Parts data records relate engineering data and part revisions.
PF	Parts family records relate part numbers to family codes.
PI	Parts information records establish EDL part numbers.
PP	Pending permit records contain file access information and issue operating system permits when the owner of the file logs into EDL.
PR	Part revision records define revision levels for each part.
PS	Part structure records define the structure of part assemblies.
PV	Parts vendor records relate part numbers and vendor codes.
RA	Release authorization records establish release information for engineering data sets.
RP	Release procedure records establish the site-defined release procedures for engineering data.
RR	Review responsibility records define reviewers and review order for each release procedure.
RS	Release signature records establish the stamp that each reviewer puts on data after completing review.
RU	Releaser records define releasers for each release procedure.
UC	User configuration records define valid terminal configuration attribute states.
UI	User information records establish EDL users and all relevant information about them.
UP	User permit records establish individual user file permits.
UV	User validation records track which users are validated to use EDL on each host.
VI	Vendor information records associate vendor names and vendor codes to EDL part numbers.

EDL Global Variables

A global variable is a mechanism that EDL uses to pass values between subprograms. Each variable is identified with a 10-character name and can contain an 80-character value. You set the value of a variable using the PUTVAR subroutine. The value of the global variable stays in effect until the variable is redefined or EDL ends. Refer to "Utility Routines" later in this chapter for additional information about the PUTVAR subroutine.

The following table lists the global variables that EDL defines at startup.

Variable	Description
USR	EDL ID of the running user
HOST	Host code of the computer system on which the user is running
AUN	User name of the EDL absolute (E125ABS)
MDB	File name of the Message and Task Database
DDB	File name of the Engineering Data Database
MUN	User name of the Message and Task Database
DUN	User name of the Engineering Data Database

Other global variables are set by data retrieval and transfer subprograms; these are listed in the following table.

Variable	Description
EDN	Internal engineering data number (EDN) of the last data selected to be retrieved or transferred
PFN	Path file name of the last data selected
UN	Operating system user name
NAME1	70-character data name
SHEET	Secondary ID
I	Name of a file containing the data name and secondary ID
PFN2	Path file name of the destination file
UN2	Operating system user name of the creator of the destination file
RENAME	*Y* if the user wants to give the transferred data a different name, otherwise *N*
NAME12	70 characters of the new data name
SHEET2	New secondary ID
J	Name of a file containing the new data name and secondary ID

Other parts of the EDL system use other global variables to communicate data between subprograms and procedures.

Using FORTRAN Interface Modules

This section describes the interface modules you can call from your own OVCAPS. OVCAPS are FORTRAN programs called by the task protocol of an EDL task to perform the following functions:

- Make calls to display messages, prompts, or option menus
- Store data in the Engineering Database
- Manipulate existing data in the Engineering Database

Creating an OVCAP

You must create an OVCAP that specifies the name of the routine you want to add. For example, if you were adding a routine call MYROUT, the relevant OVCAP would appear as follows:

```
OVCAP.
SUBROUTINE XMYROUT
CALL MYROUT
RETURN
END
```

This procedure is illustrated in the example "Adding a Task to EDL" at the end of this chapter.

Using Information Base (IB) Routines

Information Base routines allow EDL overlay capsules to perform database accesses and updates at the record level. All IB routines communicate with the EDL databases via COMMON blocks. For example, you need to take the following steps before storing a file information (FI) record:

1. Copy the FI COMMON block into your routine from the file EDLCOM.
2. Set all values (FIFIL, FIPFN, FIFUN, FIUSR, FISTA, and FIVSN).
3. Call the appropriate IB routine for storing the FI record: IBSFI.

IB routines can be grouped according to their function as follows:

IBS*xx* Store record *xx*

IBM*xx* Modify record *xx*

IBD*xx* Delete record *xx*

IBO*xxn* Obtain record *xx* via access path *xxn*

IBA*xxn* (Approximate) Obtain record *xx* or next higher via access path *xxn*

IBE*xxn* (Equivalent) Obtain the next duplicate record *xx* via access path *xx*

IBF*xxn* Obtain the first *xx* record ordered by access path *xxn*

IBN*xxn* Obtain the next *xx* record ordered by access path *xxn*

IBF*xxyy* Obtain the first member within coset *xxyy*

IBN*xxyy* Obtain the next member within coset *xxyy*

Replace the *xx* value in your IB calls with the 2-character name of the database table you want to refer to. Replace the *n* and *yy* values with the names of the access path or coset. Appendix B contains detailed definition of the EDL IB routines.

Additional IB routines with specific functions include the following:

IBCCMT Commit a concurrency parcel

IBCDRP Drop a concurrency parcel

All IB routines return a logical argument STATOK that is TRUE if the operation succeeded, FALSE if it failed.

Declaring Variables

If you call an IB routine, you must declare the variables used to contain the data fields managed by that module. The file EDLCOM contains a set of FORTRAN COMMON decks that simplify variable declaration.

- There is a file for each type of data record. Each file is named with the 2-character name of the record. You must include the appropriate file for every record you reference.
- There is a variable for every data field of the record. The variables are allocated to a labeled COMMON with the same name as the 2-character record name. Any of your own code that accesses the database variables, whether it calls an IB module or not, must include the appropriate COMMON deck.

When you retrieve data, you only need to set those variables that make up the retrieval key. For example, if you retrieve a user information (UI) record, you need only set the UIUSR field. After you call IBOUI0, all of the other fields in the UI common block will be filled.

The following sections of this chapter describe the use of IB routines in obtaining, storing, modifying, and deleting records.

Obtaining Records

You cannot unconditionally add a record to the EDL database. You must first attempt to obtain a record with the desired key fields, then call either an IBS routine if the record doesn't exist, or an IBM routine if the record does exist. For example, if you want to associate DIEDN number 27 to PROJECT 1031518 via the data description (DD) record, you would take the following steps:

1. Set DDEDN to 27 and DDATR to PROJECT.

2. Call IBODD3 to determine if another PROJECT is already assigned to DDEDN number 27.

   a. If a PROJECT already exists for record 27, the logical status code (STATOK) will be TRUE.

      1) Set DDVAL to 1031518.

      2) Call IBMDD to modify the existing data description.

   b. If DDEDN number 27 does not already have a PROJECT, STATOK is returned as FALSE.

      1) Set DDVAL to 1031518.

      2) Call IBSDD to store the new data description.

Five types of IB routines let you obtain records from the database: IBO, IBA, IBE, IBF, and IBN. Each of these obtain routines accesses records differently. The following paragraphs describe these obtain routines in detail.

Using IBO Routines

The record retrieved by an IBO routine is the first record in the database that meets the retrieval criteria. If you continue to call an IBO using the same key, you will continue to get the same record. Instead, you should use an IBE routine to move through the database. Access path IBOxx0 is always the primary key for the record. An IBO*xx*0 call retrieves only that record in the database that meets the criteria set by the *xx*0 path; therefore, an IBE*xx*0 will always return a STATOK value of FALSE.

Other access paths may retrieve more than one dataset, depending on whether those fields have been constrained to be unique. For example, you could associate the data in the preceding example to more than one project, and retrieve each by calling IBODD3 followed by IBEDD3 until STATOK is returned as FALSE.

An example of a secondary, but unique, key is made up by the DIFIL, DINAM, and DISID fields. The schema charts in appendix A indicate unique fields by the solid arrow lines above them. These arrows (called "uniqueness constraints") mean that the fields taken together form a unique key for a record, as shown in figure 3-1.

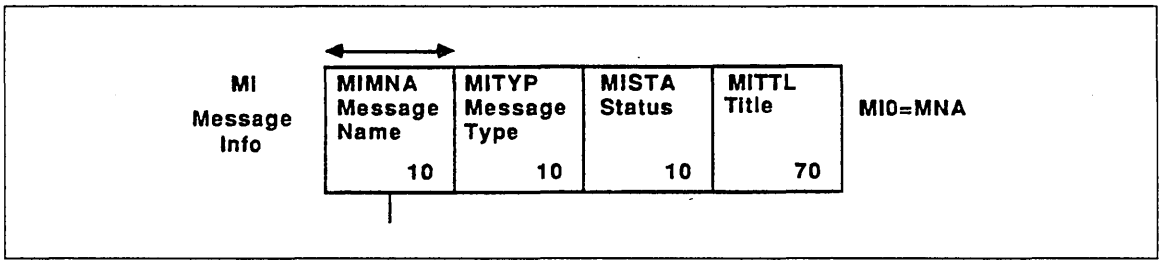


Figure 3-1. Unique Keys are Identified by the Solid Arrow Line

Using Access Paths

Each access path retrieves records via a different key. For example, if you set DIUSR equal to JONES and called IBODI6, you would obtain the first data record in the database where the DIUSR field is equal to JONES. To obtain further records related to JONES, you would use the equivalent call: IBEDI6.

Refer to schema charts in appendix A for record access paths. In order to use an access path, you must set each field used in the access path to the desired value. If you do not set the value of a field, you will most likely be using an old value, and you will not get the results you expect. On the schema chart, an access path containing fields in parentheses indicates that these fields need not be preset. The fields in parentheses are there to indicate the order in which duplicate records along the access path are sorted; the system ignores any preset values for these fields.

Obtaining Approximate Records (IBA)

IBA routines obtain the specified record or the next higher record via access path *xxn*. You should use IBA routines to initialize database retrievals at a given position in the database.

For example, you could call IBADI1 to retrieve data by data name.

1. Enter a partial data name.
2. Set DINAM to that value.
3. Call IBADI1.

The IBA routine merely finds the record that begins with, or is higher alphabetically than, the DINAM field. Your program must go on to determine whether or not the returned record meets the entered criteria. The program should next call an IBN routine to get the next record in the database, and again check against the entered criteria. In this example, you could not use an IBE routine because IBE, like IBO, looks for exact matches against the key values.

A loop to print the names of all pieces of data in the database that begin with the characters PA would look like this:

```
C
C     STATOK IS A LOGICAL VARIABLE
C
      CALL IBADI1(STATOK)
100   IF(STATOK)THEN
         IF(DINAM(1:2).EQ.'PA')THEN
            PRINT*,DINAM
            CALL IBNDI1(STATOK)
            GO TO 100
         ENDIF
      ENDIF
```

Obtaining Equivalent Records (IBE)

IBE routines obtain the next duplicate record via access path *xxn*. You call IBE routines to continue retrieving records along an access path. An IBE call differs from an IBN call in that IBE stops returning records when they no longer match the desired key. IBN routines continue to return records along an access path until there are no more records in the database.

For example, you could start a loop with an IBO call and continue with calls to IBE routines to obtain a group of records with like keys.

```
C
C     USR IS THE EDLID OF THE USER WHOSE DATA WE ARE LISTING
C     STATOK IS A LOGICAL VARIABLE
C
      DIUSR=USR
      CALL IBODI6(STATOK)
100   IF(STATOK)THEN
         PRINT*,DINAM
         CALL IBEDI6(STATOK)
         GO TO 100
      ENDIF
```

Obtaining the First Record (IBF)

IBF routines obtain the first record ordered by access path *xxn*. You should use these routines only when you want to get the first record in the database, sorted on a particular access path. IBF routines ignore all preset fields.

Obtaining the Next Record (IBN)

IBN routines obtain the next record ordered by access path *xxn*. An IBN call is similar to an IBF in that it ignores any preset fields, and returns the next record in the database, sorted by the specified access path.

You will probably never use an IBN call in conjunction with an IBO. Using an IBF routine in a loop with an IBN call gives you all records in the database. For example, a loop to list the names and EDL IDs of all users validated on the EDL database sorted by EDL ID would look like the following example.

```
C
C     STATOK IS A LOGICAL VARIABLE
C
      CALL IBFUI0(STATOK)
100   IF(STATOK)THEN
         PRINT*,UIUSR,UILNA
         CALL IBNUI0(STATOK)
         GO TO 100
      ENDIF
```

The following loop lists the names and EDL IDs of all users validated on the EDL database sorted by last name. (The only difference here is the use of access path UI1 instead of UI0.)

```
C
C     STATOK IS A LOGICAL VARIABLE
C
      CALL IBFUI1(STATOK)
100   IF(STATOK)THEN
         PRINT*,UIUSR,UILNA
         CALL IBNUI1(STATOK)
         GO TO 100
      ENDIF
```

Using Cosets to Obtain Records (IBF - IBN)

Cosets work in the same manner as IBO*xxn* and IBE*xxn*. The only difference is that cosets work implicitly with the current occurrence of the owner (*xx*) record. In general, you may find it less confusing to use IBO - IBE loops.

IBFxxyy Obtains the first member within coset xxyy.

IBNxxyy Obtains the next member within coset xxyy.

Storing Records (IBS)

To store records, set the variables that match the fields in the record's COMMON block with legal values, and call the IBS routine for the record. If the store was successful, the status code (STATOK) is returned as logical TRUE. An IBS routine returns a FALSE status code if a record with the same key field(s) already exists in the database, or if a necessary setting is not made. A call to ERRIB provides an explanation when STATOK is returned as FALSE.

NOTE

When storing new records, you need to be aware of owner-member relationships. For example, a data information (DI) record cannot be added until its corresponding file information (FI) record has been added. This is for reasons of database integrity; an FI record can exist without data having yet been put on it, but data cannot exist without a file. Refer to the database schemata in appendix A to determine owner-member relationships.

Modifying Records (IBM)

To modify an existing EDL record, call an IBM routine after first setting all fields. An IBM routine will fail if a record with corresponding key fields does not exist. You cannot use an IBM routine if it changes an identifier for a record; use an IBS routine instead.

Deleting Records (IBD)

To delete records, first obtain the record then call the appropriate IBD routine. Only the primary key fields need be set for this call. Before deleting a record you need to be aware of the owner-coset relationships. For example, in order to delete a user information (UI) record, you must first delete all of that user's data information (DI) records. The user's DI records are those in which the DIUSR field is the same as the UIUSR field of the UI record you are deleting. The database schemata in appendix A specify all owner-coset relationships.

Using Standard EDL Routines

This section of chapter 3 describes standard EDL subroutines that you can include in your OVCAPS.

- Error and status message routines
- User input routines

- Miscellaneous utility routines

Appendix C contains a description of the standard EDL subroutines.

Error and Status Messages Routines

The following subroutines allow you to display error and status messages.

SUBROUTINE ERR (MNA)

Displays an error message on the user's terminal. If no error message with the given message name is found in the menu database, the system displays the following message:

```
EDLD001 EDL INTERNAL ERROR CODE sys
```

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name of the error message

SUBROUTINE ERRSTR (MNA,MSG)

Returns a character string containing an external error code and message. If no error message with the given system code is found in the menu database, the system displays the following message:

```
EDLD000 EDL INTERNAL ERROR CODE sys
```

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name of the error
MSG	C*(*)	O	External error message string

CHARACTER(*) FUNCTION EDBE (NERR)*

Returns an EDL internal error code corresponding to the IMF diagnostic for the error that occurred on the last database operation. The value of the function may be used as the error code parameter on a call to the ERR or ERRSTR routine.

Call Parameters:

Argument	Type	I/O	Description
NERR	I	I	Diagnostic number (usually = 1)

SUBROUTINE ERRIB

Prints an error message corresponding to the IMF diagnostic for the error which occurred on the last database operation.

No parameters.

Example:

```
CALL IBSDI ( OK )
IF ( .NOT. OK ) THEN
   CALL ERRIB
ENDIF
```

SUBROUTINE MSG (MNA)

Displays a message on the user's terminal. If MNA is not the message name of a valid message in the database, the system displays an error message.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name of desired message

SUBROUTINE MSGSTR (MNA,MSG)

Returns the message text string. If MNA is not a valid message name, the system returns an error message.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Menu name of desired message
MSG	C*(*)	O	Message text

User Input Routines

You can include the following routines in your OVCAPS to control user input.

SUBROUTINE INP (OUTTXT,ICH,HELPV)

Returns the user's input to the program. INP manages the EDL type-ahead buffer by returning only a single delimited response per call, and by issuing a read request when the input buffer is empty. INP does not prompt the user; use the INTXT, INYN, or ININT routine when you need to prompt the user before returning a response.

Call Parameters:

Argument	Type	I/O	Description
OUTTXT	C*(*)	O	User-entered text returned to the calling program response
ICH	I	O	Number of characters in response
HELPV	L	O	TRUE if the user requested help; otherwise FALSE

SUBROUTINE ININT (MNA,IRESP,OK)

Displays a prompt asking the user to enter an integer and returns the value entered. If the parameter MNA is not the message name of a valid prompt in the menu database, the system displays an error message instead of the prompt, but still requires an integer response. If the user enters anything other than an integer or null carriage return, the system displays an error message and asks the user to re-enter a response.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name for the prompt
IRESP	I	O	Integer response from the user
OK	L	O	TRUE if the user entered a positive integer; FALSE if the user entered a null response

SUBROUTINE INTXT (MNA,TXT,ICH)

Prompts the user and returns a text string to the calling program. MNA may be the name of either a prompt or an option menu. If MNA is a prompt, the user enters a text string. If MNA is an option menu, the system returns the first variable value of the selected menu line to TXT.

ICH indicates the number of characters returned in TXT. If the user enters a null carriage return, TXT is blanked and ICH is set to 0. If the user enters a blank line, ICH is set to 1. If MNA is not a valid prompt or option menu, the system displays an error message but still requires a text response.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name of the desired prompt
TXT	C*(*)	O	User response string
ICH	I	O	Number of characters in the response

SUBROUTINE INYN (MNA,YES)

Prompts the user for a YES or NO response. If MNA is not a valid prompt message, the system displays an error message but still requires a Y or N response. Any response other than a null carriage return, Y, YES, N, or NO causes the system to display an error message and reprompt the user.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Message name of the desired prompt
YES	L	O	TRUE if Y or YES; FALSE if N, NO, no answer, or null return

SUBROUTINE INOPT (MNA,OK)

Displays an option menu and prompts the user for a selection. If the user enters a null response, the system selects the first line of the menu by default. This routine positions the option menu line record to the line selected by the user. After successfully calling INOPT, you can call the OPTVAL routine to retrieve the variable values corresponding to the selected option.

If the option menu has only one set of variables (parameters), you should use INTXT instead of INOPT and OPTVAL to display the menu and return the single variable value.

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Name of the option menu to be displayed
OK	L	O	TRUE if the user selected an option; FALSE if the user entered a null return, or if the menu could not be displayed

SUBROUTINE OPTVAL (POS,VAL,OK)

Returns the value of the option variable associated with the option menu line selected by the user. A successful call to INOPT is needed before you call OPTVAL. If no option variable exists in the indicated position, VAL remains unchanged and OK is set to FALSE.

Call Parameters:

Argument	Type	I/O	Description
POS	I	I	Option variable position
VAL	C*(*)	O	Value of the option variable
OK	L	O	TRUE if a variable value was returned; FALSE if no variable in the position

Utility Routines

You can include the following EDL routines in your OVCAPS to perform a variety of utility functions.

SUBROUTINE POPT

Pops and discards all remaining processes of the current task from the execution stack. This routine inhibits processing of succeeding processes when an error or condition is found that makes subsequent task processing meaningless. (No parameters.) For example, if you design a task process that runs an OVCAP that passes parameters to a procedure, an error condition in the OVCAP should call a POPT so that the procedure is not run.

SUBROUTINE CSCRN

Clears the screen of a nonscrolling terminal or resets the number of lines available for a scrolling terminal. (No parameters.)

SUBROUTINE PAUSE

Displays the message ENTER CR TO CONTINUE and waits for a user response. The system ignores any input other than a carriage return. You can use the PAUSE routine to allow users time to read a screen of information before it is scrolled off by subsequent information. (No parameters.)

SUBROUTINE COPYF (I,J)

Copies the contents of the source file (I) to the destination file (J) and erases file (I).

Call Parameters:

Argument	Type	I/O	Description
I	I	I	File number of the file to be copied from
J	I	O	File number of the file to be copied to

FUNCTION CUTNAM (NAME,SHEET)

Creates a field CUTNAM consisting of a partial drawing name and a sheet number separated by a space, a slash, and a space (/). The calling program specifies the field size of CUTNAM. For example, if CUTNAM is declared as 20 characters long in the calling program, and the sheet name consists of 2 characters, the function returns the first 15 characters of the drawing name, followed by ' / ' and the sheet number. If the drawing name does not have 15 significant characters, CUTNAM compresses the result.

Call Parameters:

Argument	Type	I/O	Description
NAME	C*(*)	I	EDL data name
CUTNAM	C*(*)	O	Partial drawing name / sheet #
SHEET	I	I	Sheet number to be appended

SUBROUTINE CUTSTR (INSTR,REMSTR,LENGTH,ALIGN)

Cuts the input string (INSTR) at a blank so that the resulting input string is less than the specified LENGTH. The remainder of the string is returned in REMSTR. ALIGN is farthest position to left to check for a blank. If no blank is found, the line is split at the specified length.

The following example illustrates the use of CUTSTR.

```
INSTR = 'THIS IS A SAMPLE OF AN INPUT STRING'
ALIGN = 10
LENGTH = 26
RESULTING INSTR = 'THIS IS A SAMPLE OF AN'
RESULTING REMSTR = 'INPUT STRING'
```

Call Parameters:

Argument	Type	I/O	Description
INSTR	C*(*)	I/O	Input string
REMSTR	C*(*)	O	Remainder of the string
LENGTH	I	I	Length the input string should be cut to
ALIGN	I	I	Farthest left position to check for a blank

FUNCTION FULLNM (USR)

Reads the UI record of the specified EDL ID, and returns the corresponding last, first, and middle names in the form: Adams, John Quincy. If the first or middle name consists of only one character (that is, an initial) FULLNM places a period after that character: Adams, John Q.

Call Parameters:

Argument	Type	I/O	Description
USR	C*(*)	I	EDL ID
FULLNM	C*(*)	O	First, middle, and last names

FUNCTION FULPER (MODE)

Spells out a single-character permission mode (W, R, I, or N) to (WRITE, READ, INFO, or NONE).

Call Parameters:

Argument	Type	I/O	Description
MODE	C*(*)	I	One-character file permission
FULPER	C*(*)	O	Spelled-out file permission

SUBROUTINE GETPRM (PRM,VAL,FOUND)

Gets the value of a task process parameter and returns it to the program. The parameter (from the TV record) may be a constant, variable, or prompt.

Call Parameters:

Argument	Type	I/O	Description
PRM	C*(*)	I	Task process parameter name
VAL	C*(*)	O	Value of the parameter
FOUND	L	O	TRUE if a parameter was returned, FALSE if a parameter was not found

SUBROUTINE GETPRN (VAL,FOUND)

Gets the next value of the task process parameter defined by GETPRM and returns it to the program. The parameter (from the TV record) may be a constant, variable, or prompt.

Call Parameters:

Argument	Type	I/O	Description
VAL	C*(*)	O	Value of the parameter
FOUND	L	O	TRUE if a parameter was returned, FALSE if a parameter was not found

CHARACTER FUNCTION LEFTJ (NUMBER)

Converts a number into a left-justified character string.

Call Parameters:

Argument	Type	I/O	Description
NUMBER	I	I	Number to be left-justified
LEFTJ	C*(*)	O	Resulting left-justified character string

SUBROUTINE LIST (MNA,INFO)

Concatenates a title and its description from the menu database and prints it as a list. LIST only lists one line each time it is called. The position of the alignment is determined by the end of the title in the menu database.

The following example shows a list created by four calls to LIST.

```
EDL USER ID  CADDATDEV
   PASSWORD  GDS43L
  USER NAME  GL0234F
 DEPARTMENT  9087
```

Call Parameters:

Argument	Type	I/O	Description
MNA	C*(*)	I	Name of message menu for title
INFO	C*(*)	I	Text to be concatenated to message

FUNCTION LSTCHR (STR)

Finds the last nonblank character working backward from the end of a string. This function is useful for concatenating strings.

Call Parameters:

Argument	Type	I/O	Description
STR	C*(*)	I	String to be examined
LSTCHR	I	O	Position of the last nonblank character in STR

SUBROUTINE NXTEDN (HOS,EDN,OK)

Finds the next available data identifier for the host. It is used to find the correct DIEDN value before adding a new DI record.

Call Parameters:

Argument	Type	I/O	Description
HOS	C*(*)	I	Host identifier
EDN	I	O	Next unused data identifier for the host
OK	L	O	TRUE if no error

SUBROUTINE NXTFIL (HOS,FIL,OK)

Finds the next available file identifier for the host. It is used to find the correct FIFIL value before adding a new FI record.

Call Parameters:

Argument	Type	I/O	Description
HOS	C*(*)	I	Host identifier
FIL	I	O	Next unused file identifier for the host
OK	L	O	TRUE if no error

SUBROUTINE PUTNAM (NAME,SHEET)

Uses PUTVAR to store DINAM in parameter NAME and DISID in parameter SHEET.

Call Parameters:

Argument	Type	I/O	Description
NAME	C*(*)	I	Data name
SHEET	I	I	Sheet number

SUBROUTINE PUTVAR (NAM,VAL)

Stores the value VAL in EDL global variable NEXT. VALUE is returned when EDL looks for a VARIABLE type parameter.

Call Parameters:

Argument	Type	I/O	Description
NAM	C*(*)	I	Parameter name to store
VAL	C*(*)	I	Parameter value to store

SUBROUTINE RETLIS (IMAX,MSGT,VAR,OK)

RETLIS displays a selection list and prompts for a choice. The selection list must be prepared by the calling program on FORTRAN unit 12. RETLIS processes the information on file EEEDL12 in the following manner:

1. The system displays the following message:

   ```
   n SELECTIONS
   ```

   where n is the number of records on EEEDL12.

2. The system displays the message specified by HEADER.

3. The system reads each record on EEEDL12.

4. The system ignores the first I characters on each line where I=len(value).

5. The system displays the next 132 characters on the line preceded by a sequence number. For example:

   ```
   "1. ADAMS, JOHN Q"
   ```

6. EDL prints the following message after displaying NL lines (where NL is the number of lines on a screen), or when the end of the list is encountered:

   ```
   "ENTER A NUMBER, E OR EXIT TO EXIT, OR CR FOR MORE"
   ```

7. Depending on the user input, EDL responds as follows:

   a. If the user enters a number J, EDL rewinds EEEDL12 and reads VALUE from the Jth line of the file.

   b. If the user enters a null response, EDL prints more of the list. If the list was ended, it is started over.

   c. If the user enters E, OK is set to FALSE and EDL returns to the calling routine.

Call Parameters:

Argument	Type	I/O	Description
IMAX	I	I	Number of records on EEEDL12
MSGT	C*(*)	I	Message identifier for the table header
VAR	C*(*)	O	Contents of the line to be returned
OK	L	O	TRUE if the user made a selection, FALSE if the user chose EXIT

Additional EDL Subroutines

Appendix C contains a list of the standard subroutines used by EDL. To see a complete task definition showing how each is used in the Message Database, use the RTASKS procedure on E125PRC (BEGIN,RTASKS,E125PRC). You can call the standard EDL subprograms from task protocols that you create, but you cannot modify or read them; source code is not provided.

Examples

This section provides examples of EDL customization involving the manipulation of the Engineering Data Database.

Adding a Task to EDL

Depending on its complexity, adding a task to EDL can involve creating a new CCL procedure and changing the Message and Task Database to use it, or it can mean writing a new FORTRAN subroutine. Chapter 2 of this manual provides an example of the simple task addition using the interactive MENUMOD utility.

CAUTION

You should always make your changes to a copy of the working databases to avoid damaging your current system. EDL prevents modification of the Message and Task Database while any user is active; the database is opened for reading, shared with all readers.

1. Use the NOS command COPYEI to make a copy of EDL.
2. Make your changes against the copy and test them out.
3. After successfully testing your modifications, implement them on your working databases.

The following example shows how to create a batch transaction data file that adds both a procedure and new code.

1. Use your text editor to prepare a batch transaction data file to change the Message and Task Database as shown in the following example.

```
A TI      MYTASK                        CUSTOMIZED TASK
A TC      MYTASK    MYCOMMAND
A TP      MYTASK              1OVCAP    XMYCODE
A TP      MYTASK              2CCL PROC MYPROC    MYPROCF
A TV      MYTASK              2         1MYPARM   VARIABLE  MYPARM
```

   This transaction file creates a task called MYTASK that is called with the command MYCOMMAND. The task MYTASK consists of two steps:

   a. It runs an OVCAP named XMYCODE.

   b. It executes a procedure called MYPROC from file MYPROCF. EDL passes a parameter called MYPARM to procedure MYPROC.

2. Create an OVCAP like the following example. (You omit this step if you are adding a task that contains only a CCL procedure.)

```
OVCAP.
SUBROUTINE XMYCODE
CALL MYCODE
RETURN
END
```

NOTE

Although the OVCAP could actually contain the code for the subroutine, we recommend that you keep your code in a separate library for ease of modification, testing, and maintenance.

3. Create a subroutine to call any necessary IB routines. You can also include any of the standard EDL routines described in this chapter to display messages, prompts, menus, or selection lists. (You omit this step if you are adding a task that contains only a CCL procedure.)

   In the following example, subroutine MYCODE prompts the user for a character string to be put into variable MSG. If the user enters a string, the PUTVAR routine is called to put the reply into a variable called MYPARM. The value in MYPARM is then available to any part of EDL and can be passed to the CCL procedure. (Remember that CCL can only handle parameters up to 40 characters in length.)

```
      SUBROUTINE MYCODE
C     THIS ROUTINE PROMPTS THE USER FOR A MESSAGE, AND WRITES IT IN A NOTE
C     TO THE TERMINAL WITH CCL PROCEDURE MYPROC ON FILE MYPROCF.
      CHARACTER MSG*30
C     'ENTER THE MESSAGE FOR A NOTE OR CR TO RETURN'
      CALL INTXT('MYCODE1',MSG,ICH)
      IF(ICH.NE.0)THEN
         CALL PUTVAR('MYPARM',MSG)
      ENDIF
      RETURN
      END
```

   This code should be put into a library. You must also remember to copy the corresponding COMMON blocks from file EDLCOM if you use any of the IB routines.

4. Create the Message and Task Database transactions needed for the MYCODE routine as shown in the following example. (You omit this step if you are adding a task that only contains a CCL procedure.)

```
A MI      MYCODE1   PROMPT    ENTER THE MESSAGE FOR A NOTE OR CR TO RETURN
A MH      MYCODE1             1WHAT YOU ENTER HERE WILL BE DISPLAYED IN A
A MH      MYCODE1             2NOTE ON YOUR TERMINAL AND IN YOUR DAYFILE.
```

5. Create the CCL procedure as follows. (You omit this step if you are adding a task that only contains an OVCAP.)

```
.PROC,MYPROC,MYPARM.
NOTE,,NR./MYPARM
REVERT. MYPROC
EXIT.
REVERT,ABORT. MYPROC
```

   Save the procedure in the file named in the menu database transaction file created in step 1. The file should be permitted in the same manner as the file E125PRC.

6. Load the routines. (You omit this step if you are adding a task that contains only a CCL procedure.)

   For example, if the OVCAP you created was written onto a file called MYOVCAP, and the routine MYCODE was put on library MYLIB, the call to LOADEDL would look like this:

   ```
   BEGIN,LOADEDL,E125PRC,F=MYOVCAP,ULIB=MYLIB.
   ```

   LOADEDL creates a new E125ABS absolute by combining the OVCAPs and routines from your original release of EDL with the OVCAPs in file MYOVCAP and the routines in library MYLIB.

   NOTE

   Keep in mind that any routines and/or OVCAPs that you have previously added will not be in this load unless they are included in the OVCAP and library files loaded here.

7. Update the Message and Task Database.

   Put the transactions you created in the preceding steps into the same file, and update EDL for those transactions as follows:

   a. Enter EDL using an ID with SYSADMIN privileges (for example, EDLID).

   b. Select the ADD INFORMATION FOR ENGINEERING DATA task (or enter command ADDINFO).

   c. Specify the application data type: EDL MDB TRANSACTIONS.

   d. Give the data a meaningful name, like MDB TRANSACTIONS TO ADD A MYTASK.

   e. Enter the MENUMGMT command.

   f. Retrieve the data that you just entered into EDL.

   g. When you select the data, the changes are automatically entered into EDL.

Creating a Site-Defined Retrieval

This example of EDL customization shows how to create a site-defined retrieval method. This modification requires the addition of reserved subroutines and menu lines. In order to actually implement this example, you must have FORTRAN 5 installed on your system.

This example alters the retrieval selection routine in EDL to process an additional option value called SITE in option menus EXTRAC and EXFRAC. When a user selects SITE, EDL calls a routine called EXTSIT. The following steps outline the procedure used to create and add this routine:

1. Write and compile a routine named EXTSIT.
2. Use the procedure LOADEDL to establish a new EDL program and link your code.
3. Change the Message and Task Database to use the your new menu changes, prompts, error messages, and help information.

4. Test and archive your work.

Building Routine EXTSIT

The EXTSIT routine retrieves data based on the type of the file that the data resides on. The line numbers in EXTSIT are for reference only (they are not part of the program). The tables that follow the program listing provide descriptions for the referenced lines.

NOTE

This example shows program statements derived from the EDLCOM data file. Though they were accurate at the time of this document's printing, they might have changed. Treat this as an example and refer to the files in EDLCOM for the current contents.

```
 1        SUBROUTINE EXTSIT(NUM)
 2  CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 3  CXX
 4  CXX    PURPOSE - RETRIEVE BY FILE TYPE CODE
 5  CXX
 6  CXX    CALL PARAMETERS -
 7  CXX    ARQUMENT  TYPE      I/O  DESCRIPTION
 8  CXX    NUM         I        O   NUMBER OF RECORDS RETRIEVED
 9  CXX
10  CXX    DATABASE USAGE -
11  CXX       DI     DATA INFORMATION RECORD
12  CXX       FT     FILE TYPE RECORDS
13  CXX       FI     FILE INFO RECORDS
14  CXX
15  CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
16  C      ENTER FILE TYPE
17  C      IF 'LIST' CALL LISFTC
18  C      SET FIFTC TO THE FILE TYPE
19  C      OBTAIN AN FI RECORD
20  C      WHILE THERE ARE FI RECORDS
21  C         USE THE FIDI COSET TO GET A DI RECORD
22  C         WHILE THERE ARE DI RECORDS
23  C            CALL EXTWRI(NUM) TO WRITE THE RECORD TO EEEDL9 IF
24  C                                          THE DATA IS PERMITTED
25  C            GET ANOTHER DI RECORD
26  C         GET ANOTHER FI RECORD
27  CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

Lines	Description
1	EDL is coded to look for the routine EXTSIT, with the parameter NUM where NUM is the number of records found. This counter is incremented by the routine called EXTWRI.
2-15	Prologue. The database usage area is the names of the database records used in this routine. These record types correspond to the common blocks which are included from the file EDLCOM.
16-27	Explanation of how EXTSIT processes.

```
C
C     EDL_COMMON BLOCK
C     EDL PRIMARY COMMON BLOCK
      COMMON /ECOM1/ HOST, USR, PWD, MDISP, SCLOCK,
     +CHELP, CLIST, CEXIT, CMENU, CCLEAR,
     +CWORK, CREL, CSUBM, CPEND,
     +CPAUSE1, CPAUSE2, CINOPT1, CEXTM1,
     +CYES, CNO,
     +NOSUN, STRDEL, INPDEL,
     +AUN, DUN, DDB, MUN, MDB, AC, IT, OT
      CHARACTER*10 HOST, USR, PWD, MDISP, SCLOCK
      CHARACTER*10 CHELP, CLIST, CEXIT, CMENU, CCLEAR
      CHARACTER*10 CWORK, CREL, CSUBM, CPEND
      CHARACTER*70 CPAUSE1, CPAUSE2, CINOPT1, CEXTM1
      CHARACTER*3 CYES,CNO
      CHARACTER*7 NOSUN
      CHARACTER*1 STRDEL, INPDEL
      CHARACTER*7 AUN, DUN, DDB, MUN, MDB
      CHARACTER*2 AC
      CHARACTER*7 IT,OT
      COMMON /ECOM2/ NSYNC, PW, PL, NL, SCROLL, ECHO
      INTEGER NSYNC, PW, PL, NL
      LOGICAL SCROLL, ECHO
C
C     DI COMMON BLOCK
C
      COMMON / DI       / DIEDN
      COMMON / R600701 / DINAM
      COMMON / R600702 / DISID  ,DIFIL
      COMMON / R600703 / DIREV  ,DIEDT  ,DIADT  ,DIUSR  ,DITTL
     *,DISTA  ,DIDATC ,DIDATM ,DIDATR ,DITIMC ,DITIMM ,DITIMR
      INTEGER DIEDN  ,DISID  ,DIFIL
      CHARACTER DINAM  *70,DIREV  *10,DIEDT  *20,DIADT  *20
     *,DIUSR  *10,DITTL  *100,DISTA  *10,DIDATC *10,DIDATM *10
     *,DIDATR *10,DITIMC *10,DITIMM *10,DITIMR *10
C
C     FI COMMON BLOCK
C
      COMMON / FI       / FIFIL
      COMMON / R601601 / FIHOS  ,FIFUN  ,FIPFN  ,FIFTC  ,FISTA
     *,FIUSR  ,FIVSN  ,FICT   ,FIMOD
      INTEGER FIFIL
      CHARACTER FIHOS  *10,FIFUN  *31,FIPFN  *100
     *,FIFTC  *20,FISTA  *10,FIUSR  *10,FIVSN  *6,FICT   *2
     *,FIMOD  *1
C
C     FT COMMON BLOCK
C
      COMMON / FT       / FTFTC  ,FTNAM  ,FTAPN  ,FTLFN
      COMMON / R602201 / FTLFNR ,FTMUL  ,FTPRT
      COMMON / R602202 / FTCHR
      LOGICAL FTLFNR ,FTMUL  ,FTPRT
      CHARACTER FTFTC  *20,FTNAM  *21,FTAPN  *20,FTLFN  *7
     *,FTCHR  *1
C
```

Lines	Description
28-82	COMMON Blocks. These declarations are included from the EDLCOM file. Blocks DI, FI, and FT are used by the IB routines. Include the corresponding file for each record type used. EDL_COMMON is the primary common block in EDL and contains constants used throughout EDL. In this example, the block must be included to provide the constant CLIST.

```
      LOGICAL OK
      CHARACTER FTC*20
100   CONTINUE
      NUM=0
C
C     GET THE FILE TYPE
C
C     'ENTER THE FILE TYPE TO BE RETRIEVED OR LIST OR CR TO RETURN'
      CALL INTXT('EXTSIT1',FTC,ICH)
      IF(ICH.NE.0)THEN
         IF(FTC.EQ.CLIST)THEN
            CALL LISFTC(FTC,OK)
            FTFTC=FTC
         ELSE
```

Lines	Description
83	The logical variable OK monitors the status of the calls to IB routines.
84	Character variable FTC stores the user's choice of file type code.
86	NUM is the count of records found. If NUM is returned to the calling routine as 0, the routine displays a message indicating that no records were found. This count is updated by routine EXTWRI.
91	Prompt for input. The INTXT routine prompts with the message named EXTSIT1 found in the message database, and puts the response in variable FTC. ICH is the character length of the response.
92	If the response was a carriage return, ICH is set to 0, the routine skips to the ENDIF on line 118, and it returns to the calling routine.
93-100	An explanation of file type (FT) records. There are two identification fields in the FT record: FTFTC and FTNAM. When EDL is released, these fields contain identical information. FTFTC is the file type code that EDL uses internally; do not change this field. FTNAM is the external name that the user sees; you can customize this field to reflect the terminology of your particular site. LISFTC is a routine that lists all FTNAM fields. When the user makes a selection, the value of the corresponding FTFTC field is returned to EDL.
93-95	If the response is equal to the constant CLIST (set to LIST when EDL is released), then the LISFTC routine is called. LISFTC lists the available FTNAM values, and prompts the user to select one. If the user chooses EXIT, OK is set to FALSE. If the user selects one of the file names, OK is set to TRUE and the FTFTC value corresponding to the FTNAM chosen by the user is put into variable FTC.

```
 97                FTNAM=FTC
 98                CALL IBOFT1(OK)
 99                IF(.NOT.OK)THEN
100  C               'THE FILE TYPE IS NOT RECOGNIZED BY EDL'
101                  CALL ERR('EXTSIT2')
102               ENDIF
103             ENDIF
104             IF(OK)THEN
105                FIFTC=FTFTC
106                CALL IBOFI4(OK)
107  200           IF(OK)THEN
108                   CALL IBFFIDI(OK)
109  300              IF(OK)THEN
110                      CALL EXTWRI(NUM)
111                      CALL IBNFIDI(OK)
112                      GO TO 300
113                   ENDIF
114                   CALL IBEFI4(OK)
115                   GO TO 200
116                ENDIF
117             ENDIF
118          ENDIF
119  900     CONTINUE
120          RETURN
121          END
```

Lines	Description
97-102	If the response is not CLIST, FTNAM is set equal to the response and IBOFT1 is called to obtain the the FT record via access path FT1. If no FT record is found, the system displays error message EXTSIT2. If the matching FT record is found, the system fills all fields in the common block FT with information from the matching record.
104	If OK was set to FALSE, either because no FT record was found or because the user exited routine LISFTC, processing skips to the ENDIF in line 117 and returns to the calling routine.
105	FIFTC (a key for the FI record) is set equal to the FTFTC value determined in line 95.
106	IBOFI4 obtains the first FI record via access path FI4 (FIFTC). OK is set to TRUE if records are found; otherwise FALSE.
107	If OK is FALSE, control goes to the ENDIF in line 116 and returns to the calling routine.
108	IBFFIDI obtains the first DI record corresponding to the FI record obtained either in line 106 or 114. OK is set to TRUE if records are found; otherwise FALSE.
109	If OK is FALSE, control goes to the ENDIF in line 113.
110	EXTWRI with the parameter NUM checks the DI record to see if the data should be in the user's retrieval list. This routine checks file permissions, application data types, and engineering categories, depending on the ADT or EDT task parameters on the task that called this routine. If the record is acceptable, the program writes data information to file EEEDL9 and increments NUM.
111-112	IBNFIDI obtains the next DI record corresponding to the current FI record. OK is set to TRUE if a record is found; otherwise FALSE. Control then goes to line 109 (statement number 300).
114-115	Once all of the corresponding DI records are found for an FI record, IBEFI4 finds any other FI record with the same FIFTC field. OK is set to TRUE if a record is found; otherwise FALSE. Control then goes to line 107 (statement number 200).

Building Routine EXFSIT

The EXFSIT routine performs further extractions (option 5 of the RETRIEVAL OPTION menu). This routine functions like EXTSIT with the exceptions as noted in lines 224 - 240.

```
122         SUBROUTINE EXFSIT(NUM)
123   CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
124   CXX
125   CXX   PURPOSE - FURTHER RETRIEVE BY FILE TYPE CODE
126   CXX
127   CXX   CALL PARAMETERS -
128   CXX   ARQUMENT  TYPE       I/O  DESCRIPTION
129   CXX   NUM          I        O   NUMBER OF RECORDS RETRIEVED
130   CXX
131   CXX   DATABASE USAGE -
132   CXX      DI    DATA INFORMATION RECORD
133   CXX      FI    FILE INFO RECORD
134   CXX      FT    FILE TYPE RECORD
135   CXX
136   CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
137   C     ENTER FILE TYPE
138   C     IF 'LIST' CALL LISFTC
139   C     SET FIFTC TO THE FILE TYPE
140   C     READ A RECORD OFF EEEDL9
141   C     GET THE CORRESPONDING DI RECORD
142   C     GET THE CORRESPONDING FI RECORD
143   C     IF THE FTC'S MATCH
144   C        WRITE THE RECORD TO EEEDL10
145   C     READ ANOTHER RECORD FROM EEEDL9
146   CXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
147   C
148   C     EDL_COMMON BLOCK
149   C     EDL PRIMARY COMMON BLOCK
150         COMMON /ECOM1/ HOST, USR, PWD, MDISP, SCLOCK,
151        +CHELP, CLIST, CEXIT, CMENU, CCLEAR,
152        +CWORK, CREL, CSUBM, CPEND,
153        +CPAUSE1, CPAUSE2, CINOPT1, CEXTM1,
154        +CYES, CNO,
155        +NOSUN, STRDEL, INPDEL,
156        +AUN, DUN, DDB, MUN, MDB, AC, IT, OT
157         CHARACTER*10 HOST, USR, PWD, MDISP, SCLOCK
158         CHARACTER*10 CHELP, CLIST, CEXIT, CMENU, CCLEAR
159         CHARACTER*10 CWORK, CREL, CSUBM, CPEND
160         CHARACTER*70 CPAUSE1, CPAUSE2, CINOPT1, CEXTM1
161         CHARACTER*3 CYES,CNO
162         CHARACTER*7 NOSUN
163         CHARACTER*1 STRDEL, INPDEL
164         CHARACTER*7 AUN, DUN, DDB, MUN, MDB
165         CHARACTER*2 AC
166         CHARACTER*7 IT,OT
167         COMMON /ECOM2/ NSYNC, PW, PL, NL, SCROLL, ECHO
168         INTEGER NSYNC, PW, PL, NL
169         LOGICAL SCROLL, ECHO
170   C
171   C     DI COMMON BLOCK
```

```
172  C
173        COMMON / DI       / DIEDN
174        COMMON / R600701 / DINAM
175        COMMON / R600702 / DISID  ,DIFIL
176        COMMON / R600703 / DIREV  ,DIEDT  ,DIADT  ,DIUSR  ,DITTL,DISTA
177       *,DIDATC ,DIDATM ,DIDATR ,DITIMC ,DITIMM ,DITIMR
178        INTEGER DIEDN  ,DISID  ,DIFIL
179        CHARACTER DINAM  *70,DIREV  *10,DIEDT  *20,DIADT  *20
180       *,DIUSR  *10,DITTL  *100,DISTA  *10,DIDATC *10,DIDATM *10
181       *,DIDATR *10,DITIMC *10,DITIMM *10,DITIMR *10
182  C
183  C     FI COMMON BLOCK
184  C
185        COMMON / FI       / FIFIL
186        COMMON / R601601 / FIHOS  ,FIFUN  ,FIPFN  ,FIFTC  ,FISTA
187       *,FIUSR  ,FIVSN  ,FICT   ,FIMOD
188        INTEGER FIFIL
189        CHARACTER FIHOS  *10,FIFUN  *31,FIPFN  *100
190       *,FIFTC  *20,FISTA  *10,FIUSR  *10,FIVSN  *6,FICT   *2
191       *,FIMOD  *1
192  C
193  C     FT COMMON BLOCK
194  C
195        COMMON / FT       / FTFTC  ,FTNAM  ,FTAPN  ,FTLFN
196        COMMON / R602201 / FTLFNR ,FTMUL  ,FTPRT
197        COMMON / R602202 / FTCHR
198        LOGICAL FTLFNR ,FTMUL  ,FTPRT
199        CHARACTER FTFTC  *20,FTNAM  *21,FTAPN  *20,FTLFN  *7
200       *,FTCHR  *1
201  C
202        LOGICAL OK
203        CHARACTER FTC*20,LINE*80
204  100   CONTINUE
205        NUM=0
206  C
207  C     GET THE FILE TYPE
208  C
209  C     'ENTER THE FILE TYPE TO BE RETRIEVED OR LIST OR CR TO RETURN'
210        CALL INTXT('EXFSIT1',FTC,ICH)
211        IF(ICH.NE.0)THEN
212           IF(FTC.EQ.CLIST)THEN
213              CALL LISFTC(FTC,OK)
214              FTFTC=FTC
215           ELSE
216              FTNAM=FTC
217              CALL IBOFT1(OK)
218              IF(.NOT.OK)THEN
219  C              'THE FILE TYPE IS NOT RECOGNIZED BY EDL'
220                 CALL ERR('EXFSIT2')
221              ENDIF
222           ENDIF
223           IF(OK)THEN
```

```
224                REWIND 9
225   200          READ(9,5000,END=900)DIEDN,LINE
226   5000         FORMAT(I10,A)
227                CALL IBODIO(OK)
228                IF(.NOT.OK)CALL ERRIB
229                FIFIL=DIFIL
230                CALL IBOFIO(OK)
231                IF(.NOT.OK)CALL ERRIB
232                IF(FIFTC.EQ.FTFTC)THEN
233                   WRITE(10,5000)DIEDN,LINE
234                   NUM=NUM+1
235                ENDIF
236                GO TO 200
237             ENDIF
238          ENDIF
239   900    CONTINUE
240          RETURN
241          END
```

Lines	Description
224-226	Rather than obtaining data records based on criteria, subsequent extractions read the current list of data records and compare them against the criteria given. All retrievals write records to the file EEEDL9. This routine reads record information from EEEDL9.
227	EDL obtains the DI record based on the DIEDN read from EEEDL9. The only reasons a record would not be there are if it had been purged, or if there was something wrong with the database. If the record cannot be found, OK is set to FALSE by the IBODI0 routine.
228	If OK was set to FALSE, the ERRIB routine is called to display the reason for the error. ERRIB is used in EDL to assist the tracking of unexpected problems in code or in the database.
232-235	If the FIFTC field for the data file matches the user entry, the program writes the record to EEEDL10 and increments NUM.
240	File EEEDL10 is copied over EEEDL9, giving a new selection list, if NUM is returned as greater than 0 when the routine terminates. If NUM is zero (meaning that no records met the new criteria), the original EEEDL9 file is left unchanged.

Updating the Message and Task Database

After creating the new subroutine EXTSIT, you need to update your Message and Task Database to include the new option. The sample batch transaction file shown in figure 3-2 performs the following updates:

1. Adds another line to the retrieval option menu. Note that the option value (OV) is SITE. EDL checks for this value when processing the retrieval menu.

2. Adds the following prompt:

   ```
   ENTER THE FILE TYPE CODE, LIST, OR CR TO RETURN
   ```

   Associated HELP messages are added if the user enters HELP in response to this prompt.

3. Adds the following error message:

   ```
   THE FILE TYPE IS NOT KNOWN TO EDL
   ```

4. Adds the same features for secondary retrievals (EXFRAC).

```
A OM      EXTRAC             13FILE TYPE CODE
A OK      EXTRAC             13FTC
A OV      EXTRAC             13          1SITE
A MI      EXTSIT1   PROMPT    ENTER THE FILE TYPE CODE, LIST, OR CR TO  RETURN
A MH      EXTSIT1             1THE FILE TYPE CODE IS DEFINED BY THE SITE TO DESCR
A MH      EXTSIT1             2USE OF A PARTICULAR TYPE OF FILE.  ENTER "LIST" TO
A MH      EXTSIT1             3A LIST OF POSSIBLE FILE TYPES.
A MH      EXTSIT1             4A CARRIAGE RETURN WILL RETURN TO THE RETRIEVAL MET
A MI      EXTSIT2   ERROR     THE FILE TYPE IS NOT KNOWN TO EDL
A OM      EXFRAC             13FILE TYPE CODE
A OK      EXFRAC             13FTC
A OV      EXFRAC             13          1SITE

A MI      EXFSIT1   PROMPT    ENTER THE FILE TYPE CODE, LIST, OR CR TO RETURN
A MH      EXFSIT1             1THE FILE TYPE CODE IS DEFINED BY THE SITE TO DESCR
A MH      EXFSIT1             2USE OF A PARTICULAR TYPE OF FILE.  ENTER "LIST" TO
A MH      EXFSIT1             3A LIST OF POSSIBLE FILE TYPES.
A MH      EXFSIT1             4A CARRIAGE RETURN WILL RETURN TO THE RETRIEVAL MET
A MI      EXFSIT2   ERROR     THE FILE TYPE IS NOT KNOWN TO EDL
```

Figure 3-2. Sample Batch Transaction File for Implementing a Site-Defined Retrieval

Adding a New Application 4

Adding a New Application 4

Adding a new application to EDL involves modifying nearly every part of the EDL system. You need an application header record in the database, CCL procedures, tasks, menus, and commands to invoke the application, tasks to retrieve application data, tasks to perform data transfers and translations, new file type and data type definitions, and perhaps application terminal configuration records to pass parameters to the application. EDL contains a procedure that automatically performs all these steps for the ICEM applications.

Application Coding Guidelines

Applications controlled by EDL must conform to certain conventions. The following paragraphs describe the coding conventions you should follow when adding new EDL applications to the database.

Data Naming

Keep the following points in mind when defining application data names.

- All file names and directory names accepted by an application should be a maximum of 100 characters long.
- Do not hard-code file names within your application.
- Your application should preserve the name assigned to a piece of data regardless of its storage format.

Application Command Line

EDL can start an application procedure and specify all parameters as parameter values. For example, `/PARTS=:udd:user:parts` might indicate the name of the user's database of parts. The application start procedure is responsible for supplying default values for parameters not specified and converting parameters into arguments, if necessary.

The CCL command line has a limit of 80 characters. If this proves to be too small, create an EDL subprogram to provide a file containing a list of parameters and values, then pass the name of this file to the application procedure.

Application Scripts

You can run your application from an input script to control certain data transfer/retrieval operations started by EDL. For example, the retrieval of a DDN drawing from the global part file is accomplished by running DDN with a script that restores the named data from the GPARTS file and makes it the working part. EDL maintains both the application data name and the actual system file name for the data. EDL can then use either name as directed by the input script.

User Profile Tasks

EDL provides a user profile default files task to preset the names of default application files/libraries. The default files list contains the preset names that are passed to CCL procedures as parameters. The value assigned to any parameter in this task must be the name of a file known to EDL. EDL ensures that the user has the appropriate permission to the named files, but does not ensure a file is exclusively open to the user; your application must handle file opening.

EDL Log File

All application filing operations should be logged in the EDLLOG file.

Batch Mode Operation

EDL may run applications as a batch task to perform certain data translations. Any application you create must be capable of running in batch. This means the NOS UPROC must not execute any operations not appropriate to batch mode.

File Locking

Because multiple users may share data, and a single user may have multiple copies of an application executing, all data files used in WRITE mode should be opened for exclusive access. Your applications must be able to detect a file already opened to another process and shut down gracefully.

File Creation

EDL can create certain files at the user's request. If your application allows EDL to create its input files, the application must be able to recognize an empty, uninitialized file and complete its creation.

Data Hierarchies

EDL manages one level of file hierarchy. Files may be designated as containing the following elements:

- One piece of engineering data (for example, a script file within the working directory)
- Multiple data, each in its own record (for example, the TAPE3 file)
- Multiple data, each in its own system file (for example, the PATTERN library)

EDL is able to identify the data name, and the file name for a piece of data stored in organization type 3. Each application must decide what information is required for it to recognize the data/file being referred to.

Example

The rest of this chapter provides an example of adding a new application to EDL. Use the following steps as an outline for adding your own site-specific applications to EDL.

1. Create the application header.

   For example, the application header for a data analyzer application might appear as follows:

```
INSERT INTO AI
  SET AIAPN = 'DATA ANALYZER',
      AIAPV = '1.0',
      AISTA = 'ACTIVE'
```

2. Define new file types and data types.

   To add new types of data to EDL, you must define the new file types (FT) on which the data resides, and then define the application data types. For example, here is how ICEM DDN drawing files and drawing data are defined in the standard EDL database.

```
INSERT INTO FT
  SET FTFTC = 'DRAWING FILE',
      FTNAM = 'DRAWING FILE',
      FTLFN = 'TAPE3',
      FTLFNR = -1,
      FTAPN = 'ICEM DDN',
      FTMUL = -1,
      FTCHR = 'B',
      FTPRT = 0
INSERT INTO AT
  SET ATADT = 'DRAWING',
      ATNAM = 'DRAWING',
      ATFTC  = 'DRAWING FILE',
      ATSIDR  = -1,
      ATTNA = 'RET-DRW'
```

3. Specify terminal configuration.

   EDL stores terminal configuration data for each user in the user configuration (UC) records. By matching the user's configuration with the application configuration (AC) records and the application information (AI) records, EDL decides what configuration parameter value to pass to the CCL procedures which execute application programs.

   Like all parameters, configuration parameters are controlled by task parameter value (TV) records for the CCL procedure task process. If the TVTYP field is CONFIG, EDL looks for all the AC records with ACPRM fields that match the TVVAL field of the TV record. If the ACATR and ACSTA fields match the user's current UCATR and UCSTA fields and the application version is active, the value in ACVAL is passed to the CCL procedure.

   EDL terminal configuration information resides in the EDL user configuration (UC) and application configuration (AC) records. ICEM DDN and ICEM Solid Modeler and Analysis applications also use the terminal configuration data in these records.

4. Include the EDL log file.

   When a standard ICEM application creates, modifies, or deletes data, it makes a log entry on file EDLLOG to inform EDL that it should update the database and prompt for additional descriptive information. The following table shows the format for each record in EDLLOG. Type C refers to character strings, type I to integer data. There are no record terminators in this format. C entries are left-justified and blank- or zero-filled; I entries are right-justified and blank-filled.

Pos	Len	Type	Description
1	1	C	Action code, A (added), C (changed), D (deleted), F (file copied), P (purged file), R (retrieved).
2	100	C	Path name of data file
102	31	C	Operating system user name
133	20	C	File Type code
153	20	C	Application Data Type code
173	100	C	Data Name
273	10	I10	Secondary Identifier (e. g. Sheet Number)

   If you want EDL to automatically track the data operated by your application, you must modify the application to create the EDLLOG file. Then you must modify EDL to include your application. Include a task process (TP) step to execute the XLOG subprogram at the end of each task that executes your application.

5. Create a default working file list.

   EDL creates a list of files for use by an application. The ATTACH subprogram can create this list. Initially, this list would contain the files specified by the user through the Default Files task. Other subprograms may add additional files to the list. The list is passed to the CCL procedure as a set of parameters in the form `lfn=pfn`, where lfn is the logical file name of the file and pfn is the path name of the actual file to be used. This list is deleted when the CCL procedure returns to EDL.

6. Provide for application data retrieval.

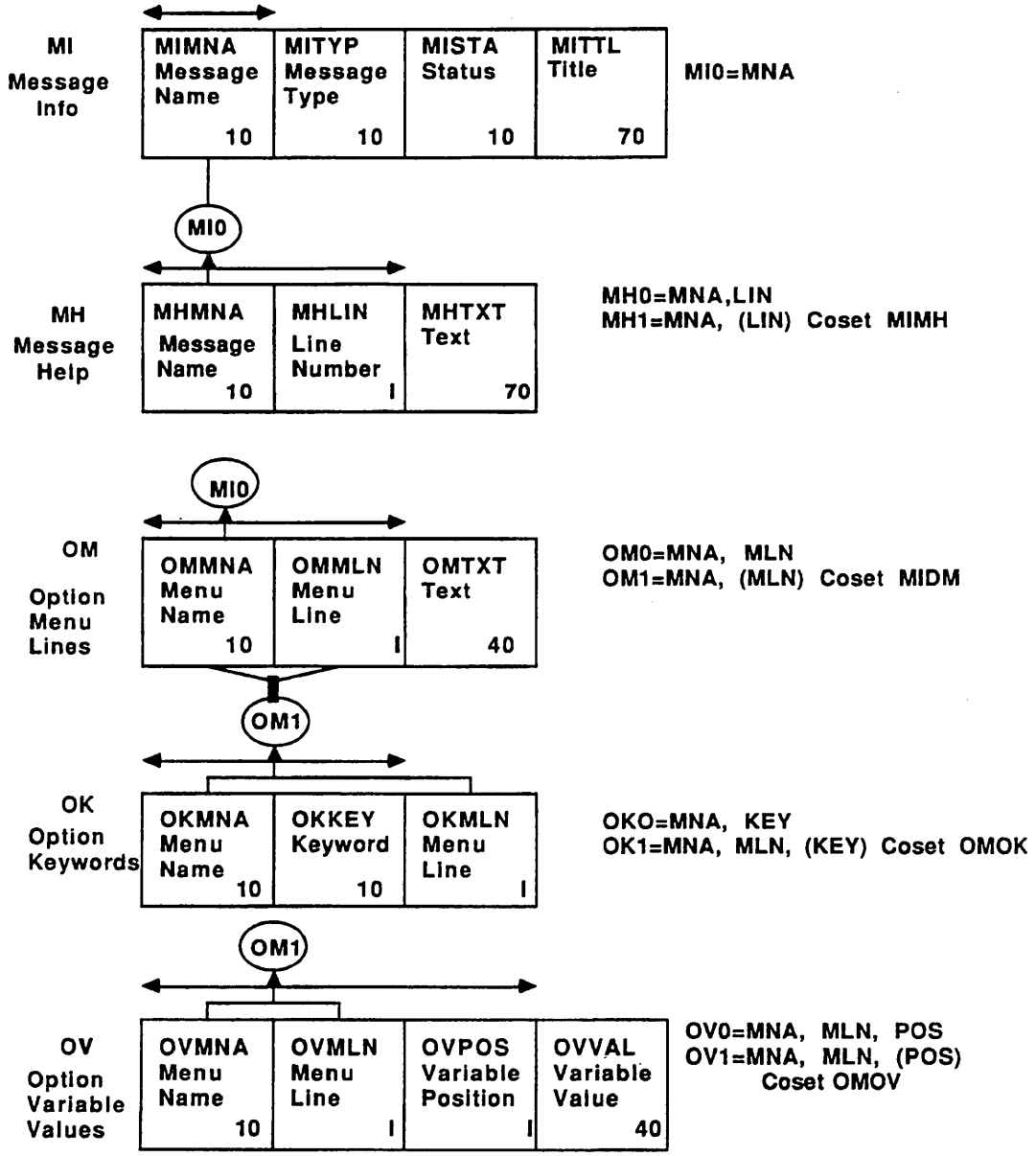
   Write a retrieval task for each type of data processed by your application. Include the name of the task in the ATTNA field when you define the data type, to enable EDL to retrieve the data with the RETRIEVE task.

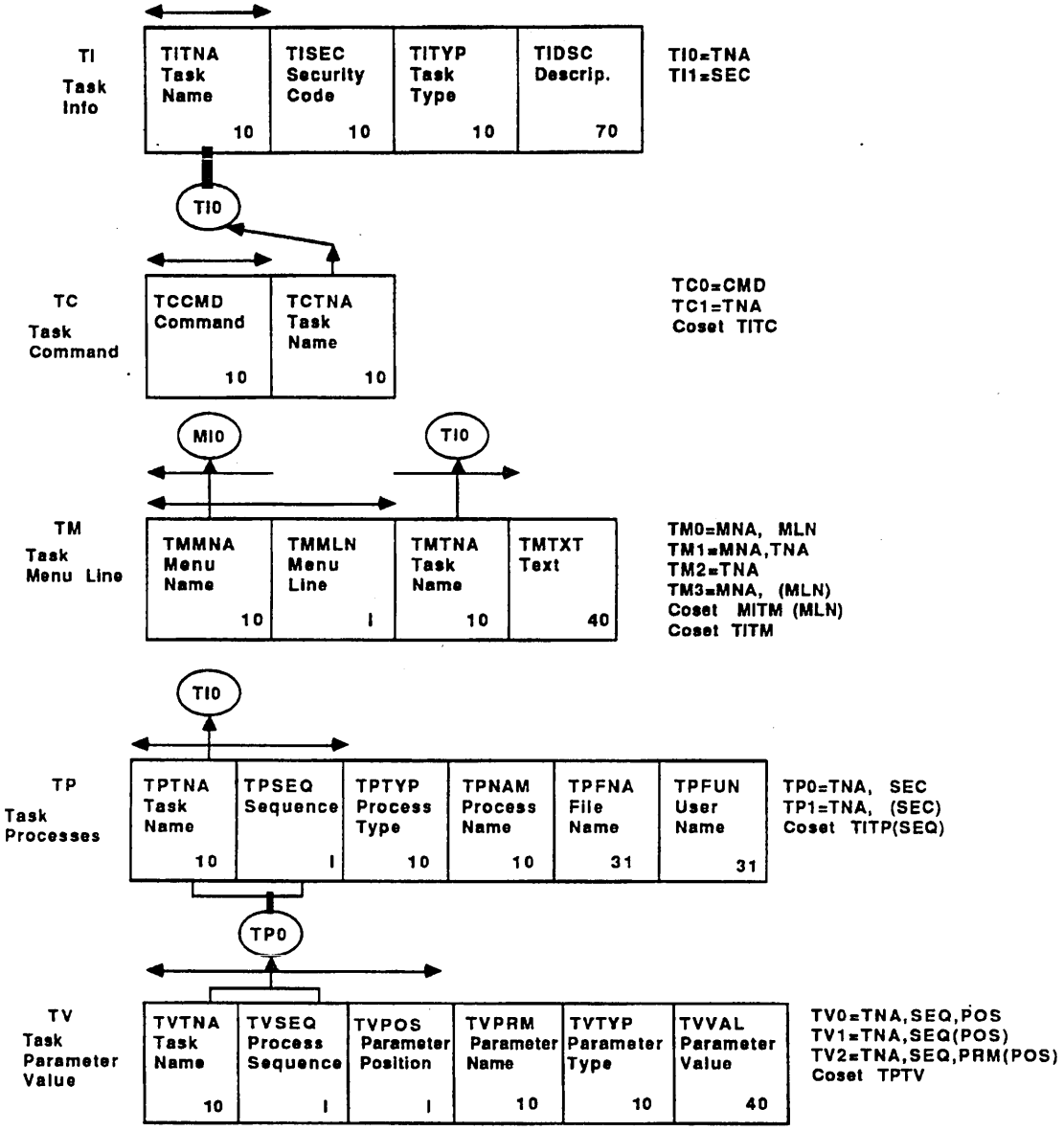
EDL Schema Definitions A

EDL Schema Definitions A

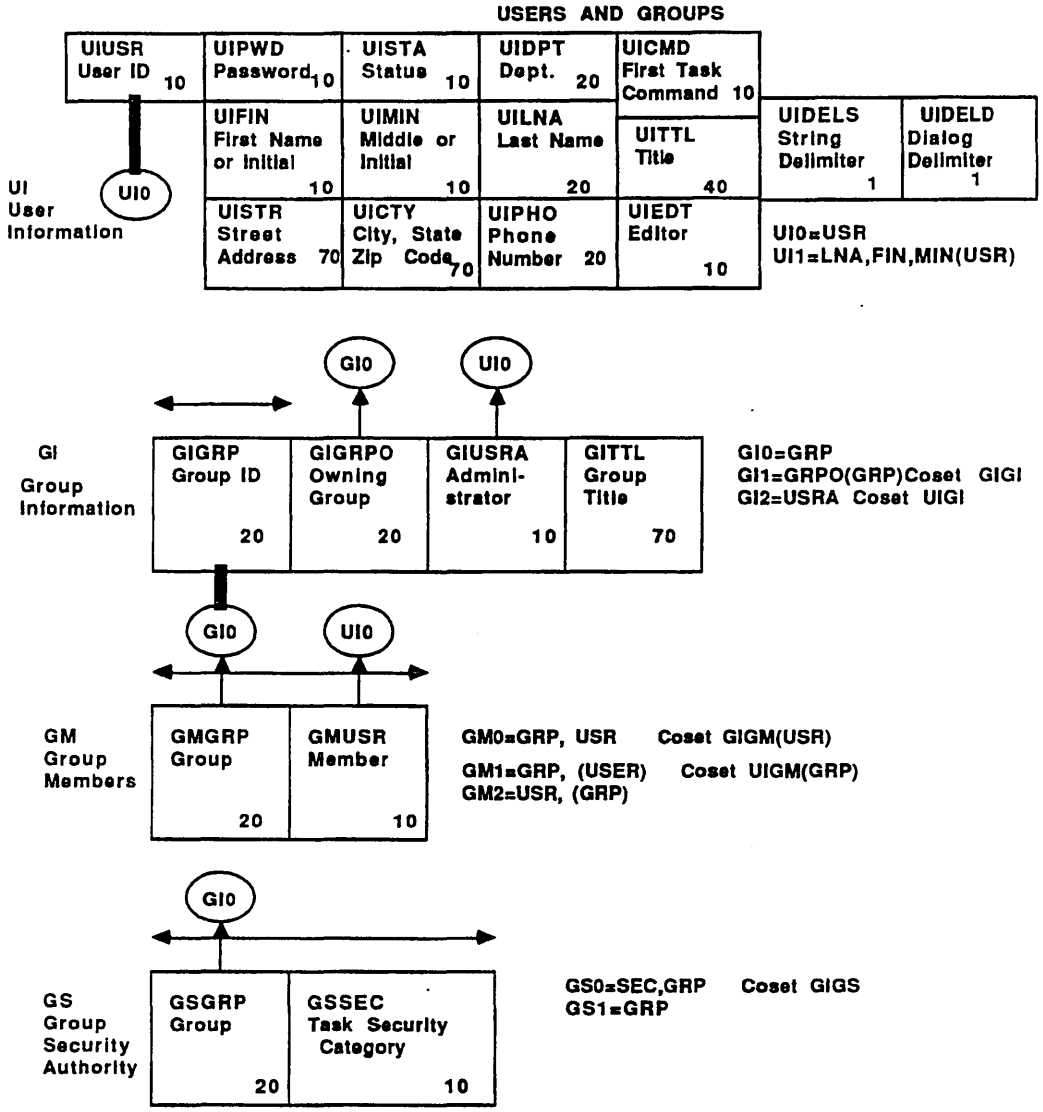
This appendix lists the external schemas for the EDL Message and Task Database (EDLMENUR and EDLMENUW) and the EDL Engineering Data Database (EDLDATAW and EDL).

EXTERNAL SCHEMAS EDLMENUR and EDLMENUW of CONCEPTUAL SCHEMA EDLMENU for EDL V1.2.5

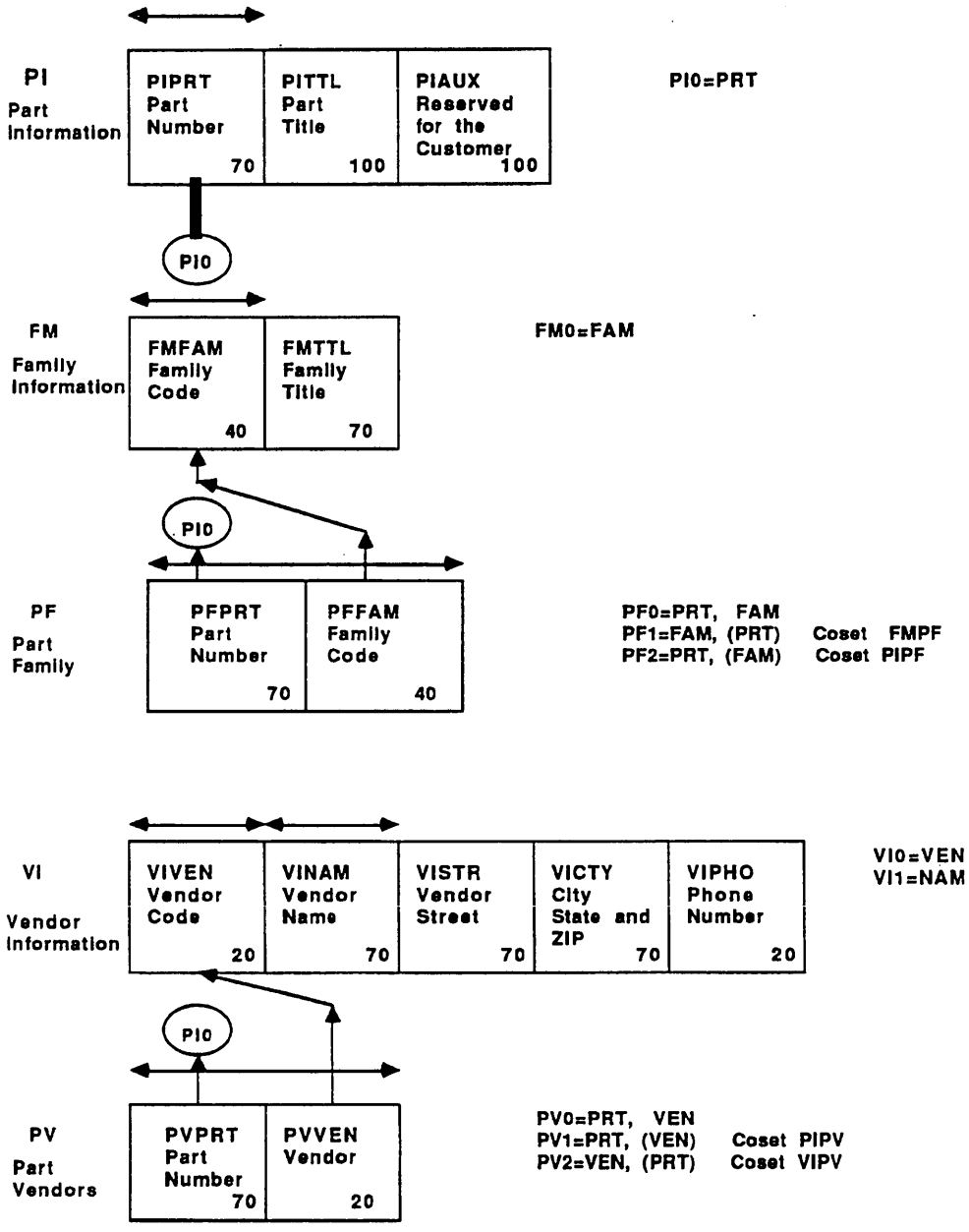



TI
Task
Info
TITNA
Task
Name
10
TISEC
Security
Code
10
TITYP
Task
Type
10
TIDSC
Descrip.
70
TI0=TNA
TI1=SEC
TI0
TC
Task
Command
TCCMD
Command
10
TCTNA
Task
Name
10
TC0=CMD
TC1=TNA
Coset TITC
MI0
TI0
TM
Task
Menu Line
TMMNA
Menu
Name
10
TMMLN
Menu
Line
1
TMTNA
Task
Name
10
TMTXT
Text
40
TM0=MNA, MLN
TM1=MNA,TNA
TM2=TNA
TM3=MNA, (MLN)
Coset MITM (MLN)
Coset TITM
TI0
TP
Task
Processes
TPTNA
Task
Name
10
TPSEQ
Sequence
1
TPTYP
Process
Type
10
TPNAM
Process
Name
10
TPFNA
File
Name
31
TPFUN
User
Name
31
TP0=TNA, SEC
TP1=TNA, (SEC)
Coset TITP(SEQ)
TP0
TV
Task
Parameter
Value
TVTNA
Task
Name
10
TVSEQ
Process
Sequence
1
TVPOS
Parameter
Position
1
TVPRM
Parameter
Name
10
TVTYP
Parameter
Type
10
TVVAL
Parameter
Value
40
TV0=TNA,SEQ,POS
TV1=TNA,SEQ(POS)
TV2=TNA,SEQ,PRM(POS)
Coset TPTV

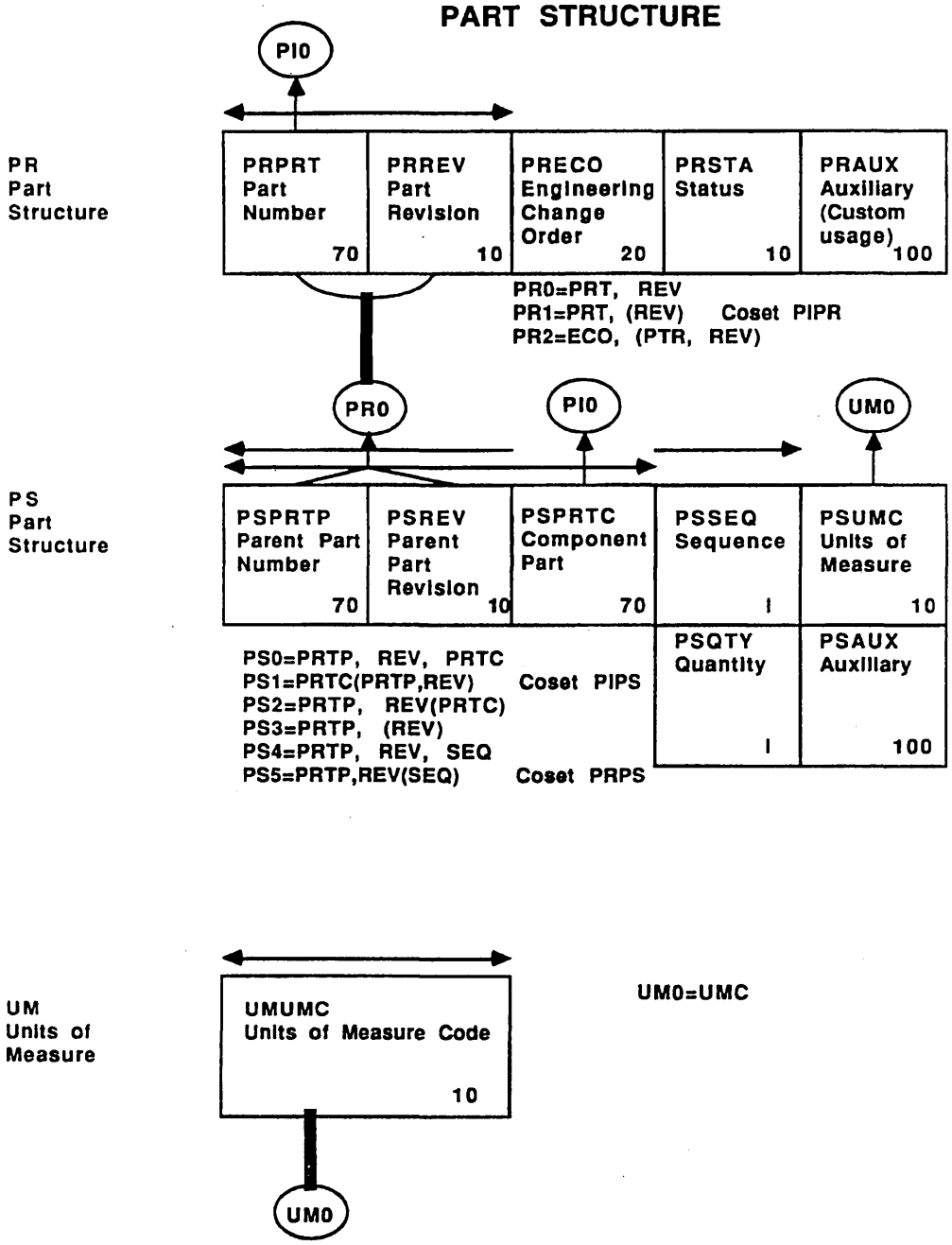
External Schemas EDLDATAW and EDL of
Conceptual Schema EDLDATA V1.2.5

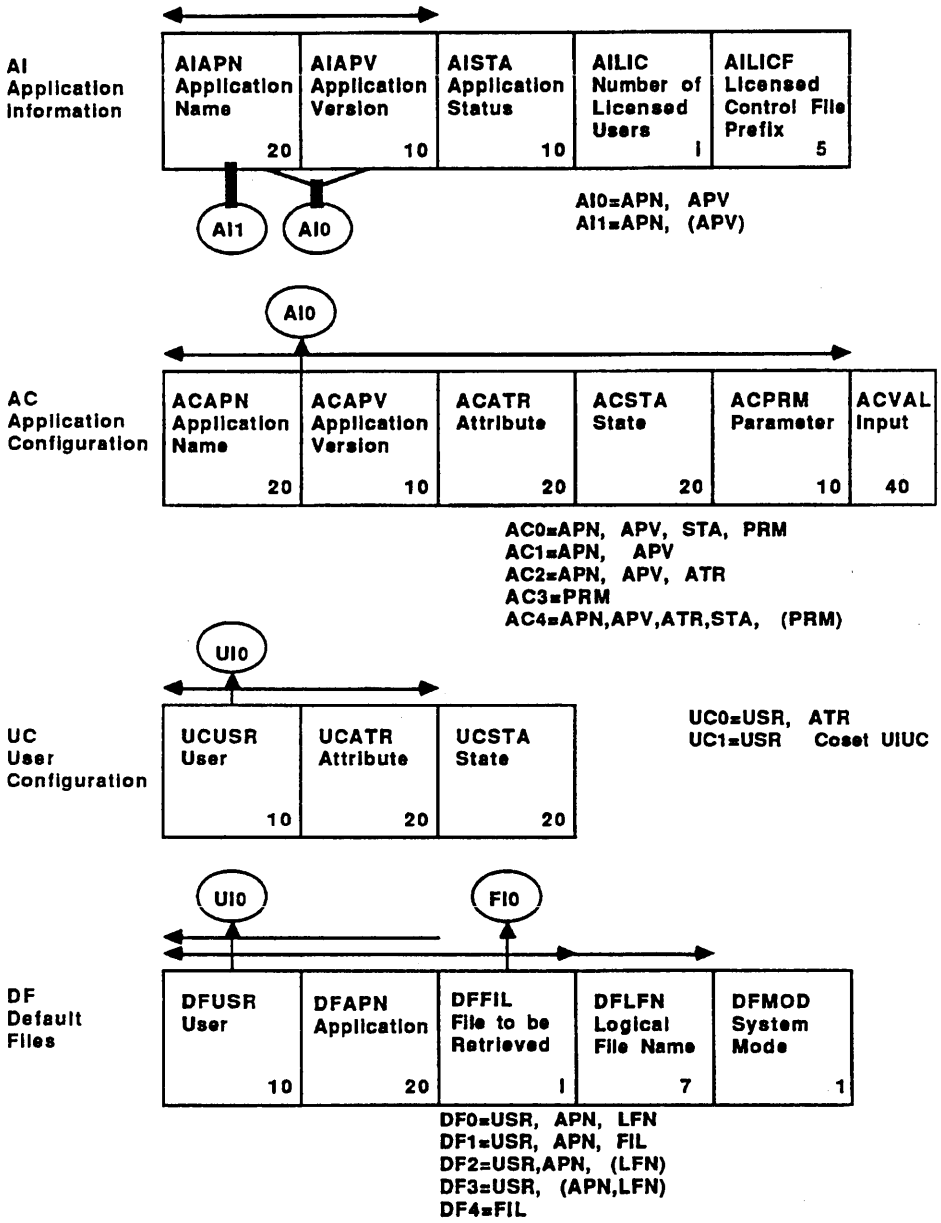


PARTS, FAMILIES, VENDORS

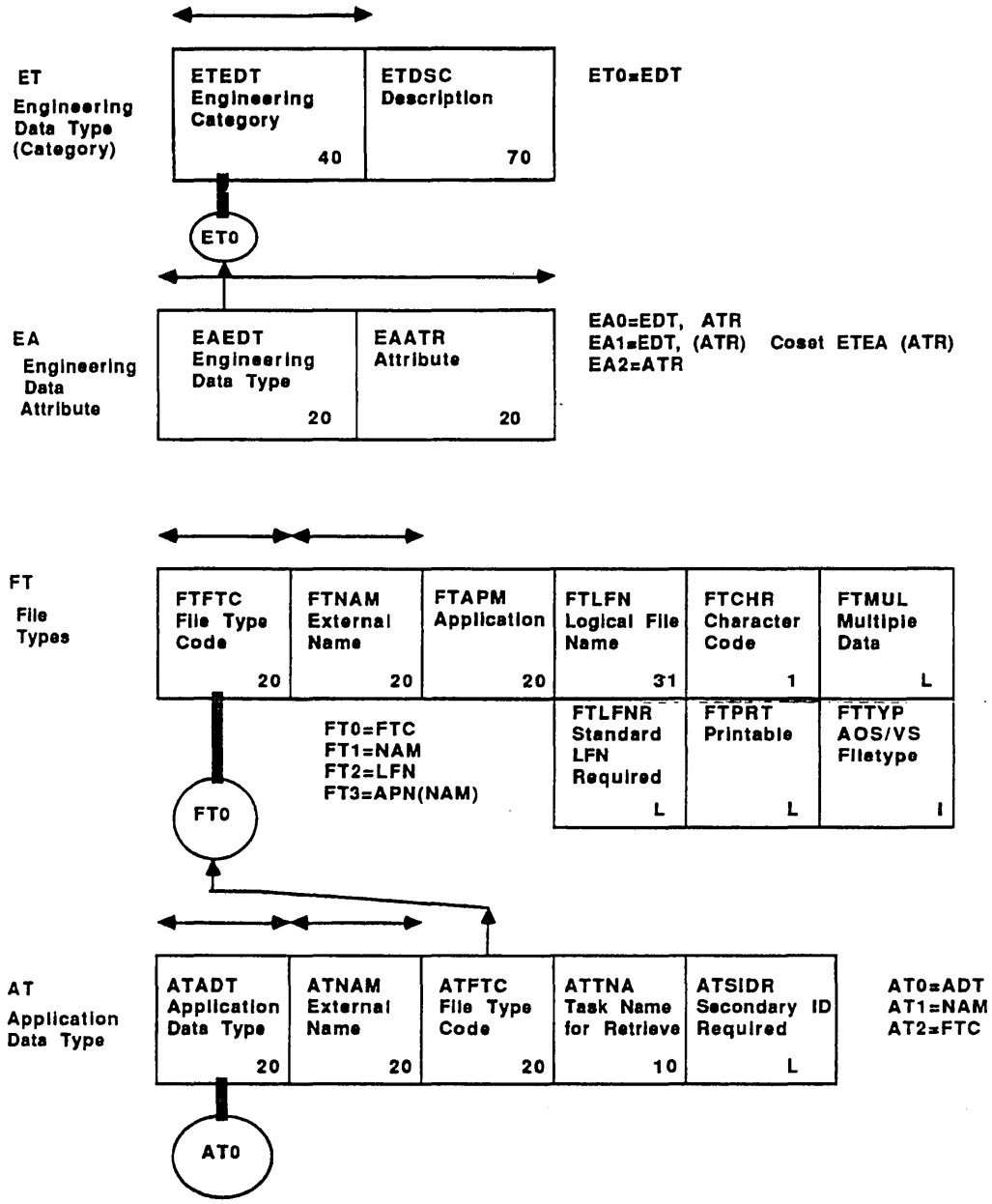


PART STRUCTURE

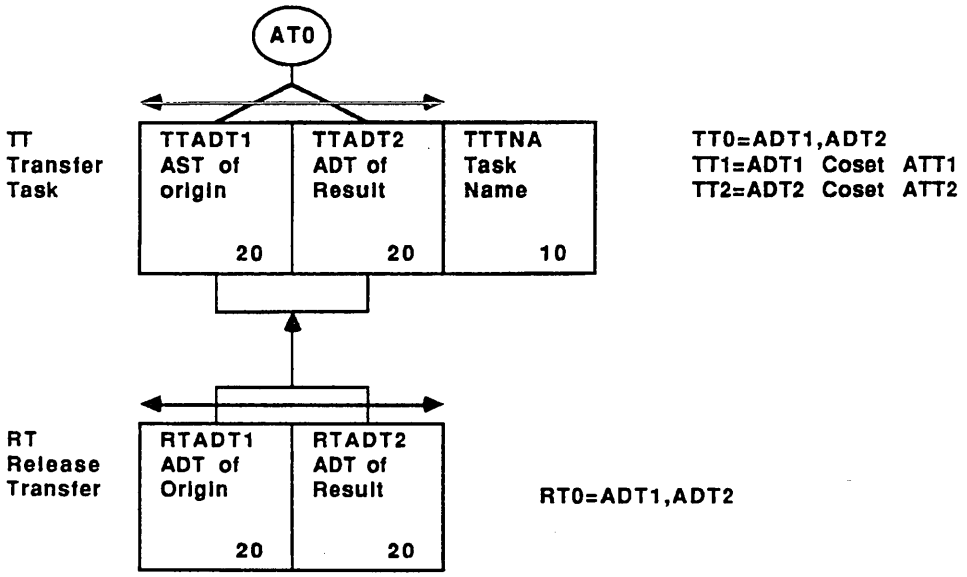



AI
Application
Information
AIAPN
Application
Name
20
AIAPV
Application
Version
10
AISTA
Application
Status
10
AILIC
Number of
Licensed
Users
1
AILICF
Licensed
Control File
Prefix
5
AI1
AI0
AI0=APN, APV
AI1=APN, (APV)
AI0
AC
Application
Configuration
ACAPN
Application
Name
20
ACAPV
Application
Version
10
ACATR
Attribute
20
ACSTA
State
20
ACPRM
Parameter
10
ACVAL
Input
40
AC0=APN, APV, STA, PRM
AC1=APN, APV
AC2=APN, APV, ATR
AC3=PRM
AC4=APN,APV,ATR,STA, (PRM)
UI0
UC
User
Configuration
UCUSR
User
10
UCATR
Attribute
20
UCSTA
State
20
UC0=USR, ATR
UC1=USR Coset UIUC
UI0
FI0
DF
Default
Files
DFUSR
User
10
DFAPN
Application
20
DFFIL
File to be
Retrieved
1
DFLFN
Logical
File Name
7
DFMOD
System
Mode
1
DF0=USR, APN, LFN
DF1=USR, APN, FIL
DF2=USR,APN, (LFN)
DF3=USR, (APN,LFN)
DF4=FIL

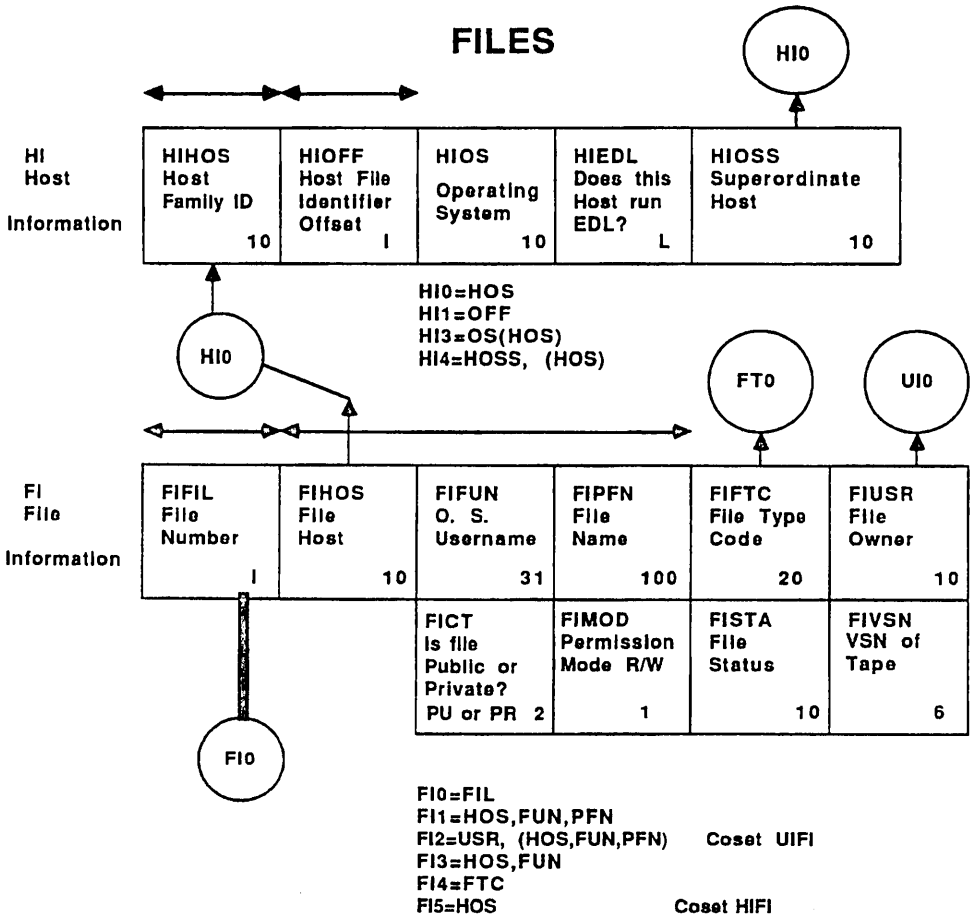
FILE AND DATA TYPES



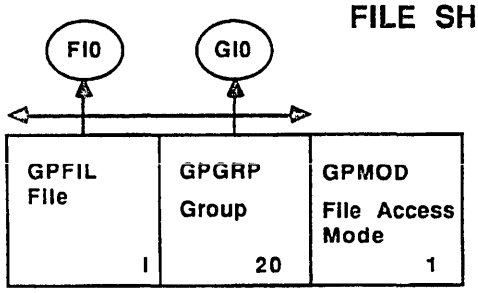
TRANSFER and TRANSLATE



FILES

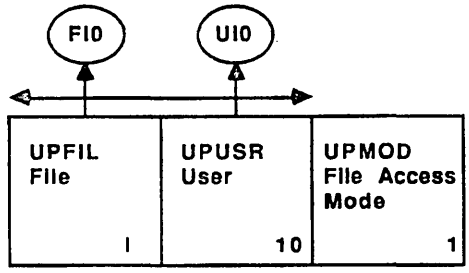

HI0
HI
Host
Information
HIHOS
Host
Family ID
10
HIOFF
Host File
Identifier
Offset
I
HIOS
Operating
System
10
HIEDL
Does this
Host run
EDL?
L
HIOSS
Superordinate
Host
10
HI0=HOS
HI1=OFF
HI3=OS(HOS)
HI4=HOSS, (HOS)
HI0
FT0
UI0
FI
File
Information
FIFIL
File
Number
I
FIHOS
File
Host
10
FIFUN
O. S.
Username
31
FIPFN
File
Name
100
FIFTC
File Type
Code
20
FIUSR
File
Owner
10
FICT
Is file
Public or
Private?
PU or PR 2
FIMOD
Permission
Mode R/W
1
FISTA
File
Status
10
FIVSN
VSN of
Tape
6
FI0
FI0=FIL
FI1=HOS,FUN,PFN
FI2=USR, (HOS,FUN,PFN) Coset UIFI
FI3=HOS,FUN
FI4=FTC
FI5=HOS Coset HIFI

FILE SHARING



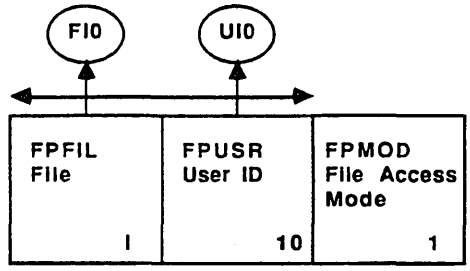
GP
Group
Permits

GP0=FIL, GRP
GP1=FIL (GRP) Coset FIGP
GP2=GRP Coset GIGP



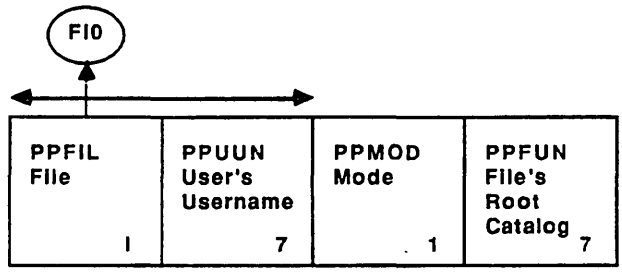
UP
User
Permits

UP0=FIL, USR
UP1=FIL, (USR) Coset FIUP
UP2=USR Coset UIUP



FP
File Permit
Result of
GP and UP

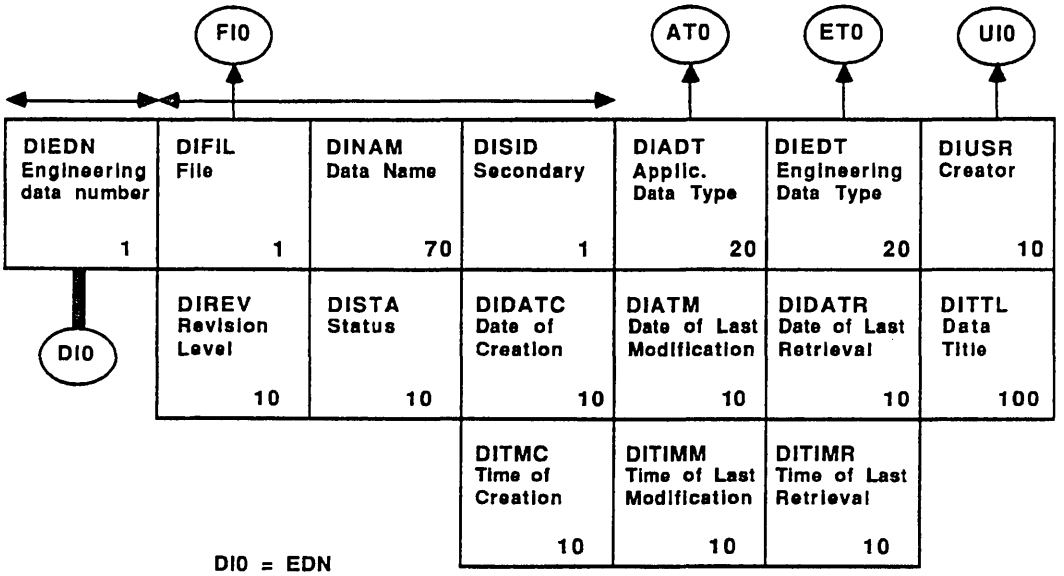
FP0=FIL,USR
FP1=FIL Coset FIFP
FP2=USR Coset UIFP



PP
Pending
Permits

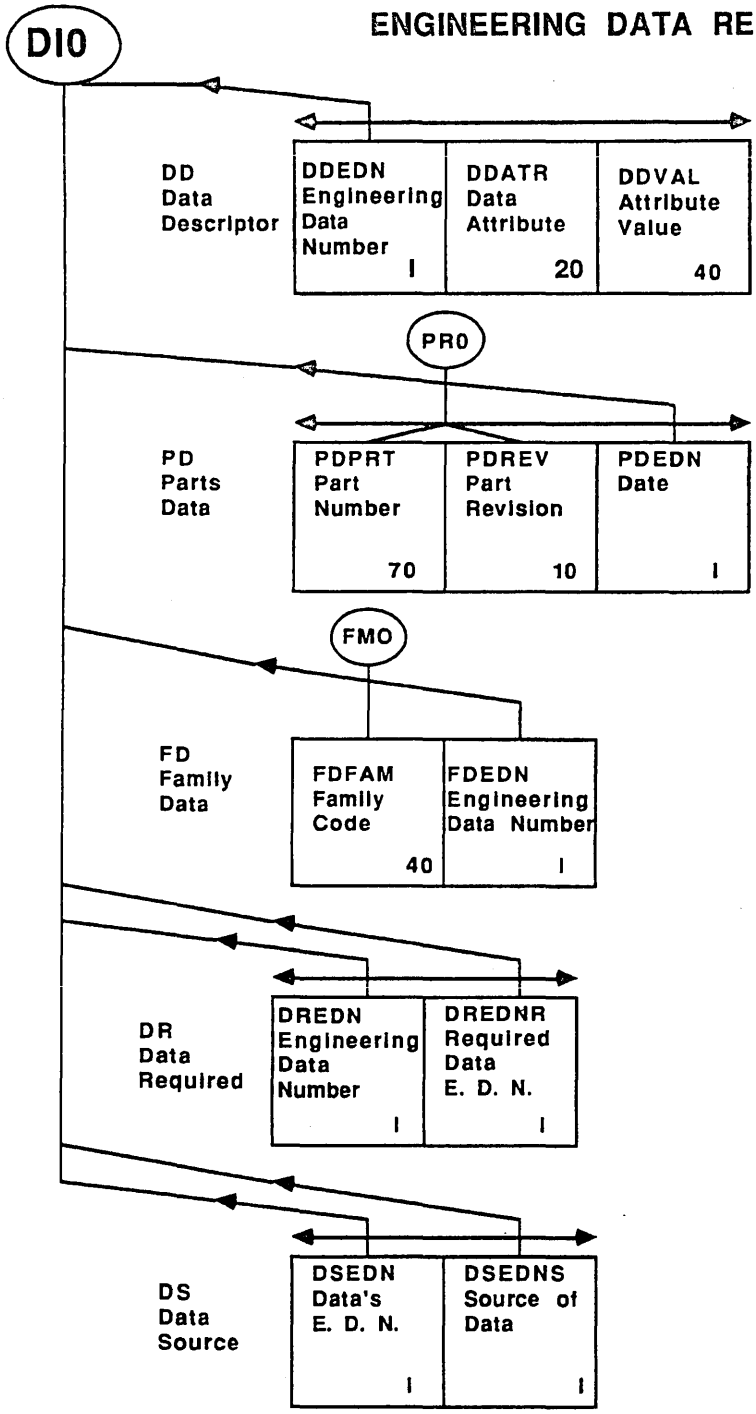
PP0=FIL, UUN
PP1=UUN
PP2=FUN
PP3=FIL

DI
Data
Information



DI0 = EDN
DI1=NAM, (SID, REV)
DI2=FIL, (NAM, SID, REV) Coset FIDI
DI3=NAM, SID, FIL
DI4=EDT Coset ETDI
DI5=ADT Coset ATDI
DI6=USR

ENGINEERING DATA RELATIONSHIPS



DD0=EDN,ATR,VAL
DD1=EDN, (ATR)
Coset DIDO
DD2=ATR, VAL
DD3=EDN, ATR

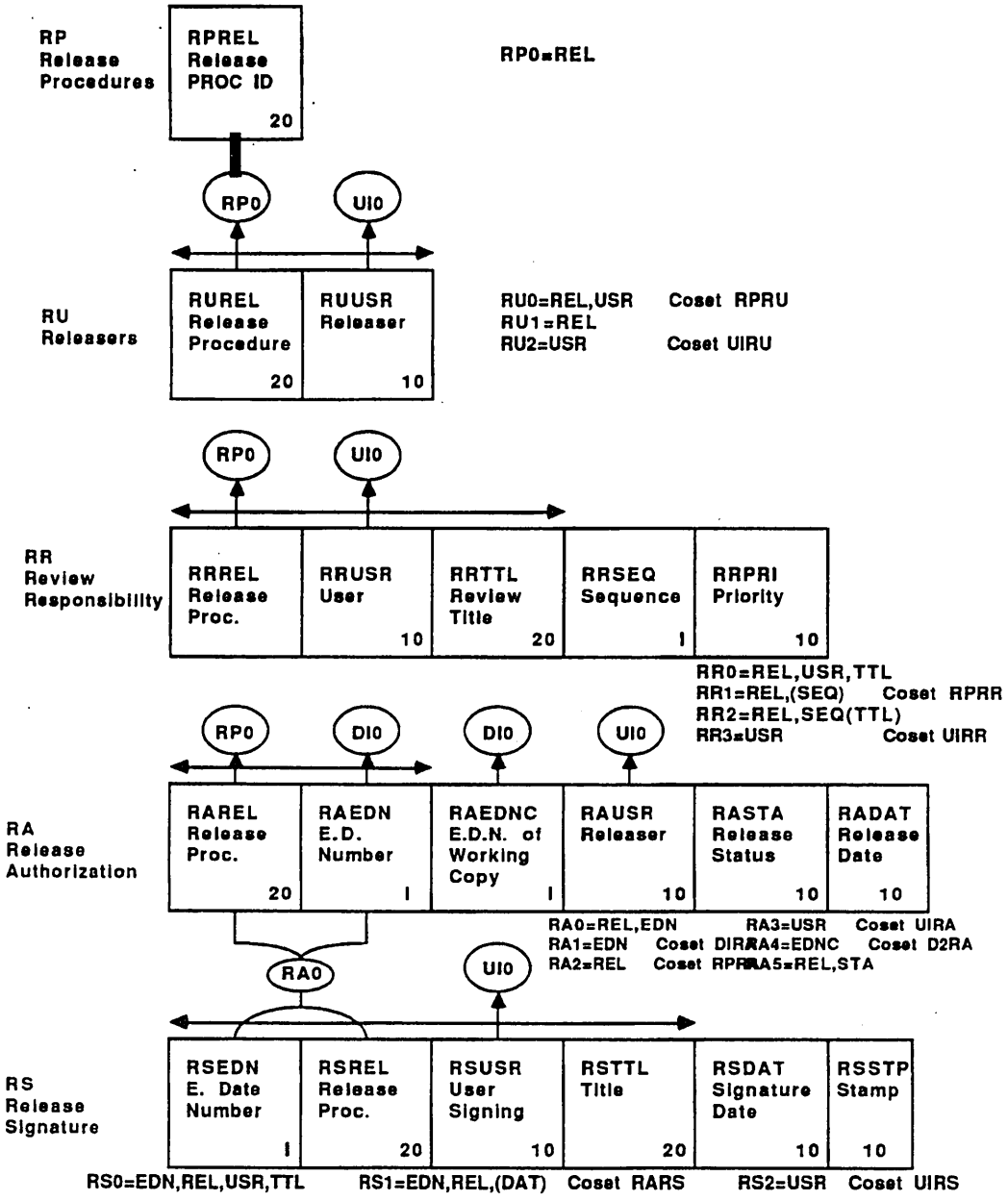
PD0=PRT, REV, EDN
Coset PRPD
PD1=PTR(REV)
Coset PIPD
PD2=EDN(PTR,REV)
Coset DIPD
PD3=PRT, REV

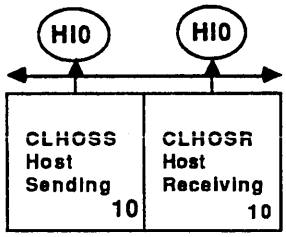
FD0=FAM,EDL
FD1=FAM Coset FMFD
FD2=EDN Coset DIFD

DR0=EDN, EDNC
DR1=EDN Coset DDR1
DR2=EDNR Coset DDR2

DS0=EDN, EDNS
DS1=EDN Coset DDS1
DS2=EDNS Coset DDS2

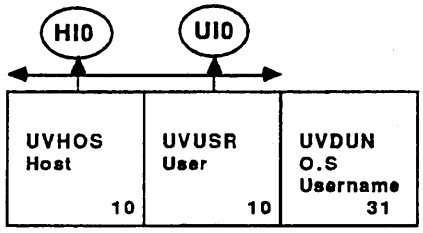
ENGINEERING DATA RELEASE





CL
Communication
Link

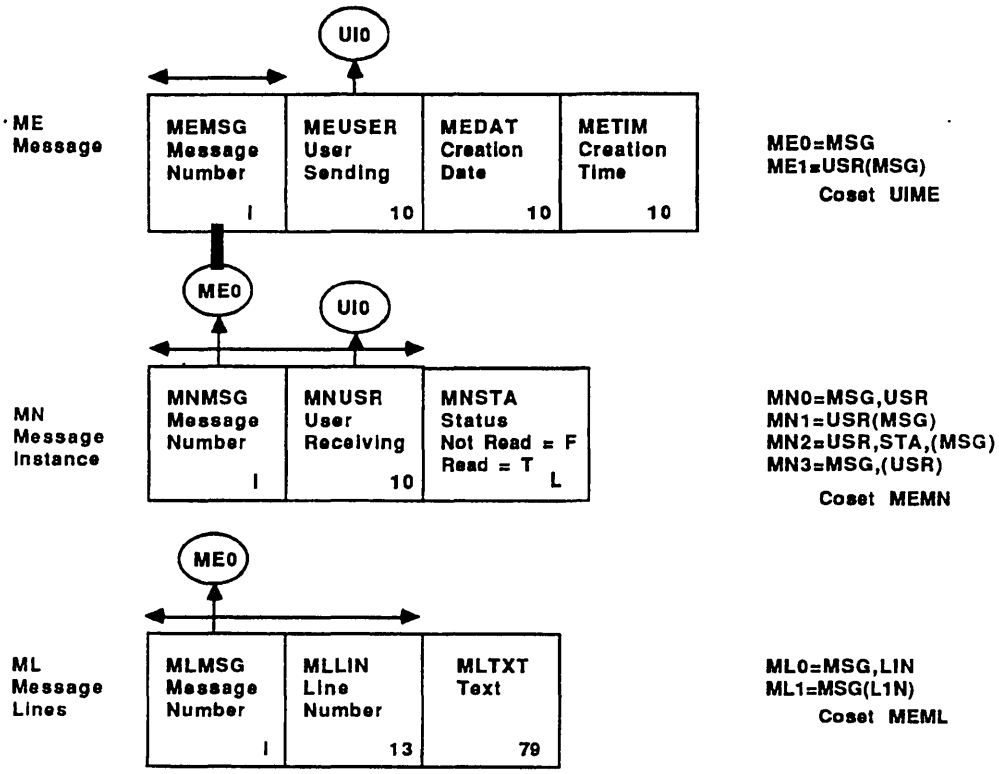
```
CL0=HOSS,HOSR
CL1=HOSS(HOSR)
CL2=HORS(HOSS)
```



UV
User
Validation

```
UV0=HOS,USR
UV1=HOS,USR,OUN
UV2=HOS,OUN
UV3=USR(HOST)     COSET UIUV
UV4=HOS(USR)      COSET HIUV
```

NOTE FACILITY



Information Base Routines B

Information Base Routines B

This appendix lists the EDL Information Base Routines sorted by table type. The routines are described in psuedo-SQL-DML format; the WHERE clause indicates which columns are key to the access, the ORDER BY clause indicates method of sorting the result set.

Application Configuration (AC) Routines

The AC table contains terminal configuration parameters that can be passed to an application.

IBSAC

```
STORE A NEW ROW IN TABLE AC.
        INSERT INTO AC IN ENGINEERING_DATA_DATABASE
        SET ACAPN = :ACAPN,
            ACAPV = :ACAPV,
            ACATR = :ACATR,
            ACSTA = :ACSTA,
            ACPRM = :ACPRM,
            ACVAL = :ACVAL
```

IBMAC

```
MODIFY AN EXISTING ROW IN TABLE AC.
        UDPATE AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV AND
              ACATR = :ACATR AND
              ACSTA = :ACSTA AND
              ACPRM = :ACPRM
        SET ACAPN = :ACAPN,
            ACAPV = :ACAPV,
            ACATR = :ACATR,
            ACSTA = :ACSTA,
            ACPRM = :ACPRM,
            ACVAL = :ACVAL
```

IBDAC

```
DELETE AN EXISTING ROW IN TABLE AC.
        DELETE FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV AND
              ACATR = :ACATR AND
              ACSTA = :ACSTA AND
              ACPRM = :ACPRM
```

IBOAC0

```
OBTAIN A ROW IN TABLE AC VIA ACCESS PATH AC0.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV AND
              ACATR = :ACATR AND
              ACSTA = :ACSTA AND
              ACPRM = :ACPRM
         ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBOAC1

```
OBTAIN A ROW IN TABLE AC VIA ACCESS PATH AC1.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV
        ORDER BY ACAPN ASC, ACAPV ASC
```

IBOAC2

```
OBTAIN A ROW IN TABLE AC VIA ACCESS PATH AC2.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV AND
              ACATR = :ACATR
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC
```

IBOAC3

```
OBTAIN A ROW IN TABLE AC VIA ACCESS PATH AC3.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACPRM = :ACPRM
        ORDER BY ACPRM ASC
```

IBOAC4

```
OBTAIN A ROW IN TABLE AC VIA ACCESS PATH AC4.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACAPN = :ACAPN AND
              ACAPV = :ACAPV AND
              ACATR = :ACATR AND
              ACSTA = :ACSTA
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBAAC0

```
OBTAIN A ROW IN TABLE AC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AC0.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE (ACAPN > :ACAPN)
          OR ((ACAPN = :ACAPN) AND (ACAPV > :ACAPV))
          OR ((ACAPN = :ACAPN AND ACAPV = :ACAPV) AND (ACATR > :ACATR))
          OR ((ACAPN  = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR) AND
              (ACSTA > :ACSTA))
          OR ((ACAPN = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR AND
              ACSTA = :ACSTA) AND (ACPRM > :ACPRM))
          OR (ACAPN = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR AND
              ACSTA = :ACSTA AND ACPRM = :ACPRM)
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBAAC1

```
OBTAIN A ROW IN TABLE AC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AC1.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE (ACAPN > :ACAPN)
          OR ((ACAPN = :ACAPN) AND (ACAPV > :ACAPV))
          OR (ACAPN = :ACAPN AND ACAPV = :ACAPV)
        ORDER BY ACAPN ASC, ACAPV ASC
```

IBAAC2

```
OBTAIN A ROW IN TABLE AC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AC2.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE (ACAPN > :ACAPN)
          OR ((ACAPN = :ACAPN) AND (ACAPV > :ACAPV))
          OR ((ACAPN = :ACAPN AND ACAPV = :ACAPV) AND (ACATR > :ACATR))
          OR (ACAPN = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR)
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC
```

IBAAC3

```
OBTAIN A ROW IN TABLE AC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AC3.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE ACPRM >= :ACPRM
        ORDER BY ACPRM ASC
```

IBAAC4

```
OBTAIN A ROW IN TABLE AC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AC4.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        WHERE (ACAPN > :ACAPN)
          OR ((ACAPN = :ACAPN) AND (ACAPV > :ACAPV))
          OR ((ACAPN = :ACAPN AND ACAPV = :ACAPV) AND (ACATR > :ACATR))
          OR  ((ACAPN  = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR) AND
              (ACSTA > :ACSTA))
          OR (ACAPN = :ACAPN AND ACAPV = :ACAPV AND ACATR = :ACATR AND
              ACSTA = :ACSTA)
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBEAC1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AC VIA ACCESS PATH AC1.
SAVE THE CURRENT POSITION IN TABLE AC.
FETCH THE NEXT ROW FROM TABLE AC.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBEAC2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AC VIA ACCESS PATH AC2.
SAVE THE CURRENT POSITION IN TABLE AC.
FETCH THE NEXT ROW FROM TABLE AC.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBEAC3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AC VIA ACCESS PATH AC3.
SAVE THE CURRENT POSITION IN TABLE AC.
FETCH THE NEXT ROW FROM TABLE AC.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBEAC4

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AC VIA ACCESS PATH AC4.
SAVE THE CURRENT POSITION IN TABLE AC.
FETCH THE NEXT ROW FROM TABLE AC.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBFAC0

```
OBTAIN THE FIRST ROW OF TABLE AC, ORDERED BY ACCESS PATH AC0.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
         ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBFAC1

```
OBTAIN THE FIRST ROW OF TABLE AC, ORDERED BY ACCESS PATH AC1.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        ORDER BY ACAPN ASC, ACAPV ASC
```

IBFAC2

```
OBTAIN THE FIRST ROW OF TABLE AC, ORDERED BY ACCESS PATH AC2.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC
```

IBFAC3

```
OBTAIN THE FIRST ROW OF TABLE AC, ORDERED BY ACCESS PATH AC3.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        ORDER BY ACPRM ASC
```

IBFAC4

```
OBTAIN THE FIRST ROW OF TABLE AC, ORDERED BY ACCESS PATH AC4.
        SELECT ACAPN, ACAPV, ACATR, ACSTA, ACPRM, ACVAL
        FROM AC IN ENGINEERING_DATA_DATABASE
        ORDER BY ACAPN ASC, ACAPV ASC, ACATR ASC, ACSTA ASC, ACPRM ASC
```

IBNAC0

```
OBTAIN THE NEXT ROW OF TABLE AC, ORDERED BY ACCESS PATH AC0.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBNAC1

```
OBTAIN THE NEXT ROW OF TABLE AC, ORDERED BY ACCESS PATH AC1.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBNAC2

```
OBTAIN THE NEXT ROW OF TABLE AC, ORDERED BY ACCESS PATH AC2.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBNAC3

```
OBTAIN THE NEXT ROW OF TABLE AC, ORDERED BY ACCESS PATH AC3.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

IBNAC4

```
OBTAIN THE NEXT ROW OF TABLE AC, ORDERED BY ACCESS PATH AC4.
        SET :ACAPN, :ACAPV, :ACATR, :ACSTA, :ACPRM, :ACVAL
```

Application Information (AI) Routines

The AI table defines applications under control of EDL.

IBSAI

```
STORE A NEW ROW IN TABLE AI.
        INSERT INTO AI IN ENGINEERING_DATA_DATABASE
        SET AIAPN = :AIAPN,
            AIAPV = :AIAPV,
            AISTA = :AISTA,
            AILIC = :AILIC,
            AILICF = :AILICF
```

IBMAI

```
MODIFY AN EXISTING ROW IN TABLE AI.
        UDPATE AI IN ENGINEERING_DATA_DATABASE
        WHERE AIAPN = :AIAPN AND
              AIAPV = :AIAPV
        SET AIAPN = :AIAPN,
            AIAPV = :AIAPV,
            AISTA = :AISTA,
            AILIC = :AILIC,
            AILICF = :AILICF
```

IBDAI

```
DELETE AN EXISTING ROW IN TABLE AI.
        DELETE FROM AI IN ENGINEERING_DATA_DATABASE
        WHERE AIAPN = :AIAPN AND
              AIAPV = :AIAPV
```

IBOAI0

```
OBTAIN A ROW IN TABLE AI VIA ACCESS PATH AI0.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        WHERE AIAPN = :AIAPN AND
              AIAPV = :AIAPV
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBOAI1

```
OBTAIN A ROW IN TABLE AI VIA ACCESS PATH AI1.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        WHERE AIAPN = :AIAPN
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBAAI0

```
OBTAIN A ROW IN TABLE AI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AI0.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        WHERE (AIAPN > :AIAPN)
          OR ((AIAPN = :AIAPN) AND (AIAPV > :AIAPV))
          OR (AIAPN = :AIAPN AND AIAPV = :AIAPV)
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBAAI1

```
OBTAIN A ROW IN TABLE AI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AI1.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        WHERE AIAPN >= :AIAPN
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBEAI1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AI VIA ACCESS PATH AI1.
SAVE THE CURRENT POSITION IN TABLE AI.
FETCH THE NEXT ROW FROM TABLE AI.
        SET :AIAPN, :AIAPV, :AISTA, :AILIC, :AILICF
```

IBFAI0

```
OBTAIN THE FIRST ROW OF TABLE AI, ORDERED BY ACCESS PATH AI0.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBFAI1

```
OBTAIN THE FIRST ROW OF TABLE AI, ORDERED BY ACCESS PATH AI1.
        SELECT AIAPN, AIAPV, AISTA, AILIC, AILICF
        FROM AI IN ENGINEERING_DATA_DATABASE
        ORDER BY AIAPN ASC, AIAPV ASC
```

IBNAI0

```
OBTAIN THE NEXT ROW OF TABLE AI, ORDERED BY ACCESS PATH AI0.
        SET :AIAPN, :AIAPV, :AISTA, :AILIC, :AILICF
```

IBNAI1

```
OBTAIN THE NEXT ROW OF TABLE AI, ORDERED BY ACCESS PATH AI1.
        SET :AIAPN, :AIAPV, :AISTA, :AILIC, :AILICF
```

Application Data Type (AT) Routines

The AT table defines the types of data controlled by EDL.

IBSAT

```
STORE A NEW ROW IN TABLE AT.
        INSERT INTO AT IN ENGINEERING_DATA_DATABASE
        SET ATADT = :ATADT,
            ATNAM = :ATNAM,
            ATFTC = :ATFTC,
            ATTNA = :ATTNA,
            ATSIDR = :ATSIDR
```

IBMAT

```
MODIFY AN EXISTING ROW IN TABLE AT.
        UDPATE AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :ATADT
        SET ATADT = :ATADT,
            ATNAM = :ATNAM,
            ATFTC = :ATFTC,
            ATTNA = :ATTNA,
            ATSIDR = :ATSIDR
```

IBDAT

```
DELETE AN EXISTING ROW IN TABLE AT.
        DELETE FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :ATADT
```

IBOAT0

```
OBTAIN A ROW IN TABLE AT VIA ACCESS PATH AT0.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :ATADT
        ORDER BY ATADT ASC
```

IBOAT1

```
OBTAIN A ROW IN TABLE AT VIA ACCESS PATH AT1.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATNAM = :ATNAM
        ORDER BY ATNAM ASC
```

IBOAT2

```
OBTAIN A ROW IN TABLE AT VIA ACCESS PATH AT2.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATFTC = :ATFTC
        ORDER BY ATFTC ASC
```

IBAAT0

```
OBTAIN A ROW IN TABLE AT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AT0.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT >= :ATADT
        ORDER BY ATADT ASC
```

IBAAT1

```
OBTAIN A ROW IN TABLE AT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AT1.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATNAM >= :ATNAM
        ORDER BY ATNAM ASC
```

IBAAT2

```
OBTAIN A ROW IN TABLE AT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH AT2.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATFTC >= :ATFTC
        ORDER BY ATFTC ASC
```

IBEAT2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE AT VIA ACCESS PATH AT2.
SAVE THE CURRENT POSITION IN TABLE AT.
FETCH THE NEXT ROW FROM TABLE AT.
        SET :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
```

IBFAT0

```
OBTAIN THE FIRST ROW OF TABLE AT, ORDERED BY ACCESS PATH AT0.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        ORDER BY ATADT ASC
```

IBFAT1

```
OBTAIN THE FIRST ROW OF TABLE AT, ORDERED BY ACCESS PATH AT1.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        ORDER BY ATNAM ASC
```

IBFAT2

```
OBTAIN THE FIRST ROW OF TABLE AT, ORDERED BY ACCESS PATH AT2.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        ORDER BY ATFTC ASC
```

IBNAT0

```
OBTAIN THE NEXT ROW OF TABLE AT, ORDERED BY ACCESS PATH AT0.
        SET :ATADT,. :ATNAM, :ATFTC, :ATTNA, :ATSIDR
```

IBNAT1

```
OBTAIN THE NEXT ROW OF TABLE AT, ORDERED BY ACCESS PATH AT1.
        SET :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
```

IBNAT2

```
OBTAIN THE NEXT ROW OF TABLE AT, ORDERED BY ACCESS PATH AT2.
        SET :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
```

IBOFTAT

```
USING COSET FTAT, OBTAIN THE ROW FROM TABLE FT THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE AT
         SELECT  FTFTC,  FTNAM,  FTAPN,  FTLFN,  FTCHR,  FTMUL, FTLFNR,
FTPRT,
           FTTYP
         INTO :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
             :FTPRT, :FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTFTC = :ATFTC
```

IBFFTAT

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE AT WITHIN COSET FTAT, USING
ACCESS PATH AT2.
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATFTC = :FTFTC
        ORDER BY ATFTC ASC
```

IBNFTAT

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE AT WITHIN COSET FTAT.
        SET :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
```

Data Descriptor (DD) Routines

The DD table contains attributes and values of data descriptors.

IBSDD

```
STORE A NEW ROW IN TABLE DD.
        INSERT INTO DD IN ENGINEERING_DATA_DATABASE
        SET DDEDN = :DDEDN,
            DDATR = :DDATR,
            DDVAL = :DDVAL
```

IBMDD

```
MODIFY AN EXISTING ROW IN TABLE DD.
        UDPATE DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DDEDN AND
              DDATR = :DDATR AND
              DDVAL = :DDVAL
        SET DDEDN = :DDEDN,
            DDATR = :DDATR,
            DDVAL = :DDVAL
```

IBDDD

```
DELETE AN EXISTING ROW IN TABLE DD.
        DELETE FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DDEDN AND
              DDATR = :DDATR AND
              DDVAL = :DDVAL
```

IBODD0

```
OBTAIN A ROW IN TABLE DD VIA ACCESS PATH DD0.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DDEDN AND
              DDATR = :DDATR AND
              DDVAL = :DDVAL
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBODD1

```
OBTAIN A ROW IN TABLE DD VIA ACCESS PATH DD1.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DDEDN
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBODD2

```
OBTAIN A ROW IN TABLE DD VIA ACCESS PATH DD2.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDATR = :DDATR AND
              DDVAL = :DDVAL
        ORDER BY DDATR ASC, DDVAL ASC
```

IBODD3

```
OBTAIN A ROW IN TABLE DD VIA ACCESS PATH DD3.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DDEDN AND
              DDATR = :DDATR
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBADD0

```
OBTAIN A ROW IN TABLE DD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DD0.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE (DDEDN > :DDEDN)
          OR ((DDEDN = :DDEDN) AND (DDATR > :DDATR))
          OR ((DDEDN = :DDEDN AND DDATR = :DDATR) AND (DDVAL > :DDVAL))
          OR (DDEDN = :DDEDN AND DDATR = :DDATR AND DDVAL = :DDVAL)
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBADD1

```
OBTAIN A ROW IN TABLE DD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DD1.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN >= :DDEDN
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBADD2

```
OBTAIN A ROW IN TABLE DD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DD2.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE (DDATR > :DDATR)
          OR ((DDATR = :DDATR) AND (DDVAL > :DDVAL))
          OR (DDATR = :DDATR AND DDVAL = :DDVAL)
        ORDER BY DDATR ASC, DDVAL ASC
```

IBADD3

```
OBTAIN A ROW IN TABLE DD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DD3.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE (DDEDN > :DDEDN)
          OR ((DDEDN = :DDEDN) AND (DDATR > :DDATR))
          OR (DDEDN = :DDEDN AND DDATR = :DDATR)
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBEDD1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DD VIA ACCESS PATH DD1.
SAVE THE CURRENT POSITION IN TABLE DD.
FETCH THE NEXT ROW FROM TABLE DD.
        SET :DDEDN, :DDATR, :DDVAL
```

IBEDD2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DD VIA ACCESS PATH DD2.
SAVE THE CURRENT POSITION IN TABLE DD.
FETCH THE NEXT ROW FROM TABLE DD.
        SET :DDEDN, :DDATR, :DDVAL
```

IBEDD3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DD VIA ACCESS PATH DD3.
SAVE THE CURRENT POSITION IN TABLE DD.
FETCH THE NEXT ROW FROM TABLE DD.
        SET :DDEDN, :DDATR, :DDVAL
```

IBFDD0

```
OBTAIN THE FIRST ROW OF TABLE DD, ORDERED BY ACCESS PATH DD0.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBFDD1

```
OBTAIN THE FIRST ROW OF TABLE DD, ORDERED BY ACCESS PATH DD1.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBFDD2

```
OBTAIN THE FIRST ROW OF TABLE DD, ORDERED BY ACCESS PATH DD2.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        ORDER BY DDATR ASC, DDVAL ASC
```

IBFDD3

```
OBTAIN THE FIRST ROW OF TABLE DD, ORDERED BY ACCESS PATH DD3.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBNDD0

```
OBTAIN THE NEXT ROW OF TABLE DD, ORDERED BY ACCESS PATH DD0.
        SET :DDEDN, :DDATR, :DDVAL
```

IBNDD1

```
OBTAIN THE NEXT ROW OF TABLE DD, ORDERED BY ACCESS PATH DD1.
        SET :DDEDN, :DDATR, :DDVAL
```

IBNDD2

```
OBTAIN THE NEXT ROW OF TABLE DD, ORDERED BY ACCESS PATH DD2.
        SET :DDEDN, :DDATR, :DDVAL
```

IBNDD3

```
OBTAIN THE NEXT ROW OF TABLE DD, ORDERED BY ACCESS PATH DD3.
        SET :DDEDN, :DDATR, :DDVAL
```

IBODIDD

```
USING COSET DIDD, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DD
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DDEDN
```

IBFDIDD

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DD WITHIN COSET DIDD, USING
ACCESS PATH DD1.
        SELECT DDEDN, DDATR, DDVAL
        FROM DD IN ENGINEERING_DATA_DATABASE
        WHERE DDEDN = :DIEDN
        ORDER BY DDEDN ASC, DDATR ASC, DDVAL ASC
```

IBNDIDD

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DD WITHIN COSET DIDD.
        SET :DDEDN, :DDATR, :DDVAL
```

Default Files (DF) Routines

The DF table contains the logical and actual file name pairs that are passed as parameters to an application. EDL users can specify their own set of such pairs for every application.

IBSDF

```
STORE A NEW ROW IN TABLE DF.
        INSERT INTO DF IN ENGINEERING_DATA_DATABASE
        SET DFUSR = :DFUSR,
            DFAPN = :DFAPN,
            DFFIL = :DFFIL,
            DFMOD = :DFMOD,
            DFLFN = :DFLFN
```

IBMDF

```
MODIFY AN EXISTING ROW IN TABLE DF.
        UDPATE DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :DFUSR AND
              DFAPN = :DFAPN AND
              DFLFN = :DFLFN
        SET DFUSR = :DFUSR,
            DFAPN = :DFAPN,
            DFFIL = :DFFIL,
            DFMOD = :DFMOD,
            DFLFN = :DFLFN
```

IBDDF

```
DELETE AN EXISTING ROW IN TABLE DF.
        DELETE FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :DFUSR AND
              DFAPN = :DFAPN AND
              DFLFN = :DFLFN
```

IBODF0

```
OBTAIN A ROW IN TABLE DF VIA ACCESS PATH DF0.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :DFUSR AND
              DFAPN = :DFAPN AND
              DFLFN = :DFLFN
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBODF1

```
OBTAIN A ROW IN TABLE DF VIA ACCESS PATH DF1.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFAPN = :DFAPN AND
              DFFIL = :DFFIL AND
              DFUSR = :DFUSR
        ORDER BY DFAPN ASC, DFFIL ASC, DFUSR ASC
```

IBODF2

```
OBTAIN A ROW IN TABLE DF VIA ACCESS PATH DF2.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :DFUSR AND
              DFAPN = :DFAPN
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBODF3

```
OBTAIN A ROW IN TABLE DF VIA ACCESS PATH DF3.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :DFUSR
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBODF4

```
OBTAIN A ROW IN TABLE DF VIA ACCESS PATH DF4.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFFIL = :DFFIL
        ORDER BY DFFIL ASC
```

IBADF0

```
OBTAIN A ROW IN TABLE DF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DF0.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE (DFUSR > :DFUSR)
          OR ((DFUSR = :DFUSR) AND (DFAPN > :DFAPN))
          OR ((DFUSR = :DFUSR AND DFAPN = :DFAPN) AND (DFLFN > :DFLFN))
          OR (DFUSR = :DFUSR AND DFAPN = :DFAPN AND DFLFN = :DFLFN)
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBADF1

```
OBTAIN A ROW IN TABLE DF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DF1.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE (DFAPN > :DFAPN)
          OR ((DFAPN = :DFAPN) AND (DFFIL > :DFFIL))
          OR ((DFAPN = :DFAPN AND DFFIL = :DFFIL) AND (DFUSR > :DFUSR))
          OR (DFAPN = :DFAPN AND DFFIL = :DFFIL AND DFUSR = :DFUSR)
        ORDER BY DFAPN ASC, DFFIL ASC, DFUSR ASC
```

IBADF2

```
OBTAIN A ROW IN TABLE DF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DF2.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE (DFUSR > :DFUSR)
          OR ((DFUSR = :DFUSR) AND (DFAPN > :DFAPN))
          OR (DFUSR = :DFUSR AND DFAPN = :DFAPN)
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBADF3

```
OBTAIN A ROW IN TABLE DF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DF3.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR >= :DFUSR
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBADF4

```
OBTAIN A ROW IN TABLE DF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DF4.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFFIL >= :DFFIL
        ORDER BY DFFIL ASC
```

IBEDF2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DF VIA ACCESS PATH DF2.
SAVE THE CURRENT POSITION IN TABLE DF.
FETCH THE NEXT ROW FROM TABLE DF.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBEDF3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DF VIA ACCESS PATH DF3.
SAVE THE CURRENT POSITION IN TABLE DF.
FETCH THE NEXT ROW FROM TABLE DF.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBEDF4

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DF VIA ACCESS PATH DF4.
SAVE THE CURRENT POSITION IN TABLE DF.
FETCH THE NEXT ROW FROM TABLE DF.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBFDF0

```
OBTAIN THE FIRST ROW OF TABLE DF, ORDERED BY ACCESS PATH DF0.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBFDF1

```
OBTAIN THE FIRST ROW OF TABLE DF, ORDERED BY ACCESS PATH DF1.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        ORDER BY DFAPN ASC, DFFIL ASC, DFUSR ASC
```

IBFDF2

```
OBTAIN THE FIRST ROW OF TABLE DF, ORDERED BY ACCESS PATH DF2.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBFDF3

```
OBTAIN THE FIRST ROW OF TABLE DF, ORDERED BY ACCESS PATH DF3.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBFDF4

```
OBTAIN THE FIRST ROW OF TABLE DF, ORDERED BY ACCESS PATH DF4.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        ORDER BY DFFIL ASC
```

IBNDF0

```
OBTAIN THE NEXT ROW OF TABLE DF, ORDERED BY ACCESS PATH DF0.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBNDF1

```
OBTAIN THE NEXT ROW OF TABLE DF, ORDERED BY ACCESS PATH DF1.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBNDF2

```
OBTAIN THE NEXT ROW OF TABLE DF, ORDERED BY ACCESS PATH DF2.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBNDF3

```
OBTAIN THE NEXT ROW OF TABLE DF, ORDERED BY ACCESS PATH DF3.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBNDF4

```
OBTAIN THE NEXT ROW OF TABLE DF, ORDERED BY ACCESS PATH DF4.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

IBOUIDF

```
USING COSET UIDF, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DF
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :DFUSR
```

IBFUIDF

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DF WITHIN COSET UIDF, USING
ACCESS PATH DF3.
        SELECT DFUSR, DFAPN, DFFIL, DFMOD, DFLFN
        FROM DF IN ENGINEERING_DATA_DATABASE
        WHERE DFUSR = :UIUSR
        ORDER BY DFUSR ASC, DFAPN ASC, DFLFN ASC
```

IBNUIDF

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DF WITHIN COSET UIDF.
        SET :DFUSR, :DFAPN, :DFFIL, :DFMOD, :DFLFN
```

Engineering Data Information (DI) Routines

The DI table contains the basic information for each user's engineering data.

IBSDI

```
STORE A NEW ROW IN TABLE DI.
        INSERT INTO DI IN ENGINEERING_DATA_DATABASE
        SET DIEDN = :DIEDN,
            DIFIL = :DIFIL,
            DINAM = :DINAM,
            DISID = :DISID,
            DIADT = :DIADT,
            DIEDT = :DIEDT,
            DIUSR = :DIUSR,
            DIREV = :DIREV,
            DISTA = :DISTA,
            DIDATC = :DIDATC,
            DIDATM = :DIDATM,
            DIDATR = :DIDATR,
            DITTL = :DITTL,
            DITIMC = :DITIMC,
            DITIMM = :DITIMM,
            DITIMR = :DITIMR
```

IBMDI

```
MODIFY AN EXISTING ROW IN TABLE DI.
        UDPATE DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DIEDN
        SET DIEDN = :DIEDN,
            DIFIL = :DIFIL,
            DINAM = :DINAM,
            DISID = :DISID,
            DIADT = :DIADT,
            DIEDT = :DIEDT,
            DIUSR = :DIUSR,
            DIREV = :DIREV,
            DISTA = :DISTA,
            DIDATC = :DIDATC,
            DIDATM = :DIDATM,
            DIDATR = :DIDATR,
            DITTL = :DITTL,
            DITIMC = :DITIMC,
            DITIMM = :DITIMM,
            DITIMR = :DITIMR
```

IBDDI

```
DELETE AN EXISTING ROW IN TABLE DI.
        DELETE FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DIEDN
```

IBODI0

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI0.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DIEDN
        ORDER BY DIEDN ASC
```

IBODI1

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI1.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DINAM = :DINAM
        ORDER BY DISID ASC, DIREV ASC
```

IBODI2

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI2.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIFIL = :DIFIL
        ORDER BY DINAM ASC, DISID ASC, DIREV ASC
```

IBODI3

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI3.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DINAM = :DINAM AND
              DISID = :DISID AND
              DIFIL = :DIFIL
        ORDER BY DINAM ASC, DISID ASC, DIFIL ASC
```

IBODI4

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI4.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDT = :DIEDT
        ORDER BY DIEDT ASC
```

IBODI5

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI5.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIADT = :DIADT
        ORDER BY DIADT ASC
```

IBODI6

```
OBTAIN A ROW IN TABLE DI VIA ACCESS PATH DI6.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIUSR = :DIUSR
        ORDER BY DIUSR ASC
```

IBADI0

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS PATH DI0.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN >= :DIEDN
        ORDER BY DIEDN ASC
```

IBADI1

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS PATH DI1.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DINAM >= :DINAM
        ORDER BY DINAM ASC, DISID ASC, DIREV ASC
```

IBADI3

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DI3.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE (DINAM > :DINAM)
          OR ((DINAM = :DINAM) AND (DISID > :DISID))
          OR ((DINAM = :DINAM AND DISID = :DISID) AND (DIFIL > :DIFIL))
          OR (DINAM = :DINAM AND DISID = :DISID AND DIFIL = :DIFIL)
        ORDER BY DINAM ASC, DISID ASC, DIFIL ASC
```

IBADI4

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DI4.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDT >= :DIEDT
        ORDER BY DIEDT ASC
```

IBADI5

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DI5.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIADT >= :DIADT
        ORDER BY DIADT ASC
```

IBADI6

```
OBTAIN A ROW IN TABLE DI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DI6.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIUSR >= :DIUSR
        ORDER BY DIUSR ASC
```

IBEDI1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DI VIA ACCESS PATH DI1.
SAVE THE CURRENT POSITION IN TABLE DI.
FETCH THE NEXT ROW FROM TABLE DI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBEDI2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DI VIA ACCESS PATH DI2.
SAVE THE CURRENT POSITION IN TABLE DI.
FETCH THE NEXT ROW FROM TABLE DI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBEDI4

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DI VIA ACCESS PATH DI4.
SAVE THE CURRENT POSITION IN TABLE DI.
FETCH THE NEXT ROW FROM TABLE DI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBEDI5

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DI VIA ACCESS PATH DI5.
SAVE THE CURRENT POSITION IN TABLE DI.
FETCH THE NEXT ROW FROM TABLE DI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBEDI6

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DI VIA ACCESS PATH DI6.
SAVE THE CURRENT POSITION IN TABLE DI.
FETCH THE NEXT ROW FROM TABLE DI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBFDI0

```
OBTAIN THE FIRST ROW OF TABLE DI, ORDERED BY ACCESS PATH DI0.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        ORDER BY DIEDN ASC
```

IBFDI26

```
OBTAIN THE FIRST ROW OF TABLE DI, ACCESS PATH DI6
                        ORDER BY  ACCESS PATH DI2
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE :DIUSR = DIUSR
        ORDER BY DINAM ASC, DISID ASC, DIREV ASC
```

IBFDI3

```
OBTAIN THE FIRST ROW OF TABLE DI, ORDERED BY ACCESS PATH DI3.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        ORDER BY DINAM ASC, DISID ASC, DIFIL ASC
```

IBFDI4

```
OBTAIN THE FIRST ROW OF TABLE DI, ORDERED BY ACCESS PATH DI4.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        ORDER BY DIEDT ASC
```

IBFDI5

```
OBTAIN THE FIRST ROW OF TABLE DI, ORDERED BY ACCESS PATH DI5.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        ORDER BY DIADT ASC
```

IBFDI6

```
OBTAIN THE FIRST ROW OF TABLE DI, ORDERED BY ACCESS PATH DI6.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        ORDER BY DIUSR ASC
```

IBNDI0

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI0.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI26

```
OBTAIN THE NEXT ROW OF TABLE DI, ACCESS PATH DI6 ORDER BY DI2
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI1

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI0.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI3

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI3.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI4

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI4.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI5

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI5.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNDI6

```
OBTAIN THE NEXT ROW OF TABLE DI, ORDERED BY ACCESS PATH DI6.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBOETDI

```
USING COSET ETDI, OBTAIN THE ROW FROM TABLE ET THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DI
        SELECT ETEDT, ETDSC
        INTO :ETEDT, :ETDSC
        FROM ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT = :DIEDT
```

IBOATDI

```
USING COSET ATDI, OBTAIN THE ROW FROM TABLE AT THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DI
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        INTO :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :DIADT
```

IBOFIDI

```
USING COSET FIDI, OBTAIN THE ROW FROM TABLE FI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DI
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        INTO :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :DIFIL
```

IBFETDI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DI WITHIN COSET ETDI, USING
ACCESS PATH DI4.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDT = :ETEDT
        ORDER BY DIEDT ASC
```

IBFATDI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DI WITHIN COSET ATDI, USING
ACCESS PATH DI5.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIADT = :ATADT
        ORDER BY DIADT ASC
```

IBFFIDI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DI WITHIN COSET FIDI, USING
ACCESS PATH DI2.
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIFIL = :FIFIL
        ORDER BY DIFIL ASC, DINAM ASC, DISID ASC, DIREV ASC
```

IBNETDI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DI WITHIN COSET ETDI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNATDI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DI WITHIN COSET ATDI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

IBNFIDI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DI WITHIN COSET FIDI.
        SET :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
```

Data Required (DR) Routines

The DR table identifies all the data sets that must be available in order for a given data set to be correctly interpreted.

IBSDR

```
STORE A NEW ROW IN TABLE DR.
        INSERT INTO DR IN ENGINEERING_DATA_DATABASE
        SET DREDN = :DREDN,
            DREDNR = :DREDNR
```

IBMDR

```
MODIFY AN EXISTING ROW IN TABLE DR.
        UDPATE DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN = :DREDN AND
              DREDNR = :DREDNR
        SET DREDN = :DREDN,
            DREDNR = :DREDNR
```

IBDDR

```
DELETE AN EXISTING ROW IN TABLE DR.
        DELETE FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN = :DREDN AND
              DREDNR = :DREDNR
```

IBODR0

```
OBTAIN A ROW IN TABLE DR VIA ACCESS PATH DR0.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN = :DREDN AND
              DREDNR = :DREDNR
        ORDER BY DREDN ASC, DREDNR ASC
```

IBODR1

```
OBTAIN A ROW IN TABLE DR VIA ACCESS PATH DR1.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN = :DREDN
        ORDER BY DREDN ASC
```

IBODR2

```
OBTAIN A ROW IN TABLE DR VIA ACCESS PATH DR2.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDNR = :DREDNR
        ORDER BY DREDNR ASC
```

IBADR0

```
OBTAIN A ROW IN TABLE DR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DR0.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE (DREDN > :DREDN)
          OR ((DREDN = :DREDN) AND (DREDNR > :DREDNR))
          OR (DREDN = :DREDN AND DREDNR = :DREDNR)
        ORDER BY DREDN ASC, DREDNR ASC
```

IBADR1

```
OBTAIN A ROW IN TABLE DR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DR1.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN >= :DREDN
        ORDER BY DREDN ASC
```

IBADR2

```
OBTAIN A ROW IN TABLE DR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DR2.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDNR >= :DREDNR
        ORDER BY DREDNR ASC
```

IBEDR1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DR VIA ACCESS PATH DR1.
SAVE THE CURRENT POSITION IN TABLE DR.
FETCH THE NEXT ROW FROM TABLE DR.
        SET :DREDN, :DREDNR
```

IBEDR2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DR VIA ACCESS PATH DR2.
SAVE THE CURRENT POSITION IN TABLE DR.
FETCH THE NEXT ROW FROM TABLE DR.
        SET :DREDN, :DREDNR
```

IBFDR0

```
OBTAIN THE FIRST ROW OF TABLE DR, ORDERED BY ACCESS PATH DR0.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        ORDER BY DREDN ASC, DREDNR ASC
```

IBFDR1

```
OBTAIN THE FIRST ROW OF TABLE DR, ORDERED BY ACCESS PATH DR1.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        ORDER BY DREDN ASC
```

IBFDR2

```
OBTAIN THE FIRST ROW OF TABLE DR, ORDERED BY ACCESS PATH DR2.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        ORDER BY DREDNR ASC
```

IBNDR0

```
OBTAIN THE NEXT ROW OF TABLE DR, ORDERED BY ACCESS PATH DR0.
        SET :DREDN, :DREDNR
```

IBNDR1

```
OBTAIN THE NEXT ROW OF TABLE DR, ORDERED BY ACCESS PATH DR1.
        SET :DREDN, :DREDNR
```

IBNDR2

```
OBTAIN THE NEXT ROW OF TABLE DR, ORDERED BY ACCESS PATH DR2.
        SET :DREDN, :DREDNR
```

IBODDR2

```
USING COSET DDR2, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DR
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DREDN
```

IBODDR1

```
USING COSET DDR1, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DR
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DREDN
```

IBFDDR2

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DR WITHIN COSET DDR2, USING
ACCESS PATH DR2.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDNR = :DIEDN
        ORDER BY DREDNR ASC
```

IBFDDR1

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DR WITHIN COSET DDR1, USING
ACCESS PATH DR1.
        SELECT DREDN, DREDNR
        FROM DR IN ENGINEERING_DATA_DATABASE
        WHERE DREDN = :DIEDN
        ORDER BY DREDN ASC
```

IBNDDR2

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DR WITHIN COSET DDR2.
        SET :DREDN, :DREDNR
```

IBNDDR1

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DR WITHIN COSET DDR1.
        SET :DREDN, :DREDNR
```

Data Source (DS) Routines

The DS table defines the data sets from which a new data set was derived. Such a data set is said to be the "source" for the new data.

IBSDS

```
STORE A NEW ROW IN TABLE DS.
        INSERT INTO DS IN ENGINEERING_DATA_DATABASE
        SET DSEDN = :DSEDN,
            DSEDNS = :DSEDNS
```

IBMDS

```
MODIFY AN EXISTING ROW IN TABLE DS.
        UDPATE DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN = :DSEDN AND
              DSEDNS = :DSEDNS
        SET DSEDN = :DSEDN,
            DSEDNS = :DSEDNS
```

IBDDS

```
DELETE AN EXISTING ROW IN TABLE DS.
        DELETE FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN = :DSEDN AND
              DSEDNS = :DSEDNS
```

IBODS0

```
OBTAIN A ROW IN TABLE DS VIA ACCESS PATH DS0.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN = :DSEDN AND
              DSEDNS = :DSEDNS
        ORDER BY DSEDN ASC, DSEDNS ASC
```

IBODS1

```
OBTAIN A ROW IN TABLE DS VIA ACCESS PATH DS1.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN = :DSEDN
        ORDER BY DSEDN ASC
```

IBODS2

```
OBTAIN A ROW IN TABLE DS VIA ACCESS PATH DS2.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDNS =\:DSEDNS
        ORDER BY DSEDNS ASC
```

IBADS0

```
OBTAIN A ROW IN TABLE DS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DS0.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE (DSEDN > :DSEDN)
          OR ((DSEDN = :DSEDN) AND (DSEDNS > :DSEDNS))
          OR (DSEDN = :DSEDN AND DSEDNS = :DSEDNS)
        ORDER BY DSEDN ASC, DSEDNS ASC
```

IBADS1

```
OBTAIN A ROW IN TABLE DS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DS1.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN >= :DSEDN
        ORDER BY DSEDN ASC
```

IBADS2

```
OBTAIN A ROW IN TABLE DS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH DS2.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDNS >= :DSEDNS
        ORDER BY DSEDNS ASC
```

IBEDS1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DS VIA ACCESS PATH DS1.
SAVE THE CURRENT POSITION IN TABLE DS.
FETCH THE NEXT ROW FROM TABLE DS.
        SET :DSEDN, :DSEDNS
```

IBEDS2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE DS VIA ACCESS PATH DS2.
SAVE THE CURRENT POSITION IN TABLE DS.
FETCH THE NEXT ROW FROM TABLE DS.
        SET :DSEDN, :DSEDNS
```

IBFDS0

```
OBTAIN THE FIRST ROW OF TABLE DS, ORDERED BY ACCESS PATH DS0.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        ORDER BY DSEDN ASC, DSEDNS ASC
```

IBFDS1

```
OBTAIN THE FIRST ROW OF TABLE DS, ORDERED BY ACCESS PATH DS1.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        ORDER BY DSEDN ASC
```

IBFDS2

```
OBTAIN THE FIRST ROW OF TABLE DS, ORDERED BY ACCESS PATH DS2.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        ORDER BY DSEDNS ASC
```

IBNDS0

```
OBTAIN THE NEXT ROW OF TABLE DS, ORDERED BY ACCESS PATH DS0.
        SET :DSEDN, :DSEDNS
```

IBNDS1

```
OBTAIN THE NEXT ROW OF TABLE DS, ORDERED BY ACCESS PATH DS1.
        SET :DSEDN, :DSEDNS
```

IBNDS2

```
OBTAIN THE NEXT ROW OF TABLE DS, ORDERED BY ACCESS PATH DS2.
        SET :DSEDN, :DSEDNS
```

IBODDS2

```
USING COSET DDS2, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DS
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DSEDN
```

IBODDS1

```
USING COSET DDS1, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE DS
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :DSEDN
```

IBFDDS2

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DS WITHIN COSET DDS2, USING
ACCESS PATH DS2.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDNS = :DIEDN
        ORDER BY DSEDNS ASC
```

IBFDDS1

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE DS WITHIN COSET DDS1, USING
ACCESS PATH DS1.
        SELECT DSEDN, DSEDNS
        FROM DS IN ENGINEERING_DATA_DATABASE
        WHERE DSEDN = :DIEDN
        ORDER BY DSEDN ASC
```

IBNDDS2

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DS WITHIN COSET DDS2.
        SET :DSEDN, :DSEDNS
```

IBNDDS1

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE DS WITHIN COSET DDS1.
        SET :DSEDN, :DSEDNS
```

Engineering Attributes (EA) Routines

The EA table defines the standard attributes used to describe data at the site. Users are prompted for these attributes when they choose to describe engineering data.

IBSEA

```
STORE A NEW ROW IN TABLE EA.
        INSERT INTO EA IN ENGINEERING_DATA_DATABASE
        SET EAEDT = :EAEDT,
            EAATR = :EAATR
```

IBMEA

```
MODIFY AN EXISTING ROW IN TABLE EA.
        UDPATE EA IN ENGINEERING_DATA_DATABASE
        WHERE EAATR = :EAATR AND
              EAEDT = :EAEDT
        SET EAEDT = :EAEDT,
            EAATR = :EAATR
```

IBDEA

```
DELETE AN EXISTING ROW IN TABLE EA.
        DELETE FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAATR = :EAATR AND
              EAEDT = :EAEDT
```

IBOEA0

```
OBTAIN A ROW IN TABLE EA VIA ACCESS PATH EA0.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAATR = :EAATR AND
              EAEDT = :EAEDT
        ORDER BY EAATR ASC, EAEDT ASC
```

IBOEA1

```
OBTAIN A ROW IN TABLE EA VIA ACCESS PATH EA1.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAEDT = :EAEDT
        ORDER BY EAEDT ASC, EAATR ASC
```

IBOEA2

```
OBTAIN A ROW IN TABLE EA VIA ACCESS PATH EA2.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAATR = :EAATR
        ORDER BY EAATR ASC
```

IBAEA0

```
OBTAIN A ROW IN TABLE EA USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH EA0.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE (EAATR > :EAATR)
          OR ((EAATR = :EAATR) AND (EAEDT > :EAEDT))
          OR (EAATR = :EAATR AND EAEDT = :EAEDT)
        ORDER BY EAATR ASC, EAEDT ASC
```

IBAEA1

```
OBTAIN A ROW IN TABLE EA USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH EA1.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAEDT >= :EAEDT
        ORDER BY EAEDT ASC, EAATR ASC
```

IBAEA2

```
OBTAIN A ROW IN TABLE EA USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH EA2.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAATR >= :EAATR
        ORDER BY EAATR ASC
```

IBEEA1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE EA VIA ACCESS PATH EA1.
SAVE THE CURRENT POSITION IN TABLE EA.
FETCH THE NEXT ROW FROM TABLE EA.
        SET :EAEDT, :EAATR
```

IBEEA2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE EA VIA ACCESS PATH EA2.
SAVE THE CURRENT POSITION IN TABLE EA.
FETCH THE NEXT ROW FROM TABLE EA.
        SET :EAEDT, :EAATR
```

IBFEA0

```
OBTAIN THE FIRST ROW OF TABLE EA, ORDERED BY ACCESS PATH EA0.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        ORDER BY EAATR ASC, EAEDT ASC
```

IBFEA1

```
OBTAIN THE FIRST ROW OF TABLE EA, ORDERED BY ACCESS PATH EA1.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        ORDER BY EAEDT ASC, EAATR ASC
```

IBFEA2

```
OBTAIN THE FIRST ROW OF TABLE EA, ORDERED BY ACCESS PATH EA2.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        ORDER BY EAATR ASC
```

IBNEA0

```
OBTAIN THE NEXT ROW OF TABLE EA, ORDERED BY ACCESS PATH EA0.
        SET :EAEDT, :EAATR
```

IBNEA1

```
OBTAIN THE NEXT ROW OF TABLE EA, ORDERED BY ACCESS PATH EA1.
        SET :EAEDT, :EAATR
```

IBNEA2

```
OBTAIN THE NEXT ROW OF TABLE EA, ORDERED BY ACCESS PATH EA2.
        SET :EAEDT, :EAATR
```

IBOETEA

```
USING COSET ETEA, OBTAIN THE ROW FROM TABLE ET THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE EA
        SELECT ETEDT, ETDSC
        INTO :ETEDT, :ETDSC
        FROM ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT = :EAEDT
```

IBFETEA

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE EA WITHIN COSET ETEA, USING
ACCESS PATH EA1.
        SELECT EAEDT, EAATR
        FROM EA IN ENGINEERING_DATA_DATABASE
        WHERE EAEDT = :ETEDT
        ORDER BY EAEDT ASC, EAATR ASC
```

IBNETEA

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE EA WITHIN COSET ETEA.
        SET :EAEDT, :EAATR
```

Engineering Categories (ET) Routines

The ET table defines the categories of engineering data descriptions used by the site. Categories provide a way of organizing and separating data based on how it is used.

IBSET

```
STORE A NEW ROW IN TABLE ET.
        INSERT INTO ET IN ENGINEERING_DATA_DATABASE
        SET ETEDT = :ETEDT,
            ETDSC = :ETDSC
```

IBMET

```
MODIFY AN EXISTING ROW IN TABLE ET.
        UDPATE ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT = :ETEDT
        SET ETEDT = :ETEDT,
            ETDSC = :ETDSC
```

IBDET

```
DELETE AN EXISTING ROW IN TABLE ET.
        DELETE FROM ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT = :ETEDT
```

IBOET0

```
OBTAIN A ROW IN TABLE ET VIA ACCESS PATH ET0.
        SELECT ETEDT, ETDSC
        FROM ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT = :ETEDT
        ORDER BY ETEDT ASC
```

IBAET0

```
OBTAIN A ROW IN TABLE ET USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH ET0.
        SELECT ETEDT, ETDSC
        FROM ET IN ENGINEERING_DATA_DATABASE
        WHERE ETEDT >= :ETEDT
        ORDER BY ETEDT ASC
```

IBFET0

```
OBTAIN THE FIRST ROW OF TABLE ET, ORDERED BY ACCESS PATH ET0.
        SELECT ETEDT, ETDSC
        FROM ET IN ENGINEERING_DATA_DATABASE
        ORDER BY ETEDT ASC
```

IBNET0

```
OBTAIN THE NEXT ROW OF TABLE ET, ORDERED BY ACCESS PATH ET0.
        SET :ETEDT, :ETDSC
```

Family Data (FD) Routines

The FD table defines the family-engineering data relationships.

IBSFD

```
STORE A NEW ROW IN TABLE FD.
        INSERT INTO FD IN ENGINEERING_DATA_DATABASE
        SET FDFAM = :FDFAM,
            FDEDN = :FDEDN
```

IBMFD

```
MODIFY AN EXISTING ROW IN TABLE FD.
        UDPATE FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM = :FDFAM AND
              FDEDN = :FDEDN
        SET FDFAM = :FDFAM,
            FDEDN = :FDEDN
```

IBDFD

```
DELETE AN EXISTING ROW IN TABLE FD.
        DELETE FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM = :FDFAM AND
              FDEDN = :FDEDN
```

IBOFD0

```
OBTAIN A ROW IN TABLE FD VIA ACCESS PATH FD0.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM = :FDFAM AND
              FDEDN = :FDEDN
        ORDER BY FDFAM ASC, FDEDN ASC
```

IBOFD1

```
OBTAIN A ROW IN TABLE FD VIA ACCESS PATH FD1.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM = :FDFAM
        ORDER BY FDFAM ASC
```

IBOFD2

```
OBTAIN A ROW IN TABLE FD VIA ACCESS PATH FD2.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDEDN = :FDEDN
        ORDER BY FDEDN ASC
```

IBAFD0

```
OBTAIN A ROW IN TABLE FD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FD0.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE (FDFAM > :FDFAM)
          OR ((FDFAM = :FDFAM) AND (FDEDN > :FDEDN))
          OR (FDFAM = :FDFAM AND FDEDN = :FDEDN)
        ORDER BY FDFAM ASC, FDEDN ASC
```

IBAFD1

```
OBTAIN A ROW IN TABLE FD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FD1.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM >= :FDFAM
        ORDER BY FDFAM ASC
```

IBAFD2

```
OBTAIN A ROW IN TABLE FD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FD2.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDEDN >= :FDEDN
        ORDER BY FDEDN ASC
```

IBEFD1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FD VIA ACCESS PATH FD1.
SAVE THE CURRENT POSITION IN TABLE FD.
FETCH THE NEXT ROW FROM TABLE FD.
        SET :FDFAM, :FDEDN
```

IBEFD2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FD VIA ACCESS PATH FD2.
SAVE THE CURRENT POSITION IN TABLE FD.
FETCH THE NEXT ROW FROM TABLE FD.
        SET :FDFAM, :FDEDN
```

IBFFD0

```
OBTAIN THE FIRST ROW OF TABLE FD, ORDERED BY ACCESS PATH FD0.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        ORDER BY FDFAM ASC, FDEDN ASC
```

IBFFD1

```
OBTAIN THE FIRST ROW OF TABLE FD, ORDERED BY ACCESS PATH FD1.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        ORDER BY FDFAM ASC
```

IBFFD2

```
OBTAIN THE FIRST ROW OF TABLE FD, ORDERED BY ACCESS PATH FD2.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        ORDER BY FDEDN ASC
```

IBNFD0

```
OBTAIN THE NEXT ROW OF TABLE FD, ORDERED BY ACCESS PATH FD0.
        SET :FDFAM, :FDEDN
```

IBNFD1

```
OBTAIN THE NEXT ROW OF TABLE FD, ORDERED BY ACCESS PATH FD1.
        SET :FDFAM, :FDEDN
```

IBNFD2

```
OBTAIN THE NEXT ROW OF TABLE FD, ORDERED BY ACCESS PATH FD2.
        SET :FDFAM, :FDEDN
```

IBODIFD

```
USING COSET DIFD, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FD
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :FDEDN
```

IBOFMFD

```
USING COSET FMFD, OBTAIN THE ROW FROM TABLE FM THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FD
        SELECT FMFAM, FMTTL
        INTO :FMFAM, :FMTTL
        FROM FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM = :FDFAM
```

IBFDIFD

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FD WITHIN COSET DIFD, USING
ACCESS PATH FD2.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDEDN = :DIEDN
        ORDER BY FDEDN ASC
```

IBFFMFD

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FD WITHIN COSET FMFD, USING
ACCESS PATH FD1.
        SELECT FDFAM, FDEDN
        FROM FD IN ENGINEERING_DATA_DATABASE
        WHERE FDFAM = :FMFAM
        ORDER BY FDFAM ASC
```

IBNDIFD

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FD WITHIN COSET DIFD.
        SET :FDFAM, :FDEDN
```

IBNFMFD

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FD WITHIN COSET FMFD.
        SET :FDFAM, :FDEDN
```

File Information (FI) Routines

The FI table contains the definitions of all user files known to EDL.

IBSFI

```
STORE A NEW ROW IN TABLE FI.
        INSERT INTO FI IN ENGINEERING_DATA_DATABASE
        SET FIFIL = :FIFIL,
            FIHOS = :FIHOS,
            FIFUN = :FIFUN,
            FIPFN = :FIPFN,
            FILNA = :FILNA,
            FIFTC = :FIFTC,
            FIUSR = :FIUSR,
            FICT = :FICT,
            FIMOD = :FIMOD,
            FISTA = :FISTA,
            FIVSN = :FIVSN
```

IBMFI

```
MODIFY AN EXISTING ROW IN TABLE FI.
        UDPATE FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :FIFIL
        SET FIFIL = :FIFIL,
            FIHOS = :FIHOS,
            FIFUN = :FIFUN,
            FIPFN = :FIPFN,
            FILNA = :FILNA,
            FIFTC = :FIFTC,
            FIUSR = :FIUSR,
            FICT = :FICT,
            FIMOD = :FIMOD,
            FISTA = :FISTA,
            FIVSN = :FIVSN
```

IBDFI

```
DELETE AN EXISTING ROW IN TABLE FI.
        DELETE FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :FIFIL
```

IBOFI0

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FIO.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :FIFIL
        ORDER BY FIFIL ASC
```

IBOFI1

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI1.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIHOS = :FIHOS AND
              FIPFN = :FIPFN AND
              FIFUN = :FIFUN AND
              FILNA = :FILNA
        ORDER BY FIHOS ASC, FIPFN ASC, FIFUN ASC, FILNA ASC
```

IBOFI2

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI2.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIUSR = :FIUSR
        ORDER BY FIHOS ASC, FIFUN ASC, FIPFN ASC, FILNA ASC
```

IBOFI3

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI3.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIHOS = :FIHOS AND
              FIFUN = :FIFUN
        ORDER BY FIHOS ASC, FIFUN ASC, FIPFN ASC
```

IBOFI4

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI4.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFTC = :FIFTC
        ORDER BY FIFTC ASC, FIPFN ASC
```

IBOFI5

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI5.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIHOS = :FIHOS
        ORDER BY FIHOS ASC, FIPFN ASC
```

IBOFI6

```
OBTAIN A ROW IN TABLE FI VIA ACCESS PATH FI6.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
      INTO :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR, :FICT,
          :FIMOD, :FISTA, :FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIPFN = :FIPFN
```

IBAFI0

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI0.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL >= :FIFIL
        ORDER BY FIFIL ASC
```

IBAFI1

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI1.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE (FIHOS > :FIHOS)
          OR ((FIHOS = :FIHOS) AND (FIPFN > :FIPFN))
          OR ((FIHOS = :FIHOS AND FIPFN = :FIPFN) AND (FIFUN > :FIFUN))
          OR  ((FIHOS  = :FIHOS AND FIPFN = :FIPFN AND FIFUN = :FIFUN) AND
              (FILNA > :FILNA))
          OR (FIHOS = :FIHOS AND FIPFN = :FIPFN AND FIFUN = :FIFUN AND
              FILNA = :FILNA)
        ORDER BY FIHOS ASC, FIPFN ASC, FIFUN ASC, FILNA ASC
```

IBAFI2

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI2.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIUSR >= :FIUSR
        ORDER BY FIUSR ASC, FIHOS ASC, FIFUN ASC, FIPFN ASC, FILNA ASC
```

IBAFI3

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI3.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE (FIHOS > :FIHOS)
          OR ((FIHOS = :FIHOS) AND (FIFUN > :FIFUN))
          OR (FIHOS = :FIHOS AND FIFUN = :FIFUN)
        ORDER BY FIHOS ASC, FIFUN ASC
```

IBAFI4

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI4.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFTC >= :FIFTC
        ORDER BY FIFTC ASC
```

IBAFI5

```
OBTAIN A ROW IN TABLE FI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FI5.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIHOS >= :FIHOS
        ORDER BY FIHOS ASC
```

IBEFI2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FI VIA ACCESS PATH FI2.
SAVE THE CURRENT POSITION IN TABLE FI.
FETCH THE NEXT ROW FROM TABLE FI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBEFI3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FI VIA ACCESS PATH FI3.
SAVE THE CURRENT POSITION IN TABLE FI.
FETCH THE NEXT ROW FROM TABLE FI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBEFI4

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FI VIA ACCESS PATH FI4.
SAVE THE CURRENT POSITION IN TABLE FI.
FETCH THE NEXT ROW FROM TABLE FI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBEFI5

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FI VIA ACCESS PATH FI5.
SAVE THE CURRENT POSITION IN TABLE FI.
FETCH THE NEXT ROW FROM TABLE FI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBFFI0

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI0.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIFIL ASC
```

IBFFI1

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI1.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIHOS ASC, FIPFN ASC, FIFUN ASC, FILNA ASC
```

IBFFI2

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI2.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIUSR ASC, FIHOS ASC, FIFUN ASC, FIPFN ASC, FILNA ASC
```

IBFFI3

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI3.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIHOS ASC, FIFUN ASC
```

IBFFI4

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI4.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIFTC ASC
```

IBFFI5

```
OBTAIN THE FIRST ROW OF TABLE FI, ORDERED BY ACCESS PATH FI5.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        ORDER BY FIHOS ASC
```

IBNFI0

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI0.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNFI1

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI1.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNFI2

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI2.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNFI3

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI3.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNFI4

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI4.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNFI5

```
OBTAIN THE NEXT ROW OF TABLE FI, ORDERED BY ACCESS PATH FI5.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBOUIFI

```
USING COSET UIFI, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FI
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :FIUSR
```

IBOHIFI

```
USING COSET HIFI, OBTAIN THE ROW FROM TABLE HI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FI
        SELECT HIHOS, HIOFF, HIOS
        INTO :HIHOS, :HIOFF, :HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIHOS = :FIHOS
```

IBFUIFI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FI WITHIN COSET UIFI, USING
ACCESS PATH FI2.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIUSR = :UIUSR
        ORDER BY FIUSR ASC, FIHOS ASC, FIFUN ASC, FIPFN ASC, FILNA ASC
```

IBFHIFI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FI WITHIN COSET HIFI, USING
ACCESS PATH FI5.
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIHOS = :HIHOS
        ORDER BY FIHOS ASC
```

IBNUIFI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FI WITHIN COSET UIFI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

IBNHIFI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FI WITHIN COSET HIFI.
        SET :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
```

Family Information (FM) Routines

The FM record contains the part family codes associated with part numbers.

IBSFM

```
STORE A NEW ROW IN TABLE FM.
        INSERT INTO FM IN ENGINEERING_DATA_DATABASE
        SET FMFAM = :FMFAM,
            FMTTL = :FMTTL
```

IBMFM

```
MODIFY AN EXISTING ROW IN TABLE FM.
        UDPATE FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM = :FMFAM
        SET FMFAM = :FMFAM,
            FMTTL = :FMTTL
```

IBDFM

```
DELETE AN EXISTING ROW IN TABLE FM.
        DELETE FROM FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM = :FMFAM
```

IBOFM0

```
OBTAIN A ROW IN TABLE FM VIA ACCESS PATH FM0.
        SELECT FMFAM, FMTTL
        FROM FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM = :FMFAM
        ORDER BY FMFAM ASC
```

IBAFM0

```
OBTAIN A ROW IN TABLE FM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH FM0.
        SELECT FMFAM, FMTTL
        FROM FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM >= :FMFAM
        ORDER BY FMFAM ASC
```

IBFFM0

```
OBTAIN THE FIRST ROW OF TABLE FM, ORDERED BY ACCESS PATH FM0.
        SELECT FMFAM, FMTTL
        FROM FM IN ENGINEERING_DATA_DATABASE
        ORDER BY FMFAM ASC
```

IBNFM0

```
OBTAIN THE NEXT ROW OF TABLE FM, ORDERED BY ACCESS PATH FM0.
        SET :FMFAM, :FMTTL
```

File Permits (FP) Routines

The FP table defines group and user permits to files.

IBSFP

```
STORE A NEW ROW IN TABLE FP.
        INSERT INTO FP IN ENGINEERING_DATA_DATABASE
        SET FPFIL = :FPFIL,
            FPUSR = :FPUSR,
            FPMOD = :FPMOD
```

IBMFP

```
MODIFY AN EXISTING ROW IN TABLE FP.
        UDPATE FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL = :FPFIL AND
              FPUSR = :FPUSR
        SET FPFIL = :FPFIL,
            FPUSR = :FPUSR,
            FPMOD = :FPMOD
```

IBDFP

```
DELETE AN EXISTING ROW IN TABLE FP.
        DELETE FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL = :FPFIL AND
              FPUSR = :FPUSR
```

IBOFP0

```
OBTAIN A ROW IN TABLE FP VIA ACCESS PATH FP0.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL = :FPFIL AND
              FPUSR = :FPUSR
        ORDER BY FPFIL ASC, FPUSR ASC
```

IBOFP1

```
OBTAIN A ROW IN TABLE FP VIA ACCESS PATH FP1.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL = :FPFIL
        ORDER BY FPFIL ASC
```

IBOFP2

```
OBTAIN A ROW IN TABLE FP VIA ACCESS PATH FP2.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPUSR = :FPUSR
        ORDER BY FPUSR ASC
```

IBAFP0

```
OBTAIN A ROW IN TABLE FP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FP0.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE (FPFIL > :FPFIL)
          OR ((FPFIL = :FPFIL) AND (FPUSR > :FPUSR))
          OR (FPFIL = :FPFIL AND FPUSR = :FPUSR)
        ORDER BY FPFIL ASC, FPUSR ASC
```

IBAFP1

```
OBTAIN A ROW IN TABLE FP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FP1.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL >= :FPFIL
        ORDER BY FPFIL ASC
```

IBAFP2

```
OBTAIN A ROW IN TABLE FP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH FP2.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPUSR >= :FPUSR
        ORDER BY FPUSR ASC
```

IBEFP1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FP VIA ACCESS PATH FP1.
SAVE THE CURRENT POSITION IN TABLE FP.
FETCH THE NEXT ROW FROM TABLE FP.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBEFP2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FP VIA ACCESS PATH FP2.
SAVE THE CURRENT POSITION IN TABLE FP.
FETCH THE NEXT ROW FROM TABLE FP.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBFFP0

```
OBTAIN THE FIRST ROW OF TABLE FP, ORDERED BY ACCESS PATH FP0.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        ORDER BY FPFIL ASC, FPUSR ASC
```

IBFFP1

```
OBTAIN THE FIRST ROW OF TABLE FP, ORDERED BY ACCESS PATH FP1.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        ORDER BY FPFIL ASC
```

IBFFP2

```
OBTAIN THE FIRST ROW OF TABLE FP, ORDERED BY ACCESS PATH FP2.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        ORDER BY FPUSR ASC
```

IBNFP0

```
OBTAIN THE NEXT ROW OF TABLE FP, ORDERED BY ACCESS PATH FP0.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBNFP1

```
OBTAIN THE NEXT ROW OF TABLE FP, ORDERED BY ACCESS PATH FP1.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBNFP2

```
OBTAIN THE NEXT ROW OF TABLE FP, ORDERED BY ACCESS PATH FP2.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBOFIFP

```
USING COSET FIFP, OBTAIN THE ROW FROM TABLE FI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FP
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        INTO :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :FPFIL
```

IBOUIFP

```
USING COSET UIFP, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE FP
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :FPUSR
```

IBFFIFP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FP WITHIN COSET FIFP, USING
ACCESS PATH FP1.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPFIL = :FIFIL
        ORDER BY FPFIL ASC
```

IBFUIFP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE FP WITHIN COSET UIFP, USING
ACCESS PATH FP2.
        SELECT FPFIL, FPUSR, FPMOD
        FROM FP IN ENGINEERING_DATA_DATABASE
        WHERE FPUSR = :UIUSR
        ORDER BY FPUSR ASC
```

IBNFIFP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FP WITHIN COSET FIFP.
        SET :FPFIL, :FPUSR, :FPMOD
```

IBNUIFP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE FP WITHIN COSET UIFP.
        SET :FPFIL, :FPUSR, :FPMOD
```

File Types (FT) Routines

The FT table defines all application types known to EDL. Each type of engineering data is stored on a file. Every file must be known as one of these types.

IBSFT

```
STORE A NEW ROW IN TABLE FT.
        INSERT INTO FT IN ENGINEERING_DATA_DATABASE
        SET FTFTC = :FTFTC,
            FTNAM = :FTNAM,
            FTAPN = :FTAPN,
            FTLFN = :FTLFN,
            FTCHR = :FTCHR,
            FTMUL = :FTMUL,
            FTLFNR = :FTLFNR,
            FTPRT = :FTPRT,
            FTTYP = :FTTYP
```

IBMFT

```
MODIFY AN EXISTING ROW IN TABLE FT.
        UDPATE FT IN ENGINEERING_DATA_DATABASE
        WHERE FTFTC = :FTFTC
        SET FTFTC = :FTFTC,
            FTNAM = :FTNAM,
            FTAPN = :FTAPN,
            FTLFN = :FTLFN,
            FTCHR = :FTCHR,
            FTMUL = :FTMUL,
            FTLFNR = :FTLFNR,
            FTPRT = :FTPRT,
            FTTYP = :FTTYP
```

IBDFT

```
DELETE AN EXISTING ROW IN TABLE FT.
        DELETE FROM FT IN ENGINEERING_DATA_DATABASE
        WHERE FTFTC = :FTFTC
```

IBOFT0

```
OBTAIN A ROW IN TABLE FT VIA ACCESS PATH FT0.
        SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
          FTTYP
        FROM FT IN ENGINEERING_DATA_DATABASE
        WHERE FTFTC = :FTFTC
        ORDER BY FTFTC ASC
```

IBOFT1

```
OBTAIN A ROW IN TABLE FT VIA ACCESS PATH FT1.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTNAM = :FTNAM
         ORDER BY FTNAM ASC
```

IBOFT2

```
OBTAIN A ROW IN TABLE FT VIA ACCESS PATH FT2.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTLFN = :FTLFN
         ORDER BY FTLFN ASC
```

IBOFT3

```
OBTAIN A ROW IN TABLE FT VIA ACCESS PATH FT3.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTAPN = :FTAPN
         ORDER BY FTAPN ASC, FTNAM ASC
```

IBAFT0

```
OBTAIN A ROW IN TABLE FT USING AN APPROXIMATE KEY VALUE AND ACCESS PATH FT0.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTFTC >= :FTFTC
         ORDER BY FTFTC ASC
```

IBAFT1

```
OBTAIN A ROW IN TABLE FT USING AN APPROXIMATE KEY VALUE AND ACCESS PATH FT1.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTNAM >= :FTNAM
         ORDER BY FTNAM ASC
```

IBAFT2

```
OBTAIN A ROW IN TABLE FT USING AN APPROXIMATE KEY VALUE AND ACCESS PATH FT2.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTLFN >= :FTLFN
         ORDER BY FTLFN ASC
```

IBAFT3

```
OBTAIN A ROW IN TABLE FT USING AN APPROXIMATE KEY VALUE AND ACCESS PATH FT3.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         WHERE FTAPN >= :FTAPN
         ORDER BY FTAPN ASC, FTNAM ASC
```

IBEFT2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FT VIA ACCESS PATH FT2.
SAVE THE CURRENT POSITION IN TABLE FT.
FETCH THE NEXT ROW FROM TABLE FT.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

IBEFT3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE FT VIA ACCESS PATH FT3.
SAVE THE CURRENT POSITION IN TABLE FT.
FETCH THE NEXT ROW FROM TABLE FT.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

IBFFT0

```
OBTAIN THE FIRST ROW OF TABLE FT, ORDERED BY ACCESS PATH FT0.
        SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
          FTTYP
        FROM FT IN ENGINEERING_DATA_DATABASE
        ORDER BY FTFTC ASC
```

IBFFT1

```
OBTAIN THE FIRST ROW OF TABLE FT, ORDERED BY ACCESS PATH FT1.
         SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
           FTTYP
         FROM FT IN ENGINEERING_DATA_DATABASE
         ORDER BY FTNAM ASC
```

IBFFT2

```
OBTAIN THE FIRST ROW OF TABLE FT, ORDERED BY ACCESS PATH FT2.
        SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
          FTTYP
        FROM FT IN ENGINEERING_DATA_DATABASE
        ORDER BY FTLFN ASC
```

IBFFT3

```
OBTAIN THE FIRST ROW OF TABLE FT, ORDERED BY ACCESS PATH FT3.
        SELECT FTFTC, FTNAM, FTAPN, FTLFN, FTCHR, FTMUL, FTLFNR, FTPRT,
          FTTYP
        FROM FT IN ENGINEERING_DATA_DATABASE
        ORDER BY FTAPN ASC, FTNAM ASC
```

IBNFT0

```
OBTAIN THE NEXT ROW OF TABLE FT, ORDERED BY ACCESS PATH FT0.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

IBNFT1

```
OBTAIN THE NEXT ROW OF TABLE FT, ORDERED BY ACCESS PATH FT1.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

IBNFT2

```
OBTAIN THE NEXT ROW OF TABLE FT, ORDERED BY ACCESS PATH FT2.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

IBNFT3

```
OBTAIN THE NEXT ROW OF TABLE FT, ORDERED BY ACCESS PATH FT3.
        SET :FTFTC, :FTNAM, :FTAPN, :FTLFN, :FTCHR, :FTMUL, :FTLFNR,
            :FTPRT, :FTTYP
```

Group Information (GI) Routines

The GI table defines the groups of EDL users.

IBSGI

```
STORE A NEW ROW IN TABLE GI.
        INSERT INTO GI IN ENGINEERING_DATA_DATABASE
        SET GIGRP = :GIGRP,
            GIGRPO = :GIGRPO,
            GIUSRA = :GIUSRA,
            GITTL = :GITTL
```

IBMGI

```
MODIFY AN EXISTING ROW IN TABLE GI.
        UDPATE GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GIGRP
        SET GIGRP = :GIGRP,
            GIGRPO = :GIGRPO,
            GIUSRA = :GIUSRA,
            GITTL = :GITTL
```

IBDGI

```
DELETE AN EXISTING ROW IN TABLE GI.
        DELETE FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GIGRP
```

IBOGI0

```
OBTAIN A ROW IN TABLE GI VIA ACCESS PATH GI0.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GIGRP
        ORDER BY GIGRP ASC
```

IBOGI1

```
OBTAIN A ROW IN TABLE GI VIA ACCESS PATH GI1.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRPO = :GIGRPO AND
              GIGRP = :GIGRP
        ORDER BY GIGRPO ASC, GIGRP ASC
```

IBOGI2

```
OBTAIN A ROW IN TABLE GI VIA ACCESS PATH GI2.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIUSRA = :GIUSRA
        ORDER BY GIUSRA ASC
```

IBAGI0

```
OBTAIN A ROW IN TABLE GI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GI0.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP >= :GIGRP
        ORDER BY GIGRP ASC
```

IBAGI1

```
OBTAIN A ROW IN TABLE GI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GI1.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE (GIGRPO > :GIGRPO)
          OR ((GIGRPO = :GIGRPO) AND (GIGRP > :GIGRP))
          OR (GIGRPO = :GIGRPO AND GIGRP = :GIGRP)
        ORDER BY GIGRPO ASC, GIGRP ASC
```

IBAGI2

```
OBTAIN A ROW IN TABLE GI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GI2.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIUSRA >= :GIUSRA
        ORDER BY GIUSRA ASC
```

IBEGI1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GI VIA ACCESS PATH GI1.
SAVE THE CURRENT POSITION IN TABLE GI.
FETCH THE NEXT ROW FROM TABLE GI.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBEGI2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GI VIA ACCESS PATH GI2.
SAVE THE CURRENT POSITION IN TABLE GI.
FETCH THE NEXT ROW FROM TABLE GI.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBFGI0

```
OBTAIN THE FIRST ROW OF TABLE GI, ORDERED BY ACCESS PATH GI0.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        ORDER BY GIGRP ASC
```

IBFGI1

```
OBTAIN THE FIRST ROW OF TABLE GI, ORDERED BY ACCESS PATH GI1.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        ORDER BY GIGRPO ASC, GIGRP ASC
```

IBFGI2

```
OBTAIN THE FIRST ROW OF TABLE GI, ORDERED BY ACCESS PATH GI2.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        ORDER BY GIUSRA ASC
```

IBNGI0

```
OBTAIN THE NEXT ROW OF TABLE GI, ORDERED BY ACCESS PATH GI0.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBNGI1

```
OBTAIN THE NEXT ROW OF TABLE GI, ORDERED BY ACCESS PATH GI1.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBNGI2

```
OBTAIN THE NEXT ROW OF TABLE GI, ORDERED BY ACCESS PATH GI2.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBOUIGI

```
USING COSET UIGI, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GI
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :GIUSRA
```

IBOGIGI

```
USING COSET GIGI, OBTAIN THE ROW FROM TABLE GI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GI
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        INTO :GIGRP, :GIGRPO, :GIUSRA, :GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GIGRP
```

IBFUIGI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GI WITHIN COSET UIGI, USING
ACCESS PATH GI2.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIUSRA = :UIUSR
        ORDER BY GIUSRA ASC
```

IBFGIGI

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GI WITHIN COSET GIGI, USING
ACCESS PATH GI1.
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRPO = :GIGRPO AND
              GIGRP = :GIGRP
        ORDER BY GIGRPO ASC, GIGRP ASC
```

IBNUIGI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GI WITHIN COSET UIGI.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

IBNGIGI

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GI WITHIN COSET GIGI.
        SET :GIGRP, :GIGRPO, :GIUSRA, :GITTL
```

Group Members (GM) Routines

The GM table lists the individual EDL users that compose a given group.

IBSGM

```
STORE A NEW ROW IN TABLE GM.
        INSERT INTO GM IN ENGINEERING_DATA_DATABASE
        SET GMGRP = :GMGRP,
            GMUSR = :GMUSR
```

IBMGM

```
MODIFY AN EXISTING ROW IN TABLE GM.
        UDPATE GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP = :GMGRP AND
              GMUSR = :GMUSR
        SET GMGRP = :GMGRP,
            GMUSR = :GMUSR
```

IBDGM

```
DELETE AN EXISTING ROW IN TABLE GM.
        DELETE FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP = :GMGRP AND
              GMUSR = :GMUSR
```

IBOGM0

```
OBTAIN A ROW IN TABLE GM VIA ACCESS PATH GM0.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP = :GMGRP AND
              GMUSR = :GMUSR
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBOGM1

```
OBTAIN A ROW IN TABLE GM VIA ACCESS PATH GM1.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP = :GMGRP
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBOGM2

```
OBTAIN A ROW IN TABLE GM VIA ACCESS PATH GM2.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMUSR = :GMUSR
        ORDER BY GMUSR ASC, GMGRP ASC
```

IBAGM0

```
OBTAIN A ROW IN TABLE GM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH GM0.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE (GMGRP > :GMGRP)
          OR ((GMGRP = :GMGRP) AND (GMUSR > :GMUSR))
          OR (GMGRP = :GMGRP AND GMUSR = :GMUSR)
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBAGM1

```
OBTAIN A ROW IN TABLE GM USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GM1.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP >= :GMGRP
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBAGM2

```
OBTAIN A ROW IN TABLE GM USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GM2.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMUSR >= :GMUSR
        ORDER BY GMUSR ASC, GMGRP ASC
```

IBEGM1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GM VIA ACCESS PATH GM1.
SAVE THE CURRENT POSITION IN TABLE GM.
FETCH THE NEXT ROW FROM TABLE GM.
        SET :GMGRP, :GMUSR
```

IBEGM2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GM VIA ACCESS PATH GM2.
SAVE THE CURRENT POSITION IN TABLE GM.
FETCH THE NEXT ROW FROM TABLE GM.
        SET :GMGRP, :GMUSR
```

IBFGM0

```
OBTAIN THE FIRST ROW OF TABLE GM, ORDERED BY ACCESS PATH GM0.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBFGM1

```
OBTAIN THE FIRST ROW OF TABLE GM, ORDERED BY ACCESS PATH GM1.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBFGM2

```
OBTAIN THE FIRST ROW OF TABLE GM, ORDERED BY ACCESS PATH GM2.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        ORDER BY GMUSR ASC, GMGRP ASC
```

IBNGM0

```
OBTAIN THE NEXT ROW OF TABLE GM, ORDERED BY ACCESS PATH GM0.
        SET :GMGRP, :GMUSR
```

IBNGM1

```
OBTAIN THE NEXT ROW OF TABLE GM, ORDERED BY ACCESS PATH GM1.
        SET :GMGRP, :GMUSR
```

IBNGM2

```
OBTAIN THE NEXT ROW OF TABLE GM, ORDERED BY ACCESS PATH GM2.
        SET :GMGRP, :GMUSR
```

IBOUIGM

```
USING COSET UIGM, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GM
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :GMUSR
```

IBOGIGM

```
USING COSET GIGM, OBTAIN THE ROW FROM TABLE GI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GM
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        INTO :GIGRP, :GIGRPO, :GIUSRA, :GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GMGRP
```

IBFUIGM

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GM WITHIN COSET UIGM, USING
ACCESS PATH GM2.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMUSR = :UIUSR
        ORDER BY GMUSR ASC, GMGRP ASC
```

IBFGIGM

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GM WITHIN COSET GIGM, USING
ACCESS PATH GM1.
        SELECT GMGRP, GMUSR
        FROM GM IN ENGINEERING_DATA_DATABASE
        WHERE GMGRP = :GIGRP
        ORDER BY GMGRP ASC, GMUSR ASC
```

IBNUIGM

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GM WITHIN COSET UIGM.
        SET :GMGRP, :GMUSR
```

IBNGIGM

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GM WITHIN COSET GIGM.
        SET :GMGRP, :GMUSR
```

Group Permits (GP) Routines

The GP table defines the file permits for groups.

IBSGP

```
STORE A NEW ROW IN TABLE GP.
        INSERT INTO GP IN ENGINEERING_DATA_DATABASE
        SET GPFIL = :GPFIL,
            GPGRP = :GPGRP,
            GPMOD = :GPMOD
```

IBMGP

```
MODIFY AN EXISTING ROW IN TABLE GP.
        UDPATE GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL = :GPFIL AND
              GPGRP = :GPGRP
        SET GPFIL = :GPFIL,
            GPGRP = :GPGRP,
            GPMOD = :GPMOD
```

IBDGP

```
DELETE AN EXISTING ROW IN TABLE GP.
        DELETE FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL = :GPFIL AND
              GPGRP = :GPGRP
```

IBOGP0

```
OBTAIN A ROW IN TABLE GP VIA ACCESS PATH GP0.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL = :GPFIL AND
              GPGRP = :GPGRP
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBOGP1

```
OBTAIN A ROW IN TABLE GP VIA ACCESS PATH GP1.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL = :GPFIL
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBOGP2

```
OBTAIN A ROW IN TABLE GP VIA ACCESS PATH GP2.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPGRP = :GPGRP
        ORDER BY GPGRP ASC
```

IBAGP0

```
OBTAIN A ROW IN TABLE GP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GP0.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE (GPFIL > :GPFIL)
          OR ((GPFIL = :GPFIL) AND (GPGRP > :GPGRP))
          OR (GPFIL = :GPFIL AND GPGRP = :GPGRP)
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBAGP1

```
OBTAIN A ROW IN TABLE GP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GP1.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL >= :GPFIL
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBAGP2

```
OBTAIN A ROW IN TABLE GP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH GP2.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPGRP >= :GPGRP
        ORDER BY GPGRP ASC
```

IBEGP1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GP VIA ACCESS PATH GP1.
SAVE THE CURRENT POSITION IN TABLE GP.
FETCH THE NEXT ROW FROM TABLE GP.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBEGP2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GP VIA ACCESS PATH GP2.
SAVE THE CURRENT POSITION IN TABLE GP.
FETCH THE NEXT ROW FROM TABLE GP.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBFGP0

```
OBTAIN THE FIRST ROW OF TABLE GP, ORDERED BY ACCESS PATH GP0.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBFGP1

```
OBTAIN THE FIRST ROW OF TABLE GP, ORDERED BY ACCESS PATH GP1.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBFGP2

```
OBTAIN THE FIRST ROW OF TABLE GP, ORDERED BY ACCESS PATH GP2.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        ORDER BY GPGRP ASC
```

IBNGP0

```
OBTAIN THE NEXT ROW OF TABLE GP, ORDERED BY ACCESS PATH GP0.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBNGP1

```
OBTAIN THE NEXT ROW OF TABLE GP, ORDERED BY ACCESS PATH GP1.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBNGP2

```
OBTAIN THE NEXT ROW OF TABLE GP, ORDERED BY ACCESS PATH GP2.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBOFIGP

```
USING COSET FIGP, OBTAIN THE ROW FROM TABLE FI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GP
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        INTO :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :GPFIL
```

IBOGIGP

```
USING COSET GIGP, OBTAIN THE ROW FROM TABLE GI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE GP
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        INTO :GIGRP, :GIGRPO, :GIUSRA, :GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GPGRP
```

IBFFIGP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GP WITHIN COSET FIGP, USING
ACCESS PATH GP1.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPFIL = :FIFIL
        ORDER BY GPFIL ASC, GPGRP ASC
```

IBFGIGP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GP WITHIN COSET GIGP, USING
ACCESS PATH GP2.
        SELECT GPFIL, GPGRP, GPMOD
        FROM GP IN ENGINEERING_DATA_DATABASE
        WHERE GPGRP = :GIGRP
        ORDER BY GPGRP ASC
```

IBNFIGP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GP WITHIN COSET FIGP.
        SET :GPFIL, :GPGRP, :GPMOD
```

IBNGIGP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GP WITHIN COSET GIGP.
        SET :GPFIL, :GPGRP, :GPMOD
```

Group Security Authorization (GS) Routines

The GS table defines the task categories that a user can use.

IBSGS

```
STORE A NEW ROW IN TABLE GS.
        INSERT INTO GS IN ENGINEERING_DATA_DATABASE
        SET GSGRP = :GSGRP,
            GSSEC = :GSSEC
```

IBMGS

```
MODIFY AN EXISTING ROW IN TABLE GS.
        UDPATE GS IN ENGINEERING_DATA_DATABASE
        WHERE GSSEC = :GSSEC AND
              GSGRP = :GSGRP
        SET GSGRP = :GSGRP,
            GSSEC = :GSSEC
```

IBDGS

```
DELETE AN EXISTING ROW IN TABLE GS.
        DELETE FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE GSSEC = :GSSEC AND
              GSGRP = :GSGRP
```

IBOGS0

```
OBTAIN A ROW IN TABLE GS VIA ACCESS PATH GS0.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE GSSEC = :GSSEC AND
              GSGRP = :GSGRP
        ORDER BY GSSEC ASC, GSGRP ASC
```

IBOGS1

```
OBTAIN A ROW IN TABLE GS VIA ACCESS PATH GS1.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE GSGRP = :GSGRP
        ORDER BY GSGRP ASC
```

IBAGS0

```
OBTAIN A ROW IN TABLE GS USING AN APPROXIMATE KEY VALUE AND ACCESS PATH GS0.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE (GSSEC > :GSSEC)
          OR ((GSSEC = :GSSEC) AND (GSGRP > :GSGRP))
          OR (GSSEC = :GSSEC AND GSGRP = :GSGRP)
        ORDER BY GSSEC ASC, GSGRP ASC
```

IBAGS1

```
OBTAIN A ROW IN TABLE GS USING AN APPROXIMATE KEY VALUE AND ACCESS PATH GS1.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE GSGRP >= :GSGRP
        ORDER BY GSGRP ASC
```

IBEGS1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE GS VIA ACCESS PATH GS1.
SAVE THE CURRENT POSITION IN TABLE GS.
FETCH THE NEXT ROW FROM TABLE GS.
        SET :GSGRP, :GSSEC
```

IBFGS0

```
OBTAIN THE FIRST ROW OF TABLE GS, ORDERED BY ACCESS PATH GS0.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        ORDER BY GSSEC ASC, GSGRP ASC
```

IBFGS1

```
OBTAIN THE FIRST ROW OF TABLE GS, ORDERED BY ACCESS PATH GS1.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        ORDER BY GSGRP ASC
```

IBNGS0

```
OBTAIN THE NEXT ROW OF TABLE GS, ORDERED BY ACCESS PATH GS0.
        SET :GSGRP, :GSSEC
```

IBNGS1

```
OBTAIN THE NEXT ROW OF TABLE GS, ORDERED BY ACCESS PATH GS1.
        SET :GSGRP, :GSSEC
```

IBOGIGS

```
USING COSET GIGS, OBTAIN THE ROW FROM TABLE GI THAT OWNS SPECIFIC ROWS IN
MEMBER TABLE GS
        SELECT GIGRP, GIGRPO, GIUSRA, GITTL
        INTO :GIGRP, :GIGRPO, :GIUSRA, :GITTL
        FROM GI IN ENGINEERING_DATA_DATABASE
        WHERE GIGRP = :GSGRP
```

IBFGIGS

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE GS WITHIN COSET GIGS, USING
ACCESS PATH GS1.
        SELECT GSGRP, GSSEC
        FROM GS IN ENGINEERING_DATA_DATABASE
        WHERE GSGRP = :GIGRP
        ORDER BY GSGRP ASC
```

IBNGIGS

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE GS WITHIN COSET GIGS.
        SET :GSGRP, :GSSEC
```

Host Information (HI) Routines

The HI table defines all host machines known to EDL. Every file known to EDL must reside on a known host.

IBSHI

```
STORE A NEW ROW IN TABLE HI.
        INSERT INTO HI IN ENGINEERING_DATA_DATABASE
        SET HIHOS = :HIHOS,
            HIOFF = :HIOFF,
            HIOS = :HIOS
```

IBMHI

```
MODIFY AN EXISTING ROW IN TABLE HI.
        UDPATE HI IN ENGINEERING_DATA_DATABASE
        WHERE HIHOS = :HIHOS
        SET HIHOS = :HIHOS,
            HIOFF = :HIOFF,
            HIOS = :HIOS
```

IBDHI

```
DELETE AN EXISTING ROW IN TABLE HI.
        DELETE FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIHOS = :HIHOS
```

IBOHI0

```
OBTAIN A ROW IN TABLE HI VIA ACCESS PATH HI0.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIHOS = :HIHOS
        ORDER BY HIHOS ASC
```

IBOHI1

```
OBTAIN A ROW IN TABLE HI VIA ACCESS PATH HI1.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIOFF = :HIOFF
        ORDER BY HIOFF ASC
```

IBOHI3

```
OBTAIN A ROW IN TABLE HI VIA ACCESS PATH HI3.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIOS = :HIOS
        ORDER BY HIOS ASC, HIHOS ASC
```

IBAHI0

```
OBTAIN A ROW IN TABLE HI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH HI0.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIHOS >= :HIHOS
        ORDER BY HIHOS ASC
```

IBAHI1

```
OBTAIN A ROW IN TABLE HI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH HI1.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIOFF >= :HIOFF
        ORDER BY HIOFF ASC
```

IBAHI3

```
OBTAIN A ROW IN TABLE HI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH HI3.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        WHERE HIOS >= :HIOS
        ORDER BY HIOS ASC, HIHOS ASC
```

IBEHI3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE HI VIA ACCESS PATH HI3.
SAVE THE CURRENT POSITION IN TABLE HI.
FETCH THE NEXT ROW FROM TABLE HI.
        SET :HIHOS, :HIOFF, :HIOS
```

IBFHI0

```
OBTAIN THE FIRST ROW OF TABLE HI, ORDERED BY ACCESS PATH HI0.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        ORDER BY HIHOS ASC
```

IBFHI1

```
OBTAIN THE FIRST ROW OF TABLE HI, ORDERED BY ACCESS PATH HI1.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        ORDER BY HIOFF ASC
```

IBFHI3

```
OBTAIN THE FIRST ROW OF TABLE HI, ORDERED BY ACCESS PATH HI3.
        SELECT HIHOS, HIOFF, HIOS
        FROM HI IN ENGINEERING_DATA_DATABASE
        ORDER BY HIOS ASC, HIHOS ASC
```

IBNHI0

```
OBTAIN THE NEXT ROW OF TABLE HI, ORDERED BY ACCESS PATH HI0.
        SET :HIHOS, :HIOFF, :HIOS
```

IBNHI1

```
OBTAIN THE NEXT ROW OF TABLE HI, ORDERED BY ACCESS PATH HI1.
        SET :HIHOS, :HIOFF, :HIOS
```

IBNHI3

```
OBTAIN THE NEXT ROW OF TABLE HI, ORDERED BY ACCESS PATH HI3.
        SET :HIHOS, :HIOFF, :HIOS
```

Message Help (MH) Routines

The MH table contains the help text associated with a message in the MI table.

IBSMH

```
STORE A NEW ROW IN TABLE MH.
        INSERT INTO MH IN MENU_DATABASE
        SET MHMNA = :MHMNA,
            MHLIN = :MHLIN,
            MHTXT = :MHTXT
```

IBMMH

```
MODIFY AN EXISTING ROW IN TABLE MH.
        UDPATE MH IN MENU_DATABASE
        WHERE MHLIN = :MHLIN AND
              MHMNA = :MHMNA
        SET MHMNA = :MHMNA,
            MHLIN = :MHLIN,
            MHTXT = :MHTXT
```

IBDMH

```
DELETE AN EXISTING ROW IN TABLE MH.
        DELETE FROM MH IN MENU_DATABASE
        WHERE MHLIN = :MHLIN AND
              MHMNA = :MHMNA
```

IBOMH0

```
OBTAIN A ROW IN TABLE MH VIA ACCESS PATH MH0.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        WHERE MHLIN = :MHLIN AND
              MHMNA = :MHMNA
        ORDER BY MHLIN ASC, MHMNA ASC
```

IBOMH1

```
OBTAIN A ROW IN TABLE MH VIA ACCESS PATH MH1.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        WHERE MHMNA = :MHMNA
        ORDER BY MHMNA ASC, MHLIN ASC
```

IBAMH0

```
OBTAIN A ROW IN TABLE MH USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH MH0.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        WHERE (MHLIN > :MHLIN)
          OR ((MHLIN = :MHLIN) AND (MHMNA > :MHMNA))
          OR (MHLIN = :MHLIN AND MHMNA = :MHMNA)
        ORDER BY MHLIN ASC, MHMNA ASC
```

IBAMH1

```
OBTAIN A ROW IN TABLE MH USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH MH1.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        WHERE MHMNA >= :MHMNA
        ORDER BY MHMNA ASC, MHLIN ASC
```

IBEMH1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE MH VIA ACCESS PATH MH1.
SAVE THE CURRENT POSITION IN TABLE MH.
FETCH THE NEXT ROW FROM TABLE MH.
        SET :MHMNA, :MHLIN, :MHTXT
```

IBFMH0

```
OBTAIN THE FIRST ROW OF TABLE MH, ORDERED BY ACCESS PATH MH0.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        ORDER BY MHLIN ASC, MHMNA ASC
```

IBFMH1

```
OBTAIN THE FIRST ROW OF TABLE MH, ORDERED BY ACCESS PATH MH1.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        ORDER BY MHMNA ASC, MHLIN ASC
```

IBNMH0

```
OBTAIN THE NEXT ROW OF TABLE MH, ORDERED BY ACCESS PATH MH0.
        SET :MHMNA, :MHLIN, :MHTXT
```

IBNMH1

```
OBTAIN THE NEXT ROW OF TABLE MH, ORDERED BY ACCESS PATH MH1.
        SET :MHMNA, :MHLIN, :MHTXT
```

IBOMIMH

```
USING COSET MIMH, OBTAIN THE ROW FROM TABLE MI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE MH
        SELECT MIMNA, MITYP, MISTA, MITTL
        INTO :MIMNA, :MITYP, :MISTA, :MITTL
        FROM MI IN MENU_DATABASE
        WHERE MIMNA = :MHMNA
```

IBFMIMH

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE MH WITHIN COSET MIMH, USING
ACCESS PATH MH1.
        SELECT MHMNA, MHLIN, MHTXT
        FROM MH IN MENU_DATABASE
        WHERE MHMNA = :MIMNA
        ORDER BY MHMNA ASC, MHLIN ASC
```

IBNMIMH

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE MH WITHIN COSET MIMH.
        SET :MHMNA, :MHLIN, :MHTXT
```

Message Information (MI) Routines

The MI table contains all the prompts, menus, and error messages.

IBSMI

```
STORE A NEW ROW IN TABLE MI.
        INSERT INTO MI IN MENU_DATABASE
        SET MIMNA = :MIMNA,
            MITYP = :MITYP,
            MISTA = :MISTA,
            MITTL = :MITTL
```

IBMMI

```
MODIFY AN EXISTING ROW IN TABLE MI.
        UDPATE MI IN MENU_DATABASE
        WHERE MIMNA = :MIMNA
        SET MIMNA = :MIMNA,
            MITYP = :MITYP,
            MISTA = :MISTA,
            MITTL = :MITTL
```

IBDMI

```
DELETE AN EXISTING ROW IN TABLE MI.
        DELETE FROM MI IN MENU_DATABASE
        WHERE MIMNA = :MIMNA
```

IBOMI0

```
OBTAIN A ROW IN TABLE MI VIA ACCESS PATH MI0.
        SELECT MIMNA, MITYP, MISTA, MITTL
        FROM MI IN MENU_DATABASE
        WHERE MIMNA = :MIMNA
        ORDER BY MIMNA ASC
```

IBAMI0

```
OBTAIN A ROW IN TABLE MI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH MI0.
        SELECT MIMNA, MITYP, MISTA, MITTL
        FROM MI IN MENU_DATABASE
        WHERE MIMNA >= :MIMNA
        ORDER BY MIMNA ASC
```

IBFMI0

```
OBTAIN THE FIRST ROW OF TABLE MI, ORDERED BY ACCESS PATH MI0.
        SELECT MIMNA, MITYP, MISTA, MITTL
        FROM MI IN MENU_DATABASE
        ORDER BY MIMNA ASC
```

IBNMI0

```
OBTAIN THE NEXT ROW OF TABLE MI, ORDERED BY ACCESS PATH MI0.
        SET :MIMNA, :MITYP, :MISTA, :MITTL
```

Option Keyword (OK) Routines

The OK table contains the keywords for choosing option menu lines.

IBSOK

```
STORE A NEW ROW IN TABLE OK.
        INSERT INTO OK IN MENU_DATABASE
        SET OKMNA = :OKMNA,
            OKKEY = :OKKEY,
            OKMLN = :OKMLN
```

IBMOK

```
MODIFY AN EXISTING ROW IN TABLE OK.
        UDPATE OK IN MENU_DATABASE
        WHERE OKKEY = :OKKEY AND
              OKMNA = :OKMNA
        SET OKMNA = :OKMNA,
            OKKEY = :OKKEY,
            OKMLN = :OKMLN
```

IBDOK

```
DELETE AN EXISTING ROW IN TABLE OK.
        DELETE FROM OK IN MENU_DATABASE
        WHERE OKKEY = :OKKEY AND
              OKMNA = :OKMNA
```

IBOOK0

```
OBTAIN A ROW IN TABLE OK VIA ACCESS PATH OK0.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        WHERE OKKEY = :OKKEY AND
              OKMNA = :OKMNA
        ORDER BY OKKEY ASC, OKMNA ASC
```

IBOOK1

```
OBTAIN A ROW IN TABLE OK VIA ACCESS PATH OK1.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        WHERE OKMNA = :OKMNA AND
              OKMLN = :OKMLN
        ORDER BY OKMNA ASC, OKMLN ASC, OKKEY ASC
```

IBAOK0

```
OBTAIN A ROW IN TABLE OK USING AN APPROXIMATE KEY VALUE AND ACCESS PATH OK0.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        WHERE (OKKEY > :OKKEY)
          OR ((OKKEY = :OKKEY) AND (OKMNA > :OKMNA))
          OR (OKKEY = :OKKEY AND OKMNA = :OKMNA)
        ORDER BY OKKEY ASC, OKMNA ASC
```

IBAOK1

```
OBTAIN A ROW IN TABLE OK USING AN APPROXIMATE KEY VALUE AND ACCESS PATH OK1.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        WHERE (OKMNA > :OKMNA)
          OR ((OKMNA = :OKMNA) AND (OKMLN > :OKMLN))
          OR (OKMNA = :OKMNA AND OKMLN = :OKMLN)
        ORDER BY OKMNA ASC, OKMLN ASC, OKKEY ASC
```

IBEOK1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE OK VIA ACCESS PATH OK1.
SAVE THE CURRENT POSITION IN TABLE OK.
FETCH THE NEXT ROW FROM TABLE OK.
        SET :OKMNA, :OKKEY, :OKMLN
```

IBFOK0

```
OBTAIN THE FIRST ROW OF TABLE OK, ORDERED BY ACCESS PATH OK0.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        ORDER BY OKKEY ASC, OKMNA ASC
```

IBFOK1

```
OBTAIN THE FIRST ROW OF TABLE OK, ORDERED BY ACCESS PATH OK1.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        ORDER BY OKMNA ASC, OKMLN ASC, OKKEY ASC
```

IBNOK0

```
OBTAIN THE NEXT ROW OF TABLE OK, ORDERED BY ACCESS PATH OK0.
        SET :OKMNA, :OKKEY, :OKMLN
```

IBNOK1

```
OBTAIN THE NEXT ROW OF TABLE OK, ORDERED BY ACCESS PATH OK1.
        SET :OKMNA, :OKKEY, :OKMLN
```

IBOOMOK

```
USING COSET OMOK, OBTAIN THE ROW FROM TABLE OM THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE OK
        SELECT OMMNA, OMMLN, OMTXT
        INTO :OMMNA, :OMMLN, :OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA = :OKMNA AND
              OMMLN = :OKMLN
```

IBFOMOK

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE OK WITHIN COSET OMOK, USING
ACCESS PATH OK1.
        SELECT OKMNA, OKKEY, OKMLN
        FROM OK IN MENU_DATABASE
        WHERE OKMNA = :OMMNA AND
              OKMLN = :OMMLN
        ORDER BY OKMNA ASC, OKMLN ASC, OKKEY ASC
```

IBNOMOK

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE OK WITHIN COSET OMOK.
        SET :OKMNA, :OKKEY, :OKMLN
```

Option Menu (OM) Routines

The OM table contains lines displayed on option menu lines.

IBSOM

```
STORE A NEW ROW IN TABLE OM.
        INSERT INTO OM IN MENU_DATABASE
        SET OMMNA = :OMMNA,
            OMMLN = :OMMLN,
            OMTXT = :OMTXT
```

IBMOM

```
MODIFY AN EXISTING ROW IN TABLE OM.
        UDPATE OM IN MENU_DATABASE
        WHERE OMMNA = :OMMNA AND
              OMMLN = :OMMLN
        SET OMMNA = :OMMNA,
            OMMLN = :OMMLN,
            OMTXT = :OMTXT
```

IBDOM

```
DELETE AN EXISTING ROW IN TABLE OM.
        DELETE FROM OM IN MENU_DATABASE
        WHERE OMMNA = :OMMNA AND
              OMMLN = :OMMLN
```

IBOOM0

```
OBTAIN A ROW IN TABLE OM VIA ACCESS PATH OM0.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA = :OMMNA AND
              OMMLN = :OMMLN
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBOOM1

```
OBTAIN A ROW IN TABLE OM VIA ACCESS PATH OM1.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA = :OMMNA
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBAOM0

```
OBTAIN A ROW IN TABLE OM USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH OM0.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        WHERE (OMMNA > :OMMNA)
          OR ((OMMNA = :OMMNA) AND (OMMLN > :OMMLN))
          OR (OMMNA = :OMMNA AND OMMLN = :OMMLN)
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBAOM1

```
OBTAIN A ROW IN TABLE OM USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH OM1.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA >= :OMMNA
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBEOM1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE OM VIA ACCESS PATH OM1.
SAVE THE CURRENT POSITION IN TABLE OM.
FETCH THE NEXT ROW FROM TABLE OM.
        SET :OMMNA, :OMMLN, :OMTXT
```

IBFOM0

```
OBTAIN THE FIRST ROW OF TABLE OM, ORDERED BY ACCESS PATH OM0.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBFOM1

```
OBTAIN THE FIRST ROW OF TABLE OM, ORDERED BY ACCESS PATH OM1.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBNOM0

```
OBTAIN THE NEXT ROW OF TABLE OM, ORDERED BY ACCESS PATH OM0.
        SET :OMMNA, :OMMLN, :OMTXT
```

IBNOM1

```
OBTAIN THE NEXT ROW OF TABLE OM, ORDERED BY ACCESS PATH OM1.
        SET :OMMNA, :OMMLN, :OMTXT
```

IBOMIOM

```
USING COSET MIOM, OBTAIN THE ROW FROM TABLE MI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE OM
        SELECT MIMNA, MITYP, MISTA, MITTL
        INTO :MIMNA, :MITYP, :MISTA, :MITTL
        FROM MI IN MENU_DATABASE
        WHERE MIMNA = :OMMNA
```

IBFMIOM

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE OM WITHIN COSET MIOM, USING
ACCESS PATH OM1.
        SELECT OMMNA, OMMLN, OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA = :MIMNA
        ORDER BY OMMNA ASC, OMMLN ASC
```

IBNMIOM

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE OM WITHIN COSET MIOM.
        SET :OMMNA, :OMMLN, :OMTXT
```

Option Value (OV) Routines

The OV table contains the values returned to the program when an option menu line is selected.

IBSOV

```
STORE A NEW ROW IN TABLE OV.
        INSERT INTO OV IN MENU_DATABASE
        SET OVMNA = :OVMNA,
            OVMLN = :OVMLN,
            OVPOS = :OVPOS,
            OVVAL = :OVVAL
```

IBMOV

```
MODIFY AN EXISTING ROW IN TABLE OV.
        UDPATE OV IN MENU_DATABASE
        WHERE OVMNA = :OVMNA AND
              OVMLN = :OVMLN AND
              OVPOS = :OVPOS
        SET OVMNA = :OVMNA,
            OVMLN = :OVMLN,
            OVPOS = :OVPOS,
            OVVAL = :OVVAL
```

IBDOV

```
DELETE AN EXISTING ROW IN TABLE OV.
        DELETE FROM OV IN MENU_DATABASE
        WHERE OVMNA = :OVMNA AND
              OVMLN = :OVMLN AND
              OVPOS = :OVPOS
```

IBOOV0

```
OBTAIN A ROW IN TABLE OV VIA ACCESS PATH OV0.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        WHERE OVMNA = :OVMNA AND
              OVMLN = :OVMLN AND
              OVPOS = :OVPOS
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBOOV1

```
OBTAIN A ROW IN TABLE OV VIA ACCESS PATH OV1.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        WHERE OVMNA = :OVMNA AND
              OVMLN = :OVMLN
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBAOV0

```
OBTAIN A ROW IN TABLE OV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH OV0.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        WHERE (OVMNA > :OVMNA)
          OR ((OVMNA = :OVMNA) AND (OVMLN > :OVMLN))
          OR ((OVMNA = :OVMNA AND OVMLN = :OVMLN) AND (OVPOS > :OVPOS))
          OR (OVMNA = :OVMNA AND OVMLN = :OVMLN AND OVPOS = :OVPOS)
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBAOV1

```
OBTAIN A ROW IN TABLE OV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH OV1.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        WHERE (OVMNA > :OVMNA)
          OR ((OVMNA = :OVMNA) AND (OVMLN > :OVMLN))
          OR (OVMNA = :OVMNA AND OVMLN = :OVMLN)
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBEOV1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE OV VIA ACCESS PATH OV1.
SAVE THE CURRENT POSITION IN TABLE OV.
FETCH THE NEXT ROW FROM TABLE OV.
        SET :OVMNA, :OVMLN, :OVPOS, :OVVAL
```

IBFOV0

```
OBTAIN THE FIRST ROW OF TABLE OV, ORDERED BY ACCESS PATH OV0.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBFOV1

```
OBTAIN THE FIRST ROW OF TABLE OV, ORDERED BY ACCESS PATH OV1.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBNOV0

```
OBTAIN THE NEXT ROW OF TABLE OV, ORDERED BY ACCESS PATH OV0.
        SET :OVMNA, :OVMLN, :OVPOS, :OVVAL
```

IBNOV1

```
OBTAIN THE NEXT ROW OF TABLE OV, ORDERED BY ACCESS PATH OV1.
        SET :OVMNA, :OVMLN, :OVPOS, :OVVAL
```

IBOOMOV

```
USING COSET OMOV, OBTAIN THE ROW FROM TABLE OM THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE OV
        SELECT OMMNA, OMMLN, OMTXT
        INTO :OMMNA, :OMMLN, :OMTXT
        FROM OM IN MENU_DATABASE
        WHERE OMMNA = :OVMNA AND
              OMMLN = :OVMLN
```

IBFOMOV

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE OV WITHIN COSET OMOV, USING
ACCESS PATH OV1.
        SELECT OVMNA, OVMLN, OVPOS, OVVAL
        FROM OV IN MENU_DATABASE
        WHERE OVMNA = :OMMNA AND
              OVMLN = :OMMLN
        ORDER BY OVMNA ASC, OVMLN ASC, OVPOS ASC
```

IBNOMOV

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE OV WITHIN COSET OMOV.
        SET :OVMNA, :OVMLN, :OVPOS, :OVVAL
```

Parts Data (PD) Routines

The PD table contains the association between part number and engineering data.

IBSPD

```
STORE A NEW ROW IN TABLE PD.
        INSERT INTO PD IN ENGINEERING_DATA_DATABASE
        SET PDPRT = :PDPRT,
            PDEDN = :PDEDN
```

IBMPD

```
MODIFY AN EXISTING ROW IN TABLE PD.
        UDPATE PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT = :PDPRT AND
              PDEDN = :PDEDN
        SET PDPRT = :PDPRT,
            PDEDN = :PDEDN
```

IBDPD

```
DELETE AN EXISTING ROW IN TABLE PD.
        DELETE FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT = :PDPRT AND
              PDEDN = :PDEDN
```

IBOPD0

```
OBTAIN A ROW IN TABLE PD VIA ACCESS PATH PD0.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT = :PDPRT AND
              PDEDN = :PDEDN
        ORDER BY PDPRT ASC, PDEDN ASC
```

IBOPD1

```
OBTAIN A ROW IN TABLE PD VIA ACCESS PATH PD1.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT = :PDPRT
        ORDER BY PDPRT ASC
```

IBOPD2

```
OBTAIN A ROW IN TABLE PD VIA ACCESS PATH PD2.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDEDN = :PDEDN
        ORDER BY PDEDN ASC, PDPRT ASC
```

IBAPD0

```
OBTAIN A ROW IN TABLE PD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PD0.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE (PDPRT > :PDPRT)
          OR ((PDPRT = :PDPRT) AND (PDEDN > :PDEDN))
          OR (PDPRT = :PDPRT AND PDEDN = :PDEDN)
        ORDER BY PDPRT ASC, PDEDN ASC
```

IBAPD1

```
OBTAIN A ROW IN TABLE PD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PD1.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT >= :PDPRT
        ORDER BY PDPRT ASC
```

IBAPD2

```
OBTAIN A ROW IN TABLE PD USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PD2.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDEDN >= :PDEDN
        ORDER BY PDEDN ASC, PDPRT ASC
```

IBEPD1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PD VIA ACCESS PATH PD1.
SAVE THE CURRENT POSITION IN TABLE PD.
FETCH THE NEXT ROW FROM TABLE PD.
        SET :PDPRT, :PDEDN
```

IBEPD2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PD VIA ACCESS PATH PD2.
SAVE THE CURRENT POSITION IN TABLE PD.
FETCH THE NEXT ROW FROM TABLE PD.
        SET :PDPRT, :PDEDN
```

IBFPD0

```
OBTAIN THE FIRST ROW OF TABLE PD, ORDERED BY ACCESS PATH PD0.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        ORDER BY PDPRT ASC, PDEDN ASC
```

IBFPD1

```
OBTAIN THE FIRST ROW OF TABLE PD, ORDERED BY ACCESS PATH PD1.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        ORDER BY PDPRT ASC
```

IBFPD2

```
OBTAIN THE FIRST ROW OF TABLE PD, ORDERED BY ACCESS PATH PD2.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        ORDER BY PDEDN ASC, PDPRT ASC
```

IBNPD0

```
OBTAIN THE NEXT ROW OF TABLE PD, ORDERED BY ACCESS PATH PD0.
        SET :PDPRT, :PDEDN
```

IBNPD1

```
OBTAIN THE NEXT ROW OF TABLE PD, ORDERED BY ACCESS PATH PD1.
        SET :PDPRT, :PDEDN
```

IBNPD2

```
OBTAIN THE NEXT ROW OF TABLE PD, ORDERED BY ACCESS PATH PD2.
        SET :PDPRT, :PDEDN
```

IBODIPD

```
USING COSET DIPD, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PD
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :PDEDN
```

IBOPIPD

```
USING COSET PIPD, OBTAIN THE ROW FROM TABLE PI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PD
        SELECT PIPRT, PITTL
        INTO :PIPRT, :PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PDPRT
```

IBFDIPD

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PD WITHIN COSET DIPD, USING
ACCESS PATH PD2.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDEDN = :DIEDN
        ORDER BY PDEDN ASC, PDPRT ASC
```

IBFPIPD

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PD WITHIN COSET PIPD, USING
ACCESS PATH PD1.
        SELECT PDPRT, PDEDN
        FROM PD IN ENGINEERING_DATA_DATABASE
        WHERE PDPRT = :PIPRT
        ORDER BY PDPRT ASC
```

IBNDIPD

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PD WITHIN COSET DIPD.
        SET :PDPRT, :PDEDN
```

IBNPIPD

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PD WITHIN COSET PIPD.
        SET :PDPRT, :PDEDN
```

Part Family (PF) Routines

The PF table contains the association between part numbers and part family codes.

IBSPF

```
STORE A NEW ROW IN TABLE PF.
        INSERT INTO PF IN ENGINEERING_DATA_DATABASE
        SET PFPRT = :PFPRT,
            PFFAM = :PFFAM
```

IBMPF

```
MODIFY AN EXISTING ROW IN TABLE PF.
        UDPATE PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM = :PFFAM AND
              PFPRT = :PFPRT
        SET PFPRT = :PFPRT,
            PFFAM = :PFFAM
```

IBDPF

```
DELETE AN EXISTING ROW IN TABLE PF.
        DELETE FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM = :PFFAM AND
              PFPRT = :PFPRT
```

IBOPF0

```
OBTAIN A ROW IN TABLE PF VIA ACCESS PATH PF0.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM = :PFFAM AND
              PFPRT = :PFPRT
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBOPF1

```
OBTAIN A ROW IN TABLE PF VIA ACCESS PATH PF1.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM = :PFFAM
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBOPF2

```
OBTAIN A ROW IN TABLE PF VIA ACCESS PATH PF2.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFPRT = :PFPRT
        ORDER BY PFPRT ASC, PFFAM ASC
```

IBAPF0

```
OBTAIN A ROW IN TABLE PF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PF0.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE (PFFAM > :PFFAM)
          OR ((PFFAM = :PFFAM) AND (PFPRT > :PFPRT))
          OR (PFFAM = :PFFAM AND PFPRT = :PFPRT)
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBAPF1

```
OBTAIN A ROW IN TABLE PF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PF1.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM >= :PFFAM
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBAPF2

```
OBTAIN A ROW IN TABLE PF USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PF2.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFPRT >= :PFPRT
        ORDER BY PFPRT ASC, PFFAM ASC
```

IBEPF1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PF VIA ACCESS PATH PF1.
SAVE THE CURRENT POSITION IN TABLE PF.
FETCH THE NEXT ROW FROM TABLE PF.
        SET :PFPRT, :PFFAM
```

IBEPF2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PF VIA ACCESS PATH PF2.
SAVE THE CURRENT POSITION IN TABLE PF.
FETCH THE NEXT ROW FROM TABLE PF.
        SET :PFPRT, :PFFAM
```

IBFPF0

```
OBTAIN THE FIRST ROW OF TABLE PF, ORDERED BY ACCESS PATH PF0.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBFPF1

```
OBTAIN THE FIRST ROW OF TABLE PF, ORDERED BY ACCESS PATH PF1.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBFPF2

```
OBTAIN THE FIRST ROW OF TABLE PF, ORDERED BY ACCESS PATH PF2.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        ORDER BY PFPRT ASC, PFFAM ASC
```

IBNPF0

```
OBTAIN THE NEXT ROW OF TABLE PF, ORDERED BY ACCESS PATH PF0.
        SET :PFPRT, :PFFAM
```

IBNPF1

```
OBTAIN THE NEXT ROW OF TABLE PF, ORDERED BY ACCESS PATH PF1.
        SET :PFPRT, :PFFAM
```

IBNPF2

```
OBTAIN THE NEXT ROW OF TABLE PF, ORDERED BY ACCESS PATH PF2.
        SET :PFPRT, :PFFAM
```

IBOPIPF

```
USING COSET PIPF, OBTAIN THE ROW FROM TABLE PI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PF
        SELECT PIPRT, PITTL
        INTO :PIPRT, :PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PFPRT
```

IBOFMPF

```
USING COSET FMPF, OBTAIN THE ROW FROM TABLE FM THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PF
        SELECT FMFAM, FMTTL
        INTO :FMFAM, :FMTTL
        FROM FM IN ENGINEERING_DATA_DATABASE
        WHERE FMFAM = :PFFAM
```

IBFPIPF

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PF WITHIN COSET PIPF, USING
ACCESS PATH PF2.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFPRT = :PIPRT
        ORDER BY PFPRT ASC, PFFAM ASC
```

IBFFMPF

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PF WITHIN COSET FMPF, USING
ACCESS PATH PF1.
        SELECT PFPRT, PFFAM
        FROM PF IN ENGINEERING_DATA_DATABASE
        WHERE PFFAM = :FMFAM
        ORDER BY PFFAM ASC, PFPRT ASC
```

IBNPIPF

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PF WITHIN COSET PIPF.
        SET :PFPRT, :PFFAM
```

IBNFMPF

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PF WITHIN COSET FMPF.
        SET :PFPRT, :PFFAM
```

Part Information (PI) Routines

The PI table defines the part numbers known to EDL.

IBSPI

```
STORE A NEW ROW IN TABLE PI.
        INSERT INTO PI IN ENGINEERING_DATA_DATABASE
        SET PIPRT = :PIPRT,
            PITTL = :PITTL
```

IBMPI

```
MODIFY AN EXISTING ROW IN TABLE PI.
        UDPATE PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PIPRT
        SET PIPRT = :PIPRT,
            PITTL = :PITTL
```

IBDPI

```
DELETE AN EXISTING ROW IN TABLE PI.
        DELETE FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PIPRT
```

IBOPI0

```
OBTAIN A ROW IN TABLE PI VIA ACCESS PATH PI0.
        SELECT PIPRT, PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PIPRT
        ORDER BY PIPRT ASC
```

IBAPI0

```
OBTAIN A ROW IN TABLE PI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PI0.
        SELECT PIPRT, PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT >= :PIPRT
        ORDER BY PIPRT ASC
```

IBFPI0

```
OBTAIN THE FIRST ROW OF TABLE PI, ORDERED BY ACCESS PATH PI0.
        SELECT PIPRT, PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        ORDER BY PIPRT ASC
```

IBNPI0

```
OBTAIN THE NEXT ROW OF TABLE PI, ORDERED BY ACCESS PATH PI0.
        SET :PIPRT, :PITTL
```

Pending Permits (PP) Routines

IBSPP

```
STORE A NEW ROW IN TABLE PP.
        INSERT INTO PP IN ENGINEERING_DATA_DATABASE
        SET PPFIL = :PPFIL,
            PPUUN = :PPUUN,
            PPMOD = :PPMOD,
            PPFUN = :PPFUN
```

IBMPP

```
MODIFY AN EXISTING ROW IN TABLE PP.
        UDPATE PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFIL = :PPFIL AND
              PPUUN = :PPUUN
        SET PPFIL = :PPFIL,
            PPUUN = :PPUUN,
            PPMOD = :PPMOD,
            PPFUN = :PPFUN
```

IBDPP

```
DELETE AN EXISTING ROW IN TABLE PP.
        DELETE FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFIL = :PPFIL AND
              PPUUN = :PPUUN
```

IBOPP0

```
OBTAIN A ROW IN TABLE PP VIA ACCESS PATH PP0.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFIL = :PPFIL AND
              PPUUN = :PPUUN
        ORDER BY PPFIL ASC, PPUUN ASC
```

IBOPP1

```
OBTAIN A ROW IN TABLE PP VIA ACCESS PATH PP1.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPUUN = :PPUUN
        ORDER BY PPUUN ASC
```

IBOPP2

```
OBTAIN A ROW IN TABLE PP VIA ACCESS PATH PP2.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFUN = :PPFUN
        ORDER BY PPFUN ASC
```

IBOPP3

```
OBTAIN A ROW IN TABLE PP VIA ACCESS PATH PP3.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFIL = :PPFIL
        ORDER BY PPFIL ASC
```

IBAPP0

```
OBTAIN A ROW IN TABLE PP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PP0.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE (PPFIL > :PPFIL)
          OR ((PPFIL = :PPFIL) AND (PPUUN > :PPUUN))
          OR (PPFIL = :PPFIL AND PPUUN = :PPUUN)
        ORDER BY PPFIL ASC, PPUUN ASC
```

IBAPP1

```
OBTAIN A ROW IN TABLE PP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PP1.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPUUN >= :PPUUN
        ORDER BY PPUUN ASC
```

IBAPP2

```
OBTAIN A ROW IN TABLE PP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PP2.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFUN >= :PPFUN
        ORDER BY PPFUN ASC
```

IBAPP3

```
OBTAIN A ROW IN TABLE PP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PP3.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        WHERE PPFIL >= :PPFIL
        ORDER BY PPFIL ASC
```

IBEPP1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PP VIA ACCESS PATH PP1.
SAVE THE CURRENT POSITION IN TABLE PP.
FETCH THE NEXT ROW FROM TABLE PP.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBEPP2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PP VIA ACCESS PATH PP2.
SAVE THE CURRENT POSITION IN TABLE PP.
FETCH THE NEXT ROW FROM TABLE PP.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBEPP3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PP VIA ACCESS PATH PP3.
SAVE THE CURRENT POSITION IN TABLE PP.
FETCH THE NEXT ROW FROM TABLE PP.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBFPP0

```
OBTAIN THE FIRST ROW OF TABLE PP, ORDERED BY ACCESS PATH PP0.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        ORDER BY PPFIL ASC, PPUUN ASC
```

IBFPP1

```
OBTAIN THE FIRST ROW OF TABLE PP, ORDERED BY ACCESS PATH PP1.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        ORDER BY PPUUN ASC
```

IBFPP2

```
OBTAIN THE FIRST ROW OF TABLE PP, ORDERED BY ACCESS PATH PP2.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        ORDER BY PPFUN ASC
```

IBFPP3

```
OBTAIN THE FIRST ROW OF TABLE PP, ORDERED BY ACCESS PATH PP3.
        SELECT PPFIL, PPUUN, PPMOD, PPFUN
        FROM PP IN ENGINEERING_DATA_DATABASE
        ORDER BY PPFIL ASC
```

IBNPP0

```
OBTAIN THE NEXT ROW OF TABLE PP, ORDERED BY ACCESS PATH PP0.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBNPP1

```
OBTAIN THE NEXT ROW OF TABLE PP, ORDERED BY ACCESS PATH PP1.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBNPP2

```
OBTAIN THE NEXT ROW OF TABLE PP, ORDERED BY ACCESS PATH PP2.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

IBNPP3

```
OBTAIN THE NEXT ROW OF TABLE PP, ORDERED BY ACCESS PATH PP3.
        SET :PPFIL, :PPUUN, :PPMOD, :PPFUN
```

Part Vendors (PV) Routines

The PV table contains the associations of part numbers with vendors.

IBSPV

```
STORE A NEW ROW IN TABLE PV.
        INSERT INTO PV IN ENGINEERING_DATA_DATABASE
        SET PVPRT = :PVPRT,
            PVVEN = :PVVEN
```

IBMPV

```
MODIFY AN EXISTING ROW IN TABLE PV.
        UDPATE PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT = :PVPRT AND
              PVVEN = :PVVEN
        SET PVPRT = :PVPRT,
            PVVEN = :PVVEN
```

IBDPV

```
DELETE AN EXISTING ROW IN TABLE PV.
        DELETE FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT = :PVPRT AND
              PVVEN = :PVVEN
```

IBOPV0

```
OBTAIN A ROW IN TABLE PV VIA ACCESS PATH PV0.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT = :PVPRT AND
              PVVEN = :PVVEN
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBOPV1

```
OBTAIN A ROW IN TABLE PV VIA ACCESS PATH PV1.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT = :PVPRT
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBOPV2

```
OBTAIN A ROW IN TABLE PV VIA ACCESS PATH PV2.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVVEN = :PVVEN
        ORDER BY PVVEN ASC, PVPRT ASC
```

IBAPV0

```
OBTAIN A ROW IN TABLE PV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PV0.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE (PVPRT > :PVPRT)
          OR ((PVPRT = :PVPRT) AND (PVVEN > :PVVEN))
          OR (PVPRT = :PVPRT AND PVVEN = :PVVEN)
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBAPV1

```
OBTAIN A ROW IN TABLE PV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PV1.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT >= :PVPRT
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBAPV2

```
OBTAIN A ROW IN TABLE PV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH PV2.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVVEN >= :PVVEN
        ORDER BY PVVEN ASC, PVPRT ASC
```

IBEPV1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PV VIA ACCESS PATH PV1.
SAVE THE CURRENT POSITION IN TABLE PV.
FETCH THE NEXT ROW FROM TABLE PV.
        SET :PVPRT, :PVVEN
```

IBEPV2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE PV VIA ACCESS PATH PV2.
SAVE THE CURRENT POSITION IN TABLE PV.
FETCH THE NEXT ROW FROM TABLE PV.
        SET :PVPRT, :PVVEN
```

IBFPV0

```
OBTAIN THE FIRST ROW OF TABLE PV, ORDERED BY ACCESS PATH PV0.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBFPV1

```
OBTAIN THE FIRST ROW OF TABLE PV, ORDERED BY ACCESS PATH PV1.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBFPV2

```
OBTAIN THE FIRST ROW OF TABLE PV, ORDERED BY ACCESS PATH PV2.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        ORDER BY PVVEN ASC, PVPRT ASC
```

IBNPV0

```
OBTAIN THE NEXT ROW OF TABLE PV, ORDERED BY ACCESS PATH PV0.
        SET :PVPRT, :PVVEN
```

IBNPV1

```
OBTAIN THE NEXT ROW OF TABLE PV, ORDERED BY ACCESS PATH PV1.
        SET :PVPRT, :PVVEN
```

IBNPV2

```
OBTAIN THE NEXT ROW OF TABLE PV, ORDERED BY ACCESS PATH PV2.
        SET :PVPRT, :PVVEN
```

IBOVIPV

```
USING COSET VIPV, OBTAIN THE ROW FROM TABLE VI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PV
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        INTO :VIVEN, :VINAM, :VISTR, :VICTY, :VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VIVEN = :PVVEN
```

IBOPIPV

```
USING COSET PIPV, OBTAIN THE ROW FROM TABLE PI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE PV
        SELECT PIPRT, PITTL
        INTO :PIPRT, :PITTL
        FROM PI IN ENGINEERING_DATA_DATABASE
        WHERE PIPRT = :PVPRT
```

IBFVIPV

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PV WITHIN COSET VIPV, USING
ACCESS PATH PV2.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVVEN = :VIVEN
        ORDER BY PVVEN ASC, PVPRT ASC
```

IBFPIPV

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE PV WITHIN COSET PIPV, USING
ACCESS PATH PV1.
        SELECT PVPRT, PVVEN
        FROM PV IN ENGINEERING_DATA_DATABASE
        WHERE PVPRT = :PIPRT
        ORDER BY PVPRT ASC, PVVEN ASC
```

IBNVIPV

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PV WITHIN COSET VIPV.
        SET :PVPRT, :PVVEN
```

IBNPIPV

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE PV WITHIN COSET PIPV.
        SET :PVPRT, :PVVEN
```

Release Authorization (RA) Routines

The RA table contains the release information for engineering data.

IBSRA

```
STORE A NEW ROW IN TABLE RA.
        INSERT INTO RA IN ENGINEERING_DATA_DATABASE
        SET RAREL = :RAREL,
            RAEDN = :RAEDN,
            RAEDNC = :RAEDNC,
            RAUSR = :RAUSR,
            RASTA = :RASTA,
            RADAT = :RADAT
```

IBMRA

```
MODIFY AN EXISTING ROW IN TABLE RA.
        UDPATE RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RAREL AND
              RAEDN = :RAEDN
        SET RAREL = :RAREL,
            RAEDN = :RAEDN,
            RAEDNC = :RAEDNC,
            RAUSR = :RAUSR,
            RASTA = :RASTA,
            RADAT = :RADAT
```

IBDRA

```
DELETE AN EXISTING ROW IN TABLE RA.
        DELETE FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RAREL AND
              RAEDN = :RAEDN
```

IBORA0

```
OBTAIN A ROW IN TABLE RA VIA ACCESS PATH RA0.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RAREL AND
              RAEDN = :RAEDN
        ORDER BY RAREL ASC, RAEDN ASC
```

IBORA1

```
OBTAIN A ROW IN TABLE RA VIA ACCESS PATH RA1.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDN = :RAEDN
        ORDER BY RAEDN ASC
```

IBORA2

```
OBTAIN A ROW IN TABLE RA VIA ACCESS PATH RA2.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RAREL
        ORDER BY RAREL ASC
```

IBORA3

```
OBTAIN A ROW IN TABLE RA VIA ACCESS PATH RA3.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAUSR = :RAUSR
        ORDER BY RAUSR ASC
```

IBORA4

```
OBTAIN A ROW IN TABLE RA VIA ACCESS PATH RA4.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDNC = :RAEDNC
        ORDER BY RAEDNC ASC
```

IBARA0

```
OBTAIN A ROW IN TABLE RA USING AN APPROXIMATE KEY VALUE AND ACCESS PATH RA0.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE (RAREL > :RAREL)
          OR ((RAREL = :RAREL) AND (RAEDN > :RAEDN))
          OR (RAREL = :RAREL AND RAEDN = :RAEDN)
        ORDER BY RAREL ASC, RAEDN ASC
```

IBARA1

```
OBTAIN A ROW IN TABLE RA USING AN APPROXIMATE KEY VALUE AND ACCESS PATH RA1.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDN >= :RAEDN
        ORDER BY RAEDN ASC
```

IBARA2

```
OBTAIN A ROW IN TABLE RA USING AN APPROXIMATE KEY VALUE AND ACCESS PATH RA2.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL >= :RAREL
        ORDER BY RAREL ASC
```

IBARA3

```
OBTAIN A ROW IN TABLE RA USING AN APPROXIMATE KEY VALUE AND ACCESS PATH RA3.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAUSR >= :RAUSR
        ORDER BY RAUSR ASC
```

IBARA4

```
OBTAIN A ROW IN TABLE RA USING AN APPROXIMATE KEY VALUE AND ACCESS PATH RA4.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDNC >= :RAEDNC
        ORDER BY RAEDNC ASC
```

IBERA0

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RA VIA ACCESS PATH RA0.
SAVE THE CURRENT POSITION IN TABLE RA.
FETCH THE NEXT ROW FROM TABLE RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBERA1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RA VIA ACCESS PATH RA1.
SAVE THE CURRENT POSITION IN TABLE RA.
FETCH THE NEXT ROW FROM TABLE RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBERA2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RA VIA ACCESS PATH RA2.
SAVE THE CURRENT POSITION IN TABLE RA.
FETCH THE NEXT ROW FROM TABLE RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBERA3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RA VIA ACCESS PATH RA3.
SAVE THE CURRENT POSITION IN TABLE RA.
FETCH THE NEXT ROW FROM TABLE RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBERA4

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RA VIA ACCESS PATH RA4.
SAVE THE CURRENT POSITION IN TABLE RA.
FETCH THE NEXT ROW FROM TABLE RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBFRA0

```
OBTAIN THE FIRST ROW OF TABLE RA, ORDERED BY ACCESS PATH RA0.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        ORDER BY RAREL ASC, RAEDN ASC
```

IBFRA1

```
OBTAIN THE FIRST ROW OF TABLE RA, ORDERED BY ACCESS PATH RA1.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        ORDER BY RAEDN ASC
```

IBFRA2

```
OBTAIN THE FIRST ROW OF TABLE RA, ORDERED BY ACCESS PATH RA2.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        ORDER BY RAREL ASC
```

IBFRA3

```
OBTAIN THE FIRST ROW OF TABLE RA, ORDERED BY ACCESS PATH RA3.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        ORDER BY RAUSR ASC
```

IBFRA4

```
OBTAIN THE FIRST ROW OF TABLE RA, ORDERED BY ACCESS PATH RA4.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        ORDER BY RAEDNC ASC
```

IBNRA0

```
OBTAIN THE NEXT ROW OF TABLE RA, ORDERED BY ACCESS PATH RA0.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNRA1

```
OBTAIN THE NEXT ROW OF TABLE RA, ORDERED BY ACCESS PATH RA1.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNRA2

```
OBTAIN THE NEXT ROW OF TABLE RA, ORDERED BY ACCESS PATH RA2.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNRA3

```
OBTAIN THE NEXT ROW OF TABLE RA, ORDERED BY ACCESS PATH RA3.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNRA4

```
OBTAIN THE NEXT ROW OF TABLE RA, ORDERED BY ACCESS PATH RA4.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBD2RA

```
USING COSET D2RA, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RA
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :RAEDN
```

IBOUIRA

```
USING COSET UIRA, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RA
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :RAUSR
```

IBORPRA

```
USING COSET RPRA, OBTAIN THE ROW FROM TABLE RP THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RA
        SELECT RPREL
        INTO :RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RAREL
```

IBODIRA

```
USING COSET DIRA, OBTAIN THE ROW FROM TABLE DI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RA
        SELECT DIEDN, DIFIL, DINAM, DISID, DIADT, DIEDT, DIUSR, DIREV,
          DISTA, DIDATC, DIDATM, DIDATR, DITTL, DITIMC, DITIMM, DITIMR
        INTO :DIEDN, :DIFIL, :DINAM, :DISID, :DIADT, :DIEDT, :DIUSR,
            :DIREV, :DISTA, :DIDATC, :DIDATM, :DIDATR, :DITTL, :DITIMC,
            :DITIMM, :DITIMR
        FROM DI IN ENGINEERING_DATA_DATABASE
        WHERE DIEDN = :RAEDN
```

IBFD2RA

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RA WITHIN COSET D2RA, USING
ACCESS PATH RA4.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDNC = :DIEDN
        ORDER BY RAEDNC ASC
```

IBFUIRA

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RA WITHIN COSET UIRA, USING
ACCESS PATH RA3.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAUSR = :UIUSR
        ORDER BY RAUSR ASC
```

IBFRPRA

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RA WITHIN COSET RPRA, USING
ACCESS PATH RA2.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RPREL
        ORDER BY RAREL ASC
```

IBFDIRA

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RA WITHIN COSET DIRA, USING
ACCESS PATH RA1.
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAEDN = :DIEDN
        ORDER BY RAEDN ASC
```

IBND2RA

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RA WITHIN COSET D2RA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNUIRA

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RA WITHIN COSET UIRA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNRPRA

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RA WITHIN COSET RPRA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

IBNDIRA

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RA WITHIN COSET DIRA.
        SET :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
```

Release Procedure (RP) Routines

The RP table contains the names of procedures used to release engineering data.

IBSRP

```
STORE A NEW ROW IN TABLE RP.
        INSERT INTO RP IN ENGINEERING_DATA_DATABASE
        SET RPREL = :RPREL
```

IBMRP

```
MODIFY AN EXISTING ROW IN TABLE RP.
        UDPATE RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RPREL
        SET RPREL = :RPREL
```

IBDRP

```
DELETE AN EXISTING ROW IN TABLE RP.
        DELETE FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RPREL
```

IBORP0

```
OBTAIN A ROW IN TABLE RP VIA ACCESS PATH RP0.
        SELECT RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RPREL
        ORDER BY RPREL ASC
```

IBARP0

```
OBTAIN A ROW IN TABLE RP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RP0.
        SELECT RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL >= :RPREL
        ORDER BY RPREL ASC
```

IBFRP0

```
OBTAIN THE FIRST ROW OF TABLE RP, ORDERED BY ACCESS PATH RP0.
        SELECT RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        ORDER BY RPREL ASC
```

IBNRP0

```
OBTAIN THE NEXT ROW OF TABLE RP, ORDERED BY ACCESS PATH RP0.
        SET :RPREL
```

Review Responsibility (RR) Routines

The RR table defines the responsibility of the reviewers involved in a given release procedure and the order in which they review the data.

IBSRR

```
STORE A NEW ROW IN TABLE RR.
        INSERT INTO RR IN ENGINEERING_DATA_DATABASE
        SET RRREL = :RRREL,
            RRUSR = :RRUSR,
            RRTTL = :RRTTL,
            RRSEQ = :RRSEQ,
            RRPRI = :RRPRI
```

IBMRR

```
MODIFY AN EXISTING ROW IN TABLE RR.
        UDPATE RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RRREL AND
              RRUSR = :RRUSR AND
              RRTTL = :RRTTL
        SET RRREL = :RRREL,
            RRUSR = :RRUSR,
            RRTTL = :RRTTL,
            RRSEQ = :RRSEQ,
            RRPRI = :RRPRI
```

IBDRR

```
DELETE AN EXISTING ROW IN TABLE RR.
        DELETE FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RRREL AND
              RRUSR = :RRUSR AND
              RRTTL = :RRTTL
```

IBORR0

```
OBTAIN A ROW IN TABLE RR VIA ACCESS PATH RR0.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RRREL AND
              RRUSR = :RRUSR AND
              RRTTL = :RRTTL
        ORDER BY RRREL ASC, RRUSR ASC, RRTTL ASC
```

IBORR1

```
OBTAIN A ROW IN TABLE RR VIA ACCESS PATH RR1.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RRREL
        ORDER BY RRREL ASC, RRSEQ ASC
```

IBORR2

```
OBTAIN A ROW IN TABLE RR VIA ACCESS PATH RR2.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RRREL AND
              RRSEQ = :RRSEQ
        ORDER BY RRREL ASC, RRSEQ ASC, RRTTL ASC
```

IBORR3

```
OBTAIN A ROW IN TABLE RR VIA ACCESS PATH RR3.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRUSR = :RRUSR
        ORDER BY RRUSR ASC
```

IBARR0

```
OBTAIN A ROW IN TABLE RR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RR0.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE (RRREL > :RRREL)
          OR ((RRREL = :RRREL) AND (RRUSR > :RRUSR))
          OR ((RRREL = :RRREL AND RRUSR = :RRUSR) AND (RRTTL > :RRTTL))
          OR (RRREL = :RRREL AND RRUSR = :RRUSR AND RRTTL = :RRTTL)
        ORDER BY RRREL ASC, RRUSR ASC, RRTTL ASC
```

IBARR1

```
OBTAIN A ROW IN TABLE RR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RR1.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL >= :RRREL
        ORDER BY RRREL ASC, RRSEQ ASC
```

IBARR2

```
OBTAIN A ROW IN TABLE RR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RR2.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE (RRREL > :RRREL)
          OR ((RRREL = :RRREL) AND (RRSEQ > :RRSEQ))
          OR (RRREL = :RRREL AND RRSEQ = :RRSEQ)
        ORDER BY RRREL ASC, RRSEQ ASC, RRTTL ASC
```

IBARR3

```
OBTAIN A ROW IN TABLE RR USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RR3.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRUSR >= :RRUSR
        ORDER BY RRUSR ASC
```

IBERR1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RR VIA ACCESS PATH RR1.
SAVE THE CURRENT POSITION IN TABLE RR.
FETCH THE NEXT ROW FROM TABLE RR.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBERR2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RR VIA ACCESS PATH RR2.
SAVE THE CURRENT POSITION IN TABLE RR.
FETCH THE NEXT ROW FROM TABLE RR.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBERR3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RR VIA ACCESS PATH RR3.
SAVE THE CURRENT POSITION IN TABLE RR.
FETCH THE NEXT ROW FROM TABLE RR.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBFRR0

```
OBTAIN THE FIRST ROW OF TABLE RR, ORDERED BY ACCESS PATH RR0.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        ORDER BY RRREL ASC, RRUSR ASC, RRTTL ASC
```

IBFRR1

```
OBTAIN THE FIRST ROW OF TABLE RR, ORDERED BY ACCESS PATH RR1.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        ORDER BY RRREL ASC, RRSEQ ASC
```

IBFRR2

```
OBTAIN THE FIRST ROW OF TABLE RR, ORDERED BY ACCESS PATH RR2.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        ORDER BY RRREL ASC, RRSEQ ASC, RRTTL ASC
```

IBFRR3

```
OBTAIN THE FIRST ROW OF TABLE RR, ORDERED BY ACCESS PATH RR3.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        ORDER BY RRUSR ASC
```

IBNRR0

```
OBTAIN THE NEXT ROW OF TABLE RR, ORDERED BY ACCESS PATH RR0.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBNRR1

```
OBTAIN THE NEXT ROW OF TABLE RR, ORDERED BY ACCESS PATH RR1.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBNRR2

```
OBTAIN THE NEXT ROW OF TABLE RR, ORDERED BY ACCESS PATH RR2.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBNRR3

```
OBTAIN THE NEXT ROW OF TABLE RR, ORDERED BY ACCESS PATH RR3.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBOUIRR

```
USING COSET UIRR, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RR
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :RRUSR
```

IBORPRR

```
USING COSET RPRR, OBTAIN THE ROW FROM TABLE RP THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RR
        SELECT RPREL
        INTO :RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RRREL
```

IBFUIRR

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RR WITHIN COSET UIRR, USING
ACCESS PATH RR3.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRUSR = :UIUSR
        ORDER BY RRUSR ASC
```

IBFRPRR

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RR WITHIN COSET RPRR, USING
ACCESS PATH RR1.
        SELECT RRREL, RRUSR, RRTTL, RRSEQ, RRPRI
        FROM RR IN ENGINEERING_DATA_DATABASE
        WHERE RRREL = :RPREL
        ORDER BY RRREL ASC, RRSEQ ASC
```

IBNUIRR

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RR WITHIN COSET UIRR.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

IBNRPRR

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RR WITHIN COSET RPRR.
        SET :RRREL, :RRUSR, :RRTTL, :RRSEQ, :RRPRI
```

Release Signature (RS) Routines

The RS table contains the release signatures and stamps made by a reviewer of engineering data.

IBSRS

```
STORE A NEW ROW IN TABLE RS.
        INSERT INTO RS IN ENGINEERING_DATA_DATABASE
        SET RSEDN = :RSEDN,
            RSREL = :RSREL,
            RSUSR = :RSUSR,
            RSTTL = :RSTTL,
            RSDAT = :RSDAT,
            RSSTP = :RSSTP
```

IBMRS

```
MODIFY AN EXISTING ROW IN TABLE RS.
        UDPATE RS IN ENGINEERING_DATA_DATABASE
        WHERE RSEDN = :RSEDN AND
              RSREL = :RSREL AND
              RSUSR = :RSUSR AND
              RSTTL = :RSTTL
        SET RSEDN = :RSEDN,
            RSREL = :RSREL,
            RSUSR = :RSUSR,
            RSTTL = :RSTTL,
            RSDAT = :RSDAT,
            RSSTP = :RSSTP
```

IBDRS

```
DELETE AN EXISTING ROW IN TABLE RS.
        DELETE FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSEDN = :RSEDN AND
              RSREL = :RSREL AND
              RSUSR = :RSUSR AND
              RSTTL = :RSTTL
```

IBORS0

```
OBTAIN A ROW IN TABLE RS VIA ACCESS PATH RS0.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSEDN = :RSEDN AND
              RSREL = :RSREL AND
              RSUSR = :RSUSR AND
              RSTTL = :RSTTL
        ORDER BY RSEDN ASC, RSREL ASC, RSUSR ASC, RSTTL ASC
```

IBORS1

```
OBTAIN A ROW IN TABLE RS VIA ACCESS PATH RS1.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSREL = :RSREL AND
              RSEDN = :RSEDN
        ORDER BY RSREL ASC, RSEDN ASC, RSDAT ASC
```

IBORS2

```
OBTAIN A ROW IN TABLE RS VIA ACCESS PATH RS2.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSUSR = :RSUSR
        ORDER BY RSUSR ASC
```

IBARS0

```
OBTAIN A ROW IN TABLE RS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RS0.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE (RSEDN > :RSEDN)
          OR ((RSEDN = :RSEDN) AND (RSREL > :RSREL))
          OR ((RSEDN = :RSEDN AND RSREL = :RSREL) AND (RSUSR > :RSUSR))
          OR ((RSEDN = :RSEDN AND RSREL = :RSREL AND  RSUSR  =  :RSUSR) AND
              (RSTTL > :RSTTL))
          OR (RSEDN = :RSEDN AND RSREL = :RSREL AND RSUSR = :RSUSR AND
              RSTTL = :RSTTL)
        ORDER BY RSEDN ASC, RSREL ASC, RSUSR ASC, RSTTL ASC
```

IBARS1

```
OBTAIN A ROW IN TABLE RS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RS1.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE (RSREL > :RSREL)
          OR ((RSREL = :RSREL) AND (RSEDN > :RSEDN))
          OR (RSREL = :RSREL AND RSEDN = :RSEDN)
        ORDER BY RSREL ASC, RSEDN ASC, RSDAT ASC
```

IBARS2

```
OBTAIN A ROW IN TABLE RS USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RS2.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSUSR >= :RSUSR
        ORDER BY RSUSR ASC
```

IBERS1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RS VIA ACCESS PATH RS1.
SAVE THE CURRENT POSITION IN TABLE RS.
FETCH THE NEXT ROW FROM TABLE RS.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBERS2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RS VIA ACCESS PATH RS2.
SAVE THE CURRENT POSITION IN TABLE RS.
FETCH THE NEXT ROW FROM TABLE RS.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBFRS0

```
OBTAIN THE FIRST ROW OF TABLE RS, ORDERED BY ACCESS PATH RS0.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        ORDER BY RSEDN ASC, RSREL ASC, RSUSR ASC, RSTTL ASC
```

IBFRS1

```
OBTAIN THE FIRST ROW OF TABLE RS, ORDERED BY ACCESS PATH RS1.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        ORDER BY RSREL ASC, RSEDN ASC, RSDAT ASC
```

IBFRS2

```
OBTAIN THE FIRST ROW OF TABLE RS, ORDERED BY ACCESS PATH RS2.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        ORDER BY RSUSR ASC
```

IBNRS0

```
OBTAIN THE NEXT ROW OF TABLE RS, ORDERED BY ACCESS PATH RS0.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBNRS1

```
OBTAIN THE NEXT ROW OF TABLE RS, ORDERED BY ACCESS PATH RS1.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBNRS2

```
OBTAIN THE NEXT ROW OF TABLE RS, ORDERED BY ACCESS PATH RS2.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBOUIRS

```
USING COSET UIRS, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RS
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :RSUSR
```

IBORARS

```
USING COSET RARS, OBTAIN THE ROW FROM TABLE RA THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RS
        SELECT RAREL, RAEDN, RAEDNC, RAUSR, RASTA, RADAT
        INTO :RAREL, :RAEDN, :RAEDNC, :RAUSR, :RASTA, :RADAT
        FROM RA IN ENGINEERING_DATA_DATABASE
        WHERE RAREL = :RSREL AND
              RAEDN = :RSEDN
```

IBFUIRS

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RS WITHIN COSET UIRS, USING
ACCESS PATH RS2.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSUSR = :UIUSR
        ORDER BY RSUSR ASC
```

IBFRARS

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RS WITHIN COSET RARS, USING
ACCESS PATH RS1.
        SELECT RSEDN, RSREL, RSUSR, RSTTL, RSDAT, RSSTP
        FROM RS IN ENGINEERING_DATA_DATABASE
        WHERE RSREL = :RAREL AND
              RSEDN = :RAEDN
        ORDER BY RSREL ASC, RSEDN ASC, RSDAT ASC
```

IBNUIRS

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RS WITHIN COSET UIRS.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

IBNRARS

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RS WITHIN COSET RARS.
        SET :RSEDN, :RSREL, :RSUSR, :RSTTL, :RSDAT, :RSSTP
```

Release Transfers (RT) Routines

The RT table defines which data transfers are usable for releasing engineering data.

IBSRT

```
STORE A NEW ROW IN TABLE RT.
        INSERT INTO RT IN ENGINEERING_DATA_DATABASE
        SET RTADT1 = :RTADT1,
            RTADT2 = :RTADT2
```

IBMRT

```
MODIFY AN EXISTING ROW IN TABLE RT.
        UDPATE RT IN ENGINEERING_DATA_DATABASE
        WHERE RTADT1 = :RTADT1 AND
              RTADT2 = :RTADT2
        SET RTADT1 = :RTADT1,
            RTADT2 = :RTADT2
```

IBDRT

```
DELETE AN EXISTING ROW IN TABLE RT.
        DELETE FROM RT IN ENGINEERING_DATA_DATABASE
        WHERE RTADT1 = :RTADT1 AND
              RTADT2 = :RTADT2
```

IBORT0

```
OBTAIN A ROW IN TABLE RT VIA ACCESS PATH RT0.
        SELECT RTADT1, RTADT2
        FROM RT IN ENGINEERING_DATA_DATABASE
        WHERE RTADT1 = :RTADT1 AND
              RTADT2 = :RTADT2
        ORDER BY RTADT1 ASC, RTADT2 ASC
```

IBART0

```
OBTAIN A ROW IN TABLE RT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RT0.
        SELECT RTADT1, RTADT2
        FROM RT IN ENGINEERING_DATA_DATABASE
        WHERE (RTADT1 > :RTADT1)
          OR ((RTADT1 = :RTADT1) AND (RTADT2 > :RTADT2))
          OR (RTADT1 = :RTADT1 AND RTADT2 = :RTADT2)
        ORDER BY RTADT1 ASC, RTADT2 ASC
```

IBFRT0

```
OBTAIN THE FIRST ROW OF TABLE RT, ORDERED BY ACCESS PATH RT0.
        SELECT RTADT1, RTADT2
        FROM RT IN ENGINEERING_DATA_DATABASE
        ORDER BY RTADT1 ASC, RTADT2 ASC
```

IBNRT0

```
OBTAIN THE NEXT ROW OF TABLE RT, ORDERED BY ACCESS PATH RT0.
        SET :RTADT1, :RTADT2
```

Releasers (RU) Routines

The RU table defines the releasers involved in a release procedure.

IBSRU

```
STORE A NEW ROW IN TABLE RU.
        INSERT INTO RU IN ENGINEERING_DATA_DATABASE
        SET RUREL = :RUREL,
            RUUSR = :RUUSR
```

IBMRU

```
MODIFY AN EXISTING ROW IN TABLE RU.
        UDPATE RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL = :RUREL AND
              RUUSR = :RUUSR
        SET RUREL = :RUREL,
            RUUSR = :RUUSR
```

IBDRU

```
DELETE AN EXISTING ROW IN TABLE RU.
        DELETE FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL = :RUREL AND
              RUUSR = :RUUSR
```

IBORU0

```
OBTAIN A ROW IN TABLE RU VIA ACCESS PATH RU0.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL = :RUREL AND
              RUUSR = :RUUSR
        ORDER BY RUREL ASC, RUUSR ASC
```

IBORU1

```
OBTAIN A ROW IN TABLE RU VIA ACCESS PATH RU1.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL = :RUREL
        ORDER BY RUREL ASC
```

IBORU2

```
OBTAIN A ROW IN TABLE RU VIA ACCESS PATH RU2.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUUSR = :RUUSR
        ORDER BY RUUSR ASC
```

IBARU0

```
OBTAIN A ROW IN TABLE RU USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RU0.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE (RUREL > :RUREL)
          OR ((RUREL = :RUREL) AND (RUUSR > :RUUSR))
          OR (RUREL = :RUREL AND RUUSR = :RUUSR)
        ORDER BY RUREL ASC, RUUSR ASC
```

IBARU1

```
OBTAIN A ROW IN TABLE RU USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RU1.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL >= :RUREL
        ORDER BY RUREL ASC
```

IBARU2

```
OBTAIN A ROW IN TABLE RU USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH RU2.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUUSR >= :RUUSR
        ORDER BY RUUSR ASC
```

IBERU1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RU VIA ACCESS PATH RU1.
SAVE THE CURRENT POSITION IN TABLE RU.
FETCH THE NEXT ROW FROM TABLE RU.
        SET :RUREL, :RUUSR
```

IBERU2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE RU VIA ACCESS PATH RU2.
SAVE THE CURRENT POSITION IN TABLE RU.
FETCH THE NEXT ROW FROM TABLE RU.
        SET :RUREL, :RUUSR
```

IBFRU0

```
OBTAIN THE FIRST ROW OF TABLE RU, ORDERED BY ACCESS PATH RU0.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        ORDER BY RUREL ASC, RUUSR ASC
```

IBFRU1

```
OBTAIN THE FIRST ROW OF TABLE RU, ORDERED BY ACCESS PATH RU1.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        ORDER BY RUREL ASC
```

IBFRU2

```
OBTAIN THE FIRST ROW OF TABLE RU, ORDERED BY ACCESS PATH RU2.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        ORDER BY RUUSR ASC
```

IBNRU0

```
OBTAIN THE NEXT ROW OF TABLE RU, ORDERED BY ACCESS PATH RU0.
        SET :RUREL, :RUUSR
```

IBNRU1

```
OBTAIN THE NEXT ROW OF TABLE RU, ORDERED BY ACCESS PATH RU1.
        SET :RUREL, :RUUSR
```

IBNRU2

```
OBTAIN THE NEXT ROW OF TABLE RU, ORDERED BY ACCESS PATH RU2.
        SET :RUREL, :RUUSR
```

IBOUIRU

```
USING COSET UIRU, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RU
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :RUUSR
```

IBORPRU

```
USING COSET RPRU, OBTAIN THE ROW FROM TABLE RP THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE RU
        SELECT RPREL
        INTO :RPREL
        FROM RP IN ENGINEERING_DATA_DATABASE
        WHERE RPREL = :RUREL
```

IBFUIRU

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RU WITHIN COSET UIRU, USING
ACCESS PATH RU2.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUUSR = :UIUSR
        ORDER BY RUUSR ASC
```

IBFRPRU

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE RU WITHIN COSET RPRU, USING
ACCESS PATH RU1.
        SELECT RUREL, RUUSR
        FROM RU IN ENGINEERING_DATA_DATABASE
        WHERE RUREL = :RPREL
        ORDER BY RUREL ASC
```

IBNUIRU

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RU WITHIN COSET UIRU.
        SET :RUREL, :RUUSR
```

IBNRPRU

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE RU WITHIN COSET RPRU.
        SET :RUREL, :RUUSR
```

Task Command (TC) Routines

The TC table contains the commands that start the tasks defined in TI.

IBSTC

```
STORE A NEW ROW IN TABLE TC.
        INSERT INTO TC IN MENU_DATABASE
        SET TCCMD = :TCCMD,
            TCTNA = :TCTNA
```

IBMTC

```
MODIFY AN EXISTING ROW IN TABLE TC.
        UDPATE TC IN MENU_DATABASE
        WHERE TCCMD = :TCCMD
        SET TCCMD = :TCCMD,
            TCTNA = :TCTNA
```

IBDTC

```
DELETE AN EXISTING ROW IN TABLE TC.
        DELETE FROM TC IN MENU_DATABASE
        WHERE TCCMD = :TCCMD
```

IBOTC0

```
OBTAIN A ROW IN TABLE TC VIA ACCESS PATH TC0.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        WHERE TCCMD = :TCCMD
        ORDER BY TCCMD ASC
```

IBOTC1

```
OBTAIN A ROW IN TABLE TC VIA ACCESS PATH TC1.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        WHERE TCTNA = :TCTNA
        ORDER BY TCTNA ASC
```

IBATC0

```
OBTAIN A ROW IN TABLE TC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TC0.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        WHERE TCCMD >= :TCCMD
        ORDER BY TCCMD ASC
```

IBATC1

```
OBTAIN A ROW IN TABLE TC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TC1.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        WHERE TCTNA >= :TCTNA
        ORDER BY TCTNA ASC
```

IBETC1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TC VIA ACCESS PATH TC1.
SAVE THE CURRENT POSITION IN TABLE TC.
FETCH THE NEXT ROW FROM TABLE TC.
        SET :TCCMD, :TCTNA
```

IBFTC0

```
OBTAIN THE FIRST ROW OF TABLE TC, ORDERED BY ACCESS PATH TC0.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        ORDER BY TCCMD ASC
```

IBFTC1

```
OBTAIN THE FIRST ROW OF TABLE TC, ORDERED BY ACCESS PATH TC1.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        ORDER BY TCTNA ASC
```

IBNTC0

```
OBTAIN THE NEXT ROW OF TABLE TC, ORDERED BY ACCESS PATH TC0.
        SET :TCCMD, :TCTNA
```

IBNTC1

```
OBTAIN THE NEXT ROW OF TABLE TC, ORDERED BY ACCESS PATH TC1.
        SET :TCCMD, :TCTNA
```

IBOTITC

```
USING COSET TITC, OBTAIN THE ROW FROM TABLE TI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TC
        SELECT TITNA, TISEC, TITYP, TIDSC
        INTO :TITNA, :TISEC, :TITYP, :TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TITNA = :TCTNA
```

IBFTITC

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TC WITHIN COSET TITC, USING
ACCESS PATH TC1.
        SELECT TCCMD, TCTNA
        FROM TC IN MENU_DATABASE
        WHERE TCTNA = :TITNA
        ORDER BY TCTNA ASC
```

IBNTITC

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TC WITHIN COSET TITC.
        SET :TCCMD, :TCTNA
```

Task Information (TI) Routines

The TI record contains the headers for EDL tasks.

IBSTI

```
STORE A NEW ROW IN TABLE TI.
        INSERT INTO TI IN MENU_DATABASE
        SET TITNA = :TITNA,
            TISEC = :TISEC,
            TITYP = :TITYP,
            TIDSC = :TIDSC
```

IBMTI

```
MODIFY AN EXISTING ROW IN TABLE TI.
        UDPATE TI IN MENU_DATABASE
        WHERE TITNA = :TITNA
        SET TITNA = :TITNA,
            TISEC = :TISEC,
            TITYP = :TITYP,
            TIDSC = :TIDSC
```

IBDTI

```
DELETE AN EXISTING ROW IN TABLE TI.
        DELETE FROM TI IN MENU_DATABASE
        WHERE TITNA = :TITNA
```

IBOTI0

```
OBTAIN A ROW IN TABLE TI VIA ACCESS PATH TI0.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TITNA = :TITNA
        ORDER BY TITNA ASC
```

IBOTI1

```
OBTAIN A ROW IN TABLE TI VIA ACCESS PATH TI1.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TISEC = :TISEC
        ORDER BY TISEC ASC
```

IBATI0

```
OBTAIN A ROW IN TABLE TI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TI0.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TITNA >= :TITNA
        ORDER BY TITNA ASC
```

IBATI1

```
OBTAIN A ROW IN TABLE TI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TI1.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TISEC >= :TISEC
        ORDER BY TISEC ASC
```

IBETI1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TI VIA ACCESS PATH TI1.
SAVE THE CURRENT POSITION IN TABLE TI.
FETCH THE NEXT ROW FROM TABLE TI.
        SET :TITNA, :TISEC, :TITYP, :TIDSC
```

IBFTI0

```
OBTAIN THE FIRST ROW OF TABLE TI, ORDERED BY ACCESS PATH TI0.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        ORDER BY TITNA ASC
```

IBFTI1

```
OBTAIN THE FIRST ROW OF TABLE TI, ORDERED BY ACCESS PATH TI1.
        SELECT TITNA, TISEC, TITYP, TIDSC
        FROM TI IN MENU_DATABASE
        ORDER BY TISEC ASC
```

IBNTI0

```
OBTAIN THE NEXT ROW OF TABLE TI, ORDERED BY ACCESS PATH TI0.
        SET :TITNA, :TISEC, :TITYP, :TIDSC
```

IBNTI1

```
OBTAIN THE NEXT ROW OF TABLE TI, ORDERED BY ACCESS PATH TI1.
        SET :TITNA, :TISEC, :TITYP, :TIDSC
```

Task Menu (TM) Routines

The TM table contains the task menu lines

IBSTM

```
STORE A NEW ROW IN TABLE TM.
        INSERT INTO TM IN MENU_DATABASE
        SET TMMNA = :TMMNA,
            TMMLN = :TMMLN,
            TMTXT = :TMTXT,
            TMTNA = :TMTNA
```

IBMTM

```
MODIFY AN EXISTING ROW IN TABLE TM.
        UDPATE TM IN MENU_DATABASE
        WHERE TMMNA = :TMMNA AND
              TMMLN = :TMMLN
        SET TMMNA = :TMMNA,
            TMMLN = :TMMLN,
            TMTXT = :TMTXT,
            TMTNA = :TMTNA
```

IBDTM

```
DELETE AN EXISTING ROW IN TABLE TM.
        DELETE FROM TM IN MENU_DATABASE
        WHERE TMMNA = :TMMNA AND
              TMMLN = :TMMLN
```

IBOTM0

```
OBTAIN A ROW IN TABLE TM VIA ACCESS PATH TM0.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMMNA = :TMMNA AND
              TMMLN = :TMMLN
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBOTM1

```
OBTAIN A ROW IN TABLE TM VIA ACCESS PATH TM1.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMMNA = :TMMNA AND
              TMTNA = :TMTNA
        ORDER BY TMMNA ASC, TMTNA ASC
```

IBOTM2

```
OBTAIN A ROW IN TABLE TM VIA ACCESS PATH TM2.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMTNA = :TMTNA
        ORDER BY TMTNA ASC
```

IBOTM3

```
OBTAIN A ROW IN TABLE TM VIA ACCESS PATH TM3.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMMNA = :TMMNA
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBATM0

```
OBTAIN A ROW IN TABLE TM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH TM0.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE (TMMNA > :TMMNA)
          OR ((TMMNA = :TMMNA) AND (TMMLN > :TMMLN))
          OR (TMMNA = :TMMNA AND TMMLN = :TMMLN)
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBATM1

```
OBTAIN A ROW IN TABLE TM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH TM1.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE (TMMNA > :TMMNA)
          OR ((TMMNA = :TMMNA) AND (TMTNA > :TMTNA))
          OR (TMMNA = :TMMNA AND TMTNA = :TMTNA)
        ORDER BY TMMNA ASC, TMTNA ASC
```

IBATM2

```
OBTAIN A ROW IN TABLE TM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH TM2.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMTNA >= :TMTNA
        ORDER BY TMTNA ASC
```

IBATM3

```
OBTAIN A ROW IN TABLE TM USING AN APPROXIMATE KEY VALUE AND ACCESS PATH TM3.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMMNA >= :TMMNA
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBETM2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TM VIA ACCESS PATH TM2.
SAVE THE CURRENT POSITION IN TABLE TM.
FETCH THE NEXT ROW FROM TABLE TM.
       SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBETM3

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TM VIA ACCESS PATH TM3.
SAVE THE CURRENT POSITION IN TABLE TM.
FETCH THE NEXT ROW FROM TABLE TM.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBFTM0

```
OBTAIN THE FIRST ROW OF TABLE TM, ORDERED BY ACCESS PATH TM0.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBFTM1

```
OBTAIN THE FIRST ROW OF TABLE TM, ORDERED BY ACCESS PATH TM1.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        ORDER BY TMMNA ASC, TMTNA ASC
```

IBFTM2

```
OBTAIN THE FIRST ROW OF TABLE TM, ORDERED BY ACCESS PATH TM2.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        ORDER BY TMTNA ASC
```

IBFTM3

```
OBTAIN THE FIRST ROW OF TABLE TM, ORDERED BY ACCESS PATH TM3.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBNTM0

```
OBTAIN THE NEXT ROW OF TABLE TM, ORDERED BY ACCESS PATH TM0.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBNTM1

```
OBTAIN THE NEXT ROW OF TABLE TM, ORDERED BY ACCESS PATH TM1.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBNTM2

```
OBTAIN THE NEXT ROW OF TABLE TM, ORDERED BY ACCESS PATH TM2.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBNTM3

```
OBTAIN THE NEXT ROW OF TABLE TM, ORDERED BY ACCESS PATH TM3.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBOTITM

```
USING COSET TITM, OBTAIN THE ROW FROM TABLE TI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TM
        SELECT TITNA, TISEC, TITYP, TIDSC
        INTO :TITNA, :TISEC, :TITYP, :TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TITNA = :TMTNA
```

IBOMITM

```
USING COSET MITM, OBTAIN THE ROW FROM TABLE MI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TM
        SELECT MIMNA, MITYP, MISTA, MITTL
        INTO :MIMNA, :MITYP, :MISTA, :MITTL
        FROM MI IN MENU_DATABASE
        WHERE MIMNA = :TMMNA
```

IBFTITM

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TM WITHIN COSET TITM, USING
ACCESS PATH TM2.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMTNA = :TITNA
        ORDER BY TMTNA ASC
```

IBFMITM

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TM WITHIN COSET MITM, USING
ACCESS PATH TM3.
        SELECT TMMNA, TMMLN, TMTXT, TMTNA
        FROM TM IN MENU_DATABASE
        WHERE TMMNA = :MIMNA
        ORDER BY TMMNA ASC, TMMLN ASC
```

IBNTITM

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TM WITHIN COSET TITM.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

IBNMITM

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TM WITHIN COSET MITM.
        SET :TMMNA, :TMMLN, :TMTXT, :TMTNA
```

Task Process (TP) Routines

The TP table defines the individual steps (task processes) that compose a task.

IBSTP

```
STORE A NEW ROW IN TABLE TP.
        INSERT INTO TP IN MENU_DATABASE
        SET TPTNA = :TPTNA,
            TPSEQ = :TPSEQ,
            TPTYP = :TPTYP,
            TPNAM = :TPNAM,
            TPFNA = :TPFNA
```

IBMTP

```
MODIFY AN EXISTING ROW IN TABLE TP.
        UDPATE TP IN MENU_DATABASE
        WHERE TPTNA = :TPTNA AND
              TPSEQ = :TPSEQ
        SET TPTNA = :TPTNA,
            TPSEQ = :TPSEQ,
            TPTYP = :TPTYP,
            TPNAM = :TPNAM,
            TPFNA = :TPFNA
```

IBDTP

```
DELETE AN EXISTING ROW IN TABLE TP.
        DELETE FROM TP IN MENU_DATABASE
        WHERE TPTNA = :TPTNA AND
              TPSEQ = :TPSEQ
```

IBOTP0

```
OBTAIN A ROW IN TABLE TP VIA ACCESS PATH TP0.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        WHERE TPTNA = :TPTNA AND
              TPSEQ = :TPSEQ
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBOTP1

```
OBTAIN A ROW IN TABLE TP VIA ACCESS PATH TP1.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        WHERE TPTNA = :TPTNA
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBATP0

```
OBTAIN A ROW IN TABLE TP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TP0.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        WHERE (TPTNA > :TPTNA)
          OR ((TPTNA = :TPTNA) AND (TPSEQ > :TPSEQ))
          OR (TPTNA = :TPTNA AND TPSEQ = :TPSEQ)
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBATP1

```
OBTAIN A ROW IN TABLE TP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TP1.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        WHERE TPTNA >= :TPTNA
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBETP1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TP VIA ACCESS PATH TP1.
SAVE THE CURRENT POSITION IN TABLE TP.
FETCH THE NEXT ROW FROM TABLE TP.
        SET :TPTNA, :TPSEQ, :TPTYP, :TPNAM, :TPFNA
```

IBFTP0

```
OBTAIN THE FIRST ROW OF TABLE TP, ORDERED BY ACCESS PATH TP0.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBFTP1

```
OBTAIN THE FIRST ROW OF TABLE TP, ORDERED BY ACCESS PATH TP1.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBNTP0

```
OBTAIN THE NEXT ROW OF TABLE TP, ORDERED BY ACCESS PATH TP0.
        SET :TPTNA, :TPSEQ, :TPTYP, :TPNAM, :TPFNA
```

IBNTP1

```
OBTAIN THE NEXT ROW OF TABLE TP, ORDERED BY ACCESS PATH TP1.
        SET :TPTNA, :TPSEQ, :TPTYP, :TPNAM, :TPFNA
```

IBOTITP

```
USING COSET TITP, OBTAIN THE ROW FROM TABLE TI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TP
        SELECT TITNA, TISEC, TITYP, TIDSC
        INTO :TITNA, :TISEC, :TITYP, :TIDSC
        FROM TI IN MENU_DATABASE
        WHERE TITNA = :TPTNA
```

IBFTITP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TP WITHIN COSET TITP, USING
ACCESS PATH TP1.
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        FROM TP IN MENU_DATABASE
        WHERE TPTNA = :TITNA
        ORDER BY TPTNA ASC, TPSEQ ASC
```

IBNTITP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TP WITHIN COSET TITP.
        SET :TPTNA, :TPSEQ, :TPTYP, :TPNAM, :TPFNA
```

Transfer and Translation Tasks (TT) Routines

The TT table defines how data is transferred from one application to another.

IBSTT

```
STORE A NEW ROW IN TABLE TT.
        INSERT INTO TT IN ENGINEERING_DATA_DATABASE
        SET TTADT1 = :TTADT1,
            TTADT2 = :TTADT2,
            TTTNA = :TTTNA
```

IBMTT

```
MODIFY AN EXISTING ROW IN TABLE TT.
        UDPATE TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 = :TTADT1 AND
              TTADT2 = :TTADT2
        SET TTADT1 = :TTADT1,
            TTADT2 = :TTADT2,
            TTTNA = :TTTNA
```

IBDTT

```
DELETE AN EXISTING ROW IN TABLE TT.
        DELETE FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 = :TTADT1 AND
              TTADT2 = :TTADT2
```

IBOTT0

```
OBTAIN A ROW IN TABLE TT VIA ACCESS PATH TT0.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 = :TTADT1 AND
              TTADT2 = :TTADT2
        ORDER BY TTADT1 ASC, TTADT2 ASC
```

IBOTT1

```
OBTAIN A ROW IN TABLE TT VIA ACCESS PATH TT1.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 = :TTADT1
        ORDER BY TTADT1 ASC
```

IBOTT2

```
OBTAIN A ROW IN TABLE TT VIA ACCESS PATH TT2.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT2 = :TTADT2
        ORDER BY TTADT2 ASC
```

IBATT0

```
OBTAIN A ROW IN TABLE TT USING AN APPROXIMATE KEY VALUE AND ACCESS PATH TT0.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE (TTADT1 > :TTADT1)
          OR ((TTADT1 = :TTADT1) AND (TTADT2 > :TTADT2))
          OR (TTADT1 = :TTADT1 AND TTADT2 = :TTADT2)
        ORDER BY TTADT1 ASC, TTADT2 ASC
```

IBATT1

```
OBTAIN A ROW IN TABLE TT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TT1.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 >= :TTADT1
        ORDER BY TTADT1 ASC
```

IBATT2

```
OBTAIN A ROW IN TABLE TT USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TT2.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT2 >= :TTADT2
        ORDER BY TTADT2 ASC
```

IBETT1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TT VIA ACCESS PATH TT1.
SAVE THE CURRENT POSITION IN TABLE TT.
FETCH THE NEXT ROW FROM TABLE TT.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBETT2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TT VIA ACCESS PATH TT2.
SAVE THE CURRENT POSITION IN TABLE TT.
FETCH THE NEXT ROW FROM TABLE TT.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBFTT0

```
OBTAIN THE FIRST ROW OF TABLE TT, ORDERED BY ACCESS PATH TT0.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        ORDER BY TTADT1 ASC, TTADT2 ASC
```

IBFTT1

```
OBTAIN THE FIRST ROW OF TABLE TT, ORDERED BY ACCESS PATH TT1.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        ORDER BY TTADT1 ASC
```

IBFTT2

```
OBTAIN THE FIRST ROW OF TABLE TT, ORDERED BY ACCESS PATH TT2.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        ORDER BY TTADT2 ASC
```

IBNTT0

```
OBTAIN THE NEXT ROW OF TABLE TT, ORDERED BY ACCESS PATH TT0.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBNTT1

```
OBTAIN THE NEXT ROW OF TABLE TT, ORDERED BY ACCESS PATH TT1.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBNTT2

```
OBTAIN THE NEXT ROW OF TABLE TT, ORDERED BY ACCESS PATH TT2.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBOATT2

```
USING COSET ATT2, OBTAIN THE ROW FROM TABLE AT THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TT
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        INTO :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :TTADT2
```

IBOATT1

```
USING COSET ATT1, OBTAIN THE ROW FROM TABLE AT THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TT
        SELECT ATADT, ATNAM, ATFTC, ATTNA, ATSIDR
        INTO :ATADT, :ATNAM, :ATFTC, :ATTNA, :ATSIDR
        FROM AT IN ENGINEERING_DATA_DATABASE
        WHERE ATADT = :TTADT1
```

IBFATT2

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TT WITHIN COSET ATT2, USING
ACCESS PATH TT2.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT2 = :ATADT
        ORDER BY TTADT2 ASC
```

IBFATT1

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TT WITHIN COSET ATT1, USING
ACCESS PATH TT1.
        SELECT TTADT1, TTADT2, TTTNA
        FROM TT IN ENGINEERING_DATA_DATABASE
        WHERE TTADT1 = :ATADT
        ORDER BY TTADT1 ASC
```

IBNATT2

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TT WITHIN COSET ATT2.
        SET :TTADT1, :TTADT2, :TTTNA
```

IBNATT1

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TT WITHIN COSET ATT1.
        SET :TTADT1, :TTADT2, :TTTNA
```

Task Parameter Value (TV) Routines

The TV table defines the parameters that are passed to CLI macros and EDL subprograms when they are executed as task processes. Task parameters can also be used to answer prompts issued by EDL subprograms.

IBSTV

```
STORE A NEW ROW IN TABLE TV.
        INSERT INTO TV IN MENU_DATABASE
        SET TVTNA = :TVTNA,
            TVSEQ = :TVSEQ,
            TVPOS = :TVPOS,
            TVPRM = :TVPRM,
            TVTYP = :TVTYP,
            TVVAL = :TVVAL
```

IBMTV

```
MODIFY AN EXISTING ROW IN TABLE TV.
        UDPATE TV IN MENU_DATABASE
        WHERE TVPOS = :TVPOS AND
              TVSEQ = :TVSEQ AND
              TVTNA = :TVTNA
        SET TVTNA = :TVTNA,
            TVSEQ = :TVSEQ,
            TVPOS = :TVPOS,
            TVPRM = :TVPRM,
            TVTYP = :TVTYP,
            TVVAL = :TVVAL
```

IBDTV

```
DELETE AN EXISTING ROW IN TABLE TV.
        DELETE FROM TV IN MENU_DATABASE
        WHERE TVPOS = :TVPOS AND
              TVSEQ = :TVSEQ AND
              TVTNA = :TVTNA
```

IBOTV0

```
OBTAIN A ROW IN TABLE TV VIA ACCESS PATH TV0.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE TVPOS = :TVPOS AND
              TVSEQ = :TVSEQ AND
              TVTNA = :TVTNA
        ORDER BY TVPOS ASC, TVSEQ ASC, TVTNA ASC
```

IBOTV1

```
OBTAIN A ROW IN TABLE TV VIA ACCESS PATH TV1.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE TVTNA = :TVTNA AND
              TVSEQ = :TVSEQ
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPOS ASC
```

IBOTV2

```
OBTAIN A ROW IN TABLE TV VIA ACCESS PATH TV2.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE TVTNA = :TVTNA AND
              TVSEQ = :TVSEQ AND
              TVPRM = :TVPRM
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPRM ASC, TVPOS ASC
```

IBATV0

```
OBTAIN A ROW IN TABLE TV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TV0.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE (TVPOS > :TVPOS)
          OR ((TVPOS = :TVPOS) AND (TVSEQ > :TVSEQ))
          OR ((TVPOS = :TVPOS AND TVSEQ = :TVSEQ) AND (TVTNA > :TVTNA))
          OR (TVPOS = :TVPOS AND TVSEQ = :TVSEQ AND TVTNA = :TVTNA)
        ORDER BY TVPOS ASC, TVSEQ ASC, TVTNA ASC
```

IBATV1

```
OBTAIN A ROW IN TABLE TV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TV1.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE (TVTNA > :TVTNA)
          OR ((TVTNA = :TVTNA) AND (TVSEQ > :TVSEQ))
          OR (TVTNA = :TVTNA AND TVSEQ = :TVSEQ)
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPOS ASC
```

IBATV2

```
OBTAIN A ROW IN TABLE TV USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH TV2.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE (TVTNA > :TVTNA)
          OR ((TVTNA = :TVTNA) AND (TVSEQ > :TVSEQ))
          OR ((TVTNA = :TVTNA AND TVSEQ = :TVSEQ) AND (TVPRM > :TVPRM))
          OR (TVTNA = :TVTNA AND TVSEQ = :TVSEQ AND TVPRM = :TVPRM)
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPRM ASC, TVPOS ASC
```

IBETV1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TV VIA ACCESS PATH TV1.
SAVE THE CURRENT POSITION IN TABLE TV.
FETCH THE NEXT ROW FROM TABLE TV.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

IBETV2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE TV VIA ACCESS PATH TV2.
SAVE THE CURRENT POSITION IN TABLE TV.
FETCH THE NEXT ROW FROM TABLE TV.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

IBFTV0

```
OBTAIN THE FIRST ROW OF TABLE TV, ORDERED BY ACCESS PATH TV0.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        ORDER BY TVPOS ASC, TVSEQ ASC, TVTNA ASC
```

IBFTV1

```
OBTAIN THE FIRST ROW OF TABLE TV, ORDERED BY ACCESS PATH TV1.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPOS ASC
```

IBFTV2

```
OBTAIN THE FIRST ROW OF TABLE TV, ORDERED BY ACCESS PATH TV2.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPRM ASC, TVPOS ASC
```

IBNTV0

```
OBTAIN THE NEXT ROW OF TABLE TV, ORDERED BY ACCESS PATH TV0.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

IBNTV1

```
OBTAIN THE NEXT ROW OF TABLE TV, ORDERED BY ACCESS PATH TV1.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

IBNTV2

```
OBTAIN THE NEXT ROW OF TABLE TV, ORDERED BY ACCESS PATH TV2.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

IBOTPTV

```
USING COSET TPTV, OBTAIN THE ROW FROM TABLE TP THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE TV
        SELECT TPTNA, TPSEQ, TPTYP, TPNAM, TPFNA
        INTO :TPTNA, :TPSEQ, :TPTYP, :TPNAM, :TPFNA
        FROM TP IN MENU_DATABASE
        WHERE TPTNA = :TVTNA AND
              TPSEQ = :TVSEQ
```

IBFTPTV

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE TV WITHIN COSET TPTV, USING
ACCESS PATH TV1.
        SELECT TVTNA, TVSEQ, TVPOS, TVPRM, TVTYP, TVVAL
        FROM TV IN MENU_DATABASE
        WHERE TVTNA = :TPTNA AND
              TVSEQ = :TPSEQ
        ORDER BY TVTNA ASC, TVSEQ ASC, TVPOS ASC
```

IBNTPTV

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE TV WITHIN COSET TPTV.
        SET :TVTNA, :TVSEQ, :TVPOS, :TVPRM, :TVTYP, :TVVAL
```

User Configuration (UC) Routines

The UC table contains the terminal configuration attributes of a given EDL user.

IBSUC

```
STORE A NEW ROW IN TABLE UC.
        INSERT INTO UC IN ENGINEERING_DATA_DATABASE
        SET UCUSR = :UCUSR,
            UCATR = :UCATR,
            UCSTA = :UCSTA
```

IBMUC

```
MODIFY AN EXISTING ROW IN TABLE UC.
        UDPATE UC IN ENGINEERING_DATA_DATABASE
        WHERE UCATR = :UCATR AND
              UCUSR = :UCUSR
        SET UCUSR = :UCUSR,
            UCATR = :UCATR,
            UCSTA = :UCSTA
```

IBDUC

```
DELETE AN EXISTING ROW IN TABLE UC.
        DELETE FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE UCATR = :UCATR AND
              UCUSR = :UCUSR
```

IBOUC0

```
OBTAIN A ROW IN TABLE UC VIA ACCESS PATH UC0.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE UCATR = :UCATR AND
              UCUSR = :UCUSR
        ORDER BY UCATR ASC, UCUSR ASC
```

IBOUC1

```
OBTAIN A ROW IN TABLE UC VIA ACCESS PATH UC1.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE UCUSR = :UCUSR
        ORDER BY UCUSR ASC
```

IBAUC0

```
OBTAIN A ROW IN TABLE UC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UC0.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE (UCATR > :UCATR)
          OR ((UCATR = :UCATR) AND (UCUSR > :UCUSR))
          OR (UCATR = :UCATR AND UCUSR = :UCUSR)
        ORDER BY UCATR ASC, UCUSR ASC
```

IBAUC1

```
OBTAIN A ROW IN TABLE UC USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UC1.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE UCUSR >= :UCUSR
        ORDER BY UCUSR ASC
```

IBEUC1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE UC VIA ACCESS PATH UC1.
SAVE THE CURRENT POSITION IN TABLE UC.
FETCH THE NEXT ROW FROM TABLE UC.
        SET :UCUSR, :UCATR, :UCSTA
```

IBFUC0

```
OBTAIN THE FIRST ROW OF TABLE UC, ORDERED BY ACCESS PATH UC0.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        ORDER BY UCATR ASC, UCUSR ASC
```

IBFUC1

```
OBTAIN THE FIRST ROW OF TABLE UC, ORDERED BY ACCESS PATH UC1.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        ORDER BY UCUSR ASC
```

IBNUC0

```
OBTAIN THE NEXT ROW OF TABLE UC, ORDERED BY ACCESS PATH UC0.
        SET :UCUSR, :UCATR, :UCSTA
```

IBNUC1

```
OBTAIN THE NEXT ROW OF TABLE UC, ORDERED BY ACCESS PATH UC1.
        SET :UCUSR, :UCATR, :UCSTA
```

IBOUIUC

```
USING COSET UIUC, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE UC
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :UCUSR
```

IBFUIUC

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE UC WITHIN COSET UIUC, USING
ACCESS PATH UC1.
        SELECT UCUSR, UCATR, UCSTA
        FROM UC IN ENGINEERING_DATA_DATABASE
        WHERE UCUSR = :UIUSR
        ORDER BY UCUSR ASC
```

IBNUIUC

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE UC WITHIN COSET UIUC.
        SET :UCUSR, :UCATR, :UCSTA
```

User Information (UI) Routines

The UI table defines EDL users and contains related profile information.

IBSUI

```
STORE A NEW ROW IN TABLE UI.
        INSERT INTO UI IN ENGINEERING_DATA_DATABASE
        SET UIUSR = :UIUSR,
            UIPWD = :UIPWD,
            UISTA = :UISTA,
            UIUUN = :UIUUN,
            UIDPT = :UIDPT,
            UICMD = :UICMD,
            UIFIN = :UIFIN,
            UIMIN = :UIMIN,
            UILNA = :UILNA,
            UITTL = :UITTL,
            UIDELS = :UIDELS,
            UIDELD = :UIDELD,
            UISTR = :UISTR,
            UICTY = :UICTY,
            UIPHO = :UIPHO,
            UIEDT = :UIEDT
```

IBMUI

```
MODIFY AN EXISTING ROW IN TABLE UI.
        UDPATE UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :UIUSR
        SET UIUSR = :UIUSR,
            UIPWD = :UIPWD,
            UISTA = :UISTA,
            UIUUN = :UIUUN,
            UIDPT = :UIDPT,
            UICMD = :UICMD,
            UIFIN = :UIFIN,
            UIMIN = :UIMIN,
            UILNA = :UILNA,
            UITTL = :UITTL,
            UIDELS = :UIDELS,
            UIDELD = :UIDELD,
            UISTR = :UISTR,
            UICTY = :UICTY,
            UIPHO = :UIPHO,
            UIEDT = :UIEDT
```

IBDUI

```
DELETE AN EXISTING ROW IN TABLE UI.
        DELETE FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :UIUSR
```

IBOUI0

```
OBTAIN A ROW IN TABLE UI VIA ACCESS PATH UI0.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :UIUSR
        ORDER BY UIUSR ASC
```

IBOUI1

```
OBTAIN A ROW IN TABLE UI VIA ACCESS PATH UI1.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UILNA = :UILNA AND
              UIFIN = :UIFIN AND
              UIMIN = :UIMIN
        ORDER BY UILNA ASC, UIFIN ASC, UIMIN ASC, UIUSR ASC
```

IBOUI2

```
OBTAIN A ROW IN TABLE UI VIA ACCESS PATH UI2.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUUN = :UIUUN
        ORDER BY UIUUN ASC
```

IBAUI0

```
OBTAIN A ROW IN TABLE UI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UI0.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR >= :UIUSR
        ORDER BY UIUSR ASC
```

IBAUI1

```
OBTAIN A ROW IN TABLE UI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UI1.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE (UILNA > :UILNA)
```

```
          OR ((UILNA = :UILNA) AND (UIFIN > :UIFIN))
          OR ((UILNA = :UILNA AND UIFIN = :UIFIN) AND (UIMIN > :UIMIN))
          OR (UILNA = :UILNA AND UIFIN = :UIFIN AND UIMIN = :UIMIN)
        ORDER BY UILNA ASC, UIFIN ASC, UIMIN ASC, UIUSR ASC
```

IBAUI2

```
OBTAIN A ROW IN TABLE UI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UI2.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUUN >= :UIUUN
        ORDER BY UIUUN ASC
```

IBEUI1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE UI VIA ACCESS PATH UI1.
SAVE THE CURRENT POSITION IN TABLE UI.
FETCH THE NEXT ROW FROM TABLE UI.
        SET :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
```

IBEUI2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE UI VIA ACCESS PATH UI2.
SAVE THE CURRENT POSITION IN TABLE UI.
FETCH THE NEXT ROW FROM TABLE UI.
        SET :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
```

IBFUI0

```
OBTAIN THE FIRST ROW OF TABLE UI, ORDERED BY ACCESS PATH UI0.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        ORDER BY UIUSR ASC
```

IBFUI1

```
OBTAIN THE FIRST ROW OF TABLE UI, ORDERED BY ACCESS PATH UI1.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        ORDER BY UILNA ASC, UIFIN ASC, UIMIN ASC, UIUSR ASC
```

IBFUI2

```
OBTAIN THE FIRST ROW OF TABLE UI, ORDERED BY ACCESS PATH UI2.
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        ORDER BY UIUUN ASC
```

IBNUI0

```
OBTAIN THE NEXT ROW OF TABLE UI, ORDERED BY ACCESS PATH UI0.
        SET :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
```

IBNUI1

```
OBTAIN THE NEXT ROW OF TABLE UI, ORDERED BY ACCESS PATH UI1.
        SET :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
```

IBNUI2

```
OBTAIN THE NEXT ROW OF TABLE UI, ORDERED BY ACCESS PATH UI2.
        SET :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
```

User Permits (UP) Routines

The UP table defines file permissions granted to individual EDL users.

IBSUP

```
STORE A NEW ROW IN TABLE UP.
        INSERT INTO UP IN ENGINEERING_DATA_DATABASE
        SET UPFIL = :UPFIL,
            UPUSR = :UPUSR,
            UPMOD = :UPMOD
```

IBMUP

```
MODIFY AN EXISTING ROW IN TABLE UP.
        UDPATE UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL = :UPFIL AND
              UPUSR = :UPUSR
        SET UPFIL = :UPFIL,
            UPUSR = :UPUSR,
            UPMOD = :UPMOD
```

IBDUP

```
DELETE AN EXISTING ROW IN TABLE UP.
        DELETE FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL = :UPFIL AND
              UPUSR = :UPUSR
```

IBOUP0

```
OBTAIN A ROW IN TABLE UP VIA ACCESS PATH UP0.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL = :UPFIL AND
              UPUSR = :UPUSR
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBOUP1

```
OBTAIN A ROW IN TABLE UP VIA ACCESS PATH UP1.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL = :UPFIL
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBOUP2

```
OBTAIN A ROW IN TABLE UP VIA ACCESS PATH UP2.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPUSR = :UPUSR
        ORDER BY UPUSR ASC
```

IBAUP0

```
OBTAIN A ROW IN TABLE UP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UP0.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE (UPFIL > :UPFIL)
          OR ((UPFIL = :UPFIL) AND (UPUSR > :UPUSR))
          OR (UPFIL = :UPFIL AND UPUSR = :UPUSR)
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBAUP1

```
OBTAIN A ROW IN TABLE UP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UP1.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL >= :UPFIL
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBAUP2

```
OBTAIN A ROW IN TABLE UP USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH UP2.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPUSR >= :UPUSR
        ORDER BY UPUSR ASC
```

IBEUP1

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE UP VIA ACCESS PATH UP1.
SAVE THE CURRENT POSITION IN TABLE UP.
FETCH THE NEXT ROW FROM TABLE UP.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBEUP2

```
OBTAIN THE NEXT DUPLICATE ROW FROM TABLE UP VIA ACCESS PATH UP2.
SAVE THE CURRENT POSITION IN TABLE UP.
FETCH THE NEXT ROW FROM TABLE UP.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBFUP0

```
OBTAIN THE FIRST ROW OF TABLE UP, ORDERED BY ACCESS PATH UP0.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBFUP1

```
OBTAIN THE FIRST ROW OF TABLE UP, ORDERED BY ACCESS PATH UP1.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBFUP2

```
OBTAIN THE FIRST ROW OF TABLE UP, ORDERED BY ACCESS PATH UP2.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        ORDER BY UPUSR ASC
```

IBNUP0

```
OBTAIN THE NEXT ROW OF TABLE UP, ORDERED BY ACCESS PATH UP0.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBNUP1

```
OBTAIN THE NEXT ROW OF TABLE UP, ORDERED BY ACCESS PATH UP1.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBNUP2

```
OBTAIN THE NEXT ROW OF TABLE UP, ORDERED BY ACCESS PATH UP2.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBOFIUP

```
USING COSET FIUP, OBTAIN THE ROW FROM TABLE FI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE UP
        SELECT FIFIL, FIHOS, FIFUN, FIPFN, FILNA, FIFTC, FIUSR, FICT,
          FIMOD, FISTA, FIVSN
        INTO :FIFIL, :FIHOS, :FIFUN, :FIPFN, :FILNA, :FIFTC, :FIUSR,
            :FICT, :FIMOD, :FISTA, :FIVSN
        FROM FI IN ENGINEERING_DATA_DATABASE
        WHERE FIFIL = :UPFIL
```

IBOUIUP

```
USING COSET UIUP, OBTAIN THE ROW FROM TABLE UI THAT OWNS SPECIFIC
ROWS IN MEMBER TABLE UP
        SELECT UIUSR, UIPWD, UISTA, UIUUN, UIDPT, UICMD, UIFIN, UIMIN,
          UILNA, UITTL, UIDELS, UIDELD, UISTR, UICTY, UIPHO, UIEDT
        INTO :UIUSR, :UIPWD, :UISTA, :UIUUN, :UIDPT, :UICMD, :UIFIN,
            :UIMIN, :UILNA, :UITTL, :UIDELS, :UIDELD, :UISTR, :UICTY,
            :UIPHO, :UIEDT
        FROM UI IN ENGINEERING_DATA_DATABASE
        WHERE UIUSR = :UPUSR
```

IBFFIUP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE UP WITHIN COSET FIUP, USING
ACCESS PATH UP1.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPFIL = :FIFIL
        ORDER BY UPFIL ASC, UPUSR ASC
```

IBFUIUP

```
OBTAIN THE FIRST ROW FROM MEMBER TABLE UP WITHIN COSET UIUP, USING
ACCESS PATH UP2.
        SELECT UPFIL, UPUSR, UPMOD
        FROM UP IN ENGINEERING_DATA_DATABASE
        WHERE UPUSR = :UIUSR
        ORDER BY UPUSR ASC
```

IBNFIUP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE UP WITHIN COSET FIUP.
        SET :UPFIL, :UPUSR, :UPMOD
```

IBNUIUP

```
OBTAIN THE NEXT ROW FROM MEMBER TABLE UP WITHIN COSET UIUP.
        SET :UPFIL, :UPUSR, :UPMOD
```

Vendor Information (VI) Routines

The VI table defines vendor names and vendor codes.

IBSVI

```
STORE A NEW ROW IN TABLE VI.
        INSERT INTO VI IN ENGINEERING_DATA_DATABASE
        SET VIVEN = :VIVEN,
            VINAM = :VINAM,
            VISTR = :VISTR,
            VICTY = :VICTY,
            VIPHO = :VIPHO
```

IBMVI

```
MODIFY AN EXISTING ROW IN TABLE VI.
        UDPATE VI IN ENGINEERING_DATA_DATABASE
        WHERE VIVEN = :VIVEN
        SET VIVEN = :VIVEN,
            VINAM = :VINAM,
            VISTR = :VISTR,
            VICTY = :VICTY,
            VIPHO = :VIPHO
```

IBDVI

```
DELETE AN EXISTING ROW IN TABLE VI.
        DELETE FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VIVEN = :VIVEN
```

IBOVI0

```
OBTAIN A ROW IN TABLE VI VIA ACCESS PATH VI0.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VIVEN = :VIVEN
        ORDER BY VIVEN ASC
```

IBOVI1

```
OBTAIN A ROW IN TABLE VI VIA ACCESS PATH VI1.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VINAM = :VINAM
        ORDER BY VINAM ASC
```

IBAVI0

```
OBTAIN A ROW IN TABLE VI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH VI0.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VIVEN >= :VIVEN
        ORDER BY VIVEN ASC
```

IBAVI1

```
OBTAIN A ROW IN TABLE VI USING AN APPROXIMATE KEY VALUE AND ACCESS
PATH VI1.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        WHERE VINAM >= :VINAM
        ORDER BY VINAM ASC
```

IBFVI0

```
OBTAIN THE FIRST ROW OF TABLE VI, ORDERED BY ACCESS PATH VI0.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        ORDER BY VIVEN ASC
```

IBFVI1

```
OBTAIN THE FIRST ROW OF TABLE VI, ORDERED BY ACCESS PATH VI1.
        SELECT VIVEN, VINAM, VISTR, VICTY, VIPHO
        FROM VI IN ENGINEERING_DATA_DATABASE
        ORDER BY VINAM ASC
```

IBNVI0

```
OBTAIN THE NEXT ROW OF TABLE VI, ORDERED BY ACCESS PATH VI0.
        SET :VIVEN, :VINAM, :VISTR, :VICTY, :VIPHO
```

IBNVI1

```
OBTAIN THE NEXT ROW OF TABLE VI, ORDERED BY ACCESS PATH VI1.
        SET :VIVEN, :VINAM, :VISTR, :VICTY, :VIPHO
```

Standard EDL OVCAP Subroutines C

Standard EDL OVCAP Subroutines C

Routine	Description
ATTACH	Prepares the user for entry into a particular application, attaches default files, ensures that required files are attached, and allows the user to attach other files.
DDNPRE	Prepares for entry into DDN.
DDNPRE2	Prepares for a DDN transfer.
DFMAIN	Allows the user to list, add, and delete the names of files that are to be attached automatically when an ICEM application is entered.
DISPLY (POP)	Displays records on EEEDL9, and provides a variety of user options.
EDITF	Calls up a file selected through RETRIEVE DATA and places the user in the appropriate editor.
EDITR	Returns the edit file called by EDITF and removes it from the list of local files.
EDLOGN	Establishes the name for the EDL log file.
EQUIT	Performs any required processing before ending the EDL session.
FIACQ	Acquires a file within EDL (same as ATTATT).
FIARCH	Archives files.
FICORR	Displays the file correction menu.
FIDEFI	Allows a user to define a file.
FIDEL	Prompts for the host information to delete a file.
FIEDIT	Prompts for the file name to edit, ensures that it is editable, and performs the PUTVARS.
FILOWN	Displays all files in a retrieval list and allows the user to select from the list for a detailed listing.
FILPER	Creates a retrieval list consisting of all permitted files on the current host, and allows the user to select from the list for a detailed listing.
FIRECL	Reclaims files.
FIREQ	Requests a tape.
FIROUT	Routes a local file known to EDL to the printer.
FIRST	Initializes the stack and pushes the user's first task.

Routine	Description
FMUPD	Displays the Family Management menu and the load option menu from the selection entered by the user.
FPMENU	Displays the File Permission menu.
GETAPN	Pushes a retrieval task onto the stack based on the APN in the DI record.
GMMGMT	Allows the user to list all members within a group, and add members to or delete members from a group.
GPLIST	Lists all groups within EDL.
GPMGMT	Allows the user to list all information about a group, and add, delete, or change groups.
GTMGMT	Allows the user to list authorized task categories for a group, and to allow or remove access privileges.
ISMLOG	Translates the Solid Modeler log file into the standard log file format.
ISMNEW	Prepares for entering the Solid Modeler when creating a new model.
LOG	Processes an application log file.
PATPRE	Prepares for entry into PATRAN.
PERSON	Allows users to change their profiles.
PFUPD	Displays the Part Family Relationship Management menu and the load option menu from the selection entered by the user.
PIUPD	Displays the Part Management menu and the load option menu from the selection entered by the user.
POSTRP	Performs general postprocessing for reports.
PVUPD	Displays the Part Vendor Relationship Management menu and the load option menu from the selection entered by the user.
RDATAF	Prepares for production of the full data report. The report is actually created by the EDL task names in the global variable RDATFU.
RDATFU	Displays the full data report.
RELACC	Creates a list of all submitted data in a specified release procedure that may be accepted by the current user.
RELADM	Allows the user to list, add, or delete release procedures, and to manage releasers and reviewers.
RELCHG	Allows reviewers to change the review signature for a pending data.

Routine	Description
RELCS	Changes a review signature.
RELF	Dislays the option menu for finalizing data and then calls the appropriate routine to release or reject the data.
RELFIN	Creates a list of all pending data that the user may finalize for a specified release procedure.
RELR	Prompts for whether to display the data to be reviewed, and displays the data if necessary.
RELREV	Creates a list of all pending data for a specified release procedure for which the user is a releaser.
RELRS	Displays the Review Disposition option menu and stores the specified release signature.
RELS	Submits data for review/release.
RELSUB	Creates a list of all of the current user's data that may be submitted for review/release.
RLPRE	Prepares for entry to the Solid Modeler when retrieving an object from a library or when retrieving a workspace.
SAVLOC	Lists local file and asks the user whether or not any of them should be saved.
TASKS	Displays a list of task commands available to the user.
TRANSF	Transfers data from one application data type to another.
TRMCON	Updates the user's terminal configuration.
UPDATA	Loads or updated engineering data information.
UPFILE	Updates data information for a file.
USMGMT	Allows the DBA to manipulate users.
VIUPD	Displays the Vendor Management menu and the Load Option menu from the selection entered by the user.

Index

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

Comments (continued from other side)

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA
Technology & Publications Division
ARH219
4201 N. Lexington Avenue
Arden Hills, MN 55126-6198

EDL Customization for NOS 60000168 A

We value your comments on this manual. While writing it, we made some assumptions about who would use it and how it would be used. Your comments will help us improve this manual. Please take a few minutes to reply.

Who are you?

- ☐ Manager
- ☐ Systems analyst or programmer
- ☐ Applications programmer
- ☐ Operator
- ☐ Other _____

How do you use this manual?

- ☐ As an overview
- ☐ To learn the product or system
- ☐ For comprehensive reference
- ☐ For quick look-up

What programming languages do you use? _____

_____

How do you like this manual? Check those questions that apply.

Yes	Somewhat	No	
☐	☐	☐	Is the manual easy to read (print size, page layout, and so on)?
☐	☐	☐	Is it easy to understand?
☐	☐	☐	Does it tell you what you need to know about the topic?
☐	☐	☐	Is the order of topics logical?
☐	☐	☐	Are there enough examples?
☐	☐	☐	Are the examples helpful? (☐ Too simple? ☐ Too complex?)
☐	☐	☐	Is the technical information accurate?
☐	☐	☐	Can you easily find what you want?
☐	☐	☐	Do the illustrations help you?

Comments? If applicable, note page and paragraph. Use other side if needed.

Would you like a reply? ☐ Yes ☐ No

From:

Name _____ Company _____

Address _____ Date _____

_____ Phone _____

Please send program listing and output if applicable to your comment.