CA 2E

Administration Guide

Release 8.7

Documentation Changes

The following documentation updates have been made since the last release of this documentation:

- Allow SQL Record Level Access

  YSQLFMT Model Value (see page 181) – Added a model value.

- Allow RLA access over DDL database
  - The Model Library (see page 15)
  - DDL Collection (see page 17)
  - Setting Model Values (see page 34)
  - SQL/DDS/DDL (see page 37)
  - Implementing DDL (see page 39)
  - Implementing SQL/DDL and DDS in the Same Model (see page 40)
  - Prompting YCRTMDLLIB (see page 47)
  - SQL/DDL Implementation (see page 175)
  - Separate View and Index Creation (see page 187)
  - Edit Access Path Auxiliaries Panel (see page 189)
  - Direct Table Access (see page 193)
  - Data Access Method Option (see page 194)
- Allow SQL/DDL generation without hard-coded schema name
  - DDL Collection (see page 17)
  - YSQLCOL Model Value (see page 182)
- Allow LVLCHK(*YES) for SQL/DDL indexes having RCDFMT keyword
  - YLVLCHK Model Value (see page 183)
  - YSQLFMT Model Value (see page 181)
- YSQLFMT override
  - Implementing DDL (see page 39)
  - YSQLFMT Model Value (see page 181)
- Meaningful Names for SQL/DDL
  - YSQLVNM Model Value (see page 180)
- Option to Generate RLA against DDL
  - YDDLDBA Model Value (see page 180)

- F7=Auxiliaries in the DDL Implementation
  - Separate View and Index Creation (see page 187)
  - Suppressing Index Generation (see page 188)
  - Edit Access Path Auxiliaries Panel (see page 189)
- Meaningful Names for SQL/DDL - Control Table vs Fields
  - Long Name Scenarios (see page 178)
  - YSQLVNM Model Value (see page 180)
  - YDDLDBA Model Value (see page 180)
- DDL Limitation
  - Implementing DDL (see page 39)
- Field names that not work for DDS, but not for SQL/DDL Implementation
  - Understanding Extended SQL Naming (see page 177)

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This chapter describes the libraries that represent CA 2E and the development environment. A description of the installation and upgrade process is also provided as well as an introduction to the Toolkit prototyping facilities.

This section contains the following topics:

CA 2E Product Libraries

The CA 2E product libraries contain the files and objects you must use products. The product libraries include the CA 2E base product library (Y2SY) and the Toolkit base product library (Y1SY).

Note: All object names begin with the letter "Y."

In addition to Y2SY and Y1SY, CA 2E ships base product libraries that contain appropriate default national language libraries, the High Level Language (HLL) source generators, the CA 2E null model, and a library containing some source.

Note: At many sites, you see only Y1SY, Y2SY, Y2SYMDL, and Y2SYSRC listed on your computer. The remaining libraries were merged into the base product libraries during installation.

The following table provides a complete list of the CA 2E product libraries:

Ship Name	Library or Folder	Suggested Restore Library or Folder
Y1SY	Toolkit base	Y1SY
Y1SYVnll1	Toolkit LDOs	Y1SY
Y2SY	base	Y2SY
Y2SYVnll1	LDOs	Y2SY
Y2SYMDL	null model	Y2SYMDL
Y2SYCBL	COBOL generators	Y2SYCBL
Y2SYRPG	RPG generators	Y2SY
Y2SYSRC	source	Y2SYSRC
YLUSLIB0	Security Library	YLUSLIB02
Notes	"nll" refers to the national language; for example, ENG for English or JPN for Japanese. The Security Library must be restored to the YLUSLIB0 library.	

The supplied source for CA 2E resides in source files in the Y2SYSRC library. The source for Toolkit resides in the Y1USRSRC file in the Y1SY library. You can obtain a list of these members with the i OS Work with Members Using PDM (WRKMBRPDM) command.

The YCRTLUSLIB command that runs during product installation automatically updates or creates the YLUSLIB library.

More Information

For more information about the WRKMBRPDM command, see Volume 5 of the i OS CL Reference.

CA 2E Development Environment

The CA 2E development environment is the model within which:

- The development team creates the application objects.
- The objects are generated.
- The objects are tested individually to determine whether they function correctly and efficiently.

Development Libraries

Each CA 2E model must reside in a single library that is called the "model library." Each model must also have a generation library that is associated with it. The model and generation library are also, when necessary, associated with a Structured Query Language (SQL) collection. The collection is a separate library similar to the generation library.

Generation Library

The generation library holds the CA 2E generated source, compiled objects, and help text. You can move generated objects from the generation library to any other library. And you can move the generation library to an environment that does not contain CA 2E product libraries.

The Model Library

Each model library holds the database files that make up the model. The associated generation library contains the source CA 2E generates for the model and the compiled objects from that source. Think of the model library and the generation library as a pair. The model library also contains other necessary i OS objects and a CA 2E job description. If the developer generates SQL or DDL, the model library also refers to an SQL collection.

The Create Model Library (YCRTMDLLIB) command creates the libraries that comprise a model.

SQL Collection

CA 2E lets you use SQL in place of or with DDS in data definition statements. You use SQL to define tables, views, rows, and columns, rather than IBM i physical files, logical files, records, and fields.

You can implement SQL at both the system level and the model level. If you:

- Implement SQL at the system level, all new models default to SQL.
- Implement SQL at the model level, all access paths, and functions default to SQL. Within a model, you can also specify SQL for individual access paths and functions.

The i OS operating system contains the execution objects necessary to execute applications you generate with SQL. However, if you want to generate and compile applications that use SQL, you must have:

- IBM's SQL/400 program products installed.
- The QSQL library that is loaded on your IBM i.

To use SQL in a model environment, an SQL collection, equivalent to an IBM i library, must be associated with the model library. The YCRTMDLLIB command can automatically create the SQL collection. This command includes the SQLLIB parameter which lets you create the collection for a particular model.

You can create an SQL collection for an existing model with the Create SQL Library (YCRTSQLLIB) command. This command updates the YSQLLIB model value for the associated model. You then use the Change Model Value (YCHGMDLVAL) command to update the YDBFGEN model value to SQL.

For more information about:

- The YCRTSQLLIB and YCHGMDLVAL commands, see the CA 2E *Command Reference Guide*.
- For differences between earlier versions of CA 2E SQL implementations and current implementations, see the appendix SQL Implementation (see page 175).
- IBM's Structured Query Language, see your IBM documentation.

DDL Collection

CA 2E lets you use DDL in place of or with DDS in data definition statements. You use DDL to define tables, indexes, rows, and columns, rather than IBM i physical files, logical files, records, and fields. There is no view generation when DDL is used as generation method.

You can implement DDL at both the system level and the model level. If you:

- Implement DDL at the system level, all new models default to DDL.
- Implement DDL at the model level, all access paths, and functions default to DDL. Within a model, you can also specify DDL for individual access paths and functions.

The i OS operating system contains the execution objects necessary to execute applications you generate with DDL. However, if you want to generate and compile applications that use DDL, you must have:

- IBM's SQL/400 program products installed.
- The QSQL library that is loaded on your IBM i.

To use DDL in a model environment, an SQL collection, equivalent to an IBM i library, must be associated with the model library. The YCRTMDLLIB command can automatically create the SQL collection. This command includes the SQLLIB parameter which lets you create the collection for a particular model.

You can create an SQL collection for an existing model with the Create SQL Library (YCRTSQLLIB) command. This command updates the YSQLLIB model value for the associated model. You then use the Change Model Value (YCHGMDLVAL) command to update the YDBFGEN model value to DDL.

For more information about:

- The YCRTSQLLIB and YCHGMDLVAL commands, see the CA 2E *Command Reference Guide*.
- For differences between earlier versions of CA 2E SQL implementations and current implementations, see the appendix SQL/DDL Implementation.
- IBM's Structured Query Language, see your IBM documentation.

Note: In earlier releases of CA 2E, it was only possible to create SQL/DDL related tables, views, and indexes into SQL Collections. However, the access path generation process has now been modified to enable creation of SQL/DDL tables, views, and indexes into normal libraries, in addition to SQL Collections. The SQL/DDL generation has now been modified to do the following tasks:

- The YSQLLIB model value can also hold a normal library (non-SQL collection).
- The user is given the option to decide whether to generate the hard coded value present in the YSQLLIB model value into the generated source, through the YSQLCOL model value.

- If the library/SQL collection is not generated into the source during generation, when a compile is submitted to create the SQL/DDL objects, the objects are created into the library/SQL collection that is specified for the YSQLLIB model value.

Prototyping an Application

Toolkit prototyping facilities let you interactively present a simulated system design. End-users can preview the prototyped system at a workstation.

Toolkit prototyping includes:

- Realistic display attributes.
- Specification of sample data values for fields.
- Function key and value-dependent branching between panels.
- Direct attachment of panels to Toolkit or CL menus.
- No compilation.
- A link with CA 2E that lets you return to CA 2E after prototyping.

The prototyping facilities use the following Toolkit commands:

- **YDSPPNL**—Display Panel Design command displays prototype panel values and sets up sample data.
- **YWRKMNU**—Work with Menus command creates menus to display prototype panels

Note: Specify the YDSPPNL command as the menu action, with the desired Toolkit panel design as the option.

- **YGO**—Go to Menu command displays menus.

Creating Prototype Panels

You can create prototype panels in the following ways. Each method converts CA 2E function device designs from a CA 2E design model into Toolkit panel designs.

- Use the CA 2E Convert Model Panel Designs (YCVTMDLPNL) command.
- Use the CA 2E animate options.

CA 2E Animation provides a direct link between CA 2E and Toolkit. This includes converting device designs to Toolkit, full access to all Toolkit panel editing and simulation functions, and the ability to return directly to your model.

Toolkit panel designs reside in a panel design file in a nominated library. The default is your model library. You create panel design files using one of the following options:

- The Toolkit Create Design File (YCRTDSNF) command.
- The CA 2E animate options. If the panel design file does not exist when you convert a model panel, it is created for you.

For more information about:

- Toolkit commands, see the Toolkit Reference.
- Details of the Toolkit prototyping, see the "Design Aids" chapter in the Toolkit Concepts guide.

Naming Prototype Panels

The YCVTMDLPNL command and the animate options create one or more prototype panel designs for each function with a device design. Each prototype panel is given the name of the source member for the program object, as specified in the Edit Function Details panel. For multiple panel designs, each design name includes a numeric suffix. For example,

- An EDTFIL function with a source member name UUJQEFR results in UUJQEFR1 as the name of the prototype panel.
- An EDTRCD function with a source member name UUJQE1R results in UUJQE1R1 for the key panel and UUJQE1R2 for the detail panel.

Using CA 2E Device Design Animation

CA 2E animation provides a direct link between CA 2E and Toolkit prototyping facilities. This lets you interactively simulate your CA 2E system design using Toolkit and easily return to your CA 2E design model.

You animate a CA 2E device design in the following ways:

- Press F2 from the CA 2E device design editor.
- Enter A to animate a selected function from the following CA 2E panels:
  - Open Functions
  - Edit Function Devices.

The CA 2E Animate Function Panels panel displays.

```
Animate Function Panels                SYMDL

 Convert Model Panel. : Y (Y-Yes, N-No)  Convert all panels : Y (Y-Yes, N-No)
   Replace Navigation : N (Y-Yes, N-No)
   Replace Action Bar : N (Y-Yes, N-No)
   Clear Narrative. . : N (Y-Yes, N-No)
   Clear Test Data. . : N (Y-Yes, N-No)

 Panel Name(s). . . . : *SRCMBR   *SRCMBR, *SELECT, *panel, name
   File . . . . . . . : YDSNPNL   Name
   Library. . . . . . : *MDLLIB   *MDLLIB, *GENLIB, *LIBL, name
   Member . . . . . . : *FILE     *FILE, name

 Display. . . . . . . : Y (Y-Yes, N-No)
   Display Option . . : 1 1-DSPDTA, 2-DSPATR, 3-CHGDTA, 4-WRKPNL

   Return to this device design . . . . . : N (Y-Yes, N-No)


Enter=Execute   F3=Exit
```

This panel, the Animate panel, is the bridge between CA 2E and Toolkit. Use it to specify:

- Whether to convert the CA 2E device design to a Toolkit panel design.
- Animation of CUA panels, action bars, and window definitions.
- Whether to keep or replace information that is associated with the Toolkit panel design, such as test data and command key and action bar definitions.
- The name and location of the Toolkit panel design.
- Whether to transfer control to Toolkit.
- Where to return within CA 2E.

The following sections discuss these actions and alternative ways to achieve them using the Toolkit commands.

Converting an CA 2E Device Design to Toolkit

You can convert a CA 2E device design into one or more Toolkit panel designs. By default, this process retains the existing Toolkit panel design's command key and action bar navigation, narrative, and data even when you download a new version of the panel design.

You convert a CA 2E device design in either of the following ways:

- Use the CA 2E Convert Model Panel (YCVTMDLPNL) command.
- Enter Y for the Convert Model Panel option on the CA 2E Animate Function Panels panel.

If you enter N for the Convert Model Panel option, control is transferred to Toolkit without converting the CA 2E device design.

Note: To convert multiple CA 2E device designs to Toolkit panel designs in one step in batch, use the YCVTMDLPNL command.

Conversion is needed in the following cases:

- A panel design corresponding to your CA 2E device design does not exist in Toolkit.
- You changed the CA 2E device design and must update the Toolkit panel design to reflect the changes. It is good practice to synchronize the Toolkit panel design with the corresponding CA 2E device design.
- You want to replace the command key or action bar navigation, narrative, or test data you previously defined for the Toolkit panel design.

Converting Multi-Screen Functions

By default, if you convert a multi-screen function, such as Edit Record, all panels are converted. The panels are automatically linked together so that you can scroll among them within Toolkit.

If you animate from a CA 2E device design by pressing F2, you can choose to convert only the panel that is displayed by setting the Convert All Panels option to N.

Replacing Toolkit Navigation and Data

By default, the conversion retains any command key, action bar, narrative, and data that is entered for the Toolkit panel. The Replace and Clear options let you replace this information. If you are using the CA 2E Animate Function Panels panel, these options are effective only when the Convert Model Panel option is Y.

Replace Navigation and Replace Action Bar

Use these options to specify whether to keep or replace the command key or action bar navigation you defined for the Toolkit panel design. The default is N (keep the Toolkit navigation).

If you enter Y, the navigation you defined in Toolkit is replaced with the standard CA 2E command key functionality; for example, F3=*EXIT, F12=*PRV. Any function-to-function navigation you defined in the action diagram creates a Toolkit navigation with the value *SAME.

Note: If you have not assigned a function key for exit on your Toolkit panel design and you do not enter Y to replace navigation, CA 2E automatically assigns the default, F3, for exit.

Clear Narrative and Clear Test Data

Use these options to keep or clear any narrative or test data you entered for the Toolkit panel design. The default is N, retain the Toolkit information

For more information about the YCVTMDLPNL command, see the CA 2E Command Reference Guide.

Transferring Control to Toolkit

You transfer control to Toolkit in the following ways:

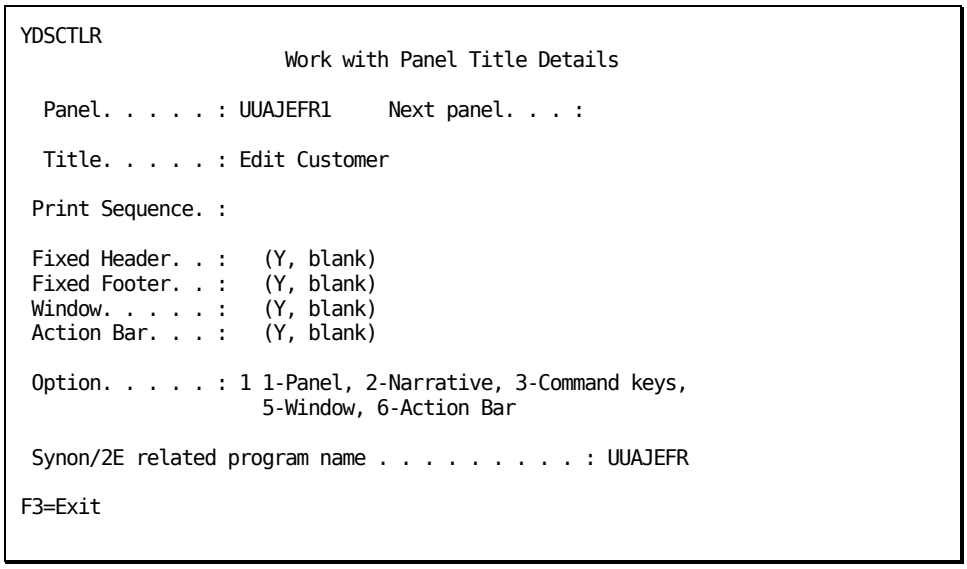
- Enter Y for the Display option on the CA 2E Animate Function Panels panel.
- If you simply want to convert your CA 2E device design to Toolkit and do not need to display the prototype, enter N to return to CA 2E. This is useful to keep your Toolkit panel design and CA 2E device design synchronized.
- Use the Toolkit Work with Panel Design (YWRKPNL) or the Display Panel (YDSPPNL) utilities. You cannot return directly to CA 2E using this method.

Working with a Toolkit Panel Design

You can work with your Toolkit panel design in one of the following ways:

- Enter 4 for the Display Option on the Animate Function Panels panel.
- From the Toolkit YWRKPNL utility, specify the panel you want to edit. For example, specify *SELECT for the panel name and enter 1 next to the appropriate panel name.

The Toolkit Work with Panel Title Details panel displays.



You can use this panel to:

- Edit the panel design.
- Enter or edit narrative.
- Define command key and action bar navigation.
- Define a window.

The Related Program Name option is the source member name of the device design that created this Toolkit panel. It is the link that allows control to be transferred back to CA 2E. This option is blank if the panel was not converted from CA 2E or was converted before CA 2E r5.0.

Note: More than one Toolkit panel can refer to the same source member

Editing the Panel

To edit your Toolkit panel design enter 1 from the Work with Panel Title Details panel. Your Toolkit panel design displays in monochrome. If you want to edit your design in color use the CA 2E device design editor.

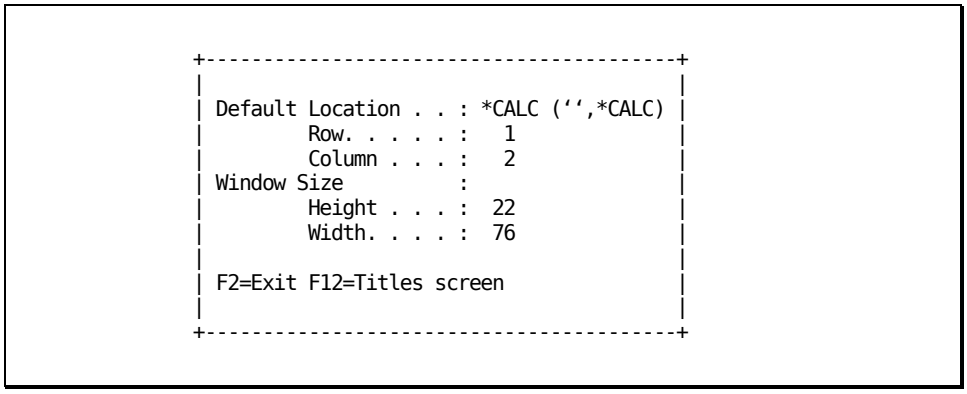
Defining Command Keys

To define command key navigation for your Toolkit panel design enter 3 from the Work with Panel Title Details panel. The Work with Panel Command Key Usage panel displays.

Note: If you are working with an action bar design, assign *ABAR to the function key that is to activate the action bar.

Building a Window Prototype

The Window option is automatically set to Y on the Work with Panel Title Details panel when you convert a CA 2E window device design. Press 5 to edit or display the window definition.



You can specify the location of the window in either of the two following ways:

- Enter specific values for Row and Column.
- Enter *CALC for the Default Location. This causes the upper left-hand corner of the window to display wherever the cursor is located at the time the window is requested. The Row and Column options are ignored.

Define the size of the window by entering values for the Height (in lines) and Width (in characters) options.

Press F12 to continue working with the panel design; press F3 to exit.

Building an Action Bar Prototype

The Action Bar option is automatically set to Y on the Work with Panel Title Details panel when you convert a CA 2E action bar device design.

Enter 6 to edit the action bar definition. The Toolkit Edit Choices panel displays.

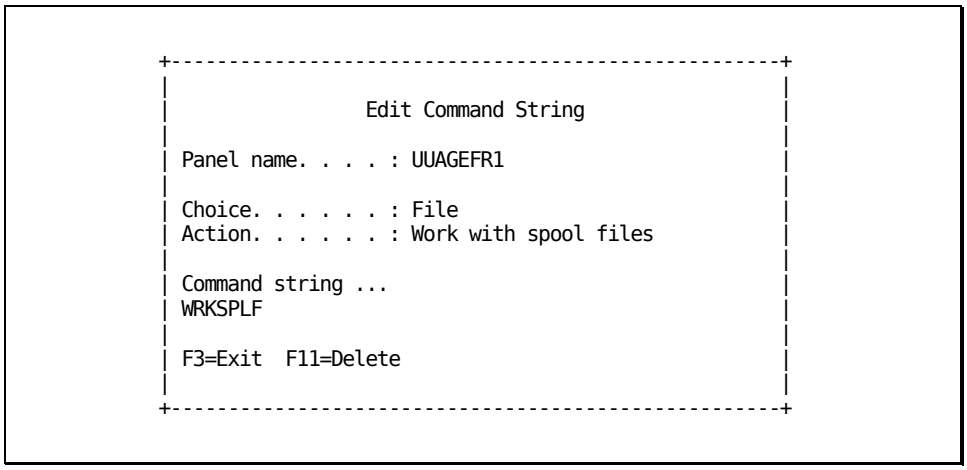
```
YDSCABC  CHANGE                                              MM/DD/YY HH:MM:ss
                            Edit Choices

Panel Name . . . . . :  UUAGEFR1
  File . . . . . :  YDSNPNL
  Library. . . . . :  SYMDL
  Member . . . . . :  YDSNPNL

Choice Sequence .      (position)

Type options, press Enter.
A=Actions  D=Delete

? Sequence Mnemonic Text
    1      E        File
    4      U        Function
   99      H        Help


F3=Exit  F4=Prompt  F9=Go to 'Add' mode  F12=Titles screen
```

The action bar menu choices that currently have actions defined display. Press F9 to add new choices. Enter A to edit the actions for an existing menu choice. The Toolkit Edit Actions panel displays.

```
YDSCABA   CHANGE                                             MM/DD/YY HH:MM:ss
                              Edit Actions

Panel Name. . . . : UUAGEFR1

Choice Sequence . :   3
Choice Text . . . : Work with spool files

Action Number .     (position)

Type options, press Enter.
C=Command String  D=Delete

?  Number  Text                    Next Panel
    2      Open                    *SAME
C   3      Work with spool files   *EXEC
   90      Exit                    *EXIT


F3=Exit  F4=Prompt  F9=Go to 'Add' mode
```

The Next Panel column indicates the result of selecting the corresponding action. For example, Next Panel can contain:

- The name of the CA 2E panel to call when the action is selected.
- *EXIT, which means to exit the panel when the action is selected.
- *EXEC, which lets you execute a command when the action is selected. To edit the command string for *EXEC, enter C for the subfile selector; the following screen displays:



Note: When defining command key navigation, assign *ABAR to the function key that is to activate the action bar. If you converted a action bar device design, the command key assigned to *ACTIONS is automatically assigned *ABAR for the Toolkit command key navigation.

Defining Color

Toolkit panel designs display in full color. Any color assignments you make within CA 2E are automatically converted to Toolkit. This is the recommended method for assigning color to Toolkit panel designs.

Note: Constants separated by a single space are treated as the same constant; as a result, constants share the color assignment given to the leftmost constant.

Alternatively, you can assign colors as you edit your panel design within Toolkit. To do so, position the cursor on the blank before the field to which you want to assign a color and press F16.

The following screen listing the available colors and attributes displays:

```
            £ Blue          q Highlight
            £ Green         q Reverse Image
            £ Pink          q Underline
            £ Red           q Blink
            £ Turquoise     q Column Separator
            £ White         q nondisplay
            £ Yellow

            F3=Exit  F12=Titles screen
```

On some terminals this screen will appear as shown below.

```
    1       1. Blue           Highlight
            2. Green          Reverse Image
            3. Pink           Underline
            4. Red            Blink
            5. Turquoise      Column Separator
            6. White          nondisplay
            7. Yellow

            F3=Exit  F12=Titles screen
```

Note: The mono display attributes display with an input field instead of a check box to the left of the attribute. You select attributes by entering a slash (/) in the corresponding input field. The slash is the default selection character; it is contained in the IBM i message, CPX5A0C, in the QCPFMSG file in the QSYS library.

Not all attributes are available for each color. The following table lists the valid combinations supported by DDS.

Color	CLR	CS	BL	UL	HI	RI	ND

Color	CLR	CS	BL	UL	HI	RI	ND
Green	GRN						
						X	
					X1		
					X1	X	
				X			
				X		X	
				X	X1		
							X
White	WHT						
						X	
				X			
Red	RED						
						X	
			X				
			X			X	
				X			
				X		X	
			X	X			
							X
Turquoise	TRQ	X					
		X				X	
		X					
		X		X		X	
Yellow	YLW	X					
		X				X	
		X		X			
Pink	PNK						
						X	
				X			

Color	CLR	CS	BL	UL	HI	RI	ND
				X		X	
Blue	BLU						
						X	
				X			
							X

"X1" indicates that the green highlight is white.

Note: Highlight is only allowed for green, Blink is only allowed for red, and column separators are required for turquoise and yellow.

Entering Sample Data

To display your Toolkit panel design in order to enter sample demonstration data, use one of the following:

- Enter 3 for the Display Option on the CA 2E Animate Function Panels panel. You can return to CA 2E by pressing the Home key from any Toolkit panel design. See the Returning to CA 2E section in this chapter for details.
- Use the Toolkit YDSPPNL command and specify *CHGDTA for the Option parameter.

All fields of your panel design will be available for input and the data you enter is retained on exit.

Displaying Prototype Panels

To display Toolkit panel designs to demonstrate your system design, use one of the following:

- Enter 1 for the Display Option on the CA 2E Animate Function Panels panel. You can return to CA 2E by pressing the Home key from any Toolkit panel design. See the Returning to CA 2E section in this chapter for details.
- Use the Toolkit Display Panel Design (YDSPPNL) command and specify *DSPDTA for the Option parameter.
- Use the Toolkit YWRKPNL utility and select option 8.

Any sample data you entered previously is displayed. You can enter data in all input fields; however, any data you enter is not retained on exit.

Returning to CA 2E

If you used CA 2E animation to transfer to Toolkit, you can return directly to the CA 2E panel or device design from which you started the animation. You return from Toolkit in one of the following ways:

- If you are editing your Toolkit panel design, press F3 to return to . You will be given an opportunity to save any changes you made.
- If you are simulating your application and have defined navigation, press the key you assigned as an exit key (usually F3) to return to CA 2E.
- If you are using Toolkit to work with your panel design, for example, viewing it, simulating it, or entering data, press the function key assigned for exit or Home to return to CA 2E.

Note: See the documentation for your terminal or computer to learn which is the Home key on your system. In addition, the Home key is designed to position the cursor on the first input-capable field on the screen. As a result, if the cursor is elsewhere on the screen, you need to press Home twice to return to .

The function to which you return depends on the value specified for the Return to this Device Design option on the Animate Function Panels panel when you transferred control to Toolkit:

- If the value was N, the default, you return to the function corresponding to the last Toolkit panel design you accessed. As a result, the corresponding function is loaded automatically into Open Functions.
- The function loaded is the function whose implementation name appears in the related program name field on the Work with Panel Title Details panel. You can edit the implementation name using this panel. This name automatically defaults when you convert a panel.
- If the value was Y, you return to the function from which you invoked the animation.

Note: If you invoked Toolkit using Toolkit commands, you cannot return directly to .

Chapter 2: Creating and Managing Your Model

This chapter describes the tasks associated with creating a model and provides procedures to use the Create Model Library (YCRTMDLLIB) command. It also describes how to use some of the CA 2E commands and i OS commands to manage models.

The tasks described in this chapter are usually assigned to a system administrator.

This section contains the following topics:

Creating a CA 2E Design Model

Each CA 2E design model is held in a set of i OS database files that must reside in a single library. A machine can hold many models, each in a different library. The YCRTMDLLIB command creates:

- A model library.
- The model library's objects.
- A Toolkit library list for the model.
- An associated library (the generation library).

Note: The generation library holds the source CA 2E generates from the model and other i OS objects.

Each model library you create and its associated generation library are a pair. The model library contains a model value, YGENLIB, that names the associated generation library. The design model can also have an SQL collection.

The YCRTMDLLIB command also contains parameters that set some of the model values for the new model.

Before You Create a Model

Before you create a model, make sure you:

- Understand how YCRTMDLLIB command parameters affect model values.
- Place the Toolkit and CA 2E libraries in your library list.
- Sign on with the correct user profile.

More Information

For more information about:

- The function of some model values, see the Setting Model Values section in this chapter.
- Adding these libraries to your library list, see the Changing Your Library List section in this chapter.
- The authority levels a user profile must have to create a model and about how the user profile affects model ownership, see the Signing on with the Correct User Profile section in this chapter.

Setting Model Values

The YCRTMDLLIB command includes parameters that specify certain model values for the new model, such as the HLL in which programs will generate. The parameters define other model values, such as the prefix for object names. You can accept the default values for these parameters or you can change the default to customize the model for a particular environment.

Before you change a model value default, make sure you understand its purpose and identify the impact that your change will have on the model environment. This section describes the following model values:

■ Default target HLL	■ Model library name
■ Design standard	■ Naming prefixes
■ Generation library name	■ National language
■ HLL naming convention	■ Open access
■ Message file name	■ SQL/DDS/DDL generation

More Information

For more information about YCRTMDLLIB model values, see the CA 2E *Command Reference Guide*.

Model Library Name

The model library contains the CA 2E design model and application design.

You define the name of the model library with the MDLLIB parameter. You must include this parameter when you execute YCRTMDLLIB.

Select the name for the model carefully. Other parameters, such as GENLIB and SQLLIB, default to options that allow CA 2E to build the names for libraries from the prefix of the model library.

Use the following conventions:

- Begin the model library name with an identifying prefix of up to seven characters.
- Follow the prefix with the characters "MDL."

Identifying prefixes might include the owner of the model, the application, or application level. For example, you might use INVMDL for an inventory model.

Generation Library Name

The model generation library contains CA 2E generated source and compiled objects.

You specify the name for the generation library with the GENLIB parameter. You can create the name yourself or you can accept the default and allow to create the name from the model library name according to the following rules:

- If the model library name contains the characters "MDL," CA 2E drops these characters and replaces them with "GEN." For example, if you name the model library "INVMDLR40," CA 2E creates the name "INVGENR40" for the generation library.
- If the model library name does not contain "MDL," CA 2E appends "GEN" to the model name. For example, if the name of the model library is "INVNTRY," creates the name "INVNTRYGEN" for the generation library.
- If the model name is more than seven to nine characters in length,CA 2E truncates the suffix. If the model name is ten characters or more,CA 2E sends an error message on model creation.

Open Access

When you enter a CA 2E model you specify one of the following three user types:

- *DSNR (Designer)
- *PGMR (Programmer)
- *USER (User)

The user type you specify determines when you can access the model and what types of changes you can make.

The Open Access parameter OPNACC lets you specify whether multiple designers (*DSNR) and programmers (*PGMR) can work in the model at the same time. This parameter sets the YOPNACC model value. The values for this parameter and the implications of each are as follows:

- ***NO**—Restricts access to the model to only one *DSNR at a time. If a designer is working in the model, programmers and other designers are denied access to the model.
- ***YES**—Allows multiple designers and programmers to work in the model concurrently. enables file and field locking to prevent two designers from updating the same file or field at the same time.

More Information

For more information about:

- Designer and programmer user types, see the Controlling User Access section in the "Using Your Development Environment" chapter in this guide.
- File and field locking, see the Locking Objects section in the "Using Your Model" chapter in this guide.

Design Standard

The *Design Standard* is the default set of values a model uses for CA 2E panel design requirements, such as function key usage and field display attributes.

Parameter DSNSTD sets the default values for all features that affect design standards. These include:

- The default standard headers and footers for primary and secondary panels.
- The default function keys.
- The initial value for the YLHSFLL model value. This value controls the appearance of field text leaders on panel designs.
- The YCUAPMT model value. This value controls whether CUA Prompt—F4 (Display a list of allowed values)—is enabled.
- The YCUAEXT model value. This value provides additional CUA device design compliance.

You can set DSNSTD to one of the following:

- ***CUATEXT**—Sets the defaults to IBM Systems Application Architecture (SAA) Common User Access (CUA) Text Subset of the Graphical Model
- ***CUAENTRY**—Sets the defaults to IBM SAA CUA Entry model standard
- ***S38**—Sets the defaults to System/38 design standard

SQL/DDS/DDL

CA 2E provides two methods for generating files:

- SQL, IBM's SAA common programming interface for database access on all SAA platforms.
- DDS, the database access method native to the IBM i platform.

The YSYSDBF system value sets the default database access method. Model value YDBFGEN defaults to the system value. Parameter DBFGEN lets you override the value for YDBFGEN to *DDS or *SQL for a particular model.

More Information

For more information about SQL/DDL and DDS, see the following IBM manuals:

- *SQL/400 Programming Guide*
- *SQL/400 Reference*
- *Database Guide*
- *DDS Reference*

Implementing SQL

If you implement SQL, each model library list and associated job description must reference an SQL collection. Parameter SQLLIB creates and sets the name of the SQL collection.

If you accept the default option *DBFGEN, CA 2E looks at the value for the DBFGEN parameter and, if it is set to *SQL, creates the collection and builds a name from the model library name according to the following rules:

- If the model library name contains the characters "MDL," CA 2E drops these characters and replaces them with the "SQL." For example, if you name the model library "INVMDLR40,"CA 2E creates the name "INVSQLR40" for the SQL collection.
- If the model library name does not contain "MDL," CA 2E appends the characters "SQL" to the model name. For example, if the name of the model library is "INVNTRY," CA 2E creates the name "INVNTRYSQL" for the SQL collection.
- If the model name is seven to nine characters in length,CA 2E truncates the suffix. If the model name is ten characters or more in length,CA 2E sends an error message.

Implementing DDL

If you implement DDL, each model library list and associated job description must reference an SQL collection. Parameter SQLLIB creates and sets the name of the SQL collection.

If you accept the default option *DBFGEN, CA 2E looks at the value for the DBFGEN parameter and, if it is set to *DDL, creates the collection and builds a name from the model library name according to the following rules:

- If the model library name contains the characters "MDL," CA 2E drops these characters and replaces them with the "SQL." For example, if you name the model library "INVMDLR40,"CA 2E creates the name "INVSQLR40" for the SQL collection.
- If the model library name does not contain "MDL," CA 2E appends the characters "SQL" to the model name. For example, if the name of the model library is "INVNTRY," CA 2E creates the name "INVNTRYSQL" for the SQL collection.
- If the model name is seven to nine characters in length,CA 2E truncates the suffix. If the model name is ten characters or more in length,CA 2E sends an error message.

Note: Irrespective of the value of the YSQLFMT model value and if the generation mode is *DDL, the RCDFMT keyword is generated.

Limitations:

- The current implementation of the DDL generation mode is not valid for the following cases:
  - Access paths that have virtual fields
  - SPN access path
  - QRY access path
  - Multi-member files

  Workaround for Virtual Fields, SPN, and QRY Access Paths: If the earlier generation mode is *DDS, revert to it and regenerate the access path. You need not regenerate the functions that use this access path. If you want to have an SQL type database, regenerate the access path using *SQL generation mode. The functions using this access path must be regenerated.

  Workaround for Multi-Member Files: If you want to have more than one member for the access paths, revert to *DDS generation mode.

Note: If you want to change an access path, which is previously defined as *DDS with a MAXMBR compiler override, to *DDL, you must revert to *DDS generation mode and must remove the compiler override, and then change back to *DDL generation mode.

Implementing SQL/DDL and DDS in the Same Model

CA 2E lets you implement both SQL/DDL and DDS within a design model. For example, if you normally use DDS but want to create a model that uses both DDS and SQL, you would create the model as follows:

1. Set the SQLLIB parameter value of the YCRTMDLLIB command to *GEN. CA 2E generates the SQL collection library and builds a name according to the rules described in the previous section.
2. Ensure that the value for the parameter DBFGEN is set to *DDS.
3. Set other model parameters as needed and create the model.

The model library list contains the SQL collection library, listed below the generation library. If you want to use SQL to implement certain CA 2E access paths in the model:

1. Set the access path details to SQL.
2. Edit the model library list so that the SQL library is listed above the generation library.
3. Update the related job description.

Note: When you use both DDS and SQL in the same model, CA 2E imposes restrictions.

For more information about using SQL/DDL and DDS, see the chapter "Setting Default Options for Your Access Paths" in the *Building Access Paths guide*.

More Information

For more information about using SQL and DDS, see the chapter "Setting Default Options for Your Access Paths" in *Building Access Paths*.

Naming Prefixes

Unless you specify otherwise,CA 2E automatically allocates names for all field and object names you define.CA 2E uses naming prefixes for the names of application objects, values list objects, and message IDs. These prefixes are derived from model values. Parameters in the YCRTMDLLIB command let you set these model values explicitly.

Application Objects

CA 2E requires that an application object prefix be added to the beginning of all member names that contain source generated by CA 2E.

Parameter OBJPFX sets the model value for the application object prefix. CA 2E uses this parameter only if autonaming is used. The default prefix is “UU.” If you define another prefix:

- Use characters that identify the user system, such as “IM” for inventory management system.
- Do not use the following characters. CA 2E reserves them for specific types of application objects:
  - Q (IBM objects)
  - (IBM S/36 environment objects)
  - Y (CA 2E and CA Xtras objects)

Values List Objects

The condition value file, accessed at program execution, stores status field values. The values list object prefix defines the first two characters of the names in the condition value file and the value selection program, called when the user enters a question mark (?) or uses F4 to prompt a status field.

The parameter VLSPFX sets the prefix for the objects in the values list file.

Message IDs

The message identifier prefix defines the first three characters of message names generated by CA 2E .

Parameter MSGPFX sets the value of the message prefix. You can set this value to *USR or *NONE:

- If you accept the default value, *USR, CA 2E prefixes each message name with the characters "USR."
- If you set the value at *NONE, you must manually prefix message IDs.

Message File Name

The message file contains message descriptions that generates. The message file name indicates where CA 2E stores messages for execution time access.

Parameter MSGVNM sets the name for the message file. This name is based on the value of the MSGPFX parameter:

- If you specify the value *USR or *NONE for the message prefix,CA 2E assigns the name QUSRMSG to the message file.
- If you specify another value for the message prefix, creates the message file name using the value of the object prefix and the message prefix, and the characters "MSG," as follows:

  object prefix + *message prefix* + MSG

  For example:

  IM + CTL + MSG = IMCTLMSG

Default Target High Level Language

The YSYSHLL system parameter sets the default for the high level language (HLL) in which to create program source. The HLLGEN parameter sets YHLLGEN, the model value for the HLL in which to create program source for the model you are creating. When you create a model,CA 2E sets the default for HLLGEN to the YSYSHLL value. You can explicitly set this parameter to generate source code for this model in RPG or COBOL. You can override the default for any specific function.

Note: CA 2E supplies RPG and COBOL separately. You will be able to generate only the languages for which your site is licensed.

High Level Language Naming Convention

The HLL you use for the model determines the naming restrictions that apply to names generated by CA 2E.

Parameter HLLVNM sets the YHLLVNM model value, which dictates naming conventions for names generated by CA 2E. You can accept the default value of *RPGCBL or you can set this value for the model.

Advanced National Language Support

Advanced National Language Support (NLS) lets you implement panel literals using external message IDs. These messages are stored in an i OS message file. By externalizing messages, you can maintain several language versions of the same application.

You can implement Advanced National Language Support at the model, function, or field level. The field level overrides the function level. The function level overrides the model level.

The device prompt generation option, PMTGEN, controls the model value YPMTGEN, which sets the externalized panel constants feature. If you implement Advanced National Language Support at the model level, you need to set this parameter to *MSGID (implement using external message IDs). The default for this parameter is *OFF (do not implement using external message IDs).

If you want to partially implement Advanced National Language Support, you can set the default to *LITERAL. You can then choose to externalize some model text at the function or field level.

More Information

For more information about implementing Advanced National Language Support at the function and field levels, see the "National Language Support" chapter in the *Generating and Implementing Applications* guide.

Signing on with the Correct User Profile

The user profile you sign on with to create a model must be authorized to use CA 2E and the following i OS commands:

- **CRTLIB**—Create a library
- **CRTPF**—Create a physical file
- **CRTLIB**—Create library
- **CRTPF**—Create physical file
- **CRTDTAARA**—Create data area
- **CRTSRCPF**—Create source file
- **CRTJOBD**—Create job description
- **CRTDUPOBJ**—Create duplicate object
- **CRTJRN**—Create journal
- **CRTJRNRCV**—Create journal receiver
- **CRTDTADCT**—Create data dictionary

The user profile used to create the model owns the model. This user can transfer ownership of the model to another user profile and can grant another user profile the authority to use the model.

More Information

For more information about model ownership, see the "Using Your Development Environment" chapter in this guide.

Changing Your Library List

The library list of the profile you are using to create the model must include the CA 2E and Toolkit libraries. You can add them to your interactive library list with the i OS Add Library List Entry (ADDLIBLE) command.

More Information

For more information about the ADDLIBLE command, see Volume 2 of the *i OS CL Reference*.

Creating a Model

You create a model library by entering the YCRTMDLLIB command and its required parameter, MDLLIB, followed by any optional parameters for model values you want to set explicitly. If you do not set a model value explicitly, the system uses the default parameter value to create the model library.

For example, to create a model (INVMDL) for an inventory control system that uses SQL to define CA 2E access paths and explicitly defines the descriptive text for the model, the object prefix, and the values list prefix:

1. Enter the command with the appropriate parameters:

```
YCRTMDLLIB  MDLLIB(INVMDL) +
   GENLIB(INVGEN) SQLLIB(INVSQL) +
   SYSTEXT(inventory-control-system) +
   OBJPFX(IN)  VLSPFX(IN)  DBFGEN(*SQL)
```

   This example includes the following optional parameters:

   - **SQLLIB(INVSQL)**—Sets the name of the library into which the SQL collection for database implementation is to be placed.
   - **SYSTEXT('Inventory Control System')**—Sets the model text value (YMDLTXT).
   - **OBJPFX(IN)**—Sets the application object prefix for all member names that contain source generated by CA 2E from the model.
   - **VLSPFX(IN)**—Sets the values list object prefix added to names for all objects used to implement the CA 2E values list facility.
   - **DBFGEN(*SQL)**—Sets the default method for implementing CA 2E access paths as database objects to SQL.

2. When you complete the parameters, execute the command by pressing Enter.CA 2E builds the model library using the parameters you specified. When the command finishes, the panel displays the following message:

   Model library (INVMDL) created

Creating the Model in Batch Mode

Creating a model generally requires at least 20 minutes. You may want to execute the YCRTMDLLIB command in batch mode using the i OS SBMJOB command. Creating the model in batch mode causes minimum impact on interactive jobs running on the system. When you submit the job, you can specify that you want the job log to provide you with a complete report when the job finishes.

For example, to create a model in batch mode and specify that messages and loggable CLP commands be logged in the job log, you would enter the following:

```
SBMJOB  CMD(YCRTMDLLIB MDLLIB(MYMDL)) + LOG(4  00  *SECLVL)  LOGCLPGM(*YES)
```

This example includes the following optional parameters:

- **LOG(4 00 *SECLVL)**—Specifies the message logging values that determine the number and type of messages logged in the job log.
- **LOGCLPGM(*YES)**—Specifies that loggable CLP commands are to be logged to the job log.

You can also prompt this command by entering the command and pressing F4. For example:

```
SBMJOB  CMD(YCRTMDLLIB  MDLLIB(MYMDL))
```

More Information

For more information about the SBMJOB command, see Volume 5 of the *i OS CL Reference*.

Prompting YCRTMDLLIB

You can also create a model by letting the system prompt you for the model values you want to use.

1. Enter YCRTMDLLIB and press F4. A prompt asks you to enter a name for the model.
2. Enter the name for the model and press Enter. The Create Model Library (YCRTMDLLIB) panel displays the parameters with their default settings.

```
                    Create Model Library (YCRTMDLLIB)

Type choices, press Enter.

Library for data model            MYMDL_____    Name, *NONE
Library for generation            *GEN______    Name, *NONE, *GEN
Library for SQL collection        *DBFGEN___    Name, *DBFGEN, *NONE, *GEN
System text or *SYSTEXT           *MDLLIB___
Design standard                   *SYS______    *SYS, *CUAENTRY, *CUATEXT...
Prefix for app. objects           UU________    Name
Prefix for value list objects     Y2________    Name, *OBJPFX
Prefix for new message id's       USR_______    Name, *NONE
Workstation implementation        *NPT______    *NPT, *GUI, *JVA, *VB
Application folder for GUI        *NONE_____    Character value, *MDL, *NONE
Library partitioning for GUI      *AUTO_____    Character value, *AUTO
Device prompt implementation      *SYS______    *SYS, *LITERAL, *MSGID, *OFF
Database implementation           *SYS______    *SYS, *DDS, *SQL, *DDL
Default target HLL language       *SYSHLL___    *SYSHLL, *RPG, *CBL, *RPGIV
HLL naming convention             *RPGCBL___    *RPGCBL, *RPG, *CBL
Binding Directory                 YBNDDIR___    Name, *NONE
                                                                  MORE...

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
```

1. Change the displayed parameters as necessary or accept the defaults. Use these function keys to display:
   - Additional parameters (F10).
   - All parameters (F9).
2. Press Enter. The system builds the model library. When the command finishes, the panel displays the following message:

Model library (*model-library-name*) created

Managing CA 2E Models

CA 2E commands help you manage the models in your work environment. This section describes how to use CA 2E commands to:

- Reorganize a model
- Clear a model
- Rename a model

For some model management tasks, you use i OS commands. This section describes how to use i OS commands to save (SAVLIB) and delete (DLTLIB) a library.

More Information

For more information about these CA 2E commands, see the CA 2E *Command Reference Guide*.

Executing the Commands in Batch Mode

You can execute commands in batch mode. Submitting the job to batch causes minimum impact on interactive jobs running on the system. When you submit the job, you can specify that you want the job log to provide you with a complete report when the job finishes.

For example, to delete a model in batch mode and specify that messages and loggable CLP commands be logged in the job log, you would enter the following:

```
SBMJOB  CMD(DLTLIB  LIB(MYMDL))  LOG(4 00 + *SECLVL)  LOGCLPGM(*YES)
```

This example includes the following optional parameters:

- **LOG(4 00 *SECLVL)**—Specifies the message logging values that determine the number and type of messages logged in the job log.
- **LOGCLPGM(*YES)**—Specifies that loggable CLP commands are to be logged to the job log.

You can also prompt this command by entering the command and pressing F4. For example:

```
SBMJOB  CMD(DLTLIB  LIB(MYMDL))
```

More Information

For more information about the SBMJOB command, see Volume 5 of the *i OS CL Reference*.

Reorganizing an CA 2E Model

The Reorganize Model (YRGZMDL) command is a housekeeping function that recovers disk space used by deleted records and may help improve performance. The YRGZMDL command executes the i OS Reorganize Physical File Member (RGZPFM) command.

Note: You cannot use a model while it is being reorganized.

To reorganize a model, enter the following and press Enter.

```
YRGZMDL  MDLLIB(model-name)
```

CA 2E sends messages as the command rebuilds access paths. When the model is reorganized, CA 2E sends the following message:

```
Model library (model-library-name) reorganized.
```

More Information

For more information about the RGZPFM command, see Volume 4 of the *i OS CL Reference*.

Checking an CA 2E Model

The Check Model (YCHKMDL) command is a housekeeping function that checks your model for inactive internal records and un-resolvable model object references.

In CA 2E, whenever you create an object in the model, a corresponding model object reference called a surrogate number is created that the product uses to recognize the object. When the object is deleted, the surrogate number should also be deleted. At times the surrogate number may not be deleted. When this happens the condition which occurs is known as a dangling surrogate. This condition can be caused by such events as a power outage which results in an incomplete update; a developer makes an error when using YWRKF to change model file data; a session is closed without exiting the model properly by a developer or by network security; a model relation name is typed over or field exit is used rather than deleting the relation.

Work can be performed in a model for months without this condition being detected. It most likely manifests itself when generating, coping or editing an object that references the deleted object surrogate number.

Consequently it is very important to use the YCHKMDL *Update action periodically to maintain a healthy model. The YCHKMDL *UPDATE cleans up dangling references and produces a report of objects it was able to delete. For large models it is recommended to incorporate it with weekly backup procedures. Using YCHKMDL *UPDATE once a month on smaller models will be sufficient. For details about the YCHKMDL command, see the *Command Reference Guide*.

Clearing an CA 2E Model

The Clear Model (YCLRMDL**)** command drops all user-defined data in the specified model library. You can also clear the generation library and, if it exists, the SQL collection of user objects.

The YCLRMDL command is often used during training to allow new users to create and use a model and then clear it to use for production.

To clear a model library and its generation library, enter the following and press Enter:

```
YCLRMDL  MDLLIB(model-library-name) + GENLIB(generation-library-name)
```

As CA 2E clears the model, messages display. When the process is complete, CA 2E sends the following message:

```
Model library (model-library-name) for (model) has been cleared.
```

Prompting the YCLRMDL Command

You can also clear a model by allowing the system to prompt you for the libraries you want to clear.

1. Enter YCLRMDL and press F4. The Clear Model (YCLRMDL) panel displays with the selection prompts.
2. Enter the names for the libraries you want to clear and press Enter. As CA 2E clears the model, messages display. When the process is complete, the system sends the message:

Model library (*model-library-name*) for (*model*) has been cleared.

Saving an CA 2E Model Library

The i OS Save Library (SAVLIB) command saves a copy of up to fifty specified libraries or all libraries. This command saves the entire library, including the library description, the object descriptions, and the contents of the objects in the library. To protect changes to your models, use SAVLIB regularly as part of your batch job process.

Note: Always save libraries before upgrading to a new version.

You can save to an off-line storage medium, such as a cartridge, or to an online save file using DEV(*SAVF) on the SAVLIB command.

Considerations

To save libraries, you must have either the *SAVSYS special authority in your user profile or you must have read authority or ownership of each library specified and object existence authority for each object in the library. If you do not have this authority, the command saves only those libraries and objects for which you do have authority.

Using the SAVLIB Command

To save a model library and its generation library to a tape in a specified device, enter the following and press Enter:

```
SAVLIB  LIB(model-library-name/generation- + library-name)  DEV(device).
```

For example, to save a model library named INVMDL and its generation library INVGEN to tape device TAP01, you would enter the following and press Enter:

```
SAVLIB  LIB(INVMDL  INVGEN)  DEV(TAP01)
```

You can also save libraries by allowing the system to prompt you for the libraries you want to save and the parameters you want to specify. Enter the command and press F4.

More Information

For more information about the SAVLIB command, see Volume 5 of the *i OS CL Reference*.

Deleting an CA 2E Model Library

The i OS Delete Library (DLTLIB) command deletes all objects in the specified library and then deletes the library.

Before you use the DLTLIB command to delete either a generation or an SQL library, you first need to ensure the library does not have journals or journal receivers.

This section describes how to check for and delete journals and journal receivers and provides examples of how you might delete model, generation, and SQL libraries.

Considerations

To delete a library, you must have use and object existence authority for the library and object existence authority for all objects in the library:

- If you do not have object existence authority for the library, the command does not delete the library or its objects.
- If you do not have object existence authority for one or more objects in the library, the command does not delete those objects or the library.

You cannot delete a library while it is on the library list of any job active on the system.

Deleting a Model Library

To delete a CA 2E model library, enter the following and press Enter.

```
DLTLIB  LIB(model-library-name)
```

Deleting a Generation Library

You can use the DLTLIB command to delete the generation library associated with a model library. Deleting a generation library requires that you first delete any journal receivers and journals within the generation library before you delete the generation library.

Deleting Journal Receivers and Journals

If the generation library contains a journal, the DLTLIB command will not complete. Before you delete a generation library, make sure it does not contain journals. You delete journals by first deleting the journals and then deleting the journal receivers.

Checking for Journals

To find out if the generation library contains journals, perform the following steps:

1. To access the Programming Development Manager (PDM), enter STRPDM and press Enter. The IBM i Programming Development Manager (PDM) panel displays.
2. Select the Work with objects option. The Specify Objects to Work With panel displays.
3. Specify that you want to work with journals in the generation library you are deleting:
    a. a. In the Library field, enter the name of the generation library you are deleting.
    b. b. In the Type field, enter *JRN and press Enter. The Work With Objects Using PDM panel displays.
    c. c. Record the name of the journal in your generation library.
4. Return to a command line. Press F3 until a command line displays.

End Journaling

You must end journaling for access paths and physical files before you delete the journal receivers and journals. Perform these steps:

1. End journaling for access paths. Enter the following and press Enter.

ENDJRNAP FILE(*ALL) + JRN(*GEN-library-name*/(*your-journal-name*)

1. End journaling for physical files. Enter the following and press Enter.

ENDJRNPF FILE(*ALL) + JRN(*GEN-library-name*/(*your-journal-name*)

1. When you have ended journaling for access paths and physical files, return to the Work with Objects Using PDM panel.
2. Mark the journals for deletion. Enter 4 to the left of all of the journals and press Enter**.** The system asks you to confirm the deletions. Press Enter to confirm.
3. When you finish deleting the journals, return to the Specify Objects to Work With panel. Press F3. You are ready to delete the journal receivers.

Delete the Journal Receivers

To delete journal receivers, perform the following steps:

1. Specify that you want to work with journal receivers in the generation library you are deleting:
   a. In the Library field, enter the name of the generation library you are deleting.
   b. In the Type field, enter *JRNRCV and press Enter. The Work with Objects Using PDM panel displays.
2. Make sure the generation library does not contain journal receivers:
   - If no journal receivers are listed, delete the generation library. Continue with the Deleting a Generation Library section.
   - If journal receivers are listed, continue with the next step.
3. Delete the journal receivers. Enter 4 to the left of all receivers and press Enter. The system asks you to confirm the deletions. Press Enter to confirm.
4. If a break message displays, indicating the journal receiver has not been saved, enter I (Ignore) and press Enter.
5. When the process finishes, delete the generation library.

End Journaling

You must end journaling for access paths and physical files before you delete the journal receivers and journals.

1. End journaling for access paths. Enter the following and press Enter.

ENDJRNAP FILE(*ALL) + JRN(*SQL-library-name/your-journal-name*)

1. End journaling for physical files. Enter the following and press Enter.

ENDJRNPF FILE(*ALL) + JRN(*SQL-library-name/your-journal-name*)

1. When you have ended journaling for access paths and physical files, return to the Work with Objects Using PDM panel.
2. Delete the journals. Enter 4 to the left of all journals and press Enter. The system asks you to confirm the deletions. Press Enter to confirm.
3. When you finish deleting the journals, return to the Specify Objects to Work With screen. Press F3. You are ready to delete the journal receivers.

Delete Journal Receivers

To delete journal receivers, perform the following steps:

1. Specify that you want to work with journal receivers in the SQL collection you are deleting:
   a. In the Library field, enter the name of the SQL collection you are deleting.
   b. In the Type field, enter *JRNRCV and press Enter. The Work with Objects Using PDM panel displays.
2. Check to make sure the library does not contain journal receivers.
   - If no journal receivers are listed, you can delete the SQL collection. Continue with the next section, Deleting the SQL Collection.
   - If journal receivers are listed, continue with the next step.
3. Delete the journal receivers. Enter 4 to the left of all receivers and press Enter. The system asks you to confirm the deletions. Press Enter to confirm.
4. If a break message displays, indicating the journal receiver has not been saved, enter I (Ignore) and press Enter.
5. When the process finishes, delete the SQL collection.

Deleting the Generation Library

To delete the generation library, enter the following and press Enter.

```
DLTLIB  LIB(generation-library-name)
```

Deleting an SQL Collection

You can use the DLTLIB command to delete an SQL collection associated with the model library. Deleting an SQL collection requires that you delete any journal receivers and journals, including access paths and physical files, within the SQL collection before you delete the collection.

Deleting Journal Receivers and Journals from the SQL Collection

If the SQL collection contains a journal, the DLTLIB command will not complete. Before you delete an SQL collection, make sure it does not contain a journal.

Checking for Journals

To check for journals in an SQL collection, perform the following steps:

1. To access the Programming Development Manager (PDM), enter STRPDM and press Enter. The IBM i Programming Development Manager (PDM) panel displays.
2. Select the Work with objects option. The Specify Objects to Work With panel displays.
3. Specify that you want to work with journals in the SQL library you are deleting:
   a. In the Library field, enter the name of the SQL collection you are deleting.
   b. In the Type field, enter *JRN and press Enter. The Work With Objects Using PDM panel displays.
   c. Record the name of the journal in your SQL collection.
4. Return to a command line. Press F3 until a command line displays.

Deleting the SQL Collection

To delete the SQL collection, enter the following and press Enter.

```
DLTLIB  LIB(SQL-library-name)
```

More Information

For more information about the DLTLIB command, see Volume 3 of the *i OS CL Reference*.

Renaming an CA 2E Model

The Rename Model (YRNMMDL) command lets you change the model name. This command also updates the generation library and any library list or model values that use the model or generation library name. This command does not rename an SQL collection or change the YMHPLBA data area (location of Help text).

You can use the YRNMMDL command to rename a model as it progresses through its life cycle.

Use the YRNMMDL command to rename a model rather than the i OS Rename Library (RNMLIB) command. If you use RNMLIB, the internal model values will be inaccurate.

Note: You cannot use a library while it is being renamed.

For example, to rename a model and its generation library, enter the following and press Enter.

```
YRNMMDL  MDLLIB(current-model-library-name) +
   NEWMDLLIB(new-model-library-name) +
   GENLIB(current-generation-library-name) +
   NEWGENLIB(new-generation-library-name)
```

CA 2E sends messages as the command updates the library lists. When the model has been renamed,CA 2E sends the following message:

```
Model (model-library-name) renamed to (new-model-library-name).
```

You can also prompt this command by entering the command and pressing F4

Chapter 3: Using Your Model

This chapter describes the CA 2E Main Menu; how to access your model; features common to CA 2E programs, such as narrative text, online help, and the Display Services Menu; and features invoked with line selection values.

This section contains the following topics:

Menus

The CA 2E menu system uses the Toolkit menu facility. This facility lets you execute programs or CLP commands, or to display other menus and panels by selecting menu options.

For more information about the Toolkit menu facility, see the *Toolkit Concepts guide*.

More Information

For more information about the Toolkit menu facility, see the *Toolkit Concepts* guide.

Main Menu

The CA 2E Main Menu is the product menu from which you access your model, change the library list to work with another model, invoke all CA 2E commands, and access CA 2E and various related CA Xtras products.

To start CA 2E and display the Main Menu, enter the following Toolkit command and press Enter:

```
YGO  *Y2
```

Or, if you intend to work with a model:

YSTRY2 *model-name*

The Main Menu displays.

```
MAIN                  CA 2E Main Menu
Level . : 1
                                                          System:  SYNONDV1
Select one of the following:

 Design Model         1. Display Designer (*DSNR) menu
                      2. Display Programmer (*PGMR) menu
                      3. Display User (*USER) menu

                      8. Work with Model Object Lists
                      9. Change to work with another model

 Commands            50. CA 2E commands in alphabetical order

                     51. Commands to set up or alter a model
                     52. Commands to copy a model
                     53. Commands to create an application
                     54. Commands to document a model
                                                                    More...
Selection or command
===>

F3=Exit  F6=Messages  F8=Rev retrieve  F9=Retrieve  F10=Cmd Entry  F14=Sbm jobs

Maximum capability to access model MYMDL is *DSLK.
```

The first panel of the Main Menu provides a set of Design Model options and access to commands grouped according to function.

Scroll to view the second panel of the Main Menu for access to more commands and various CA Xtras products.

```
MAIN                    CA 2E Main Menu
Level . : 1
                                                          System:  SYNONDV1
Select one of the following:
                      55. Commands to work with Model Objects
                      56. Commands to work with Model Object Lists
                      57. Change Control Facilities commands

Other Products        60. Change Management (CM)
                      61. 400 Toolkit
                      62. Model Investigator (MI)
                      63. Gateway
                      64. Performance Expert (PE)
                      65. Translator

                      90. Signoff
                      99. Endpasthr
                                                                      Bottom
Selection or command
===>

F3=Exit  F6=Messages  F8=Rev retrieve  F9=Retrieve  F10=Cmd Entry  F14=Sbm jobs

Maximum capability to access model MYMDL is *DSLK.
```

More Information

For more information about Design Model options, see the next section, Design Model Options, in this chapter.

Design Model Options

Although the function of most of the options listed on the Main Menu is clear from the name of the option, the first set of options, Design Model, require additional explanation. These options provide the following capabilities:

- A set of submenus listing the common tasks for each of the three user types: designer, programmer, and user. Your user type determines when you can access the model and what types of modifications you can effect. These submenus are described in the following sections.
- Access to the Work with Model Lists utility.
- The ability to change to work with another model by invoking the Change Library List (YCHGLIBL) command.

More Information

For more information about:

- User types and their related authorities, see Controlling User Access in the chapter "Using Your Development Environment."
- Model object lists and the Work with Model Lists utility, see the chapter "Managing Model Objects" in the *Generating and Implementing Applications* guide.
- Library lists, see Managing Model Lists in the chapter "Using Your Development Environment."

User-Type Submenus

Following are descriptions of the three submenus listing the tasks available for each of the three user types: designer, programmer, and user. Your user type determines when you can access the model and what types of modifications you can effect.

Designer (*DSNR) Menu

The CA 2E Designer (*DSNR) Menu provides a list of tasks available to users with *DSNR authority. Use either of the following methods to access this menu.

- Select the Display Designer (*DSNR) menu option from the CA 2E Main Menu.
- Enter the following YSTRY2 command at a command line:

YSTRY2 *model-library-list name* MENU(DSNR)

The first panel of the Designer (*DSNR) Menu displays.

```
DSNR                CA 2E Designer (*DSNR) Menu
Level . : 1                                                System:  SYNONDV1

Select one of the following:

 Enter Model          1. Edit Database Relations
                      2. Services Menu
                      3. Edit Default Model Object List
                      4. Edit Session List (changed objects)
                      5. Work with Model Objects
                      6. Load model and display command line

                      8. Work with Model Object Lists
                      9. Change to work with another model

 Open Access:       ? 10. Change Open Access Model Value
  enter with *NO      11. Edit Database Relations
                      12. Services Menu                                More...

Selection or command
===>

F3=Exit  F6=Messages  F8=Rev retrieve  F9=Retrieve  F10=Cmd Entry  F14=Sbm jobs

Maximum capability to access model MYMDL is *DSLK.
```

Scroll down to display the second panel of the Designer (*DSNR) Menu.

```
DSNR                CA 2E Designer (*DSNR) Menu
 Level . :   2
                                                              System:    MARVINS
 Select one of the following:
                      13. Edit Default Model Object List
                      14. Work with Model Objects
                      15. Load model and display command line

                      19. Synchronise model

  Model Profile       20. Edit Model Profile for Model
                      21. Edit Default Model Profile for Model

  Authority           30. Edit Model Authority (access to model)

  Library list      ? 40. Change name of library list for model
                      41. Edit library list for model

                                                                        More..
 Selection or command
 ===>

 F3=Major menu  F6=Msg  F8=Rev retrieve  F9=Retrieve  F10=Cmd entry  F24=More
 Maximum capability to access model HTSD71MDL is *DSLK.
```

Scroll down once again to see the third panel.

```
DSNR                CA 2E Designer (*DSNR) Menu
 Level . :   2
                                                              System:    MARVINS
 Select one of the following:
  Auto naming
  YEDTNXTMNC        ? 50. Edit Next Mnemonic

                                                                        Bottom
 Selection or command
 ===>

 F3=Major menu  F6=Msg  F8=Rev retrieve  F9=Retrieve  F10=Cmd entry  F24=More
 Maximum capability to access model HTSD71MDL is *DSLK.
```

The *DSNR tasks shown on this menu are grouped functionally as follows:

Enter Model Options

The Enter Model options let you choose which of several panels to display first when you enter your model. The following panels and options are described in this chapter.

- **Edit Database Relations**—This option is the same as specifying *EDTDBREL on the YEDTMDL command.
- **(Display) Services Menu**—This option is the same as specifying *SERVICES as the entry point on the YEDTMDL command.
- **Load model and display command line**—This option loads the model and displays a command line. Use it to execute a series of commands without needing to reload the model for each command. It is the same as specifying *NONE as the entry point on the YEDTMDL command.
- **Change (the library list) to work with another model**—This option invokes the Change Library List (YCHGLIBL) command.

The options that display panels that let you work with model objects and model object lists are described in the *Generating and Implementing Applications* guide.

More Information

For more information about:

- The YEDTMDL command, see the Accessing Your CA 2E Model section in this chapter.
- The YCHGLIBL command, see the Managing the Model Library Lists section in the "Using Your Development Environment" chapter in this guide.
- Model object list panels, see the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide.

Open Access: Enter with *NO Options

These options give you exclusive access to the model; in other words, they let you temporarily override the Open Access (YOPNACC) model value. If other designers or programmers are working in the model when you try to use these options, you will be denied access. Most of the options are the same as the Enter Model options and let you choose which of several panels to display first when you enter the model.

In addition, there are two options for tasks that can only be performed by a designer (*DSNR) with exclusive access to the model.

- **Change Open Access Model Value**—The YOPNACC model value controls whether programmers and designers can use the model at the same time.
- **Synchronize Model**—Synchronizing the model ensures that any changes a designer makes to the relations in a model are reflected in all CA 2E file entries of the model.

More Information

For more information about:

- Open access and the YOPNACC model value, see the Creating a CA 2E Design Model section in the "Creating and Managing Your Model" chapter in this guide.
- Synchronizing your model, see the File Locks section in this chapter.

Model Profile Options

Each user in a model has an associated model profile that defines defaults for various processes and file specifications for an interactive session. The default model profile is shipped with CA 2E and is usually the source for creating model profiles for individual users. The Model Profile options let you modify both the default and user model profiles in order to tailor the development environment. However, note that only a designer can change the default model profile.

Authority Option

This option lets you edit authority to i OS objects that CA 2E uses to control user access to the model.

Library List Options

The Library List options include:

- **Change name of library list for model**—This option lets you reset the name of the model's library list stored in the YLIBLST model value.
- **Edit library list for model**—This command invokes the Toolkit Edit Library List **(**YEDTLIBLST**)** command and displays the Edit Library List Entries panel where you can edit the current library list.

Programmer (*PGMR) Menu

The CA 2E Programmer (*PGMR) Menu provides a list of tasks available to users with *PGMR authority. Use either of the following methods to access this menu.

- Select the Display Programmer (*PGMR) menu option from the CA 2E Main Menu.
- Enter the following YSTRY2 command at a command line:

YSTRY2 *model-library-list name* MENU(PGMR)

This menu contains a subset of the options available to designers (*DSNR). The Enter Model options let you choose which of several panels to display first when you enter your model. However, note that a programmer cannot create or maintain files, fields, relations, or entries. As a result, the Edit Database Relations panel will be view only.

This menu also lets you change the current library list to work with another model and to edit your own model profile.

User (*USER) Menu

The CA 2E User (*USER) Menu provides a list of tasks available to users with *USER authority. Use either of the following methods to access this menu.

- Select the Display User (*USER) menu option from the CA 2E Main Menu.
- Enter the following YSTRY2 command at a command line:

YSTRY2 *model-library-list name* MENU(USER)

This menu provides a restricted set of the options available to a programmer (*PGMR). Note that all options on this menu provide view-only access to a model.

Accessing Your CA 2E Model

In addition to using the Toolkit menu facility described in the previous section, you can access a model with the following CA 2E commands:

- The Start CA 2E (YSTRY2) command lets you set the library list you want to work with and to display a CA 2E menu. The default is the CA 2E Main Menu.
- The Edit Model (YEDTMDL) command lets you access the model. You can invoke this command from the CA 2E Main Menu.

The remainder of this section describes these two commands in detail and explains how to set and edit your model library list.

Access Your Model Using the YSTRY2 Command

Access your model by entering the Start CA 2E (YSTRY2) command at a command line, as is shown in the following examples:

- If you know the name of the model library list, enter the following and press Enter:

YSTRY2 (*model-library-list-name*)

For example:

YSTRY2 MYMDL

The command changes the current library list to the library list for the model and displays the CA 2E Main Menu.

- If you do not know the name of the model library list, you can select from a list. Enter the following and press Enter:

YSTRY2 *SELECT

A panel displays a list from which you can select the library list you want. When you select a list, the command changes the current library list to the list you selected and displays the CA 2E Main Menu.

- If you want to display a menu other than the CA 2E Main Menu, you can use the Menu option either to select from a list of existing menus or specify another menu. For example, to display the CA 2E Designer (*DSNR) Menu, enter the following:

YSTRY2 MYMDL MENU(DSNR)

To select from a list of menus enter:

YSTRY2 MYMDL MENU(*SELECT)

Accessing Your Model Using the YEDTMDL Command

Access your model by entering the Edit Model (YEDTMDL) command at a command line. The following describes the options available on this command:

- **User Type**—The user designation determines when you can access the model and what types of modifications you can effect. Designer is the default user designation for YEDTMDL. To invoke the YEDTMDL command as a:
  - Designer, enter *****DSNR for the User option and press Enter.
  - Programmer, enter *****PGMR for the User option and press Enter.
  - User, accessing the model in a view-only mode, enter *****USER for the User option and press Enter.

Note: After entering a model as either *DSNR or *PGMR, do not attempt to enter another model without exiting the current model. Always exit and save your changes before changing to another model's library list.

In addition to user type, you can also specify the following when invoking the YEDTMDL command:

- **Job List**—This is the name of the job list containing the names of source members to be generated and/or compiled. *MDLPRF defaults to the job list name specified in your model profile.
- **Model entry point**—This specifies which of the following panels to display on entry to the model:
  - *EDTDBREL (Edit Database Relations, the default)
  - *EDTMDLLST (Edit Model Object List)
  - *SERVICES (Display Services Menu)
  - *NONE (This option loads the model and displays a command line. Use it to execute a series of commands without needing to reload the model for each command. In addition, this option can be used to lock a user into the model based on a user designation.)
- **Session List**—This specifies the model object list that is to be the target for model objects that are changed during the editing session. *MDLPRF defaults to the session list specified in your model profile.
- **Open Access to the Model**—This option allows a *DSNR exclusive access to the model. The option displays only if the YOPNACC model value is set to *YES and can be overridden only by a *DSNR with *LOCK authority.
- **Model Object List**—This is the name of the model object list to edit on the Edit Model Object List panel when the model entry point is *EDTMDLLST. *MDLPRF defaults to the list specified in your model profile. Specify *ALLOBJ to edit model objects rather than model object list entries.

When you select the options you want, the CA 2E logo window displays with the user type you selected displayed in a window in the center of the panel. Following the CA 2E logo, the panel corresponding to the entry point you selected displays. You are ready to edit the model.

More Information

For more information about:

- User access and control, see the "Using Your Development Environment" chapter in this guide.
- Open access, see the Open Access section in the "Creating and Managing Your Model" chapter in this guide.
- Model object lists and the model profile, see the "Managing Model Objects" chapter in the CA 2E *Generating and Implementing Applications* guide.

Setting and Editing the Library List for Your Model

The YEDTMDL command defaults to the library list of the current job. If you use the YSTRY2 command to access your model, the list is changed to the specified library list as YSTRY2 executes.

Setting the Library List with the Change Library List Command

You can change your library list with the Change Library List (YCHGLIBL) command. This command replaces the current library list with the list you select. You can invoke the YCHGLIBL command either from a command line or from the CA 2E Main Menu with the option Change to work with another model.

Editing the Library List

You can edit the current library list using the CA 2E Designer (*DSNR) Menu.

To access the *DSNR Menu:

1. Enter the following from any command line:

YSTRY2 *model-library-list-name* MENU(DSNR)

1. Select the Edit library list for model option.

YLIBLST Model Value

The Create Model Library (YCRTMDLLIB) command sets the YLIBLST model value to the library list specified by the LIBLST parameter when a model is created.

When you invoke a command that loads a model,CA 2E checks the current job's library list for the specified model library. If it cannot find the specified model library,CA 2E automatically replaces the current job's library list with the library list specified by the YLIBLST model value for the requested model. When the command completes,CA 2E resets the current library list back to the original library list.

More Information

For more information about changing and editing library lists, see the Managing the Model Library Lists section in the "Using Your Development Environment" chapter in this guide.

Edit Database Relations Panel

The Edit Database Relations panel is the center of CA 2E.This panel enables the designer to enter relation statements that define a model. Each statement uses file and field names that assert a declaration about your model or application; for example, Order Known by Order Code.

```
EDIT DATABASE RELATIONS            SYMDL
=>                               Rel Lvl:
 ? Typ Object                    Relation     Seq Typ Referenced object
    FIL Order                    Known by      10 FLD Order code
    FIL Order                    Has           20 FLD Order date
    FIL Order                    Has           30 FLD Order status
    FIL Order                    Refers to     40 FIL Customer
    FIL Order                    Refers to     50 FIL Employee
    FIL Order                    Refers to     60 FIL Product
    FIL Order Detail             Owned by      10 FIL Order
    FIL Order Detail             Known by      20 FLD Order line number
    FIL Order Detail             Has           30 FLD Order quantity
    FIL Order Detail             Has           40 FLD Line total
    FIL Order Detail             Refers to     50 FIL Product


                                                                    More...
Z(n)=Details   F=Functions   E(n)=Entries   S(n)=Select   F23=More options
F3=Exit   F5=Reload   F6=Hide/Show   F7=Fields   F9=Add/Change   F24=More keys
```

The Edit Database Relations panel has a number of edit and navigation aids that expedite model design and maintenance.

More Information

For information about a utility that serves as an alternate entry point to your model, see the Edit Model Object List Panel section in this chapter.

Edit Aids

Edit aids include:

- **Multi-line full screen entry**—You enter data from a subfile display that lets you enter many relations at one time.
- **Interactive validation**—The program carries out a number of checks to ensure that you used the correct syntax.
- **Unordered entry**—You can enter relations as they occur to you.CA 2E sorts them in alphabetical order by file and into default order of relations within the file.
- **Abbreviated entry of keywords**—You can use a single letter abbreviation for relations.
- **Duplication and reference functions**—By using selection facilities, you can duplicate values from the previous line or select from a list of existing values.

Navigation Facilities

Navigation facilities include:

- **Reordering**—The system displays the relations in alphabetical order by subject line.
- **Repositioning**—You can position the display to show relations starting at a particular name or partial name.
- **Grouping**—You can control which relations display.
- **Selection**—You can determine the allowed values for controlling the display.
- **Explosion**—The system helps you visualize what the implemented database will look like by resolving the relations for a file into entries for that file.

Subsidiary Facilities of Edit Database Relations Panel

Entering the relation statements is the top level activity of using CA 2E. The top level panel is concerned with only the names of the CA 2E objects in the model and the relations between them. From the Edit Database Relations panel, you can branch to a number of supporting functions to add the details of your design.

For example, you can:

- Add explanatory text.
- Create new CA 2E objects.
- Add details about CA 2E objects.
- Add virtual fields to relations.
- Display all entries on a file.
- Define additional CA 2E functions to operate on the objects.
- Examine the field data dictionary.
- Display existing CA 2E objects and valid code types.

Grouping Facilities

From the Edit Database Relations panel, you can also specify selection criteria to control which part of the model displays to you for editing. You can use any of the following criteria:

- Application area
- CA 2E object name or partial name
- Relation usage group
- Referenced CA 2E object type
- Referenced CA 2E object name or partial name

Branching to Other Facilities

From the Edit Database Relations panel, you can branch to all other CA 2E facilities:

- Function keys let you branch to general facilities, such as the Display Services Menu.
- Selection values let you invoke facilities that apply to a particular relation or object.

More Information

For more information about this feature, see the Using Line Selection Values section in this chapter.

Exiting Edit Database Relations Panel

When you are ready to exit from Edit Database Relations, press F3.

The Exit Edit Relations window displays with the exit options. If you updated the model, the window displays the following message:

```
** Model  is  not  synchronized **
```

Enter the number for the exit option you want and press Enter. The option you select depends on whether you need to resynchronize the model. Synchronizing a model ensures that the CA 2E file entries of the model reflect any changes to the relations in the model.

If you:

- Do not want to resynchronize the model, select the option, Exit without resynchronizing.
- Want to resynchronize the model, select the option Exit and resynchronize data model. You can select this option only if you are a *DSNR with exclusive access to the model.
- Want to return to editing, select the option Return to editing.
- Select an exit option, the Main Menu displays.

More Information

For more information about synchronizing a model, see the File Locks section in this chapter.

Edit Model Object List Panel

The Edit Model Object List panel serves as an alternate entry point into your model. It has a PDM-like interface which you can use to perform most functions available from the Edit Database Relations panel, not including editing relations and creating model objects.

You can temporarily transfer to the Edit Database Relations panel from the Edit Model Object List panel by entering YEDTMDL or Y2 on the command line. When you finish your editing, press F3 to return to the Edit Model Object List panel.

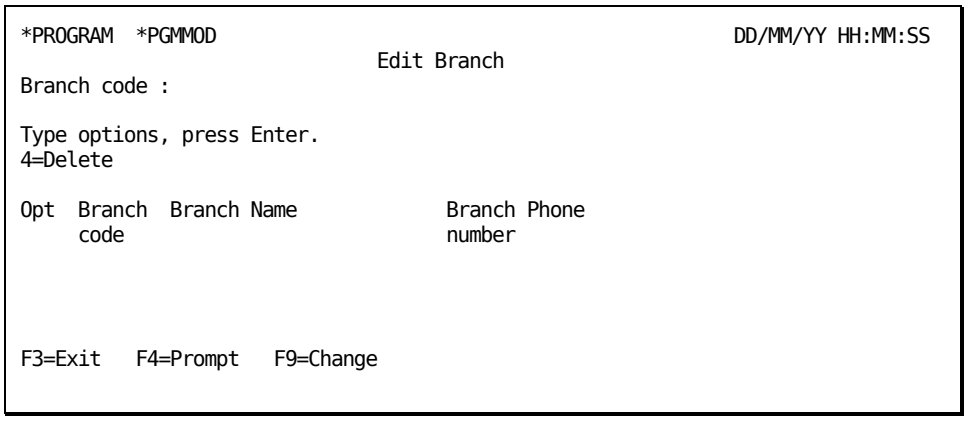
More Information

For more information about the Edit Model Object List panel, see the "Working with Model Objects" chapter in the CA 2E *Generating and Implementing Applications Guide*.

Edit Device Design Panel

The Edit Device Design panel lets you design device formats; that is, panel and report layouts, for your functions.CA 2E presents you with a full image of the panel you are designing with the correct data attributes for each field type.

creates a default device layout as a starting point. The default layout provides a placement of the fields according to function type and brings in all relevant field default values.



Edit Device Design Facilities

The Edit Device Design panel includes a number of facilities that let you design panels and reports efficiently. You can:

- Vertically align fields with other fields.
- Split lines into two parts or combine two lines into one.
- Choose from three positions for the object text: Above the field, to the left of the field, as a column heading. Moving a field moves the associated text.
- Hide fields on the panel.
- Move fields relative to one another within their format.
- Add function fields and/or constants to the panel. You can search the data dictionary for existing fields.
- Override and/or condition field display attributes.

Exiting the Edit Device Design Panel

When you use the Edit Device Design panel to alter a panel or report,CA 2E does not permanently store your changes until you exit from the editor.

1. When you are ready to exit from an Edit Device Design panel, press F3. The Edit Function Devices panel displays.
2. Press F3 to display the exit options on the Exit Function Definition panel. Leave or change the defaults.
3. When you finish, press Enter to exit the panel.

Edit Action Diagram Panel

The Edit Action Diagram panel lets you specify the processing steps for a CA 2E function. For standard functions, a default action diagram displays. By using line commands, you can insert additional actions at certain points, called user points.

```
 EDIT ACTION DIAGRAM            Edit     SYMDL       Customer
FIND=>                                                Edit Customer
I(C,I,S)F=Insert construct          I(X,0)F=Insert alternate case
I(A,E,Q,*,+,-,=,=A)F=Insert action  IMF-Insert message
    >Edit Customer
    .--
    . ...Initialize                                                    <--
    . .=REPEAT WHILE
    .  -*ALWAYS
    . | ...Load first subfile page                                     <--
    . | PGM *Reload subfile = CND.*NO
    . | > Conduct screen conversation
    . |  =REPEADT WHILE
    . | |-PGM.*Reload subfile is *NO
    . | | Display screen
    . | | ...Process response                                        <--
    . | .-ENDWHILE
    . .-ENDWHILE
    . ...Closedown                                                     <--
    .--

F3=Prev block   F5=User points   F6=Cancel pending moves   F23=More options
F7=Find         F8=Bookmark      F9=Parameters             F24=More keys
```

Edit Action Diagram Facilities

The Edit Action Diagram panel includes a number of facilities that let you manipulate an action diagram. You can:

- Add, delete, copy, or move constructs.
- Control which portions of the action diagram display.
- Insert calls to other functions. Symbols on the right side of the panel indicate the user points at which you can add your own processing.

In addition, function keys let you:

- Display a pull-down menu that lists the user points. From the menu, you can transfer to any user point.
- Display the date a user point or function was updated.
- Display the Action Diagram Services panel. This panel lets you search for any occurrence of a function, field (including where the field is used as a parameter to a function), or change date. From this panel, you can also reset change dates.

  Note: The Usage field allows for I (Input), O (Output), B (Both), and U (Updated).
- Display the Edit Device Design panel.
- Display the Edit Function Parameters panel. This panel lets you update the parameters associated with the function.

Exiting the Edit Action Diagram Panel

When you use the Edit Action Diagram panel to alter an action diagram,CA 2E does not permanently store your changes until you exit from the editor.

1. When you are ready, exit from the Edit Action Diagram panel. The Exit Function Definition panel displays with the exit options.
2. Leave or change the defaults.
3. When you finish, exit the panel. Press Enter.

Action Diagram Line Commands and Function Keys

Line Commands

The following table lists and describes the Edit Action Diagram Line Commands.

Note: Use the value in parentheses when you want the command to prompt you.

Line Command	Command and Prompt	Description
IS	(ISF)	Insert sequence
IA	(IAF)	Insert action
IC	(ICF)	Insert case condition
IO	(IOF)	Insert *OTHERWISE condition
IX	(IXF)	Insert new condition within case
II	(IIF)	Insert iteration
I*	(I*F)	Insert comment
I+	(I+F)	Insert *ADD built-in function
I-	(I-F)	Insert *SUB built-in function
I=	(I=F)	Insert *MOVE built-in function
I=A		Insert *MOVE ALL built-in function
IE	(IEF)	Insert a *EXIT PROGRAM built-in function
IQ	(IQF)	Insert *QUIT built-in function
IM	(IMF)	Insert message function
F		Edit action or condition details for line
FF		Edit action parameters line
C		Copy this construct to a point indicated by ‘A’ or ‘B.’
CC		Block copy boundary Bound by another CC
M		Move this construct to a point indicated by ‘A’ or ‘B.’
MM		Block Move boundary Bound by another MM
A		Place copied or moved construct after this line

Line Command	Command and Prompt	Description
D		Delete this construct
DD		Block Delete boundary Bound by another DD
B		Place copied or moved construct before this line
H		Hide construct
N		Edit object narrative
PR		Protect a block
R		Display function/message references
U		Display function/message usages
S		Show construct
T		Return to top level of action diagram
U		Unzoom out of construct to previous construct
V		View summary of selected block
Z		Zoom into construct
?		Prompt line commands
*		Activate/Deactivate construct
**		Block activate/deactivate construct
NR		Copy to notepad and replace existing notepad contents
NRR		Block copy Replace to notepad Bound by another NRR
NA		Copy to notepad and append to existing notepad contents
NAA		Block Copy Append to notepad Bound by another NAA
NI		Insert entire current notepad contents after this line

Function Keys

The following table lists and describes Edit Action Diagram Product Function Key settings.

Function Key	Description
F3	Return to place of previous zoom or exit
F5	Display user points
F6	Cancel pending moves
F7	Find
F8	Create a bookmark
F9	Edit parameters
F12	Unzoom, one block at a time
F13	Exit action diagram
F14	Display CA 2E Map
F15	Opens functions
F16	Display date user point or function was updated
F17	Display Action Diagram Services menu
F18	Access Notepad and exit Notepad
F19	Edit device design
F20	Display bookmarks
F21	Toggle implementation names and function types
F13	More line commands
F24	More keys

Using Application Areas

The CA 2E application area feature lets you divide your model into groups of files. You can display, edit, or operate on these groups in isolation from the rest of your model. You can create many types of application areas. For example, you can create an application area for your accounts payable files.

Note: Application areas apply only to files.CA 2E model objects lists and associated panels and commands provide a method for grouping model objects that is independent of object type. You can access this facility using the Edit Model Object List (YEDTMDLLST) command.

You define application areas using a panel accessed from the Edit Database Relations panel. Once you define an application area, you can use it to control the type of information that displays on CA 2E interactive panels or prints when you invoke documentation commands.

Each application area must have a valid name and a three-character code. The name and code must be unique within the model. For example, you can represent orders by ORD.

Each application area can contain one or more files. A file can be in more than one application area.

More Information

For more information about model object lists and the YEDTMDLLST command, see the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide.

System Application Area

CA 2E automatically defines a special system application area. This area contains all CA 2E files in the model. Model value YMDLTXT specifies the text for the system application area. You can attach narrative text to the system application area to describe the model. You cannot delete the system application area.

Application Area Codes

You can use application area codes as:

- Selection values at the top of CA 2E interactive panels.
- Selection parameters for CA 2E documentation commands.

Application Areas as Selection Values

When an application area code is a selection value displayed at the top of a panel, only CA 2E objects belonging to that area are displayed. For example:

- The Edit Database Relations panel displays the relations for only those CA 2E files that belong to that object.
- The Display All Access Paths panel displays the access paths for only those files within that application area.
- The Display All Functions panel displays the functions for only those CA 2E functions that belong to files within the application area.

  Note: All of the functions that are available from the EDIT FUNCTIONS panel are also available from this panel, to include C-Copy, Y=Y2CALL, U=Usages, R=References, and O=Open.

Application Areas as Selection Parameters

When an application area code is a selection parameter for CA 2E documentation or job list commands, only objects that belong to the specified application area are included in the result. For example:

- Document Model Relations (YDOCMDLREL)
- Document Model Files (YDOCMDLF)
- Document Model Functions (YDOCMDLFUN)
- Document Model Access Paths (YDOCMDLACP)
- Document Model Application Areas (YDOCMDLAPP)
- Build Job List (YBLDJOBLST)

Displaying/Editing Application Areas

The application area feature includes a panel, Edit Application Areas, that lets you display and update existing application areas. You can access this panel from the Edit Database Relations panel by either of the following methods:

- Enter a question mark (**?**) in the application area code field and press Enter. This field is located at the top of the panel, with an arrow to the left of the field.
- Access the Display Services Menu. Select either the Display all access paths or Display all functions options. From the panel that displays, enter **?** for the Application area option at the upper left of the panel.

The Edit Application Areas panel displays. From this panel you can:

- Delete an application area.
- Create, display the details for, and edit an application area.
- Add narrative text.

You can begin the list with a particular application area by entering the code for the application area in the Area field, identified by Position Display. When you press Enter, the display begins with the selected application area.

More Information

For more information about:

- Deleting an application area, see the Deleting an Application Area section in this chapter.
- Creating, displaying the details for, and editing an application area, see the Creating/Editing an Application Area section in this chapter.
- Adding narrative text, see the Using Narrative Text section in this chapter.

Deleting an Application Area

You can delete an application area by entering **D** in the line selection field to the left of the application area you want to delete and pressing Enter. The message ***DELETED** displays in the descriptive text field for the application area you deleted.

Creating/Editing an Application Area

From the Edit Application Areas panel, you can access the Edit Application Details panel, from which you can create an application area or edit an existing area.

Displaying Files

The Edit Application Details panel displays the CA 2E files currently in an existing application area. To display:

- Files beginning with a particular file, place the cursor in the File Name field, identified as Position display**,** and enter the name of the file with which you wish to begin. When you press Enter, the display begins with the selected file.
- All files regardless of application area, press F9. To display only the files currently in this application area, press F9 again. The message **SELECTED FILES** displays in the upper right corner of the panel.

Creating an Application Area

To create an application area:

1. From the Edit Application Areas panel, select the option to add an application area. Press F9. The Edit Application Details panel displays.
2. Enter a three-character code and a name for the application area.
3. Select the files you want to include in the area. Enter a plus sign (**+**) in the selection lines for the file names. If you are not sure you want to include a file, you can display narrative text by entering **N** in the selection line for the file and pressing Enter.
4. When you have defined the application area, accept your entries. Press Enter. The selected files are highlighted and identified with an asterisk to the left of the file.
5. Exit the panel. Press F3. The Edit Application Areas panel displays with the application area you created added to the list.

Editing an Application Area

To edit an application area:

1. From the Edit Application Areas panel, select the application area you want to edit. Enter Z in the selection line for the application area and press Enter. The Edit Application Details panel displays.
2. Edit the panel as necessary:
   - To change the code and/or name for the application area, press F8 and enter over the information you want to change.
   - To display or add narrative text for a file, enter N in the selection line for the file and press Enter.
   - To select or deselect a file, enter a plus sign (+) to select or a minus sign (-) to deselect in the selection line for the file.
3. When you finish editing, press Enter.
4. When you are ready to return to the Edit Application Areas panel, press F3 (Exit). Make sure you press Enter first to accept the changes you made.

More Information

For more information about narrative text, see the Using Narrative Text section in this chapter.

Using Line Selection Options

Certain features common to CA 2E interactive programs let you perform tasks on the objects listed on a particular panel. Some of these features—locked objects, narrative text, and model object cross references—are discussed in detail in this chapter.

You can invoke these features where a panel of items includes a selection option column to the left of the item column. A value, called a line selection value, attaches to each feature. You invoke the feature by entering the line selection value next to the item and pressing Enter.

The bottom of the panel displays a list of the line selection values available from that panel. Each possible value represents an action. The following table lists the standard line selection values.

Value	Description
A	Associated access path
	Select access path for functions
	Animate

Value	Description
C	Copy this item
D	Delete this item
E	Edit source for this item
	Entries
EO	Redirection
F	Display functions for this item Action diagram Function references
G	Generate source interactively
H	Hold or release this item
J	Generate source in batch
L	Display locks for this item Lock item
M	Mapping field parameters
N	Edit narrative text for this item
O	Prompt overrides for this item Add to open functions list
P	Display parameters for this item
Q	Resequence
R	Relations
	Redirect
	Replace
	File dependencies
S	Device designs
	send message
T	Trim
	Structure
	Reset default for F4 function
	Subset by referenced object
U	Display usages for this item
V	Edit virtual fields for this item
	Virtualize
X or 1	Select this item

Value	Description
Z	Show details for this item

Alternatively, the Edit Model Object List panel provides a set of numeric line selection options for performing actions on your model objects.

More Information

For more information about the numeric line selection options, see the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide.

Locking Objects

Locks are used to prevent two developers from updating and/or accessing the same object at the same time; they apply to both interactive and batch jobs. An object is locked while it is in use.CA 2E provides the following categories of locks:

- **Object locks**—Object locks apply to functions and access paths and can be:
  - Temporary—A developer is using or updating an access path or function.
  - Permanent—A designer is preventing all changes to an access path or function.
- **File locks**—File locks apply only to files and can be:
  - *READ—(Read Lock) A developer is using the file.
  - *EXCL—(Exclusive Lock) A designer is updating the file.
  - *SYNC—(Synchronize Lock) A designer has left the model without expanding an updated file; the existing *EXCL lock is converted to a *SYNC lock.
- **Field locks**—Field locks are functionally similar to object locks and apply only to fields. They can be:
  - *READ—(Read Lock) A developer is using the field.
  - *EXCL—(Exclusive Lock) A designer is updating the field. An *EXCL field lock can cause implicit *EXCL locks to be set on other fields and files.

The following table gives an overview of the three categories of locks. Each is discussed in more detail in this section.

Lock Category	Objects that Can Be Locked	Lock Type	Meaning of Lock	Authority Needed to Set Lock
Object Locks	Access Paths Functions	Temporary	A developer is using or updating an access path or function.	*ANY
		Permanent	A developer is using a file.	*DSNR with additional rights
File locks	Files	*READ		*ANY
		*EXCL	A designer is updating a file.	*DSNR
		*SYNC		*DSNR
Field Locks	Fields and related files and fields	*READ	A developer is using a field.	*ANY

Lock Category	Objects that Can Be Locked	Lock Type	Meaning of Lock	Authority Needed to Set Lock
		*EXCL	A designer is updating a field.	*DSNR

Object Locks

Objects locks apply to functions and access paths.CA 2E supports both temporary and permanent object locks:

- Temporary locks prevent individual objects such as CA 2E functions or access paths from being changed by more than one user at a time. An object is locked while it is in use.
- Permanent locks prevent users from changing a CA 2E object.

The locking feature is available from panels that display the line selection value assigned to this feature.

You can edit but not update locked functions. When you exit, a message offers you the option to create a new function.

Displaying Object Locks

You display the Object Lock panel by typing L in the selection line of the object whose lock you want to display and pressing Enter. The Display Object Lock panel shows:

- The name of the job holding the lock.
- The name of the user profile holding the lock.
- The number of the job holding the lock.
- The date and time the lock was placed.

Adding/Removing Object Locks

If you have the required authority level, you can add or remove a permanent object lock. You must be a designer (*DSNR) with *ALL rights to the YMDLLIBRFA data area.

You can add or remove a permanent lock from the Object Lock panel.

- If an object is unlocked and you want to lock it, press F8. The message "Permanent lock added" displays.
- If an object is locked and you want to unlock it, press F8. The message "Permanent lock removed" displays.

You may find that an object is locked, although you did not request to lock it. A file or an access path can become locked when a request to generate source (YGENSRC) terminates with the end option *IMMED. This failure is caused by an event such as a subsystem ending.

More Information

For more information about authority, see the "Using Your Development Environment" chapter in this guide.

File Locks

File locks allow designers and programmers to work in a model concurrently by preventing them from updating the same file at the same time. File locking occurs only if the Open Access (YOPNACC) model value is set to *YES.

A summary of file lock functions appears at the end of this section.

More Information

For more information about the YOPNACC model value and open access, see the "Creating and Managing Your Model" chapter in this guide.

Setting File Locks

File locks can be *READ, which allows multiple users to access the file at the same time, *EXCL, which gives exclusive use of the file to a single *DSNR for update, and *SYNC, which are converted *EXCL locks set when a *DSNR leaves the model without expanding an updated file. In general,CA 2E sets file locks automatically based on your user designation and the operation you are performing. A designer can also set *EXCL file locks explicitly.

*READ File Locks

*READ file locks let several developers use a file at one time and prevent *DSNRs from updating the file. Each developer using the file sets a separate *READ lock. A *DSNR must wait until all *READ locks are removed before setting an *EXCL lock to update the file. A *READ lock cannot be changed to another lock type.

Operations that set *READ locks include, viewing access paths for a file, loading a function, and specifying a file or access path as a parameter detail.

If a *DSNR attempts to use a file when the model is unsynchronized, before setting the requested *READ lock on the file,CA 2E expands the file and temporarily sets an *EXCL lock on all files in the expansion path. When the expansion completes, the *EXCL locks are removed, a *READ lock is set on the file, and the requested operation is performed.

Note: If CA 2E finds a lock on a file in the expansion path, the *DSNR will not be able to perform the operation until the lock is removed.

More Information

For more information about file expansion, see the Considerations for Using File Locks section in this chapter.

Implicit *EXCL File Locks

Implicit locks are set automatically when a *DSNR performs an action that updates a relation or has the intent of updating a file. Operations that set *EXCL locks include adding, changing, or deleting a relation, and adding a virtual entry.

Once a file is updated, only a *DSNR can use the file until the file is expanded or the model is synchronized. Depending on the type of update, other files can also be locked.

- Adding a relation to a file sets an *EXCL lock on the current file only.
- If you delete or change a non-key relation on a file, sets an *EXCL lock on the file and any files having entries derived from the relation; for example, virtual entries.
- If you delete or change a key relation for a file,CA 2E sets *EXCL locks on all files that use the file. For example, if you delete the first relation shown below,CA 2E sets *EXCL locks on the Order, Order detail, Order detail line, and Invoice files.

Order	Known by	Order Code
Order detail	Owned by	Order
Order detail line	Owned by	Order detail
Invoice	Refers to	Order

Order	Known by	Order Code

Explicit *EXCL File Locks

Most file locks are set implicitly. However, if you are a *DSNR you can set an explicit file lock by entering **L** against the file on the Edit Database Relations panel. The model need not be synchronized.

For example, an explicit file lock lets you add several relations to a file and ensures that you will be able to update the file when you finish. An explicit lock prevents all other developers from using the file while you are adding the relations.

*SYNC File Locks

A *SYNC file lock is a converted *EXCL file lock. It is set when a *DSNR leaves the model without expanding an updated file. A *SYNC lock is automatically converted to an *EXCL lock when any *DSNR locks the file for update.

Note: We recommend that you run the Synchronize Model (YSNCMDL) command at the end of each day to synchronize the model and clear all outstanding file locks.

File Dependencies

To display a list of the files that will be affected by a change to a relation, enter **R** against the relation you intend to update on the Edit Database Relations panel. This displays the Display File Dependencies panel.

This panel lists the following information:

- File name
- Lock type
- User profile
- Job name
- Date
- Time

You can also use this panel to send a message to a user requesting use of a file.

Displaying File Locks

Display existing file locks by pressing F22 on panels where it is available; for example, the Edit Database Relations panel. The Display File Locks panel displays.

```
DISPLAY FILE LOCKS                SYMDL
File. . . :

    File                    Lock   User      Job        Date     Time
    Branch                  EXCL   JAR       JARS2    02/02/02 13:31:04
    Customer                EXCL   JAR       JARS2    02/02/02 13:31:04



SEL: S-Send Message
F3=Exit   F5=Reload
```

Enter S against any file listed to send a message to the user that locked the file. For example, you can request that the user release the lock.

Removing File Locks

You remove *EXCL and *SYNC file locks by:

- Expanding the file. For example, enter F, Z, E, or Q against the file on the Edit Database Relations panel.
- Synchronizing the model.
- Completing the update on the field that set the lock. See the Field Locks section in this chapter for details.

You can remove an explicit *EXCL file lock you set by entering **U** against the file on the Edit Database Relations panel.

*READ file locks are removed when you exit the panel where the lock was set; for example, when you return to the Edit Database Relations panel.

Considerations for Using File Locks

Following are special topics and considerations related to file locking.

Expanding Files

Expanding a file means that CA 2E resolves all of the file's relations. If the file Refers to another file, that file's relations are resolved also.CA 2E follows the path of file-to-file relations until all files in the path are expanded. File expansion relates to file locking in the following ways:

- One way to remove an *EXCL lock is to expand the file. For example, enter F, Z, E, or Q against the file on the Edit Database Relations panel.
- If the model is unsynchronized and a *DSNR performs an operation that uses a file,CA 2E expands the file and sets an *EXCL lock on each file in the expansion path momentarily. During the expansion, other developers are prevented from accessing the locked files.

  When the expansion is complete,CA 2E removes the *EXCL locks, sets a *READ lock on the selected file, and performs the requested operation. If CA 2E finds a lock on any file during the expansion, the requested operation is denied.

Synchronizing the Model

Synchronizing the model ensures that any changes to the relations in a model are reflected in the CA 2E file entries of the model. This process removes all file locks and can only be done by a *DSNR with exclusive access to the model. A *DSNR can synchronize the model when exiting the model, select the resynchronize data model option.

Note: This option is not available if the YOPNACC model value is *YES unless you enter the model using the YEDTMDL command and override the OPNACC option to *NO. This gives the *DSNR exclusive access to the model and the option of synchronizing the model when exiting. After synchronization, YOPNACC is automatically reset to *YES.

- Access the Display Unreferenced Fields panel. (Press **F11** from the Display Fields panel.) This panel implicitly synchronizes the model.
- Use one of the following options on the Display Services menu. These options implicitly synchronize the model.
  - Submit model create request (YSBMMDLCRT)
  - Job list menu. From this menu, select one of the following options:
- YBLDJOBLST (Build job list for model)
- YCHKJOBLE (Check job list entries)
- YSBMMDLCRT (Submit model create requests)
- Use the Synchronize Model (YSNCMDL) command outside of the model.

  We recommend that you run the YSNCMDL command at the end of each day to synchronize the model and clear all outstanding file locks.

More Information

For more information about the YSNCMDL command, see the CA 2E *Command Reference Guide*.

File Locks Summary

The following table summarizes the functions related to file locks.

	***READ Lock**	***EXCL Lock**	***SYNC Lock**
Setting	As *DSNR or *PGMR, perform an action that *uses* a file; e.g., edit an action diagram or generate source.	Implicit: (As *DSNR) Change, add, or delete a relation. Access a panel with intent to update a file, e.g., Edit Virtual Field Entries panel. Update a field on the file. See the next section, Field Locks, for more information.	As *DSNR, exit the model without expanding an updated file; the existing *EXCL lock is converted to a *SYNC lock.
		Explicit: (As *DSNR) Enter **L** against file on EDR1; the model does not have to be synchronized.	
Displaying	Press F22=File Locks where available; e.g. on EDR (Edit Database Relations panel). **Note:** F22=File Locks displays only if the model value YOPNACC is set to *YES.	Press F22=File Locks where available; e.g. on EDR1.	Press **F22**=File Locks where available; e.g. on EDR1.

	*READ Lock	*EXCL Lock	*SYNC Lock
Removing	Exit panel where lock has been set; for example, return to Edit Database Relations.	Implicit: (As *DSNR) Expand file; e.g., enter F, Z, E, or Q against file. Exit and resynchronize the model. If file locked as a result of a field update, finish the update.	As *DSNR, Synchronize the model. Expand the locked file. **Note:** We recommend that you run the **YSNCMDL** command at the end of each day to synchronize the model and clear all outstanding file locks.
		Explicit: (As *DSNR) Enter **U** against file on EDR1 to unlock an explicit lock.	
Notes:	Multiple *READ locks can exist on a file. An *EXCL lock cannot be placed on a file until all removed.	Only one *EXCL lock can be set on a file at a time. Access path or function generation will fail if an *EXCL lock is set on the based-on file.	A *SYNC lock is automatically converted to an *EXCL lock when any *DSNR locks the file for update.

Field Locks

Field locks apply to fields and are only set implicitly. A field lock can be exclusive (*EXCL) or read (*READ). An *EXCL field lock can cause CA 2E to lock files and other fields. Field locking occurs only if the YOPNACC model value is set to *YES.

A summary of field lock functions appears in the table at the end of this section.

Setting Field Locks

If a *DSNR accesses the Edit Field Details panel,CA 2E sets an *EXCL lock on both the field and its referencing file, if any. If the field is referenced by other fields, all reference (REF) fields in the domain are also locked, including the REF field's referencing files. These prevent other developer's from accessing an incorrect field definition while the field is being updated.

If a *PGMR accesses the Edit Field Details panel,CA 2E sets a *READ lock on the field, but does not lock files. This prevents *DSNRs from updating the field while it is used by a *PGMR.

Displaying Field Locks

You display field locks by entering L against a field on most panels that access the Edit Field Details panel. This displays the Display Object Locks panel which lists current locks and the name of the job and the user profile that placed each lock. This is useful if you cannot access the Edit File Details panel because a lock is set by another job.

Removing Field Locks

Field locks, and all associated locks, are automatically removed when you exit the Edit Field Details panel.

Field Locks Summary

The following table summarizes functions related to field locks.

	***READ Lock**	***EXCL Lock**
Setting	As *DSNR or *PGMR, perform an action that uses a field. In this case, only the field is locked.	As *DSNR, access the Edit Field Details panel. An *EXCL lock is also set on the field's referencing file, if any.
Displaying	Enter **L**=Locks against a field (when available) to view locked fields using the Object Lock panel. **Note:** L=Locks is available on most panels that access the Edit Field Details panel.	To view field locks, enter **L**=Locks against a field (when available) to display the Object Lock panel. To view associated file locks, press **F22**=File Locks (when available). **Note:** F22=File Locks is displayed only if the model value YOPNACC is set to *YES.
Removing	Exit panel where lock has been set; e.g., return to the Edit Database Relations panel.	Press **Enter** and exit the Edit Field Details panel. All associated field and file locks are also removed.
Notes:	Multiple *READ locks can exist on a field. An *EXCL lock cannot be set on a field until all *READ locks are removed.	Only one *EXCL lock can be set on a field at a time. If the locked field is referenced by other fields, all (REF) fields in the domain are also locked.

Using Narrative Text

Narrative text describes the purpose of a CA 2E object within the design. Each object can include an unlimited amount of narrative text.

Narrative text is used to:

- Document the model. All CA 2E documentation commands have a PRTTEXT parameter that lets you specify whether you want to include text in a listing and, if so, which type of text.
- Describe the model interactively. The developer programmer can examine text at any time when using the model.
- Generate Help text for CA 2E functions
- Generate program synopses

Types of Narrative Text

Narrative text can be functional or operational:

- **Functional** narrative text describes the purpose of the design object, restrictions associated with the object, and notes about the reasons for design decisions. Functional text is available through the documentation commands.
- **Operational** narrative text describes the function of the object to the end user. Operational text is incorporated into Help text. If you do not specify operational text in addition to functional text,CA 2E uses the functional text as Help text.

Creating/Editing Narrative Text

You can create new narrative text for an object or edit existing narrative text. You access the narrative text feature from panels that display the line selection value for this feature.

Accessing Narrative Text

To access the narrative text feature from the Edit Database Relations panel, enter **N** in the selection line for the object you want to describe and press Enter. The Edit Narrative Text panel displays.

Other panels let you access narrative text by pressing F20. This function key also lets you switch between functional and operational narrative text.

User Interface Manager

If you generate Help text using User Interface Manager (UIM), the Help text you create in narrative text formats according to the UIM defaults set up for your system. You can create special formatting effects, such as columns or tables, by embedding UIM tags in the narrative text.

More Information

For more information about:

- Generating objects with UIM, see the "Implementing Your Application" chapter in the *Generating and Implementing Applications* guide.
- Tailoring Help text using UIM, see the *Application System/400 Guide to Programming Displays* guide.

Entering/Editing the Text

You can add an unlimited amount of narrative text, with 24 lines per page. You edit both functional and operational text in the same manner. You can edit the text from the Edit Narrative Text panel, or if the IBM S/38 Text Management product for the IBM i is installed on your system, you can use a function key (F7) to convert the text to a temporary EDTTXT document. The following table lists the active function keys and their function.

Function Key	Action	Comment
F3	Save changes and exit program.	
F4	Make current line top of text display.	
F5	Return to last saved version of text.	
F6	Cancel block for pending copy, move, or delete.	Blocked text is no longer highlighted.
F7	Edit text using TM/38.	Convert text to temporary TM/38 EDTTXT document and edit with EDTDOC commands. Reconvert document to CA 2E text format. EDTTXT paragraph Ids and text emphasis information are not saved.

Function Key	Action	Comment
F8	Mark line or block.	To block text, place the cursor on the first line you want to block and press F8. Then place the cursor on the last line you want to block and press F8. You can block any number of pages. If you do not block a line, the copy, move, or delete function operates on the line where the cursor is.
F9	Add blank line above current line.	
F10	Display text from beginning.	
F11	Delete text or block and close up text.	
F13	Exit without saving changes.	All changes you made in the session are lost.
F14	Display CA 2E product map.Display product map.	
F15	Display last line of text.	
F16	Copy marked line or block.	Places the text on the line immediately preceding the line with the cursor. To copy one line, place the cursor on the line and press F16 twice.
F17	Move marked line or block.	Places the text on the line immediately preceding the line with the cursor. To move one line, place the cursor on the line and press F17 twice. If the cursor is within a block, both the move and the block are canceled.
F19	Edit text using SEU.	Convert the text to a temporary source member. Edit using SEU commands. Reconvert to CA 2E text format.
F20	Toggle between operational and functional narrative text.	Saves changes
F24	Display more keys	

When you finish editing, press Enter to accept the text. Then save changes and exit by pressing F3 (Exit) to return to the panel from which you accessed the narrative text.

Displaying Model Object Cross References

CA 2E provides a cross-reference feature that helps you identify dependencies between CA 2E objects. Using this feature you can identify all other model objects that use or are used by a selected object.

The information you receive depends on the object you select:

- For a **physical access path**, the panel displays all references to the access path by other non-physical access paths, built over it or joined to it.
- For **other access path types**, the panel displays the associated physical access paths.
- For a **function**, the panel displays a list of all functions that reference the function.
- For a **field**, the panel displays references by CA 2E files, functions and parameters.
- For a **condition**, the panel displays usage in a CA 2E access path, to specify selection criteria, in an action diagram, or for a field, to control the validation and default values.

More Information

For more information about model object cross references based on object type, see the *Command Reference* guide.

Accessing the Cross References Utility

Following are ways to access this utility:

- Enter the Display Model Usages (YDSPMDLUSG) command at a command line.
- Enter the Display Model References (YDSPMDLREF) command at a command line.
- Use options on the Edit Model Object List panel.
- Enter **U** in the selection line to the left of the object from the interactive panels shown in the following table and press Enter.

Usage on	Invoked from Panel
CA 2E Access Paths	Edit File Details

Usage on	Invoked from Panel
	Display Access Paths
	Display Access Path References
CA 2E Functions Functions	Edit Functions
	Display Functions
	Display Access Path Functions
CA 2E Fields Fields	Display Fields

In all cases the Display Model Usages or Display Model References panel displays.

```
Gen objs :    1            Display Model Usages      Model. . : SYMDL
Total. . :    2                                      Level. . : 001
Object . : Change Employee            Owner . : Employee
Type . . : FUN Attribute: RPG   Include inactive code: *Yes  Exclude system objs .
*YES
Scope. . : *NEXT    Filter. . . : *ANY     Current objects only. *YES
Object . :                            Type. .         Reason. . *FIRST

2=Edit    3=Copy  4=Delete object  5=Display  8=Details  10=Action Diagram
13=Parms  14=GEN batch             15=GEN interactive    16=Y2CALL

Opt Object                 Typ Attr Owner                Lvl Reason
    Change Employee        FUN DBF Employee              000 *OBJECT
    Edit Employee          FUN RPG Employee              001 *DFTDBF


                                                                    Bottom
F3=Exit   F5=Refresh   F9=Command line   F12=Previous         F15=Top level
F16=Build model list   F21=Print list    F22=File locks       F23=More options
```

More Information

For more information about the Display Model Usages panel and model object cross references, see the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide.

Using the Display Services Menu

The Display Services Menu lets you access CA 2E support functions while you are using a model. This section describes how to access and exit the Display Services Menu and to use two of the options available on this menu:

- Documentation menu
- Display system parameters

Accessing/Exiting the Display Services Menu

You access the Display Services Menu by pressing function key F17. This function key is available from a number of CA 2E panels. You can also use the YEDTMDL command to access the Display Services Menu. Enter the following at a command line:

```
YEDTMDL ENTRY(*SERVICES)
```

When you finish using the tasks on the Display Services Menu exit by pressing F3 (Exit) to return to the panel from which you accessed the menu.

```
DISPLAY SERVICES MENU                SYMDL

Generation          1. Submit model create request (YSBMMDLCRT)
                    2. Convert model data menu

Documentation       6. Documentation menu
                    7. Convert model panel designs (YCVTMDLPNL)

Model               8. Display all access paths
                    9. Display all functions
                   10. Display model values (YDSPMDLVAL)
                   11. Edit model pprofile (YEDTMDLPRF)
                   12. Work with model lists (YWRKMDLLST)
                   13. Edit model list (YEDTMDLLST *SESSION)
                   14. Impact analysis menu

Change Control     21. Go to Synon/CM menu

           Option:    (press F4 to prompt commands)

F3=Exit  F6=Messages  F8=Submitted jobs  F9=Command line  F10=Display job log
```

Invoking Documentation Commands from the Documentation Menu

CA 2E includes commands that print the documentation for a data model. The Display Services Menu contains an option, Documentation Menu, that lets you invoke documentation commands from a menu. These commands include:

- YDOCMDLREL—Document model relations
- YDOCMDLF—Document model files
- YDOCMDLACP—Document model access paths
- YDOCMDLFLD—Document model fields
- YDOCMDLMSG—Document model messages
- YDOCMDLAPP—Document applications

To execute a documentation command:

1. From the Display Services Menu, select the option to display the documentation menu and press Enter. The Display Documentation Menu panel displays.
2. Enter the number for the command you want to invoke. A panel displays that includes:
   - Selection criteria allowing you to identify the CA 2E objects you want to document.
   - Print options allowing you to identify the information you want to include for the selected objects.
   - The PRTTEXT parameter lets you specify whether you want to include narrative text in the listing and, if so, whether you want to include the functional or the operational text .
3. Edit the prompts as necessary.
4. When you are ready to execute the command, press Enter. The system is inhibited while the command executes.

   Note: These commands can all be run in batch mode.

When the command is finished, the Display Documentation Menu redisplays with a completion message.

For more information about the documentation commands, see the *Command Reference guide*.

More Information

For more information about the documentation commands, see the *Command Reference* guide.

Displaying CA 2E System and Model Values

The Display Services Menu includes an option, Display model values (YDSPMDLVAL), that lets you display all model values:

1. From the Display Services Menu, select the option to Display model values (YDSPMDLVAL) and press Enter. The Display Model Values menu displays.
2. To display all model values, accept the default and press Enter. The Design Option Values panel displays.
3. When you finish, return to the Display Model Values Menu. Press F3 (Exit).

From the Display Model Values menu, you can select another option or press F3 (Exit) to return to the Display Services Menu.

Viewing/Editing Panel Default Attributes

From the Display All Values panel, you can display and edit default display attributes.

From the Display All Values panel, press F10**.** The Edit Default Display Attributes panel displays. This panel displays a matrix with the default attributes for fields on panels and reports created in CA 2E.

You can edit, as well as view, these attributes:

- If you want only to view the attributes, press F3 when you are ready to return to the Design Option Values panel. The panel displays the message:

Default screen attributes displayed for model (*model name*).

- If you want to change a default attribute:
  - Y indicates the attribute is active for the field type.
  - Blank indicates the attribute is not active for the field type.

When you finish changes, accept the edits. Press Enter. Then press F3 to return to the Design Option Values panel. The panel displays the message

`Default screen attributes edited for model (`*`model name`*`).`

Using Online Help

CA 2E contains a comprehensive online help facility. Features include:

- Diagnostic messages
- Selection displays
- Help text on commands and panels
- Product Map

Diagnostic Messages

CA 2E sends messages when a long-running process is executing or when an error occurs. Messages appear on line 24 of the displayed panel. Second level Help text is usually available by pressing Help.

Selection Displays

You can display a list of allowed values for most fields by entering a question mark (**?**) into each field and pressing Enter.

Help text

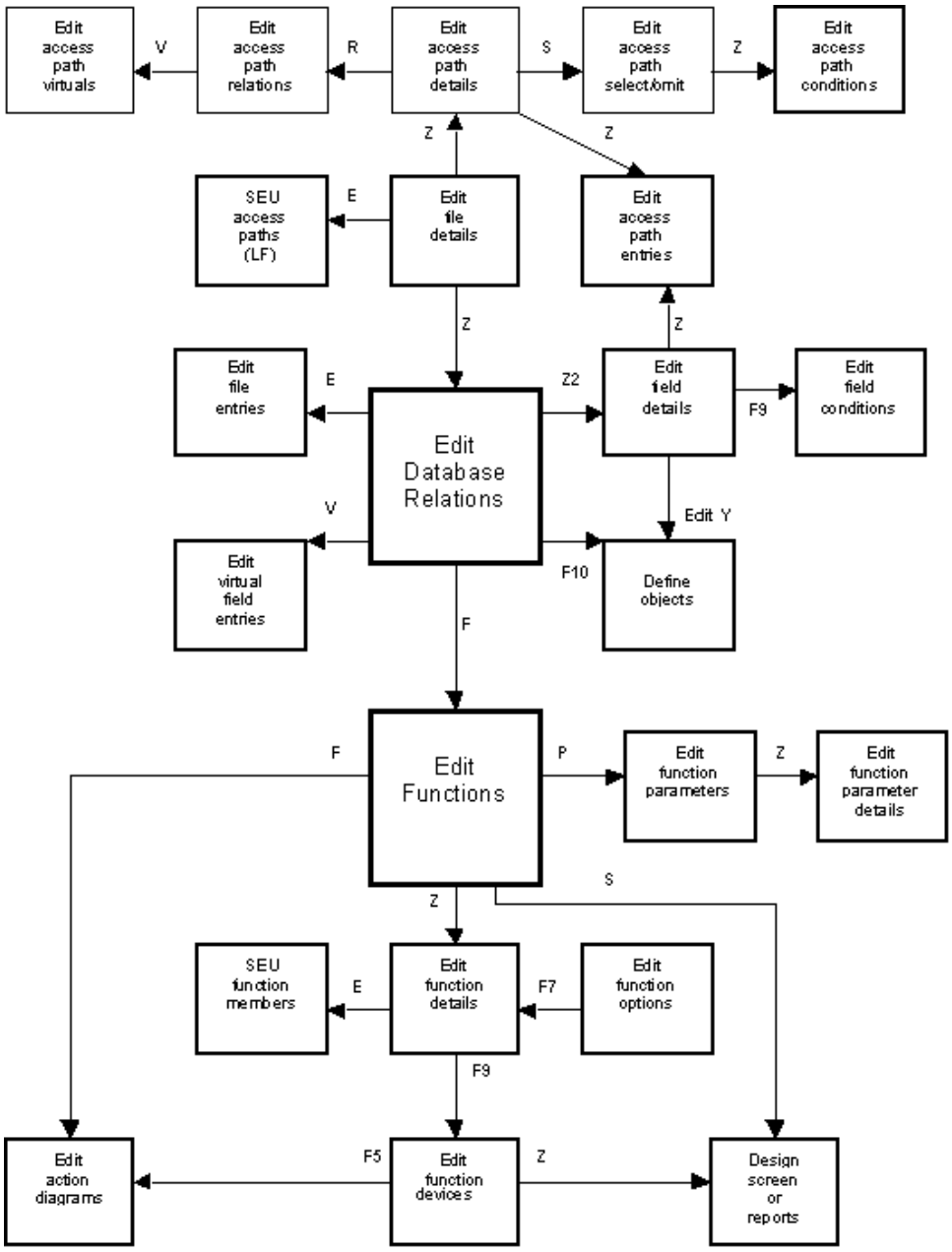
Each functional panel in CA 2E includes online Help text. Help provides detailed information about the display down to the field level. You access help by pressing the Help key.

CA 2E provides several means to navigate through help. Where information is extensive, the Help text provides an index. You can also scroll through the text panel-by-panel, pressing Page Up to display the previous panel and Page Down to display the next one.

Product Map

The product map helps you navigate through CA 2E. This feature shows where you are in relation to other functions in the product. You display the product map using function keys:

- To display the product map, press F14. The product map shows an outline map of the functions in the CA 2E program you are using. The map displays the functions you can invoke from the top level display. The function you are currently using is blinking. The functions you passed through are in high intensity.
- To display a detail map, press F12. Use this function key to toggle between the product map and the detail map.
- To return to the program, press F3.


Edit access path virtuals
V
Edit access path relations
R
Edit access path details
S
Edit access path select/omit
Z
Edit access path conditions
Z
Z
SEU access paths (LF)
E
Edit file details
Edit access path entries
Z
Z
Edit file entries
E
Edit Database Relations
Z2
Edit field details
F9
Edit field conditions
V
Edit Y
Edit virtual field entries
F10
Define objects
F
F
Edit Functions
P
Edit function parameters
Z
Edit function parameter details
S
Z
SEU function members
E
Edit function details
F7
Edit function options
F9
Edit action diagrams
F5
Edit function devices
Z
Design screen or reports

Chapter 4: Using Your Development Environment

This chapter describes tasks you perform to set up a model environment after you create a model. These tasks include using CA 2E commands to build and store the model library list, controlling user access to a model, granting authority for users to perform various tasks, and customizing the model to meet the requirements of the application you are building.

This section contains the following topics:

Managing the Model Library Lists

Library lists provide an efficient means to control the objects a job can access. An i OS library list, however, lasts only until the job finishes or until it is overwritten with another list.

Toolkit library lists are designed to overcome that limitation. You can store a Toolkit library list to use in every session desired. You can retrieve the list and apply it to user profiles and other jobs. You can edit the list, and you can copy it to use as the library list for another model.

Toolkit library lists are a Toolkit feature. They are stored by default in the Toolkit product library. You can use the Toolkit Move Toolkit User Data Objects (YMOVY1DTA) command to move the file containing your Toolkit library lists as well as other user objects to another library.

More Information

For more information about the YMOVY1DTA command, see the *Toolkit Reference*.

System-Wide Values

System-wide values apply to all models on a machine. These values are set during installation. You can override them once the product is installed. System-wide values include:

- The action diagram symbols (YACTSYM)
- The default HLL (YHLLGEN)
- The name of the null model (YNLLMDL)
- The IBM i or S/38 editing environment (YCRTENV)
- The system prompts (YSYSPMT)
- The system database access method (YSYSDBF)

Data Areas: Company Name Versus Company Text

The company text model value (YCMPTXT) contains the name of the model that displays in the top right corner of each panel for that model and in the banner for generated code.

The company text model value is different from the company name that prints on reports or displays on panels of generated programs. The company name is retrieved at execution time from the data area YYCOTXA located in the generation library.

Model Specific Values

Model-specific values set design standards for a particular model. These values include:

- Model use and code generation values, such as model name, automatic naming, and HLL options.
- Code generation values, such as date validation and default function option values.
- Implementation values, such as the working environment and the database access.

Setting Up the Model Library List

When you use the YCRTMDLLIB command to create a model,CA 2E creates a Toolkit library list. The name of this library list is stored in the Library List (YLIBLST) model value. By default, the name of the library list is the same as the name of the model library.

To use a model, your library list must contain the data model, generation libraries, and the CA 2E and Toolkit product libraries. The list may also include the library for the generated source.

Setting Up the Library List for RPG, COBOL, or RPG/COBOL

The YCRTMDLLIB command also sets the default for the HLL used to generate source. Unless you have merged the HLL objects into the base product library at install time, the library list created with this command contains the libraries for the HLL s you will use:

- To generate source in RPG, the library named Y2SYRPG.
- To generate source in COBOL, the library named Y2SYCBL.
- To generate source in both RPG and COBOL, both libraries Y2SYRPG and Y2SYCBL.

Setting Up the Library List for Other National Languages

If you are using a national language that is different from the one installed in the base product library, your library list must include the target national language library. The library must display above the other product libraries.

More Information

For more information about Advanced National Language Support, see the "National Language Support" chapter in the CA 2E *Generating and Implementing Applications* guide.

Using the Change Library List (YCHGLIBL) Command

The YCRTMDLLIB command automatically creates a Toolkit library list that contains the libraries needed to use the model. It also sets the Library List (YLIBLST) model value to the library list specified by the LIBLST parameter. By default, the name of the library list is the same as the name of the model library. You can set your library list from the CA 2E-created list.

One way to set your library list is to use the i OS Change Library List (CHGLIBL) command. This command makes the CA 2E-created list your list for the duration of the session.

You can also use the Toolkit Change Library List (YCHGLIBL) command. This command uses the named library list to replace your library list. You can invoke this command either from the CA 2E Main Menu or from a command line.

More Information

For more information about the CHGLIBL command, see Volume 2 of the *i OS CL Reference*.

Invoking YCHGLIBL from the Main Menu

To invoke the command from the CA 2E Main Menu:

1. From a command line, enter YGO *Y2 and press Enter. The Main Menu displays.

   Note: You can also invoke the YCHGLIBL command from each of the submenus associated with the three user types. You can access these submenus from the Main Menu or you can use the YSTRY2 command. For example, to access the CA 2E Programmer (*PGMR) Menu, enter the following at a command line:

YSTRY2 *library-list-name* MENU(PGMR)

1. Select the Change to work with another model option. The Change Library List (YCHGLIBL) panel displays.
2. Select the library list you want to use to create your library list by doing one of the following:
   - Entering the name of the list.
   - Pressing Enter to display the Select Library List panel. From this panel you can display the contents of library lists and select the list you want.
3. When you are ready to execute the command, press Enter. The command replaces your current library list with the list you selected.

Invoking YCHGLIBL from a Command Line

You can replace your library list by entering YCHGLIBL from a command line. For example, to set your library list from a model library list named MYLIB, enter the following and press Enter:

```
YCHGLIBL  LIBLST(MYLIB)
```

You can also use the short form of the command:

```
R  LIBLST(MYLIB)
```

When the command finishes, your library list might look something like the following:

- QTEMP
- CA 2E generation library
- CA 2E SQL library (optional)
- CA 2E model library
- QGPL
- Toolkit National Language library (Y1SYV*xxx*)
- Toolkit product library (Y1SY)
- CA 2E generator library (Y2SYRPG or Y2SYCBL)
- CA 2E National Language library (Y2SYV*xxx*)
- CA 2E product library (Y2SY)

CA 2E Commands and the Model Library List

Some CA 2E commands automatically change the library list. When you invoke a command that loads a model,CA 2E checks the current job's library list for the specified model library as the first model library in the list. If it is not found,CA 2E automatically replaces the current job's library list with the library list specified by the YLIBLST model value for the requested model. When the command completes,CA 2E resets the current library list back to the original library list.

More Information

For more information about CA 2E commands and how they affect the library list, see the CA 2E *Command Reference Guide*.

Editing the Library List

Once you set your library list, you may need to customize it to meet the requirements of a particular model.

The i OS provides a set of commands that lets you add or remove a single library list entry and edit multiple library list entries. These commands include:

- ADDLIBLE (Add library list entry)
- RMVLIBLE (Remove library list entry)
- CHGLIBL (Change library list)
- EDTLIBL (Edit library list)

The changes you make with these commands are valid only for the duration of the session.

The Toolkit Edit Library List (YEDTLIBLST) command calls an interactive program that lets you edit a Toolkit library list and save the edited list. The YEDTLIBLST command lets you create library lists that are tailored for the needs of particular models. You can use this command to add libraries to, reorder libraries in, or delete libraries from a library list. A model library list can contain up to 25 libraries.

More Information

For more information about these commands, see the *i OS CL Reference*.

Invoking the YEDTLIBLST Command

1. Invoke the YEDTLIBLST command from a command line or by using the CA 2E Designer (*DSNR) Menu as follows:

   a. Enter the YEDTLIBLST command at a command line:

YEDTLIBLST LIBLST(*name-of-library-list*)

Note: You can also use the letter "L" as an abbreviation for the YEDTLIBLST command. The following has the same effect as the previous command:

L YLIBLST(*name-of-library list*)

Examples:

YEDTLIBLST LIBLST(MYLST)

To edit the library list for the current job, enter:

YEDTLIBLST

   a. Display the CA 2E Designer (*DSNR) Menu using one of the following:

      - Select the Display Designer (*DSNR) menu option from the CA 2E Main Menu.
      - Enter the following at a command line:

YSTRY2 *library-list-name* MENU(DSNR)

Select the Edit Library List for Model option.

1. The Edit Library List Entries panel displays. You can display additional library information by pressing F11. This function key toggles between views of the list. You can use this panel to:
   - Edit the entries in the library list.
   - Edit the Current Library used for this list.
   - Edit the model job description used for this list.
   - Completely replace the User Library and Current Library portions of the displayed list with another library list.
2. When you finish all edits on the Edit Library List panel, exit from the panel. Press F3. The Exit Edit Library List Entries panel displays.

   This panel displays the values for the current library list and job description, and indicates whether you changed entries on the list. You can accept the entries or change them. If you want to print the library list, you can also execute the Document Library List command from this panel.
3. When you are ready to execute YEDTLIBLST, press Enter. Depending on the options you select when you exit, the edited library list replaces the current library list:
   - Interactively

- In the specified job list
- In the Toolkit library list file YLIBLST

More Information

For more information about:

- Editing the entries, see the Editing Library List Entries section in this chapter.
- Editing the Current Library for this list, see the Editing the Current Library for the List section in this chapter.
- Editing the model job description, see the Editing the Model Job Description for the List section in this chapter.
- Replacing the User Library and Current Library portions, see the Retrieving a Library List from Another Source section in this chapter.

Editing Library List Entries

To edit library list entries, perform the following steps:

1. Edit the entries for the library list by doing the following:
   - Add a library by placing the cursor at the first available line and enter the name of the library you want to add. To place the library in a different position, add a sequence number to the left of the library name.
   - Remove a library by placing the cursor on the first character of the library you want to delete and delete the name from the line.
   - Replace a library by placing the cursor on the first character of the library you want to replace and enter the new name over the displayed name.
2. When you finish editing library list entries, accept the changes. Press Enter. If you added libraries, the libraries resequence according to the sequence numbers you typed.

Editing the Current Library for the List

The YEDTLIBLST command lets you change the Current Library portion of the library list. On the Edit Library List panel:

1. Place the cursor on the entry field for the Current Library prompt.
2. Enter the name of the library that you want to assign as the Current Library when the list is to be used to change the library list for a job.

Editing the List for the Model Job Description

The YEDTLIBLST command lets you change the library list you want to use in a job description. On the Edit Library List panel:

1. Place the cursor on the entry field for the Job Description prompt.
2. Enter the name of the job description you want to associate with the list. This should match the job description defined for the model in the YCRTJBD model value.
3. At the entry field for the Library prompt, enter the name of the library that contains the job description you selected.

Retrieving a Library List from Another Source

You can retrieve the User Library and Current Library portion of your library list from another source. You can change only the User Library and Current Library portions of the list. All other portions remain the same.

1. From the Edit Library List panel, press F13. The Edit Library List Entries - Services panel displays.
2. Select the source to be used to refresh the Edit Library List panel. You can reload the list from:
   - The User Library and Current Library of the current job
   - The system value QUSRLIBL.
   - The User Library and Current Library of a specified Toolkit library list. You specify the list by name.
3. When you finish, press Enter. The library list is refreshed with the list you specified.

Controlling User Access

CA 2E user access and ownership security features protect the models within a development environment from unauthorized access and change.

Through Model Ownership

The user profile used to create the model owns the model library, its associated generation library, and i OS objects. The owner can be an individual user or a user group. The owner can transfer ownership to another user.

The owner of a model can control how users or groups of users can access and edit the model by granting or revoking appropriate authority to access certain data areas and to update the i OS objects that make up the model. The same considerations that apply to granting and revoking authority in the i OS environment also apply to granting and revoking authority in a CA 2E environment.

The primary means to control access to the model is by setting authority on the YMDLLIBRFA data area. To edit this data area you can use the Edit Model Authority option on the CA 2E Designer (*DSNR) Menu.

More Information

For more information about:

- The YMDLLIBRFA data area, see the Through Authority section in this chapter.
- The CA 2E Designer (*DSNR) Menu, see the "Using Your Model" chapter in this guide.

Changing Model Ownership

The owner of a model can transfer ownership with the Toolkit Change Object Ownership (YCHGOBJOWN) command.

For example, to change the ownership of all objects in the MYLIB library to a new YOURNAME owner, you would enter the following and press Enter:

```
YCHGOBJOWN  OBJ(MYLIB/*ALL) OBJTYPE(*ALL) NEWOWN(YOURNAME) CHGLIBOWEN(*YES)
```

This example includes the following required parameters:

- **OBJ(MYLIB/*ALL)**—Sets the generic name of the objects whose ownership is to be changed followed by which objects in the list are to be changed.
- **OBJTYPE(*ALL)**—Sets the type of objects whose ownership is to be changed.
- **NEWOWN(YOURNAME)**—Sets the new owner of the objects. Must be the name of an existing user profile.
- **CHGLIBOWN(*YES)**—Ensures that the new owner owns the library as well as all objects in the library.

More Information

For more information about the YCHGOBJOWN command, see the *Command Reference Guide*.

Through Authority

The owner profile can grant or revoke authority for additional users to access the model, to update specific objects in the model, and to generate and compile the source for the model.

Types of User

CA 2E lets you access a model as one of three different user types: designer, programmer, or user. Designers and programmers are referred to collectively as "developers." Your user type determines the limitations placed on you when you access and edit the model.

Designer User Type

A designer (*DSNR) can change any aspect of the model, both data relationships and functional specifications. Whether more than one designer can use the model at one time depends on the current setting of the Open Access (YOPNACC) model value. If YOPNACC is set to:

- *NO, only one user can use the model as a designer. In this case, no other user can access the model while the designer is using it.
- *YES, multiple designers and programmers can use the model at the same time. In addition,CA 2E enables file and field locking to prevent two designers from updating database relations at the same time.

You can control Open Access either by setting the YOPNACC model value or by using the Edit Model (YEDTMDL) command.

A designer can also have *LOCK authority (*DSLK). This authority allows the designer to place a permanent, exclusive lock on a function or access path that only a designer can unlock. In addition, only a designer with *LOCK authority can change the YOPNACC model value.

Designers are also responsible for synchronizing the model after updating a database relation. Synchronizing ensures that any changes to the relations in a model have been reflected in the CA 2E file entries of the model. To synchronize the model, a designer must have exclusive use of the model.

The CA 2E Designer (*DSNR) Menu provides a list of tasks available to users having *DSNR authority. You can access this menu by entering the following at a command line.

```
YSTRY2 library-list-name MENU(DSNR)
```

More Information

For more information about the *DSNR menu, see the Menus section in the "Using Your Model" chapter in this guide.

Programmer User Type

A programmer (*PGMR) can create, change, and delete any access paths, arrays, functions, including all work with action diagrams, and field conditions for database and function fields but cannot alter the database files or fields. Several programmers may simultaneously use a model. Whether programmers can use a model while a designer is using it depends on the current setting of the Open Access (YOPNACC) model value.

If YOPNACC is set to:

- *NO, programmers cannot use a model while a designer is using it and cannot use an unsynchronized model.
- *YES, multiple designers and programmers can use the model at the same time.

The CA 2E Programmer (*PGMR) Menu provides a list of tasks available to users having *PGMR authority. You can access this menu by entering the following at a command line.

```
YSTRY2 library-list-name MENU(PGMR)
```

More Information

For more information about the *PGMR menu, see the "Using Your Model" chapter in this guide.

'User' User Type

A user (*USER) is limited to viewing the model and cannot change it.

The CA 2E User (*USER) Menu provides a list of tasks available to users having *USER authority. You can access this menu by entering the following at a command line:

```
YSTRY2 library-list-name MENU(USER)
```

More Information

For more information about the *USER menu, see the "Using Your Model" chapter in this guide.

User Authority Advantage

You control the ability of each of the three user types to access a model by granting different levels of authority to the YMDLLIBRFA data area. This is explained in detail in the Granting Authority section in this chapter. The YSTRY2 command displays your user authority as a message at the bottom of the CA 2E menus.

The following table shows the minimum authority needed for each user type.

User Type	Minimum Authority Needed
*DSNR with *LOCK (*DSLK)	*ALL
*DSNR	*CHANGE plus *OBJMGT
*PGMR	*CHANGE
*USER	*USE

Note: *LOCK authority gives you the capability to lock functions and access paths and also to control the Open Access (YOPNACC) model value.

More Information

For more information about:

- The YOPNACC model value, see the Open Access section in the "Creating and Managing Your Model" chapter.
- Locking objects and synchronizing the model, see the Locking Objects section in the "Using Your Model".
- Data area authority levels, see the Editing Authority to Access Data Areas section in this chapter.

Granting Authority

You grant authority to i OS objects with the i OS Grant Object Authority (GRTOBJAUT) command.

More Information

For more information about the GRTOBJAUT command, see Volume 4 of the *i OS CL Reference*.

Granting Authority to Update Objects

To allow a user to update a model, grant the following authority:

- Use (*CHANGE) authority for the model library. For example:

GRTOBJAUT OBJ(QSYS/MYMDL) OBJTYPE(*LIB) USER(YOU) AUT(*CHANGE)

- Data authority to change (*CHANGE) for the objects in the model library. For example:

GRTOBJAUT OBJ(MYMDL/*ALL) OBJTYPE(*ALL) USER(YOU) AUT(*CHANGE)

- The following minimum authorities are required to gain access to a CA 2E data model:
  - Object management authority is required to certain i OS database files in the model library to which additional members may be added:
- YJOBLSTRFP
- YMDLLSTRFP
  - Change authority is required to the following i OS database files in the model library:
- YMDLPRFRFP
- YMDLOBJRFP
  - Object existence authority is required to the following i OS database files in the model library:
- YOBJLCKRFP
- YSSNLCKRFP
- YSSNDTARFP
- YLSTSGTNXP

  For example:

GRTOBJAUT OBJ(MYMDL/YJOBLST*) + OBJTYPE(*FILE) USER(YOU)

AUT(*ALL)

GRTOBJAUT OBJ(MYMDL/YMDLLST*) + OBJTYPE(*FILE) USER(YOU)

AUT(*ALL)

GRTOBJAUT OBJ(MYMDL/YMDLPRF*) + OBJTYPE(*FILE) USER(YOU)

AUT(*ALL)

GRTOBJAUT OBJ(MYMDL/YMDLOBJ*) + OBJTYPE(*FILE) USER(YOU)

AUT(*ALL)

```
GRTOBJAUT  OBJ(MYMDL/YOBJLCKRFP) + OBJTYPE(*FILE)  USER(YOU)  AUT(*ALL)
 GRTOBJAUT  OBJ(MYMDL/YSSNLCKRFP) + OBJTYPE(*FILE)  USER(YOU)  AUT(*ALL)
 GRTOBJAUT  OBJ(MYMDL/YSSNDTARFP) + OBJTYPE(*FILE)  USER(YOU)  AUT(*ALL)
 GRTOBJAUT  OBJ(MYMDL/YLSTSGTNXP) + OBJTYPE(*FILE)  USER(YOU)  AUT(*ALL)
```

- Appropriate authority to use data area YMDLLIBRFA in the data model library.

More Information

For more information about data area authority levels, see the Editing Authority to Access Data Areas section in this chapter.

Granting Authority to Generate Source

To allow a user to generate source, in addition to the rights listed in the Granting Authority to Update Objects section in this guide, grant object existence authority to the source files in the generation library. For example:

```
GRTOBJAUT  OBJ(MYGEN/*ALL)  OBJTYPE(*FILE) + USER(YOU)  AUT(*ALL)
```

Granting Authority to Compile Source

To allow a user to compile source, in addition to the rights listed in Granting Authority to Update Objects section in this chapter, grant data authority to update the generation library. For example:

```
GRTOBJAUT OBJ(QSYS/MYGEN) OBJTYPE(*LIB) USER(YOU)  AUT(*ALL)
```

Editing Authority to Access Data Areas

When a user attempts to access a model,CA 2E checks to ensure the user has the authority to access data areas YMDLLIBRFA and YGENLIBRFA with the specified user type before allowing access to the model.

Designer User Type—To allow a user access to a model as a designer, grant at least CHANGE and OBJMGT authority. To allow the designer to lock and unlock objects, grant ALL authority. For example:

```
EDTOBJAUT  OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA)
```

Press Enter and F11. Update the panel that displays by assigning authorities to the appropriate user as follows:

- For a designer without lock authority, enter *CHANGE in the Object Authority column and X in the Mgt column.
- For a designer with lock authority, enter *ALL in the Object Authority column. For example:

EDTOBJAUT OBJ(MYMDL/YMDLLIBRFA) OBJTYPE(*DTAARA)

Press Enter. Update the panel that displays by assigning *CHANGE authority. For example:

```
EDTOBJAUT  OBJ(MYMDL/YMDLLIBRFA) OBJTYPE(*DTAARA)
```

Press Enter. Update the panel that displays by assigning *USE authority.

The following illustration shows the authorities needed to the YMDLLIBRFA data area to access a model:

- User JXY has designer with lock access.
- User DSH has designer access.
- User OPK has programmer access.
- User RMG has user access.

```
                          Edit Object Authority
Object . . . . . :  YMDLLIBRFA      Owner  . . . . . : DEV
  Library. . . . . :    MYMDL         Primary group. . . . : *NONE
Object type. . . . :  *DTAARA

Type changes to current authorities, press Enter.

  Object secured by authorization list . . . . . *NONE

                          Object    ----------Object-----------
User        Group       Authority  Opr  Mgt  Exist  Alter  Ref
QSYS                    *ALL_____   X    X     X      X     X
*GROUP      DEV         *ALL_____   X    X     X      X     X
RMG                     *USE_____   X
OPK                     *CHANGE__
DSH                     *USER DEF   _    X     _      _     _
JXY                     *ALL_____   X    X     X      X     X

                                                                      Bottom
F3=Exit   F5=Refresh   F6=Add new users    F10=Grant with reference object
F11=Display data authorities   F12=Cancel  F17=Top   F18=Bottom
Object authorities changed.
```

```
                          Edit Object Authority
Object . . . . . :  YMDLLIBRFA      Owner  . . . . . : DEV
  Library. . . . . :    MYMDL         Primary group. . . . : *NONE
Object type. . . . :  *DTAARA

Type changes to current authorities, press Enter.

  Object secured by authorization list . . . . . *NONE

                          Object    ----------Object-----------
User        Group       Authority  Opr  Mgt  Exist  Alter  Ref
QSYS                    *ALL_____   X    X     X      X     X
*GROUP      DEV         *ALL_____   X    X     X      X     X
RMG                     *USE_____   X                       X
OPK                     *CHANGE__   X    X     X      X     X
DSH                     *USER DEF        X
JXY                     *ALL_____   X    X     X      X     X

F3=Exit   F5=Refresh     F6=Add new users   F10=Grant with reference object
F11=Nondisplay detail    F12=Cancel         F17=Top   F18=Bottom
```

Both designers and programmers must have *ALL authority to access YGENLIBRFA. For example:

```
GRTOBJAUT  OBJ(MYMDL/YGENLIBRFA) + OBJTYPE(*DTAARA)  USER(BERT)  AUT(*ALL)
```

You can grant *USE authority to those users who need to view, but not update, generated source. For example:

```
GRTOBJAUT  OBJ(MYMDL/YGENLIBRFA) + OBJTYPE(*DTAARA)  USER(USER)  AUT(*USE)
```

The following illustration shows the authorities needed to the YGENLIBRFA data area to access a generation library:

- User JXY and DSH have designer access.
- User OPK has programmer access.
- User RMG has user access.

```
                          Edit Object Authority

Object . . . . . :  YGENLIBRFA      Owner . . . . . :  DEV
  Library. . . . . :    MYMDL         Primary group . . . :  *NONE
Object type. . . . . :  *DTAARA

Type changes to current authorities, press Enter.

  Object secured by authorization list. . . . . :  *NONE

                        Object    ----------Object-----------
User        Group       Authority  Opr  Mgt  Exist  Alter  Ref
QSYS                    *ALL_____   X    X     X      X     X
OPK                     *ALL_____   X    X     X      X     X
DSH                     *ALL_____   X    X     X      X     X
RMG                     *USE_____   X    _     _      _     _
JXY                     *ALL_____   X    X     X      X     X
*GROUP      DEV         *ALL_____   X    X     X      X     X
*PUBLIC                 *EXCLUDE_   _    _     _      _     _
                                                                    Bottom
F3=Exit   F5=Refresh   F6=Add new users     F10=Grant with reference object
F11=Display data authorities   F12=Cancel   F17=Top   F18=Bottom
```

```
                          Edit Object Authority

Object . . . . . :  YGENLIBRFA      Owner . . . . . :  DEV
  Library. . . . . :    MYMDL         Primary group . . . :  *NONE
Object type. . . . . :  *DTAARA

Type changes to current authorities, press Enter.

  Object secured by authorization list. . . . . :  *NONE

                        Object    --------------Data---------------
User        Group       Authority  Read  Add  Update  Delete  Execute
QSYS                    *ALL_____   X     X     X       X       X
OPK                     *ALL_____   X     X     X       X       X
DSH                     *ALL_____   X     X     X       X       X
RMG                     *USE_____   X     _     _       _       _
JXY                     *ALL_____   X     X     X       X       X
*GROUP      DEV         *ALL_____   X     X     X       X       X
*PUBLIC                 *EXCLUDE_   _     _     _       _       _

                                                                    Bottom
F3=Exit   F5=Refresh    F6=Add new users   F10=Grant with reference
F11=Nondisplay detail   F12=Cancel         F17=Top   F18=Bottom
```

If you want to grant authority for many users to use YGENLIBRFA, you can set the USER parameter to *PUBLIC rather than granting separate authorities to each user.

Revoking Authority

Sometimes, the owner of a model may want to prevent a user profile from accessing as a designer or unauthorized users from accessing the model.

To set up these limitations, use the Revoke Object Authority (RVKOBJAUT) command.

Revoke *ALL authority for user USER1 to data area YMDLLIBRFA.

For example:

```
RVKOBJAUT  OBJ(MYMDL/YMDLLIBRFA) + OBJTYPE(*DTAARA)  USER(USER1)  AUT(*ALL)
```

More Information

For more information about the RVKOBJAUT command, see Volume 4 of the *i OS CL Reference.*

Compiling Objects in a Multi-Programmer Environment

If all the developers who use a model are not part of the same group profile, one developer cannot recompile objects created by a developer in another group. When the developer invokes the command to generate and recompile, YSBMMDLCRT, the process fails because the developer does not have the authority to delete existing objects. The *PUBLIC authority given by YSBMMDLCRT to each object in the model is only *CHANGE.

The i OS Change Command Default (CHGCMDDFT) command lets you change the default value on the YSBMMDLCRT command to allow anyone authorized to use the data model to recompile all programs in the model. Change the default value from *CHANGE to *ALL.

For example:

```
CHGCMDDFT  CMD(Y2SY/YSBMMDLCRT) + NEWDFT('AUT(*ALL)')
```

Note: If you have separate LDO libraries, this command will reside in the CA 2E LDO library (Y2SYVENG).

Should you find later that you do not want to grant universal recompile privileges, you can restore the authority value to *CHANGE.

A developer can also temporarily change the authority for YSBMMDLCRT when invoking the command from the Display Services Menu. The command is stored in the Y2MSG message file in the two message IDs, Y2R0030 and Y2R0062.

1. To invoke the command, select the Submit model create request (YSBMMDLCRT) option and press F4. A panel displays the parameters for the command.
2. Page to the second panel to display additional parameters. Press F10.
3. Change the PUBLIC AUTHORITY (AUT) value to *ALL.
4. Accept the change and execute the command. Press Enter.

The new authority affects only those objects submitted while using the option. When the command finishes, the PUBLIC AUTHORITY parameter returns to the default value.

More Information

For more information about the CHGCMDDFT command, see Volume 2 of the *i OS CL Reference*.

Setting Up the User Environment

When you create a model, you create a template for your user environment. This template is a copy of the null model. Before you can begin the tasks involved in building an application, you need to change aspects of this model so that it meets the requirements and standards of your development environment. In addition, you may need to perform other tasks that prepare your environment. This section discusses some of these tasks:

- Controlling how names are allocated at both the system and the model level
- Customizing the default device design
- Setting up common routines and utilities

Every user in a model has an associated model profile that can be modified to tailor the development environment. Additional capabilities are available using CA Xtras Change Management (CM).

More Information

For more information about:

- Setting up a default environment, see the Model Profile section in the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide
- CM, see the *CA Xtras Change Management User Guide*.

The Null Model

The null model is the base for new models. It contains no user-defined objects. The null model is installed with CA 2E and reinstalled when you upgrade to a new version of CA 2E.

When you execute YCRTMDLLIB, the command creates a copy of the null model with the default system values for your development environment. YCRTMDLLIB parameters let you modify these defaults. You can then tailor the model to meet the requirements of your development environment.

Shipped System Files (*)

Each model contains a number of CA 2E system files, fields, and functions. The names of all CA 2E objects begin with an asterisk (*). Shipped system objects include:

- ***Built-in functions**—The CA 2E file to which all built-in functions are attached.
- ***Field attribute types**—The CA 2E file to which all field data types, such as DTE, CDE, or TXT, attach. You can add user-defined data types to this file. The validation functions for user-defined data types are attached to this file.
- ***Job data**—A CA 2E file that provides access to job status data, such as job date and user profile name, for use in CA 2E functions. The *Job data shipped file contains system fields that supply execution time information about the job, such as the user name, the job name, or the job start time. The *Job data file contents are fixed. You cannot add fields to this file.

  You can use fields in the JOB context only for input to other functions. You cannot change these fields.

  The JOB context is available in the action diagram of all function types.
- ***Messages**—The CA 2E file to which all message functions attach.CA 2E is shipped with a number of commonly required messages.
- ***Program data**—A CA 2E file that provides access to program status data, such as *Program mode and *Return code, for use in CA 2E functions. The *Program data shipped file contains CA 2E system fields that control the execution of a CA 2E function, such as the *Program mode. The *Program data file contents are fixed. You cannot add fields to this file.

  The PGM context is available in the action diagram of all function types.
- ***Standard header/footer**—A CA 2E file to which any functions defining headers, footers, windows, and action bars used by device functions attach. The file includes five default functions: four of type DFNSCRFMT to define panel design headers and one of type DFNRPTFMT to define report design headers.
- ***Template**—A CA 2E file that provides a method for defining a template for any standard function. A function based on this file is known as a template function.
- ***Synon reserved program data**—A CA 2E file that contains special fields for use in all function types.

Default Model Profile

The default model profile defines defaults for various processes and file specifications for an interactive session. It is shipped with the null model. When you create a new model, this default model profile is automatically copied to the new model library.

You can edit the default model profile to create a default model profile tailored to your environment. To do so, select the Edit Default Model Profile for Model option from the CA 2E Designer (*DSNR) Menu.

More Information

For more information about:

- Setting up a default environment, see the Model Profile section in the "Managing Model Objects" chapter in the Generating and Implementing Applications guide.
- The *DSNR menu, see the Menus section in the "Using Your Model" chapter.

System and Model Values

CA 2E model values control optional features of the product. Changing model values lets you create a model that meets your needs. You can override many model values at the function or file level.

Model values fall into two classifications:

- System-wide
- Model-specific

You can display the model values for a model using the Display Services Menu option Display model values (YDSPMDLVAL)

More Information

For more information about:

- All model values, see the description of the YCHGMDLVAL command in the CA 2E Command Reference Guide.
- Model values related to functions, see the "Setting Default Options for Your Functions" chapter in the CA 2E Building Applications guide.
- The Display system parameters option, see the "Using Your Model" chapter in this guide.

Naming Control

When you think of names in CA 2E, you need to consider two types:

- The name that CA 2E automatically allocates to the i OS objects when generating source code, provided you are using automatic name allocation
- The textual names that you assign when you create objects within a model

This section describes how CA 2E automatically allocates names and provides guidelines for assigning names when you create objects.

Automatic Naming of Generated Code

A model value, YALCVNM, lets you control whether or not CA 2E automatically allocates names within a model. If you set this value to *YES,CA 2E automatically assigns a name for every database file, device file, format field, and program you define. Automatic naming ensures that names generated for model objects are unique across the model.

In CA 2E, the implementation names are issued by a number of CL programs. The source for these programs is available. You can replace these programs with your own routines.

Name Allocation Programs

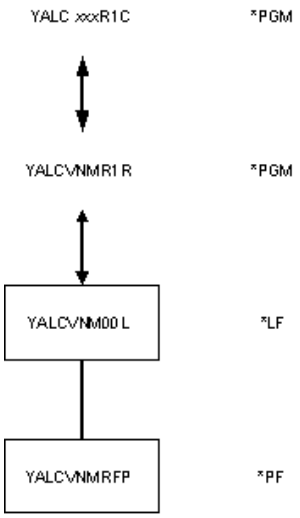
The following table provides the name allocation programs.

CA 2E Object Object	CL Program to Name	Model Entity
Access Paths	YALCPHYR1C	Physical file names
	YALCLGLR1C	Logical file names
	YALCQPFR1C	Query physical file names
	YALCIDXR1C	SQL index names
	YALCVIWR1C	SQL view names
Access Path Formats	YALCFMTR1C	PF and LF format names
Field Names	YALCFLDR1C	Field names
Functions	YALCFUNR1C	Program names
	YALCMSGR1C	Message names
	YALCDSPR1C	Display files
	YALCPRTR1C	Printer files
	YALCHLPR1C	Help text member and help panel group names
	YALCSCMR1C	Screen message identifier
User Interface Manager (UIM) Help Modules	YALCUIMR1C	UIM help module names

Most of the name allocation programs make use of an RPG program, YALCVNMR1R, to retrieve the last used mnemonic for the object type from a file in the model, YALCVNMRFP. The source for the RPG program is also shipped.

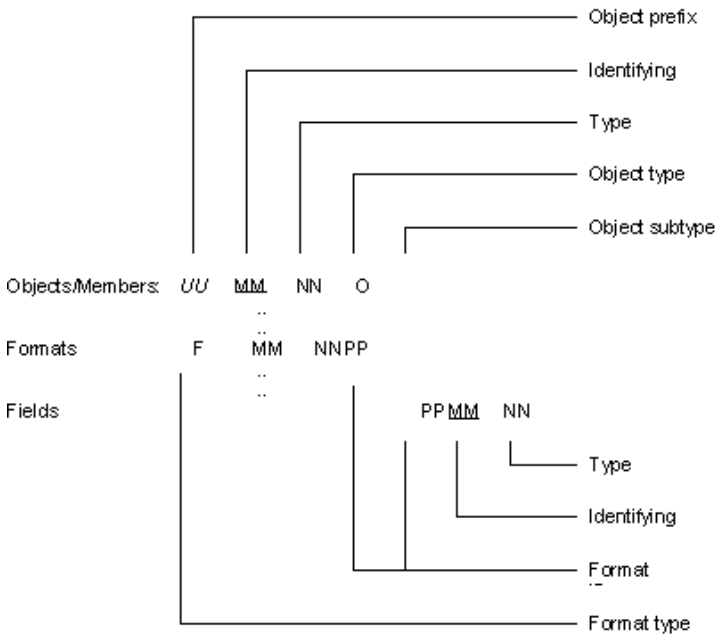
CA 2E ships source code for a number of CL programs that assign names. The source resides in the library Y2SYSRC. These programs provide an Application Program Interface (API) to naming routines. You can modify these programs to achieve any naming convention that fits the constraints of the implementation language.CA 2E checks for duplication and calls the naming programs until they either return a unique name or signal failure by returning blanks.

Automatic Naming: Last-Used Mnemonics



Automatic Naming Algorithm

If you use automatic naming,CA 2E assigns a name according to an algorithm when you create a new entity, such as a field, file, or function, and ensures that the name is unique across the model. The following diagram summarizes how names are assembled for Objects/Members (programs/files), Formats, and Fields.



The following explains the elements in the illustration:

***UU*—The object prefix is derived from the model value YOBJPFX which contains the value stored in the file YALCVNMRFP.**

MM—The identifying mnemonics are a unique, arbitrary pair of letters read from the file YALCVNMRFP. Each object type has a different set of identifying mnemonics that correspond to different records in the file YALCVNMRFP.

For example, physical and logical files will use the next available value in Object type FIL. Functions, including the CLP for query access paths, will use the value in Object type MSG.

Model value YFILPFX contains the last identifying mnemonic used for defining files; see the note following this list for more information.

NN—The type mnemonics are a pair of letters allocated according to the CA 2E object attribute. For example, fields of type CDE are CD and query access path CLP programs are QF.

O—The object type is a single letter that identifies object types; for example, P (physical file), L (logical file), R (RPG program), or C (CLP program). You can set up the values to be used by the object types by using the Edit Generation Types panel. For functions, the object sub-type distinguishes between the help and display file associated with the PGM object type.

X—The object sub-type is a single letter that depends on the object type. For logical files, this letter indicates the access path number. For display files and help member or panel groups, you can set up the values to be used by the object sub-types by using the Edit Generation Types panel.

F—The format type is a single letter that identifies the format type; for example, @ for database files or # for display files. You can set the values to be used by the object types using the Edit Generation Types panel.

PP—The format ID (PP) is an arbitrary set of letters unique for each physical file.

Presetting Automatic Naming Identifiers

Two-character automatic naming identifier that is incremented each time a new name is assigned, ranges from AA to T9 for file names and from AA to Z9 for other objects. This limits the product to 684 file names and 936 names for each of the other object types.

To avoid the inconvenience of reaching these limits you can preassign the next unique code to be used by the automatic naming algorithm for each object type using the Edit Next Mnemonic (YEDTNXTMNC) command. This command displays the following interactive panel.

Note: You need to be out of your model when you run YEDTNXTMNC and you need *DSNR authority to edit the panel. If multiple models use the same naming library, ensure that no one is using any of the models.

```
Change Next Object Prefix (OBJPFX Model Value) for Member Names
      ID Mnemonic Current  Next
       FIL MSG    Prefix  Prefix
       AH  A8      MY      YY

Change Next Field Type Mnemonic
Field Identifying Type     Next
Type   Mnemonic   Code     Code
 CDE      AI       CD       CE
 DTE      AA       DT       DE
 NBR      AF       NB       NR
 NAR      AA       NA       NE
 QTY      AB       QT       QY
 STS      AE       ST       SS
 TME      AA       TM       TE
 TXT      AH       TX       TT
 VAL      AD       VA       VL
 VNM      AA       VN       VM
                                                          MORE ...
F3=Exit   F5=Refresh   F20=Override
```

This panel is divided into two parts as follows:

- **Change Next Object Prefix for Member Names**—The top section of the panel applies to physical and logical files (FIL) and functions (MSG). The mnemonic for these objects is the system prefix; namely, the OBJPFX model value. This table describes the panel options:

Option		Description
ID Mnemonic	FIL	Last 2-character identifier used for a file name. Use this to determine how close you are to the limit (T9)
	MSG	Last 2-character identifier used for a program name. Use this to determine how close you are to the limit (Z9)

Current Prefix	The current OBJPFX model value.
Next Prefix	The mnemonic to use when the naming algorithm runs out of names. When the name limit is reached for either files or programs, the OBJPFX model value is updated to this value.

Assign the next mnemonic to be used when the automatic naming algorithm runs out of names by typing a two-character mnemonic for the 'Next Prefix' option and press Enter.

When the name limit is reached for either files or programs, the 'ID Mnemonic' is reset to AA for both FIL and MSG objects and the value you entered becomes the new mnemonic when assigning file and program names. The OBJPFX model value is also updated to this value.

- **Change Next Field Type Mnemonic**—The bottom section of the panel applies to field types. This table describes the panel options:

Option	Description
Field Type	Three-character field type code.
Identifying Mnemonic	Last 2-character identifier used when assigning a field name for the corresponding field type. Use this to determine how close you are to the limit of Z9.
Type Code	The current Type mnemonic used for the corresponding field type.
Next Code	The Type mnemonic to use when the naming algorithm runs out of names for the corresponding field type.

Assign the next mnemonic to be used when the automatic naming algorithm runs out of names for a field type by typing a two-character mnemonic for the 'Next Code' option and press Enter.

When the name limit is reached for the field type, the 'Identifying Mnemonic' is reset to AA and the value you entered becomes the new mnemonic when assigning field names for this field type.

Note:

- Use the F20 function key on the YEDTNXTMNC panel with care. Like the Enter key, it updates the YALCVNMRFP file but without performing error checking. It is intended for use only in unusual circumstances; for example, if you need to use the same mnemonic for two different field types.
- Since format names are combined with the associated file name identifier, the two-character identifier for formats is simply reset to AA when the name limit is reached.

Reserved Format Identifiers

CA 2E reserves certain letters to use as field prefixes so that you do not inadvertently duplicate field names in HLL programs. The program you use to allocate format prefixes, Edit Generation Type Details, contains a check to ensure that you do not use the reserved values.

RPG	COBOL/ RPGCBL	Reserved for
P	P	Passed parameter fields
W	W	Program work fields
X	X	User service subroutine fields
Y	Y	CA 2E internal use
U	U	Use in user EXCUSRSRC fields
V	V	External date fields
Z	Z	CA 2E service subroutine fields
#	X	Use by panel fields
$	Z	Use by printer fields
@	Y	Use by special fields

Source File Names

Implementation names should be appropriate for the HLL and DDS source code you are generating.

In the shipped system, new implementation names depend on the YHLLVNM model value; for example *RPGCBL, and the allocation characters for each source type. The Edit Generation Types panel displays a table that provides allocation characters. These characters should be appropriate for your target HLL. To display this table:

1. From the Display Services Menu, select the Display model values (YDSPMDLVAL) option.
2. From the Display Model Values panel, select the Display Name allocation values option.
3. Press F10 to display the Edit Generation Types panel. Modify the allocation character and source file name as necessary.

High Level Language Naming Restrictions

Different CA 2E HLLs impose different restrictions on the names they allow and, therefore, to the names you can give to files, formats, and fields. CA 2E also requires that you reserve the first two characters in field names for a format prefix.

Other restrictions depend on your target HLL. For example, restrictions on length are generally more severe in RPG than in COBOL. However, RPG lets you use some non-alphabetic characters, such as #, &, or @, in names, that COBOL does not allow.

The HLL name validation model value, YHLLVNM, lets you specify that names must satisfy RPG restrictions, COBOL restrictions, or both. If you are generating a database for use with both COBOL and RPG programs, use names that satisfy both sets of restrictions.

The default for the model value YHLLGEN is *RPGCBL. The CA 2E convention using *RPGCBL produces names of 8 or less characters that would be valid names on most platforms.

The following table describes YHLLVNM and HLL name restrictions:

Restricted	DDS	RPG	COBOL	#RPG	#RPGCBL	#CBL
File Name Max Length	10	8	30	8	8	10
Format Name Max Length	10	8	30	8	8	10
Field Name Max Length	10	8	30	2 + 4*	2 + 4*	2 + 4*
Allow #, @, etc.	Yes	Yes	No	Yes	No	No

*Where 2 is the length of the prefix and 4 or 8 is the remainder of the name.

When you create a new model, you set the initial value for the YHLLVNM model value. The default for this model value is the same as the value set for the HLLGEN parameter.

You can change the model value for the YHLLVNM using the Change Model Value (YCHGMDLVAL) command.

Note: If your HLL is COBOL, the first five characters of the name must be unique.

If you have a model originally implemented with model value YHLLVNM set to RPG and you want to regenerate it in COBOL, you must convert the implementation names so that they satisfy COBOL restrictions.

For more information about:

- The YCHGMDLVAL command, see the *Command Reference guide*.
- Converting a model from one HLL to another, see the “Preparing for Generation and Compilation” chapter in the *Generating and Implementing Applications guide*.

More Information

For more information about:

- The YCHGMDLVAL command, see the Command Reference Guide.
- Converting a model from one HLL to another, see the "Preparing for Generation and Compilation" chapter in the Generating and Implementing Applications guide.

Device Field Names

When you create a new model,CA 2E automatically sets up a table with values appropriate for the HLL specified by the YHLLGEN model value. When you implement a CA 2E device function in DDS and an HLL,CA 2E uses the implementation names for the device file formats and fields specified by the table. The Edit Device Generation Details panel provides this table. The allocation characters in the table should be valid for your target HLL. You can change these values.

To display the table:

1. From the Display Services Menu, select the option Display model values (YDSPMDLVAL).
2. From the Display Model Values panel, select the Display Name allocation values option.
3. Press F22. The Edit Device Generation Details panel displays. Modify the GEN name as necessary.

Adopting Naming Conventions for File and Function Design

Name allocation lets you set naming standards at the system level. Naming conventions let you set naming standards at the model level.

All objects originating from an application must have a uniquely identifiable name. Users who add objects to an application should take care that object names follow a set of naming standards. The Copy Model Object (YCPYMDLOBJ) command uses the text description to determine whether to add an object or copy over an existing object.

CA 2E requires that all names you create follow these general rules:

- Names can be up to 25 characters.
- Letters within a name can be both upper and lower case.
- Names can include embedded blanks and special characters.

Fields

Field names should be complete words or abbreviations that are a concise, meaningful definition of the field. Following are some suggestions for creating field names:

- The format should be *object name* + *field type*:
  - The object name should be words or abbreviations separated by spaces.
  - The field type is optional and should describe the type of data stored in the field. The field type is useful when the type is not evident from the object name. For example, Date might be Statement Date.
- Develop a method to capitalize consistently. For example, capitalize the first letter of the first word, or the first letter of each meaningful word.
- Develop standard abbreviations that conform to site vocabulary. For example, you could use "CUST" for customer or "PROD" for product.
- Develop rules for naming common status field types. For example, status fields used as binary flags should be referenced to a single status field with the conditions Yes or No.
- Set up base fields for all common field types and reference these fields by most other field types. This convention helps to ensure uniformity and allows changes to be made more efficiently. To place base fields at the top of the list, preface them with the letter A. Name base fields to reflect the field attributes. For example, A yes or no (Y|N) status or A Text Desc 30.
- When you create the field, add narrative text. Add functional narrative to describe the field to designers, and if applicable, add operational narrative to describe the field to end users. Use a standard template.

  Keep in mind the effect of the narrative on help text. If you enter operational text, it becomes the help text for the field; however, if you enter only functional text, it becomes the help text.
- To facilitate panel generation, set appropriate column headings for each field.
- Where sequencing is necessary, sequence numbers by five.

More Information

For more information about narrative text, see the "Using Your Model" chapter in this guide.

Function Fields

Following are some suggestions for creating function field names:

- Function field names should be the same as the related database field. Prefix USR function fields with a common set of characters and, if necessary, a number to make the name unique. Reference the function field to the same base field as the database field. For example, use Order Amount #. Suffix other function fields according to their type, such as, Order Amount SUM, Order Amount MAX.
- Add functional narrative to describe the field to programmers, and if applicable, add operational narrative to describe the field to end users.
- Develop similar conventions for function fields not related to database fields.

Files

File names should consist of complete words or abbreviations and should provide a concise and meaningful definition of each file.

Following are some suggestions for creating file names:

- Do not use the word file in a file name.
- Name files in the singular, rather than the plural, tense. For example, use Customer rather than Customers.
- Develop a method to capitalize consistently. For example, capitalize the first letter of the first word, or the first letter of each meaningful word.
- Develop standard abbreviations that conform to site vocabulary.
- To avoid truncation of default function names, restrict file names to 18 characters where possible.
- When you create a file, add narrative text. Use a standard template. Keep in mind the effect of the narrative on help text.

Relations

Following are some suggestions for creating relation names:

- Create meaningful names for foreign key fields by adding For Text to ALL relationships in order to define the relationship with the target file.
- When you create a file-to-file relation, add narrative text. Use a standard template. Keep in mind the effect of the narrative on help text.

Access Paths

Following are some suggestions for creating access path names:

- When you add an extra access path, create a meaningful name. Avoid including redundant information. Where possible, indicate selection and sequence details. The name of a span file should indicate the two files the span is over. If necessary, abbreviate the file name rather than reduce other information. For example, Customers Active by Postcode, Orders US by Product.
- It may be useful to set a convention that indicates where a procedure is being followed to control the implementation of joins within separate logical files.

Functions

Function names should consist of complete words or abbreviations and should provide a concise and meaningful definition of each function.

Following are some suggestions for creating function names:

- Develop a method to capitalize consistently. For example, capitalize the first letter of the first word, or the first letter of each meaningful word.
- Number functions that are sequenced to perform a process. Use the number in numeric form. For example, Calc Daily Rate 2.

Database Maintenance Functions:

Following are some suggestions for creating database maintenance function names:

- Use the Process name + Object name format. The process name should describe the type of process the function performs. You may want to abbreviate the name:
- The object name can be a file name, generic data name, or specific data attribute name. The name should conform with other naming standards. To indicate how many objects will be processed, use singular and plural tense.

Versions of Functions and Messages

For versions of functions and messages there is a shipped program for generating names if you specify *GENERATE as the To model object name on the Create Model Version (YCRTMDLVSN) command. You can define your own naming convention by amending the CL program YOBJNAMR1C.

More Information

For more information about versions, see the "Working with Model Objects" chapter in the CA 2E *Generating and Implementing Applications* guide.

Panel Titles

All functions with a device design have a screen title on the header/footer. For *SAA format, this field is fifty characters. For other formats, this field is thirty-seven characters. The default for this field is the function name.

- Expand the default to further describe the function.
- Allow upper and lower case letters.

Condition Names

Condition names are referenced in action diagrams and for selection criteria on active paths.

- Make condition names as meaningful as possible. Do not abbreviate unless absolutely necessary.
- Condition names should be independent of the functions that use them.

Message Names

CA 2E messages consist of three parts:

- Message name
- First level text
- Second level text

Message Name

The message name is unique, 25-character field that identifies the message texts in the CA 2E model. The message name forms the default for the first level text. The message name should indicate the intended first level text and the message type.

First Level Text

The first level text can be up to 76 characters and should be meaningful to the system user. You might want to add variable data to make the message more flexible and expand the user’s understanding.

Second Level Text

Second level text displays when the user presses the Help key while first level text displays. Second level text describes why the message displayed and what actions the user can take. Since many functions may use this text, it should not be specific to the operation.

The following table shows CA 2E message types and their components.

Type	Message Name	1st Level Text	2nd Level Text
ERR	Yes	Yes	Yes
INF	Yes	Yes	Yes
CMP	Yes	Yes	Yes
STS	Yes	Yes	Yes
EXC	Yes	Yes	Yes
RTV	Yes	Yes	Yes

Messages of the EXC type should contain the command they process. For example, SBMJOB Customer Report.

Design Control

A device design is a display format for either a panel or a report that refers to a particular function. Establishing and modifying a device design lets you control the user interface for a function.CA 2E translates device designs into DDS.

Model values set when the model is created determine the initial defaults for device design attributes.CA 2E lets you change these defaults to customize the device design for a particular model.

This section introduces CA 2E design options and standard user interfaces, such as standard function key meanings, line selection values, headers and footers, and screen/report display attributes.

More Information

For more information about device design and procedures to customize the design for your environment, see the CA 2E *Building Applications Guide*.

Design Options

When a model is created, the model value YSAAFMT sets the default display conventions for screen designs within a mode. The default can be one of the following:

- CUA Text Subset
- CUA Entry
- S/38

The display convention determines such aspects of display as:

- Standard user interfaces
- Standard headers and footers
- Screen and report attributes

The following sections discuss these features.

Standard User Interfaces

The generated user interface includes features that affect the appearance of the screens and reports you design and generate with CA 2E. These features include:

- Standard function key meanings
- Standard line-selection value meanings
- Standard headers and footers
- Default screen and report attributes

Model values control these features of the user interface. By resetting model values, you can change from, for example, S/38 to SAA standards. In addition, CA 2E provides programs that let you customize the user interface for a particular application.

Standard Function Keys

CA 2E provides a number of standard function key meanings, such as *Exit, *Delete, or *Next page. Each meaning is assigned a function key. These meanings conform to the design standards. To change the user interface for an application, you reassign the function key and regenerate the code.

Each function key has a specific value out of a list of standard values. The standard function key meanings are controlled by list (LST) conditions attached to the *CMD key field. Each available function key is specified by a value (VAL) condition attached to the same field. Assigning a particular function key value condition to a given list condition assigns it a particular meaning. The CA 2E screen design and generator programs use the assigned value.

The following table shows the shipped function key value defaults.

For...	Command Key	Description
All Programs	F1	Display Help text
	Enter	Validate Input
	F3(Exit)	exit without update
Where appropriate	F12	Previous screen
	F4	prompt
	F5	Reset
	F7,Roll Up	Next page
	F8,Roll Down	Previous page
	F9	Add/Change
	F11	Delete

You can display standard function key meanings by accessing the Edit Field Conditions panel. You can access this panel from an action diagram or from the Edit Field Details panel.

Note: A good practice is to define all function keys referenced in action diagrams as lists (LST) and not values (VAL). You can then change the description of the function key where ever it displays by changing the list condition.

Standard Line Selection Values

CA 2E identifies a number of standard line selection value meanings, such as *Zoom, *Delete, or *Select.

Standard line selection meanings are controlled by list (LST) conditions attached to the *SFLSEL field. Each available option value is specified by a value (VAL) condition attached to the same field. Assigning a particular line selection value condition to a given list condition assigns it to have a particular meaning. The CA 2E screen design and generator programs then use the assigned value.

The length of the *SFLSEL field can be either one or two; it is shipped with a length of one. A designer (*DSNR) can change the model-wide length using the Edit Field Details panel; a developer can override the model-wide length for an individual function on the Edit Screen Entry Details panel.

The Edit Field Conditions panel for the *SFLSEL field shows the standard line selection values. You can display this panel from an action diagram. You can also edit a list condition from this panel.

More Information

For more information about overriding the length of the *SFLSEL field, see the "Modifying Device Designs" chapter in the CA 2E *Building Applications* guide.

Standard Headers and Footers

CA 2E functions Define Screen Format (DFNSCRFMT) and Define Report Format (DFNRPTFMT) usually define the top and bottom lines for device designs. The function options show the name of the standard header function associated with each function. Unless you explicitly name a particular function, this name defaults to the name of the default header function.

You can change the appearance of the header and footer of all your device designs by doing one of the following:

- Change the layout or your default standard header, using the Edit Device Design displays.
- Specify an alternate default standard header.

Specifying the Default Standard Header

If you have several DFNSCRFMT functions, you can control which is the default standard header function by accessing a CA 2E panel, Edit Function Options, from the DFNSCRFMT panel and setting the Use as default header flag. Only one DFNSCRFMT should be flagged as the default.

Panel/Report Display Attributes

Field attributes determine how a field displays in a device design. Attributes include such features as reverse image, highlighting, and underlining.

For each CA 2E field data type, you can set default values for various field properties, such as length and edit codes. Any new fields will be given these values.

You set default values for field properties with the Edit Field Attribute Defaults panel. Some values on this panel are protected, such as the data type. You can change other values. Changing the value does not affect existing fields. The value is the default for any new fields created in the model after the change.

You can change other default attributes at the individual field level with the CA 2E Edit Default Display Attributes (YEDTDFTATR) command.

Environment

CA 2E lets you create objects and generate application systems on an IBM i as if you were in an S/38 environment. Three model values control the environment in which applications are designed, created, and executed:

- YEDTEXC sets the environment within which CA 2E is used.

  This value is determined automatically by the machine, S/38 or IBM i.

- YCRTENV sets the default object creation environment.

  This value determines whether System/38 or native IBM i code is generated; that is, whether machine- specific references in the source should be in IBM i syntax or System/38 syntax. The source attribute, native IBM i or System/38, is set according to this model value.

  This value also controls whether the native or the System/38 environment compilers are invoked to create objects from the generated source.

- YEXCENV sets the environment within which CA 2E applications are to execute. This value determines the default for any machine-dependent syntax that is brought or built at execution time; for example, messages to use in QCAEXEC or QCMDEXEC.

You can change these values with the YCHGMDLVAL command. These values can be either QCL for S/38 or QCMD for IBM i.

More Information

For more information about the YCHGMDLVAL command, see the CA 2E *Command Reference Guide*.

Setting Up Common Routines/Utility Functions

Certain routines and utilities are common to many CA 2E functions. These include:

- Date/Time stamp update for CRTOBJ and CHGOBJ
- Date handling
- Authority level checking
- Report distribution
- Menus
- User defined attributes and mapping functions
- Function fields

IBM supplied programs

Chapter 5: Setting Up a Multi-Modeling Environment

This chapter describes commonly used multi-model structures and provides an overview of the tasks that may be required to set up a multi-model environment. This chapter also describes how to copy an entire model and portions of a model.

This section contains the following topics:

Before You Begin

Multi-model environments requires planning to set up and maintain. Weigh the benefits you expect to receive against the resources you expect to invest.

As a first step, read this chapter. It provides an overview of common multi-model structures and describes the tasks involved in setting up a multi-model structure. If you decide that a multi-model structure is appropriate for your development environment, you may want to get information and advice from other sources.

Multi-Model Structures

The multi-model structure is a modeling environment that allows multiple models to exist in development and production environments. You can copy object definitions from among participating models. And you can create a multi-model structure made up of:

- A **standards model**, which is the foundation for all new models in the environment. This model contains any system-wide standards, such as structure files, field types, and header/footer designs. This model is used to create all other models.
- A **core model**, which contains the database relations of the complete database.
- **Application models**, which contain database relations for a particular application plus functions. Many environments may not need application models. The application models may be integrated into a production model.

In a multi-model structure, object definitions are intended to be shared. The structure can be treated as one model and eventually consolidated. In some environments, database changes to the design may be prevented in the core model.

Considerations

A multi-model structure is designed for development environments in which:

- A large project must be split so that users can work separately.
- A product is being tailored for a client but must be compatible with the standards.
- Each model in the structure represents a phase of development.
- Development is split over IBM i without a remote link.

In deciding whether your development environment would benefit from using a multi-model structure, consider these factors:

- **Development Schedule**—A multi-model structure benefits environments where data design and programming overlap.

  Note: If you are using Change Management (CM), you should consider using Check-out, User capabilities, and locks to achieve control in this kind of environment rather than using multi-models.

- **Application Requirements**—Identify which interfaces to other applications are required; for example, which files must be shared between applications. Identify functions that are common to or must be shared with multiple applications. Loosely integrated or unrelated applications work well in separate models.

- **Organization Requirements**—You may want to separate models for security reasons, or you may want to separate or merge staff responsibilities.

  Note: If you have CA Xtras Change Management, consider partitioning the model with model object lists authorized to groups of staff and further refine access capabilities within just the one model.

If you decide that a multi-model structure is appropriate for your environment, use the requirements you identified to design a structure that meets those requirements. This section describes some common configurations, though other configurations may best meet the needs of your environment.

In addition to identifying requirements, plan how you will maintain the structure you designed. Consider such factors as:

- Standards and procedures:
  - Shared naming library
  - Standardized functions
  - Coding techniques
  - Project time
- Staff needs:
  - Database administration
  - Operations

- System requirements:
  - Hardware
  - JOBQs
  - Subsystems
  - Pool allocations

CA 2E Change Management (CM)

CM, available as a separate product, is a change control system for the IBM i. In a multi-model environment, CM can automate and track the flow of design objects, generated source, and compiled objects between development, test, and production models. In a single model, CM can control versions of objects, user access capabilities, locking of objects, and concurrent development.

More Information

For more information about CM, see the *CA 2E Change Management User Guide.*

Shared Name Environment

CA 2E requires that object names of the same type be unique across a model. In a multi-model environment, where the models will eventually be consolidated, names of different objects of the same type must be unique across all models. Similarly, the names of the same object, both CA 2E names and implementation names, should be the same across all models.

CA 2E's autonaming function ensures that names are unique for all objects in a particular model. A shared name environment ensures that names are unique even when created in different models. You set up a shared name environment when you create the models that make up the multi-model environment.

Note: If you override an automatically allocated name,CA 2E can check only that the new name is unique within the application model. We strongly suggest that you use autonaming in a multi-model environment and rename objects only when absolutely required. Renaming of either the CA 2E name or the object name is the most common cause for the same object having different names in different models.

More Information

For more information about autonaming, see the "Setting Up Your Development Environment" chapter in this guide.

Setting Up a Shared Name Environment

You can set up a shared name environment by creating a separate shared name library or by designating a library in a particular model as the shared name library. Then edit the designated library to include only the names of shared objects listed in the procedure. This section describes how to set up a separate shared name library.

Creating a Separate Shared Name Library

Use the i OS Create Library (CRTLIB) command to create a shared name library that contains the objects required by the name allocation routines.

The shared name library objects should include:

- The logical file YALCVNM00L, which contains the last used sequence values for names of files, programs, and fields, and the model value object prefix. The based-on physical file is YALCVNMRFP.
- Data area YMSGPFXRFA, which contains the message prefix.
- Data area YMSGNBRRFA, which contains the last sequential message number.
- Data area YMSGVNMRFA, which contains the message file name.

More Information

For more information about the CRTLIB command, see Volume 3 of the *i OS CL Reference*.

Advanced National Language Support in a Shared Name Environment

If you are using the National Language Support feature, that is, the model value YPMTGEN is set to *MSGID or *LITERAL, you need to move the data area YPMTNBRRFA into the shared name library so all models can share the naming for message identifiers.

Library List Considerations

For any model that shares the name environment, the shared name environment library name must precede the name of the model in both the interactive and batch library lists. To edit the lists and associated job description for each model, use the Edit Library List (YEDTLIBLST) command and specify the library list for each model (named in the YLIBLST model value).

Object Prefixes

For a shared name environment, we recommend that you assign an object prefix to shared objects. Program YALCVNMR1R assigns the object prefix using the SYS entry in the YALCVNMRFP file. All models in the environment then share the prefix.

Note: The source for the YALCVNMR1R name allocation program is shipped in source file QRPGSRC in library Y2SYSRC.

If you want object prefixes to indicate the model origin of each object, you need to assign prefixes differently. A typical method is to create a new data area for the prefix in each model. Change the YALCVNMR1R program to retain the data area rather than to access the file. The library list then determines the object prefix.

Note: For environments with many objects, you may need to reset the YALCVNMRFP file.CA 2E then starts allocating object names from the next prefix. Make sure you allocate unique prefixes for each model. It is possible to assign duplicate names.CA 2E implements the names until no duplicates exist for the current model, but does not check the models for duplicates.

Common Multi-Model Configurations

This section describes several common multi-model configurations and outlines the steps required to configure each type:

- Database administrator
- Development/test/production
- Split application

Database Administrator Configuration

A database administrator (DBA) configuration is a shared name environment that enables developers to design a data model in one model and design functions in another. All data modeling changes take place in the database design model, and the object definitions are copied to the application model. All procedural coding takes place in the application model. The database design model does not require a generation library.

The DBA configuration generally has two variations:

- **One** database design model with several application models. This configuration may be appropriate when an environment needs a cross-application data model but application-specific programming.
- **Multiple** database design models with a single application model. This configuration may be appropriate in a maintenance environment or in an environment developing interfaces between applications in separate models.

To set up a DBA configuration:

1. Create the required models. Specify the same model values for all models. Designate the database design model and application design models.
2. Using the database design model as the name library, set up a shared name environment. See the Shared Name Environment section in this chapter for more information.
3. Prevent access by programmers to the database design model.
4. Optionally, prevent access by designers to the application models.

You can generate files and access paths using the database design or the application model. You should not generate them from both models. If you use the database design model, any differences between the models tends to become obvious when the functions are created. In this situation, job descriptions for the application models must contain the database generation library. Differences between the model definition of the access path and the actual model can also cause spurious image compare errors during application execution.

You can define additional access paths either in the application or in the database models. Do not define them in both models. Because of the significant impact on performance, ensure that you control the number and types of access paths if you define them in the application model.

More Information

For more information about granting authority to use a model, see the "Using Your Development Environment" chapter in this guide.

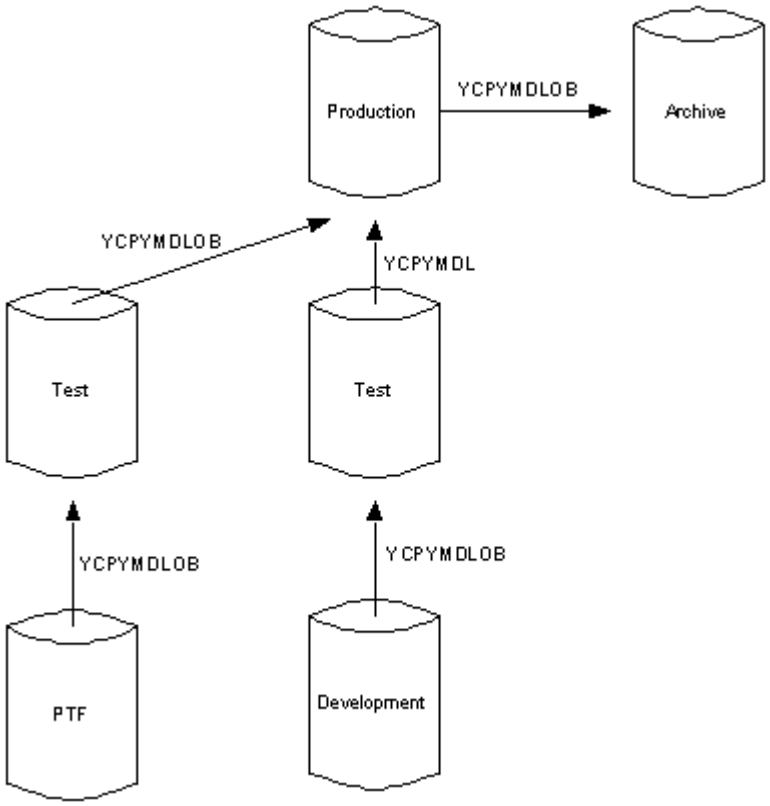
Development/Test/Production Configuration

A development/test/production configuration is a shared name environment that lets you develop in one model and to install production-level definitions in another.

Note: This is the recommended configuration when using CM. CM automates and controls the promotion of objects from development through to production.

You set up a development/test/production configuration as you would set up a DBA configuration except that you:

- Specify a different generation library for each model.
- Use the production library as the shared name library.
- Prohibit access to the production model and place object definitions into it only by copying from the development model using the Copy Model Objects (YCPYMDLOBJ) command.

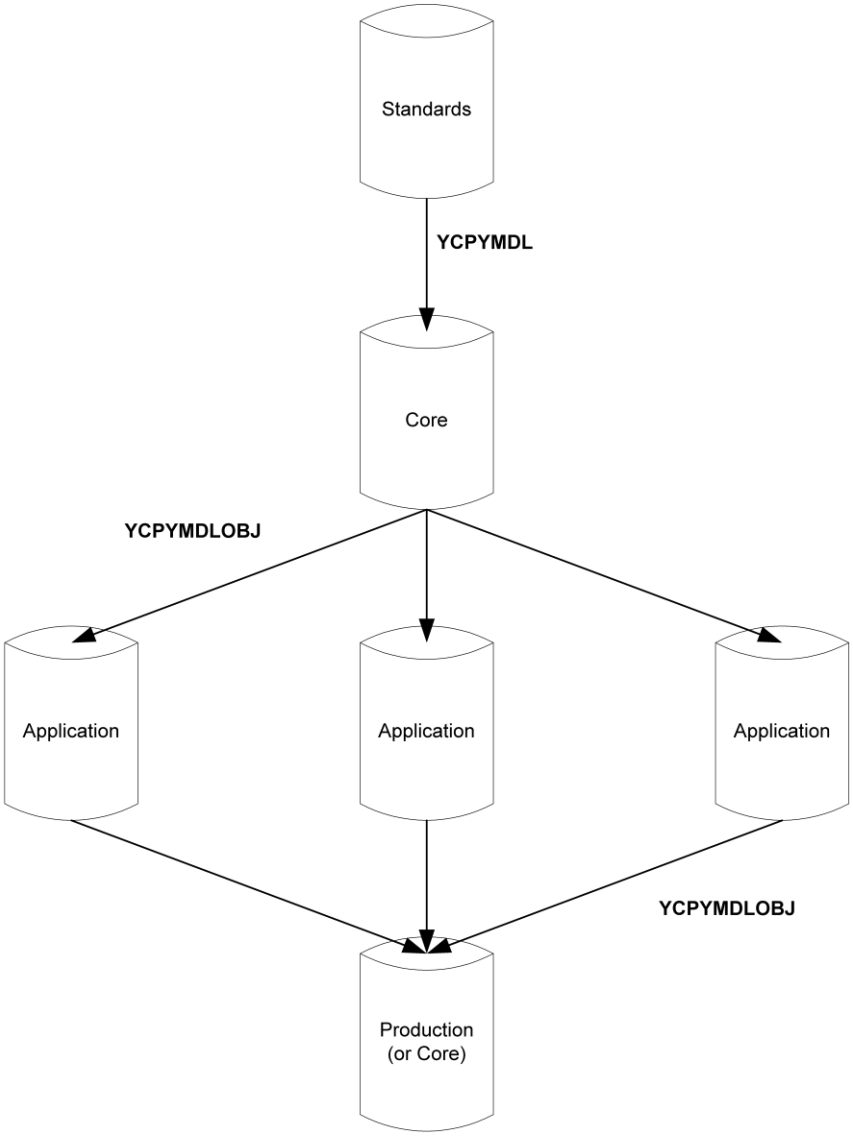


Split Application Configuration

The split application configuration is a hybrid of the DBA configuration. The split application configuration has two model levels:

- A **Core Model level** with the core model that contains the files common to multiple applications. These files are maintained at the core model level and copied to the application models. The core model may have been created from a standards model.
- An **Application Model level** with multiple application models for application-specific DBA configurations.

The split application configuration provides flexibility with control in environments that need application-specific data models.



You set up a split application configuration as you would set up a DBA configuration:

1. Create the core model.
2. Set up the application models.

If you want to specify a generation library that is different from the generation library at the core level, add the core model generation library to the application level library lists after the application model generation library.

Copying a Model

CA 2E provides two methods to copy a model. You can copy part of a model or a complete model.

Copy Part of a Model

To copy only selected objects from one CA 2E data model to another, use the CA 2E Copy Model Objects (YCPYMDLOBJ) command. For example, a company that develops software for its own use might want to apply some changes to the production model before the entire development model is ready to be copied.

You copy model objects by creating a model object list of all objects in the model you want to copy from. You can use the Edit Model List for Copy (YEDTCPYLST) command to perform additional setup tasks related to copying and to view the results of a copy. For a description of this process, see the Copying Part of a Model section in this chapter.

Copying part of a model involves the following steps:

1. Prepare a model object list containing the model objects you want to copy. For example:
   - Use the Build Model List (YBLDMDLLST) command.
   - Use an existing model object list; for example, your session list where CA 2E has automatically logged all changed model objects.
   - Use other model object list commands to create the list, such as the Filter a Model Object List (YFLTMDLLST) command to select objects changed since a certain date.
2. Ensure the list entries for the objects you want to copy are flagged with CPYOBJ flag *SELECTED. You can do this directly from the YBLDMDLLST or YFLTMDLLST commands, with the YCHGMDLLE command, or interactively using the YEDTCPYLST panel (see below).

   Note: To prepare a list for copying you should use only the new model object list commands and/or the YEDTCPYLST command. The Build Copy List (YBLDCPYLST) command is available only for backward compatibility with previous releases of CA 2E and should not be used.
3. Optionally, use the YEDTCPYLST command to invoke the Edit Model Object List for Copy panel. You can specify which objects you want to copy.

   You can assign new object names that will be used both to identify the same object in the target model and to be the name of the copied object in the target model. This step is often required if an object has been renamed that exists in both the target and source models.
4. Use the YCPYMDLOBJ command to copy the CA 2E objects indicated by the edited model object list to another model. You can copy the objects, or you can run a prepass check for discrepancies between the objects in the source and target models.

The remainder of this section discusses this process in more detail.

More Information

For more information about:

- Model object lists, see the "Managing Model Objects" chapter in *Generating and Implementing Applications*.
- Model object list commands, see the CA 2E *Command Reference Guide*.

Copying an Entire Model

When you want to copy everything in a model to another model, use the CA 2E Copy Model (YCPYMDL) command. The YCPYMDL command copies an entire CA 2E model library into a new model or to an existing model. When you copy using this command, you cannot add to the contents of an existing model. The source model overwrites the target model if it exists.

If the target model does not exist, the YCPYMDL command creates the target model when it copies from the source model. Using this method is significantly faster than creating the target model first with the Create Model Library (YCRTMDLLIB) command, then separately executing the YCPYMDL command.

Note: The YCPYMDL command does not create the generation library for the target model.

You can use the YCPYMDL command to create models from your standards model. For example, to copy model MYSTDMDL to a model MYCOREMDL, that is created by the copy, you would enter the following and press Enter:

```
YCPYMDL  FROMMDLLIB(MYSTDMDL) + TOMDLLIB(MYCOREMDL)  CRTOPT(*YES)
```

More Information

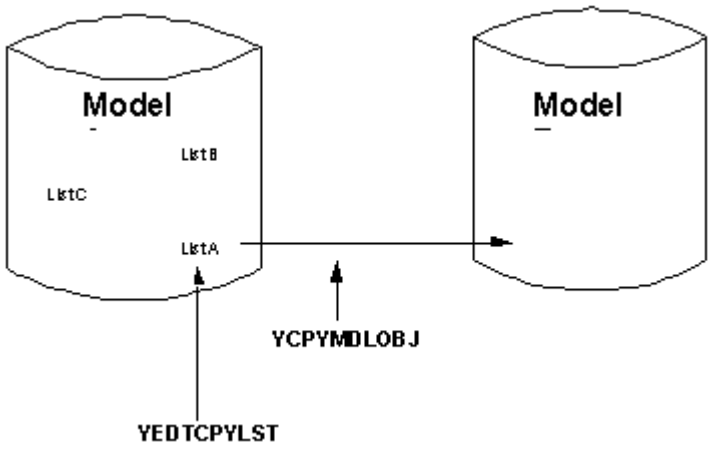
For more information about the YCPYMDL command, see the CA 2E *Command Reference Guide*.

Understanding Model Object Lists

A model object list is a list of CA 2E objects in a CA 2E data model. In a multi-model environment, using a model object list for copying provides a method for you to control which objects are copied to the target model.

Note: The terms *model list* and *copy list* are often used interchangeably; both refer to the same thing.

Model Object List Commands Used for Copying Objects



Before You Copy

Associations between objects in a CA 2E model are complex. Before you copy, you should be aware of how the copy process handles objects associated with objects referenced in a model object list and conflicting versions of the same object in both the source and target models.

Referenced Objects

A model object may have other required or dependent objects that it refers to. These objects will also need to be in the target model for the target object to be complete.

The YCPYMDLOBJ command automatically expands the model object list to include implicitly required (or referenced) objects. When the copy takes place, any implicitly required objects not in the target model must also be copied. You can intervene to examine the expanded list before continuing with the copy, and you can control the expansion of the complex interrelationships between functions using the CPYSUBFUN and EXPRQDOBJ parameters.

Note: You can also use options and function keys from the Edit Model Object List for Copy panel to expand a model object list to include dependent objects (references).

When copying a model object that references other model objects, the command uses existing dependent objects whenever possible. Any object you **explicitly** select is added or replaced. Any object you do not explicitly select, but is **implicitly** required, is copied only if it does not exist in the target model.

The explicit flag shows as an asterix (*) in the Copy Select column on the YEDTCPYLST panel. The implicit flag shows as an exclamation (!). Entries will be flagged as implicitly required during the expansion phase of the YCPYMDLOBJ command.

Note: The expanded list does not automatically include functions of type SELRCD implicitly referenced by other functions.

More Information

For more information about:

- Dependent model objects, see the Edit Model Object Lists and Impact Analysis sections in the "Managing Model Objects" chapter in *Generating and Implementing Applications*.
- The YCPYMDLOBJ command, see the CA 2E *Command Reference Guide*.
- Dependent objects and a table showing model object types and their possible dependent objects, see the CA 2E *Command Reference Guide*, the YDSPMDLREF command.
- Objects selected for copy on the YEDTCPYLST panel, see the Editing the Model Object List for Copy section in this chapter.

Conflicting Object Names Across Models

When YCPYMDLOBJ copies the selected entries to the target it matches objects between the two models by owner, name and type. By default, an object in the source model is copied to the target model using the same object name. You can override this default by specifying a different name for an object in the target model; two reasons you may need to do this are:

- A different object in the target model has the same name as an object in the source model that you want to copy.
- An object in the source model has a different name in the target model.

The YCPYMDLOBJ command includes a prepass option that lets you compare objects by name within object type. A prepass report generates with diagnostic messages that indicate discrepancies. The default selection for this option causes the copy process to stop so that you can resolve discrepancies before you proceed with the copy. If you want the object to be copied but do not want to overwrite the corresponding object in the target model, you can rename it in the model object list. The object is copied into the target model without replacing the existing version.

Note: The object in the source model is **not** renamed.

To rename an object in a model object list:

1. Use selection option 7 from the Edit Model Object List for Copy panel.
2. Enter the name in the Name for purposes of copy field.
3. Press F8 to view a list of all renamed objects.

You can make the rename more permanent by changing the Copy name for an object using the Change Model Object Description (YCHGMDLOD) command; as a result, any new model object lists containing the object will automatically be renamed for purposes of copy.

To avoid problems when copying objects from one model to another, consider the following:

- The YCPYMDLOBJ command can detect that two objects are the same object in both models only if the objects have the same name and type. Remember to rename a shared object in all models where it is defined. Otherwise, the command assumes it is a different object and copies it, creating a new instance of the object.
- When you delete a shared object from one model, remember to delete it from all models where it is defined.
- To avoid overlaying the correct definition for an object, copy the object from its owning model to other models.

- Remember that access path names are reassigned. If you delete or manually rename an access path,CA 2E automatically reassigns the original name when a new access path is created. Make sure you update or delete all references to the original name in the owning or referencing models.

More Information

For more information about renaming objects for copying, see the Editing the Model Object List for Copy section in this chapter.

Building the Model Object List

As the first step in the copy model object process, you need to ensure the objects you want to copy have entries in the model object list. One way to do this is to use the Build Model List command. This section describes how the Build Model List (YBLDMDLLST) command functions and provides an example of how you might use the command.

Note: This step is optional. You can use any model object list as the source for the copy; for example, you could use your session list of changed model objects.

The YBLDMDLLST command builds a list of CA 2E objects in a model, including CA 2E files, fields, access paths, and functions in the source model. The model object list resides in the model library.

For example, to build a model object list, MYMDLLST, of all access paths and functions in the model, MYMDL, you would enter the following and press Enter:

```
YBLDMDLLST MDLLST(MYMDL/MYMDLLST) + OBJNAM((*ANY *ALL *ACP)(*ANY *ALL *FUN))
```

The YBLDMDLLST command includes optional parameters that let you specify how CA 2E should build the model object list and which objects in the model object list should be selected. For example, to automatically select for copying all objects added by the YBLDMDLLST command, specify the OUTCPYOBJ(*SELECTED) parameter.

More Information

For more information about the YBLDMDLLST command, see the CA 2E *Command Reference Guide*.

Editing the Model Object List for Copy

You can edit the model object list using the YEDTCPYLST command. This command invokes the Edit Model Object List for Copy interactive panel. This section describes how YEDTCPYLST functions and provides an example of how you might use the command.

The YEDTCPYLST command lets you select from the model object list the CA 2E objects you want to copy to the target model. Any renamed objects are copied to the target model under their new names.

For example, to edit the model object list, MYMDLLST, in the model SYMDL, you would enter the following and press Enter:

```
YEDTCPYLST  MDLLST(MYMDL/MYMDLLST)
```

The Edit Model List for Copy panel displays.

The explicitly selected objects have an '*' in the Copy Select column. Objects that were implicitly selected during a previous use of the YCPYMDLOBJ command are shown with an '!'. You can explicitly select additional objects for copy using option 1. You can deselect explicitly selected objects from this panel or by using the CPYOBJ parameter of the YCHGMDLLE command. Press F9 to subset the display so only entries that have been selected display.

```
                  Edit Copy List - Selected Entries
                                                          System . : SYNONDV1
  Model . : SYMDL
  List. . : MYMDLLST    List MYMDLLST in SYMDL created by user JAR

Type options, press Enter.
1=Select     4=Delete entry     5=Display              7=Rename
8=Details    9=Deselect        12=Resolve conflicts   19=Work with versions

Opt    Object                       Owner                    Type
     ! Customer phone number                                 FLD
     ! Cusomer status                                        FLD
     * Customers by name            Customer                 ACP
     ! Existing                     Customer status          CND
     ! Former                       Customer status          CND
     ! New                          Customer status          CND
     ! Physical file                Customer                 ACP
     * Retrieval index              Customer                 ACP
     * Update index                 Customer                 ACP
                                                                    More...
F3=Exit       F4=Prompt             F5=Refresh         F6=Build
F7=Position   F8=Display renames    F23=More options   F24=More keys
This is a subsetted list.
```

More Information

For more information about the YEDTCPYLST command, see the CA 2E *Command Reference Guide* and the "Managing Model Objects" chapter in *CA 2E Generating and Implementing Applications*.

Renaming Objects for Purposes of Copy

By default, YCPYMDLOBJ copies an object in the source model to the target model using the Copy name specified in the model object's description. If you need to copy an object in the source model to another name in the target model, you can rename the object in the model object list for purposes of copy. For example, you might need to do this to avoid overwriting an object in the target model that has the same object name and type but a different definition. Note that this does not change the name of the object in the source model.

To rename a model object for a particular copy, use selection option 7 on the YEDTCPYLST panel. Enter the name of the object in the target model in the Name for purposes of copy field.

Note: The object in the source model has a different name in each of several target models, you need to rename the object each time you copy it to a different target model.

Press F8 to view a list of all objects that have been renamed for purposes of copy.

Copying the Model Objects

After editing the model object list, you use the YCPYMDLOBJ command to copy the objects. As a safety check, the target model must be the first model in the library list. The source model must be the second. This section describes how the YCPYMDLOBJ command functions and provides an example of how you might use the command.

The YCPYMDLOBJ command does the following:

1. Expands the model object list. Each of the selected objects in the list is examined for referenced objects that are required for each object to be complete. These implicitly required objects must be copied if they do not exist in the target model.
2. Adds to or updates the referenced objects in the list and flags them as implicitly selected.
3. Checks for discrepancies between the objects.
4. Depending on the option you specify in the CPYOPT parameter, does one of the following:
   - If you specified the prepass check option (*PREPASS), generates a prepass report with discrepancies and stops so that you can resolve them before you continue the copy. Note that only exceptions are reported during this procedure.
   - If you specified the copy option (*COPY) and the procedure does not detect errors, copies the objects in the model object list to the target model and generates a prepass report of warnings and an audit report that lists all objects copied.

For example, to copy model objects from model object list MYMDLLST of the objects in model MYOLDMDL to the model MYMDL, you would enter the following and press Enter:

```
YCPYMDLOBJ  FROMMDLLIB(MYOLDMDL)+ TOMDLLIB(MYMDL) CPYLST(MYMDLLST)
```

This example leaves the CPYOBJ parameter at the default *PREPASS option.

More Information

For more information about the YCPYMDLOBJ command, see the CA 2E *Command Reference Guide*.

Using the Prepass Check Option

The YCPYMDLOBJ command includes a parameter, CPYOPT, with an option that lets you check the objects before they are copied.

The default CPYOPT(*PREPASS**)** option lets you check for discrepancies between the models before copying the objects. The command expands the list of objects specified for copying and compares the selected objects against those in the target model but does not copy the objects. A report generates, which lists any discrepancies between the models. These discrepancies fall into two categories:

- **Warning**—Severity less than 30

  For example, the target model may contain fewer referenced objects than the source model.CA 2E can resolve the discrepancy by replacing the existing object in the target model with the new version from the source model.

- **Error**—**Severity** 30 or greater

  A direct conflict exists between the object in the source model and the one in the target model.CA 2E cannot resolve the discrepancy. You must resolve the conflict before the copy can continue. Errors are identified by the message *ERROR.

All messages reside in the Y2MSG file. You can obtain more information about a message by using the Display Message Description (DSPMSGD) command to examine the second level text for the message.

If the report contains only warnings or no discrepancies, you can rerun YCPYMDLOBJ, using the ***COPY option. You must resolve all errors before you can successfully rerun the copy. Actions you can take include:

- Editing one of the models directly using the Edit Model (YEDTMDL) command.
- Renaming objects using YEDTCPYLST and then copying them into the target model.

To reduce the volume of the report you can use the SEVFLT parameter to omit various categories of warnings.

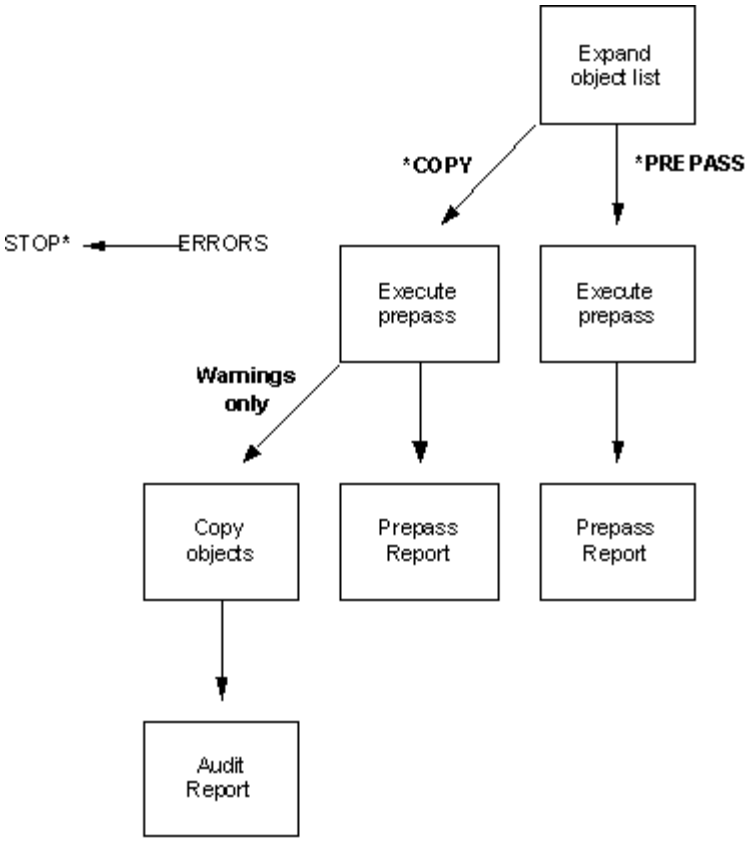
Using the Copy Option

If you select CPYOPT(*COPY**)**, the command expands the list of specified objects, generates the prepass report and, if no errors exist, copies the objects. An audit report generates, which lists each object copied to the target model and details, such as implementation names and any renamed source members. If errors exist, the copy does not take place.

Merging Implementation Names

The YCPYMDLOBJ command includes a parameter that lets you specify whether you want to copy the implementation names of CA 2E objects that exist in the target model. The command always copies implementation names for objects new to the target model.

The YCPYMDLOBJ Command



Appendix A: SQL/DDL Implementation

This appendix contains examples illustrating differences between earlier versions of CA 2E SQL implementations and current implementations. Use these examples as guidelines to understanding how SQL/DDL is implemented in your generated application.

The following SQL/DDL features are covered. Each is discussed in detail in the sections that follow.

- **Extended SQL Naming**—Supports long table and column names.
- **Separate View and Index Creation**—Lets the designer specify whether to generate a view, an index, or both a view and index for an access path.
- **Reduced Number of SQL SELECTs**—Suppresses SELECTs prior to update, delete, or insert.
- **Row Level Locking**—Lets the designer specify what lock capability to use when updating rows.
- **Restrictor and Positioner Functionality**—Lets the designer determine whether to use NOT or OR logic to implement restrictor and positioner capability.
- **Direct Table Access**—Lets the designer specify whether to access data using a view or directly from the table.
- **Cursor Name Length**
- **SQL SELECT in CRTOBJ**

Extended SQL Naming

Prior to Release 5.2 the SQL Data Definition Language (DDL) and Data Manipulation Language (DML) generated for CA 2E used the name allocation mechanisms available for traditional DDS file definition used in RPG- or COBOL-generated function access. Specifically, names were limited to six-character field names by RPG naming rules and to 10-character names by IBM i object naming rules. As a result, names of the SQL tables and columns generated by CA 2E did not allow for meaningful reference to the implemented tables and their associated columns.

Example of Pre-Release 5.2 SQL Naming

The following example shows generated SQL DDL prior to Release 5.2 to create a view and an index for an access path over a table called ORDP.

```
EXEC SQL
       CREATE VIEW RWCSQL.ORDL1 (
                ABABCD
               ,ABADCD
               ,ABACTX
               ,ABABST
               ,ABABDT
                                    )
       AS SELECT
                X1.ABABCD
               ,X1.ABADCD
               ,X2.ACACTX
               ,X1.ABABST
               ,X1.ABABDT
       FROM       RWCSQL.ORDPX1
                   ,RWCSQL.CUSPX2
       WHERE    X1.ABADCD = X2.ACADCD
END-EXEC
EXEC SQL
       COMMENT ON TABLE RWCSQL.ORDL1 IS
'Orders                 Retrieval index'
END-EXEC
```

Understanding Extended SQL Naming

```
EXEC SQL
     COMMENT ON RWCSQL.ORDL1 (
           ABABCD   IS   'Order code'
          ,ABADCD   IS   'Customer code'
          ,ABACTX   IS   'Customer name'
          ,ABABST   IS   'Order status'
          ,ABABDT   IS   'Order date'
                                        )
END-EXEC

EXEC SQL
     LABEL ON TABLE RWCSQL.ORDL IS
         'Orders       Retr'
END-EXEC

EXEC SQL
     CREATE UNIQUE INDEX RWCSQL.ORDL1I ON RWCSQL.ORDP
         (ABABCD        ASC
          )
END-EXEC
```

The optional extended SQL naming feature assigns longer, more meaningful names as follows:

- **SQL Tables and Columns**—These are assigned the 25 character unique file and field names currently found in the model. Since names for CA 2E model files and fields can be any combination of allowable characters and blanks as long as the name is unique within the model, you need to ensure that no illegal characters are used.

  The CA 2E generators map special characters as shown in the following table.

Special Character in Model	Mapped to in the SQL Name
~ ! ‘ % & * ( ) { } | " \ / < > = - + : ; . , ? [ ] ¢ blank	Underscore (_)
Lowercase letters	Uppercase letters

Note: When you generate the SQL/DDL with certain column names, they fail to generate and throw error messages. However, using the same field names for the DDS implementation result in successful generation.

Example: If you have a field starting with a number such as “2nd_Order”, the SQL/DDL fails to generate a table with such column name with the following error:

```
“SQL0103 - Numeric constant 2nd_Order not valid.”
```

However, the DDS implementation with the same field name is successfully generated.

- **SQL Views**—Views are assigned the access path source member name.
- **SQL Indexes**—Index names are assigned at generation time by appending an ‘ to the access path source member name. They are a maximum of ten characters long.

Long Name Scenarios

The following chart depicts the way tables get generated in 2E based on the following three factors and the corresponding impact on the functions.

1. YSQLVNM model values
2. Whether the model file name being generated is a valid system name
3. Whether the file level option “Enhance SQL Naming” flag is set to Y or N

Enhance SQL Name ?	YSQLVNM Value	Model File Name	Table name	Table Implementation Name	Implementation name available for the table?	Table Name used by RLA function	Table Name used by SQL function	Function re-compilation required after enhanced table name?
Y	*LNG	Product	Product_TABLE	UUA2REP	Y	UUA2REP	UUA2REP	N
N	*LNG	Product	Product	UUA2REP	N	UUA2REP	UUA2REP	N/A (*)
Y	*LNG	Purchased_Products	Purchased_Products	UUB2REP	Y	UUB2REP	UUB2REP	N
N	*LNG	Purchased_Products	Purchased_Products	UUB2REP	Y	UUB2REP	UUB2REP	N
Y/N	*SQL	Product	Product	UUA2REP	N	UUA2REP	Product	N
Y/N	*SQL	Purchased_Products	Purchased_Products	N/A	N	UUA2REP	Purchased_Products	N
Y/N	*DDS	Product	N/A	UUA2REP	Y	UUA2REP	UUA2REP	N

Enhance SQL Name ?	YSQLVNM Value	Model File Name	Table name	Table Implementation Name	Implementation name available for the table?	Table Name used by RLA function	Table Name used by SQL function	Function re-compilation required after enhanced table name?
Y/N	*LNF	Product	UUA2REP	UUA2REP	Y	UUA2REP	UUA2REP	N
Y/N	*LNF	Purchased_Products	UUB2REP	UUB2REP	Y	UUB2REP	UUB2REP	N
Y	*LNT	Product	Product_TABLE	UUA2REP	Y	UUA2REP	UUA2REP	N
N	*LNT	Product	Product	UUA2REP	N	UUA2REP	UUA2REP	N/A (*)
Y/N	*LNT	Purchased_Products	Purchased_Products	UUB2REP	Y	UUB2REP	UUB2REP	N

(*) E* line generated in the ACP source

Example:

When the YSQLVNM Model value is set to *LNG or *LNT and the model file name is “Product” and the file level option, Enhance SQL naming flag is set to **Y**, then the table is generated with a suffix “_TABLE” (that is, the table is generated with the name “Product_TABLE”). This table is also accessible using its implementation name, UUA2REP. The functions built over the access paths (Table, views and so on) use the implementation name of the table/views and need not be recompiled when the Enhance SQL Naming flag is switched from **N** to **Y**.

More Information

For more information about:

- Naming SQL indexes, see the next section, "Separate View and Index Creation."
- SQL name conflicts, see the "SQL Name Conflicts" section in this appendix.

YDDLDBA Model Value

The Database Access Method (YDDLDBA) model value specifies a method of accessing the database (RLA or SQL) when a function's Generation Mode option is set to A(ACPVAL) or M(MDLVAL), which resolves to DDL type.

***RLA**

Specifies that the external function generates with RLA access.

***SQL**

Specifies that the external function generates with SQL access.

Note: To generate or regenerate a function with RLA code for DDL database, set the YSQLVNM model value to *DDS or *LNG or *LNT, or *LNF and set the YDDLDBA model value to *RLA.

YSQLVNM Model Value

The SQL Naming (YSQLVNM) model value specifies whether to use the extended SQL naming capability. The valid values are:

***DDS**

Use DDS names. The shipped default.

***SQL**

Use the names of the CA 2E objects in the model.

***LNG**

Use the long names of the CA 2E objects in the model along with the DDS or implementation names.

***LNF**

Use the long field names of the CA 2E objects in the model along with the DDS or implementation names.

***LNT**

Use the long table names of the CA 2E objects in the model along with the DDS or implementation names.

It is up to you when and whether to use extended SQL naming. It is anticipated that most designers will choose to convert all tables, views and indexes, including the functions built over them, as a unit; however, incremental migration is also possible with careful planning.

YSQLFMT Model Value

The Generate SQL RCDFMT clause (YSQLFMT) model value specifies whether the RCDFMT keyword must be generated for SQL tables, views, and indexes. The possible values are *NO, and *YES. The shipped default is *NO.

If YSQLFMT is specified as *NO, the record format is the same as the table, index, or view name (if YSQLVNM (*DDS) is specified) or will be generated by the system (if YSQLVNM(*SQL) is specified). If YSQLFMT is specified as *YES, the RCDFMT value is calculated using the same rules as are used when DDS files are generated.

Note: Irrespective of the value of the YSQLFMT model value and if the generation mode is *DDL, the RCDFMT keyword is generated.

Important!

If YSQLFMT is set to *YES or *NO and a DDL index is generated and created, the index is created with LVLCHK(*NO).

If YSQLFMT is set to *YES and an SQL index is generated and created, the index is created with LVLCHK(*NO).

If YSQLFMT is set to *NO and an SQL index is generated and created, the index is created with LVLCHK(*YES).

If you want to change the LVLCHK attribute of the index to LVLCHK(*YES), the model value YLVLCHK must be set to *YES, and the corresponding index-related access path must be regenerated and re-created. Upon regeneration of the access path, an additional line "Y* CHGLF LVLCHK(*YES)" is generated in the header portion, which informs YEXCSQL to create the corresponding index with LVLCHK(*YES).

Note: If YSQLFMT is set to *YES, YLVLCHK is set to *YES and RUNSQLSTM is used to create an index (SQL or DDL), the index would still be created with LVLCHK(*NO). The current functionality does not cater to the RUNSQLSTM command. Tables and Views (SQL) and Tables (DDL) are created with LVLCHK(*YES) by default, irrespective of the YSQLFMT model value. Therefore, YSQLFMT and YLVLCHK model values have no effect on tables and views regarding the LVLCHK attribute.

YSQLLEN Model Value

The SQL Naming Length (YSQLLEN) model value is a numeric value that controls the length of the extended SQL name. Its maximum value is 25. This model value is used only when YSQLVNM is *SQL.

YSQLLEN lets you trim the SQL name so that it conforms to the parameters of the version of i OS or a third party RDBMS; for example, SQL as supported by i OS permits extended names up to 10 characters prior to V3R1 and supports 30 characters under V3R1 and beyond.

Note: If you are running i OS V3R1 or beyond and set YSQLLEN to a value greater than 10, you also need the QDBRTVSN IBM API in order to successfully submit IBM i physical file access paths for generation and compilation. This API is used to retrieve the ten-byte character name from the long SQL name. If the API is not present, the submission will fail.

YSQLCOL Model Value

The Generate SQL Collection/Library Name (YSQLCOL) model value specifies whether a hard coded SQL Collection/Library name should be generated for tables, indexes and views. The possible values are *YES and *NO. The shipped default is *YES.

If YSQLCOL is specified as *YES, the SQL Collection/Library specified for the YSQLLIB model value is generated into the tables, indexes and views, by default, as is the case now. Subsequently when YEXCSQL is executed to create tables, indexes and views, they are created into the hard coded SQL Collection/Library. If YSQLCOL is specified as *NO, the SQL Collection/Library specified for the YSQLLIB model value is not generated into the tables, indexes and views. However, when YEXCSQL is executed subsequently, the tables, indexes and views are generated into the SQL Collection/Library specified for the YSQLLIB model value.

Note: If YSQLCOL is set to *NO and the access paths are generated, another change that can seen in the source, apart from the absence of hard coded SQL collection/Library name is, the previously generated Z* line "Z* YEXCSQL NAMING(*SQL)" is now generated as "Z* YEXCSQL NAMING(*SYS)".

YLVLCHK Model Value

The Generate IDX with LVLCHK(*YES) (YLVLCHK) model value specifies whether an Index (SQL or DDL), when generated with the RCDFMT keyword in it, is created with LVLCHK(*YES). The possible values are *NO and *YES. The shipped default is *NO.

If YLVLCHK is specified as *NO, then existing defaults around LVLCHK are retained when an SQL or DDL index is generated and created. The existing defaults for the LVLCHK attribute in the case of SQL and DDL are as follows.

- When a table or view is generated and created, it is generated with LVLCHK(*YES), irrespective of the existence of the RCDFMT keyword.
- When an index (SQL or DDL) is generated and created without the RCDFMT keyword, it is created with LVLCHK(*YES).
- When an index (SQL or DDL) is generated and created with the RCDFMT keyword, it is created with LVLCHK(*NO).

If YLVLCHK is specified as *YES (in addition to YSQLFMT set as *YES), upon subsequent generation and creation of an index (SQL or DDL), the index is created with LVLCHK(*YES).

Note: If YLVLCHK is set to *YES (along with YSQLFMT set to *YES), upon re-generation of the access path, an additional line "Y* CHGLF LVLCHK(*YES)" is generated into the header portion. This informs YEXCSQL to create the corresponding index with LVLCHK(*YES). For any other combination of YLVLCHK and YSQLFMT, there is no change to the existing processing.

SQL Name Conflicts

When you use extended SQL naming, it is possible for two or more file or field names to resolve to the same SQL name. Such name conflicts are not automatically resolved; it is your responsibility to ensure that name conflicts do not occur. Two ways name conflicts can occur are:

- Model names contain special characters.

  For example, two fields named "< start'" and "> start'" both resolve to the SQL name "_START'."

- The YSQLLEN model value contains a value less than 25.

  For example, if YSQLLEN is 24, two fields named "is a very long name1'" and "is a very long name2'" both resolve to the SQL name "_IS_A_VERY_LONG_NAME'."

Examples of Extended SQL Naming

The following two examples show the generated SQL DDL and SQL DML that results if you set YSQLVNM to *SQL and YSQLLEN to 25. Note that the view name remains unchanged, but the table name becomes ORDERS (from ORDP) and columns are assigned the names for the corresponding fields in the CA 2E model.

Note: Do not use reserved SQL keywords when selecting names. The name ORDERS is used because ORDER is a reserved SQL keyword and cannot be used as a valid table name.

Example of Extended SQL DDL Naming

```
EXEC SQL
     CREATE VIEW RWCSQL.ORDL1 (
           ORDER_CODE
          ,CUSTOMER_CODE
          ,CUSTOMER_NAME
          ,ORDER_STATUS
          ,ORDER_DATE
                                 )
     AS SELECT
           X1.ORDER_CODE
          ,X1.CUSTOMER_CODE
          ,X2.CUSTOMER_NAME
          ,X1.ORDER_STATUS
          ,X1.ORDER_DATE
     FROM     RWCSQL.ORDERS X1
              RWCSQL.CUSTOMER X2
     WHERE    X1.CUSTOMER_CODE = X2.CUSTOMER_CODE
END-EXEC

EXEC SQL
     COMMENT ON TABLE RWCSQL.ORDL1 IS
          'Orders       Retrieval index'
END-EXEC

EXEC SQL
     COMMENT ON RWCSQL.ORDL1 (
           ORDER_CODE          IS      'Order code'
          ,CUSTOMER_CODE       IS      'Customer code'
          ,CUSTOMER_NAME       IS      'Customer name'
          ,ORDER_STATUS        IS      'Order status'
          ,ORDER_DATE          IS      'Order date'
                                       )
END-EXEC

EXEC SQL
     LABEL ON TABLE RWCSQL.ORDL1 IS
          'Orders             Retr'
END-EXEC

EXEC SQL
     CREATE UNIQUE INDEX RWCSQL.ORDL1I ON RWCSQL.ORDERS
          ORDER_CODE ASC
END-EXEC
```

Example of Extended SQL DML Naming

```
EXEC SQL
     UPDATE ORDL0
          SET     CUSTOMER_CODE  =  :ABADCD
              ,   ORDER_STATUS   =  :ABABST
              ,   ORDER_DATE     =  :ABABDT
     WHERE  (  ORDER_CODE = :ABABCD  )
END-EXEC
```

Impact on Other Areas of the Product

Extended SQL naming affects the following areas of the product:

- **Impact Analysis and Change Type**—Because CA 2E file and field names are used in generated DDL and DML, when you change a field name you now need to regenerate any SQL-implemented functions and access paths that use the field. In other words, the type of change caused by changing a field name becomes *PRIVATE rather than *OBJONLY when you use extended SQL naming.

  Use option 94 (Simulate *PRIVATE change) on the Display Model Object Usages panel to identify the functions and access paths you need to regenerate.

- **National Languages**—Because names of files and fields in non-English models often contain inflections and other "invalid" characters, you might experience difficulty when using extended SQL naming.

  Note: Extended SQL naming is not valid for models using DBCS (double byte character set).

- **DRDA Applications**—Because view names are still limited to ten characters and CA 2E still uses the existing member name in the distributed configuration file table, extended SQL naming does not affect DRDA (Distributed Relational Database Architecture) Applications.

More Information

For more information about change type, impact analysis, and simulating a change, see the CA 2E *Generating and Implementing Applications* guide.

Separate View and Index Creation

Beginning with Release 5.2 you can specify not to generate an index and also retain *IMMED maintenance capability for the access path on the IBM i. Prior to Release 5.2, if you specified *IMMED maintenance for an SQL access path, SQL DDL to create an index was unconditionally generated into the source member of the access path following the DDL for the view.

In the current implementation of SQL you can:

- Suppress the generation of an index by specifying *NONE or blank for the index name on the Edit Access Path Auxiliaries panel.
- Rename an index.
- Generate an index without a view.
- For DDL implementation, *only* an index with the same name as source member name is created and view does not get created because, current implementation of DDL allows database access methods of *TABLE only.
- Auxiliaries are not applicable for DDL type file as Views are not created for DDL type file and *only* Index with the same name as the source member name is created.

A discussion of these capabilities follows.

Suppressing Index Generation

When an access path that is to be implemented in SQL/DDL has *IMMED index maintenance, you can press F7 from the Edit Access Path Details panel to access the Edit Access Path Auxiliaries panel. Previously this panel was available only for query (QRY) access paths.

Note: F7=Auxiliaries is not applicable for the DDL implementation.

```
EDIT ACCESS PATH DETAILS                SYMDL
File name . . . . . . . . . : Customer                      Attribute . : REF
Access path name. . . . . : Retrieval index                 Type. . . . : RTV
Unique or duplicate order : U (U-Unique, F-FIFO, L-LIFO, C-FCFO, ' '-Undefined)
Index maintenance option  : I (I-IMMED, D-DLY, R-REBLD)
Alternate Collating table :
Allow select/omit . . . . :   (S-Static, D-Dynamic, ' '-None)
Generation mode . . . . . : S (M-MDLVAL, D-DDS, S-SQL, L-DDL)
Source member name. . . . : UUADREL1
Source member text. . . . : Customer                  Retrieval index

Data access method. . . . :   (M-MDLVAL, G-DBFGEN, T-TABLE)

      Format      GEN     Format text               Associated
? Seq name        pfx     (Based on file)           Updated access path
    1 FADREA0     AD      Customer                  Update index



SEL: Z-Entries, R-Relations, S-Select/omit, A-Assoc.acps, T-Trim, V-Virtualize
F3=Exit  F7=Auxiliaries  F8=Change name  F20=Narrative
```

Edit Access Path Auxiliaries Panel

Because an SQL index is considered to be an access path auxiliary, this panel provides a method to suppress index generation. To suppress the generation of the SQL index, either enter *****NONE in the name' field or leave it blank. In both cases, the index will not be generated, regardless of the Maintenance Option'. As a result, you can create *IMMED DDS access paths and also suppress generation of an SQL index.

```
EDIT ACCESS PATH AUXILIARIES          SYMDL

File name . . . . . . . . : Customer
Access path name. . . . . : Retrieval index
Source library. . . . . . : SYGEN


SQL access path auxiliaries :
? Src member Type  Text
  UUADREL1   RTV   Customer               Retrieval index


  Index Name Type  Text
  UUADREL1I   IX1  Index SQL DDL LOCATED IN : UUADREL1



SEL: e-STRSEU.
F3=Exit
```

Note: The name' is automatically set to *NONE when the Maintenance Option' is either R or D. You cannot use this facility to create a REBLD or a DELAY DDS access path and also create an SQL index.

For DDL implementation, the index is generated with the same name as source member name.

Auxiliaries are not applicable for DDL type file as Views are not created for DDL type file and *only* Index with the same name as the source member name is created.

Generating an Index Only

You can specify T for the Access Method' option on the Edit Access Path Details panel to generate an SQL index and not a view for a specific access path.

More Information

For more information about the Access Method option, see the Direct Table Access section in this appendix.

Reducing the Number of SQL SELECTs

Before Release 5.2, an SQL SELECT was always performed prior to the insert, update, or delete for the CRTOBJ, CHGOBJ, and DLTOBJ function types. However, where a prior read is not required, not performing the SQL SELECT can result in significant performance gains.

In current releases the SQL SELECT is suppressed for CHGOBJ, DLTOBJ, and CRTOBJ unless any of the conditions shown in the following table are true.

Database Function	Conditions in which the SQL SELECT Is Required	Reason SQL SELECT Is Required
CHGOBJ	Null update suppression is on.	To compare images to see if the record needs to be updated.
	The CHGOBJ is embedded in the DBF Record' user point in the EDTFIL, EDTRCD(1-3), or EDTTRN function types.	To compare images to ensure that the record has not been modified by another user.
	There is action diagram code in any of the user points in the CHGOBJ.	Action diagram code in CHGOBJ usually indicates that changes to the DB1 context are needed prior to update
	There are output parameters specified for the CHGOBJ.	To return data for those fields that are not being updated but merely returned to the calling function.
CRTOBJ	The CRTOBJ is embedded in the DBF Record' user point in the EDTFIL, EDTRCD(1-3), or EDTTRN function types.	To check for a duplicate key and to issue a message to the screen in order to terminate the write attempt.
	There is action diagram code in the if data record already exists' user point	To allow the action diagram logic to be executed.
DLTOBJ	The DLTOBJ is embedded in the DBF Record' user point in the EDTFIL, EDTRCD(1-3), or EDTTRN function types.	To compare images to ensure that the record has not been modified by another user.

Row Level Locking

Prior to Release 5.2, a row to be updated by a CHGOBJ function implemented in SQL was locked at the time the row was updated. As a result, it was possible for the row to be updated by another user between the time it was read (SELECT) and the time it was updated (UPDATE).

The SQL Locking (YSQLLCK) model value in Release 5.2 lets you specify whether a row to be updated will be locked at the time it is read or at the time it is updated. Note that locking the row at the time of the read requires that an SQL cursor be declared, which incurs performance penalties.

YSQLLCK Model Value

The SQL Locking (YSQLLCK) model value lets you specify what level of locking is appropriate for your environment. It has three possible values:

- ***UPD**—Row locking occurs at time of update. This is the shipped default.
- ***FET**—Row locking occurs at time of read. Note that if no SELECT is to be performed, the lock will occur at time of update. See the Reducing the Number of SQL SELECTs section in this chapter for conditions that require a SELECT.
- ***IMG**—Row locking occurs at time of read. This applies only to CHGOBJ's that are embedded in the default DBF Record' user point of standard EDTFIL, EDTRCD(1-3), or EDTTRN functions, which will contain code that performs image compares.

Implementing Restrictor and Positioner Functionality

Restrictor and positioner functionality can be implemented in SQL using WHERE clauses that contain either NOT or OR logic. For example, the two following SQL examples are equivalent: both declare a cursor that satisfies one restrictor and two positioners.

Example of WHERE Clause Containing OR Logic

```
EXEC SQL
     DECLARE MYVIEWCURSOR CURSOR FOR
          SELECT * FROM MYVIEW
               WHERE    KEY1   =   :HOSTVAR1
               AND    ((KEY2   >   :HOSTVAR2     )
               OR      (KEY2   =   :HOSTVAR2  AND
                        KEY3   >=  :HOSTVAR3     ))
          ORDER BY      KEY1   ASC,
                        KEY2   ASC,
                        KEY3   ASC
END-EXEC
```

Example of WHERE Clause Containing NOT Logic

```
EXEC SQL
     DECLARE MYVIEWCURSOR CURSOR  FOR
          SELECT 8 FROM MYVIEW
               WHERE       KEY1   =   :HOSTVAR1
               AND NOT    (KEY1   =   :HOSTVAR1  AND
                           KEY2   <   :HOSTVAR2     )
               AND NOT    (KEY1   =   :HOSTVAR1  AND
                           KEY2   =   :HOSTVAR2  AND
                           KEY3   <   :HOSTVAR3     ))
          ORDER BY         KEY1   ASC,
                           KEY2   ASC,
                           KEY3   ASC
END-EXEC
```

The relative efficiency of these two methods depends on the target RDBMS; for example, Oracle tables with large amounts of data perform better using NOT logic.

YSQLWHR Model Value

The SQL Where Clause (YSQLWHR) model value lets you specify the method that is more efficient for your target database. The two valid values are:

- ***OR**—Generate WHERE clauses using OR logic to implement restrictor and positioner functionality. This is the shipped default.
- ***NOT**—Generate WHERE clauses using NOT logic to implement restrictor and positioner functionality.

The end result of the executing application is the same whether you specify *OR or *NOT for YSQLWHR. All that changes is the structure of WHERE clauses in the generated application. This can result in improved performance depending on the target RDBMS.

Note: The OR and NOT logical operators are used only when there is more than one positioner. For example, if there were two restrictors and one positioner, both *OR and *NOT would generate identical SQL statements.

Direct Table Access

If an access path and the table over which it is based contain the same fields, it is generally more efficient to access data directly from the table rather than through a view. The Database Access Method (YDBFACC) model value and an option on the Edit Access Path Details panel let you specify which method is most appropriate for your environment.

Current implementation of DDL, allows database access method of *TABLE only.

Note: Direct table access is not supported for DRDA (Distributed Relational Database Architecture) Applications.

YDBFACC Model Value

The two possible values for the Database Access Method (YDBFACC) model value are:

***DBFGEN—Access data using a view. This is the default.**

***TABLE—Access data directly from the table rather than using a view.**

Data Access Method Option

You can set the Access Method' option on the Edit Access Path Details panel to specify how data is to be accessed for a specific access path. Specify G to access data using a view; specify T to access data directly from the table; M uses the YDBFACC model value.

```
EDIT ACCESS PATH DETAILS              SYMDL
File name . . . . . . . . . : Customer                      Attritue . : REF
Access path name. . . . . : Retrieval index                 Type . . . : RTV
Unique or duplicate order : U (U-Unique, F-FIFO, L-LIFO, C-FCFO, ' '-Undefined)
Index maintenance option  : I (I-IMMED, D-DLY, R-REBLD)
Alternate collating table :
Allow select/omit . . . . :   (S-Static, D-Dynamic, ' 'None)
Generation mode . . . . . : S (M-MDLVAL, D-DDS, S-SQL, L-DDL)
Source member name. . . . : UUADREL1
Source member text. . . . : Customer                  Retrieval index

Data access method. . . . : T (M-MDLVAL, G-DBFGEN, T-TABLE)

      Format      GEN     Format text               Associated
? Seq name        pfx     (Based on file)           Update access path
    1 FADREA0     AD      Customer                  Update index


SEL: Z-Entries, R-Relations, S-Select/omit, A-Assoc.acps, T-Trim, V-Virtualize
F3=Exit  F7=Auxiliaries  F8=Change name  F20=Narrative.
```

Only functions in which the generation mode is SQL/DDL use this access path option.

Note: If you specify *TABLE as the data access method, be sure that the access path does not contain a join; in other words, be sure it contains no virtual fields. Otherwise, either compile or run time errors may occur.

When you specify direct table access, no SQL DDL is generated to create the view. As a result, you can use this feature to suppress the view and generate SQL DDL for an index only.

Note: Current implementation of DDL, allows database access method of *TABLE only.

More Information

For more information about controlling the generation of SQL DDL of an access path, see the Separate View and Index Creation sections in this chapter.

Cursor Name Length

The suffix used to generate an SQL cursor name in CA 2E applications changed in Release 5.2 from "CURSOR" to "CSR"; for example, the cursor name associated with a view named ORDL1 will be ORDL1XXCSR rather than ORDL1XXCURSOR.

SQL SELECT in CRTOBJ

The SQL SELECT performed in a CRTOBJ function is used to determine if a row currently exists with the key value of the data to be written. Before Release 5.2, the SELECT loaded all host variables belonging to the view as shown in the following figure.

SQL SELECT Before Release 5.2

```
EXEC SQL
     SELECT * INTO
          :ABABCD,
          :ABADCD,
          :ABABST,
          :ABABDT
          FROM ORDL0
          WHERE  (  ABABCD = :ABABCD  )
 END-EXEC
```

In Release 5.2, the SELECT was rewritten as follows to avoid overwriting current values of the host variables belonging to the view.

Current SQL SELECT Implementation

```
EXEC SQL
     SELECT * INTO
          :ABABCD
          FROM  ORDL0
          WHERE  (  ORDER_CODE = :ABABCD  )
 END-EXEC
```

Index

N

O

P

R

S

U

Y