

# CA 2E

## Implementation Guide

r8.5



This documentation and any related computer software help programs (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the documentation for their own internal use, and may make one copy of the related software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the product are permitted to have access to such copies.

The right to print copies of the documentation and to make a copy of the related software is limited to the period during which the applicable license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

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

The use of any product referenced in the Documentation is governed by the end user's applicable license agreement.

The manufacturer of this Documentation is CA.

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

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2009 CA. All rights reserved.

## Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at <http://ca.com/support>.



# Contents

---

## Chapter 1: CA 2E and Application Development

Efficient Design with CA 2E .....	1-1
Objectives .....	1-1
Design Principles .....	1-2
Data-Driven Design .....	1-2
Object-Based Design .....	1-3
Understanding the Application Development Life Cycle .....	1-3
Rapid Prototyping Methodology .....	1-4
Maintaining Applications .....	1-5
Features .....	1-5
Change Control Facilities .....	1-7
Cross-Reference Facilities .....	1-7
Change Management .....	1-7
Technical Documentation .....	1-7
National Language Support .....	1-8
Integrating with Existing Databases and Applications .....	1-9
Getting Help .....	1-9
CA Technology Services: Delivering Business Value On Your Terms .....	1-10
CA Education Services: Ready When You Are .....	1-10
CA: Commitment, Quality, Innovation .....	1-11
For More Information .....	1-12

## Chapter 2: Components and Installation

Product Libraries .....	2-1
Installing CA 2E .....	2-2
Upgrading CA 2E .....	2-2
Development Environment .....	2-3
Development Libraries .....	2-3
The Model Library .....	2-3
Generation Library .....	2-4
SQL Collection .....	2-4

## Chapter 3: CA 2E Quick Tour

Before You Begin .....	3-1
------------------------	-----

---

Development Task Quick Tour .....	3-1
Analysis and Design .....	3-2
Entering the Design Model .....	3-3
The Display Services Menu .....	3-6
Leaving the Design Model .....	3-6
Prototyping an Application .....	3-7
Creating Prototype Panels .....	3-7
Naming Prototype Panels .....	3-8
Using Device Design Animation .....	3-8
Converting a Device Design to Toolkit Panel Designs .....	3-9
Converting Multiscreen Functions .....	3-10
Replacing Toolkit Navigation and Data .....	3-10
Transferring Control to the Toolkit .....	3-11
Working with a Toolkit Panel Design .....	3-11
Editing the Panel .....	3-12
Defining Command Keys .....	3-12
Building a Window Prototype .....	3-12
Building an Action Bar Prototype .....	3-13
Defining Color .....	3-15
Entering Sample Data .....	3-17
Displaying Prototype Panels .....	3-17
Returning to CA 2E .....	3-18
Defining Data Models .....	3-19
Enter Relations .....	3-19
Define Objects .....	3-20
File Entries .....	3-21
Documenting Relations .....	3-22
Field Details and Conditions .....	3-22
Relationship Extension .....	3-25
Virtual Fields .....	3-26
Defining Access Paths .....	3-27
Default Access Paths .....	3-27
View Default Access Paths .....	3-27
Defining Functions .....	3-29
Default Reference File Functions .....	3-30
New Functions .....	3-31
Displaying Functions .....	3-32
Function Options .....	3-33
Linking Functions .....	3-35
Default Panels .....	3-38
Default Action Diagram .....	3-41
Code Generation .....	3-42
Generate Access Paths .....	3-42

---

Generate External Functions Source Code .....	3-45
Program Compilation .....	3-46
Testing Your Programs .....	3-48





# Chapter 1: CA 2E and Application Development

---

CA<sup>®</sup> 2E is an application development tool used with the IBM i server. This dynamic tool provides a complete range of interactive facilities to design, generate, document, and maintain applications software.

CA 2E includes CA 2E Toolkit—a set of implementation support and system utilities incorporating menu creation, object management tools, and a full range of automated technical documentation facilities.

## Efficient Design with CA 2E

CA 2E is an application development tool that allows you to design, develop, implement, and maintain applications software more efficiently, rigorously, and effectively than current third generation methods permit. Using CA 2E, you can create programs without having to know a programming language. These programs build applications that manipulate your data according to your business requirements.

An application development tool is more than just a program generator. It helps you determine how your business or organization can be best represented on a computer with respect to the data you need to store and to the functions that operate upon the data.

The data modeling and procedural specification facilities of CA 2E enable you to do precisely this. In other words, your application is stored as business design objects, such as data definitions, interface designs, and action diagrams.

## Objectives

CA 2E incorporates a number of industry standard methodologies in application development such as entity-relationship modeling and object-based design. Although CA 2E is a modeling and specification tool, your end product is not just a design; it is a working application system.

CA 2E separates the process of design from implementation. In addition to clarity in development, you benefit in the following ways:

- Changes to the specification can be applied automatically throughout the application design

- Improvements or changes to the implementation can be made independently of the specification
- The same design can be used to implement applications for several different operating systems and environments

Using CA 2E, you receive the following benefits:

- Easy design of applications that are closer to your end user's requirements
- Support in designing higher quality applications
- Create applications that are easy to maintain and enhance as your business evolves
- Design applications independently of the language in which you implement them
- Develop systems that make an effective use of the architecture for which they are targeted
- Enable you to design and generate applications multiple times faster than traditional methods

## Design Principles

The approach of CA 2E to the design of application systems embodies important design concepts:

- Implementation independence
- Data-driven design
- Object-based design

Each of these concepts is described briefly in the following sections.

### Data-Driven Design

The prerequisite of a successful system design is a correctly designed database. If the application data can be defined and described in a careful, organized manner, it is possible to streamline the creation of processing functions that work with the information. To produce an adequate database, you need to rigorously analyze the data you want to store in the system. This can be achieved through relational database design.

Traditional application development has often been a process-driven approach in which the structure of the data is imposed by the processing of each program. Updating the database for business environment changes is difficult since manual checking may be required to ensure consistency.

Within a data-driven design, the business model of your application is held in the data model, enabling database integrity from the outset. Changes to this central design automatically cascade throughout the application design.

For more information on data-driven design, see the section Data Modeling in the chapter “CA 2E Quick Tour” of this guide.

## Object-Based Design

One of the key concepts of the i OS operating system is the *object* that actions are performed on. Objects provide:

- A mechanism for modularization. Details of how a process is implemented can be separated from the purpose of the process.
- A way to describe software entities that is easily understood.

At a basic level most languages have a verb-object syntax. Developers tend to think in such terms as Add a Customer or Send an Invoice. CA 2E incorporates this natural language syntax into the design process. The resulting design is expressed in terminology that is easily understood by both the designer and the end users.

The design of CA 2E provides for this object-based approach. This design concept is incorporated in CA 2E where the objects are database entities and the actions are CA 2E functions. These functions specify a system in terms of simple processes operating on entities, independently of how the processes are implemented.

CA 2E focuses on reusable components. Actions and objects are defined once and reused. For example, a subroutine to update a record is defined once. All functions that update records can use this common routine.

## Understanding the Application Development Life Cycle

The design principles CA 2E uses are applied throughout the life cycle of application development, including all of the tasks required to start, complete, and maintain an application. CA 2E supports a structured, data-driven approach in the application development life cycle.

To help eliminate errors, CA 2E provides a structured notation that can be interpreted accurately.

CA 2E captures and describes application specifications that reflect users' requirements. After the requirements are represented by the design model and agreed to by the users, the other tasks in the life cycle use this information to ensure the efficient development of the application.

## Rapid Prototyping Methodology

Rapid Prototyping Methodology (RPM) is one approach to using CA 2E in the application development process. CA 2E RPM is a data-driven, pragmatic approach to rapid application creation that provides a framework for using CA 2E to build quality systems. The CA 2E RPM life cycle is divided into the following four phases:

- Analyze
- Design
- Construct
- Implement

### Analyze

In the analyze phase you can:

- Begin system analysis
- Complete the entity relationship diagram
- Complete the CA 2E data model
- Design primary access paths, function flows, and device designs

### Design

In the design phase you can:

- Continue development and prototype application
- Review with the user and update the design
- Continue function definitions, action diagrams, and complete database design

### Construct

In the construct phase you can generate the final version of the application and complete unit testing.

### Implement

In the implement phase you can:

- Complete integration test, system test, and user acceptance test
- Convert the system

The CA 2E toolset automates many of the traditionally manual, error-prone design and coding processes. The features and capabilities of CA 2E are the driving forces behind CA 2E RPM.

## Maintaining Applications

To modify an application you can change your design model as necessary, revisit any function designs that need to take account of the changes, and regenerate all elements dependent on the changes to the files.

For instance, let us say that you want to add a field to a file. CA 2E will automatically add the field to all the necessary access paths (except for those you have earmarked as being unalterable) and make it available on all of the device designs dependent on the access paths. You can use the panel and report design tools to place the field where you want it on these existing designs. The appropriate logic will be added to the function to handle referential integrity checking where appropriate.

You can assess the implications of the changes using interactive inquiries as well as the model documentation reports.

In most cases, regeneration becomes a routine matter. Using the batch generation facilities of CA 2E, it is possible to have an entire system regenerate itself to take account of a database modification.

For example, when a field is used in a function, the function keeps a reference to the field that is resolved when the function is generated. If you make a change, such as to the length of the field, it is reflected within the function's program logic and panel or report design when you next generate the function. This unique feature of CA 2E allows a massive reduction in the maintenance of CA 2E generated applications.

## Features

CA 2E allows you to design, develop, implement, and maintain applications software for the IBM i.

CA 2E includes the following features:

- Entity Relationship Specification
  - Automatic creation of a data dictionary
  - Support for free-format text to describe any of the design objects for both designers and end users
  - Automatic documentation of the design model
  - Specification of domain checks
- Access Paths
  - Design of logical views to support functions
  - Retrieval of existing i OS database descriptions

- Functions
  - Automatic creation of action diagram and panel or report design
  - Modular (object-based) design
  - Procedural specification through reusable components and action diagrams
- Panels and Reports
  - Automatic design of panel and report layouts combined with interactive device painting facilities
  - Panel design prototyping facilities
  - Automatic provision of SAA Common User Access design standards
- HLL code generation
  - Generation of Data Description Specification (DDS) source or Structured Query Language (SQL) data definition language for the database
  - Generation of source code for panels and reports
  - Generation of source code (RPG or COBOL) for IBM i stand-alone programs
  - Automatic generation of help text
- Other features include:
  - Automatic implementation of object naming standards
  - Automatic implementation of panel design standards
  - Automated change control of design objects, including change tracking, multi-level impact analysis, model object lists, and function versioning
  - Automated documentation of generated systems
  - Cross reference facilities
  - Integration of user source code within generated functions
  - Regeneration of source code if the model is changed with the preservation of user modifications to the processing specifications
  - Integration of the native facilities of i OS for commitment control, journaling, and error handling
  - Generation of code quickly and easily when the design is changed
  - Automatic generation of referential integrity checks and domain checking
- Documentation features include:
  - Online help text
  - Extensive technical documentation

## Change Control Facilities

Change control facilities are a set of features and functions supplied with the CA 2E product for managing design objects. They include:

- Capability of building lists of CA 2E design objects, known as model object lists, for auditing and input to specialized model list commands
- Impact analysis tools, including component change processing
- Model object cross-reference facilities
- Support for versions of functions and messages
- Redirection of functions and messages
- Copying model objects between models
- Centrally recorded model object information

You can use the change control facilities as follows:

- Manage your design objects with specific tools
- Manually control changes to your design model
- Provide an integrated, automated change management solution

## Cross-Reference Facilities

CA 2E has a number of facilities to help you identify dependencies between CA 2E model objects. These can be of use when you need to identify the object affected by a particular change.

## Change Management

Change Management (CM) automates control of changes to CA 2E design objects and generated applications. CM tracks and controls changes from development to production, enabling impact analysis, concurrent development, and rigorous promotion of objects.

For more information on change management, see the *Change Management User Guide*.

## Technical Documentation

You can produce documentation from CA 2E models using the CA 2E documentation commands.

## Documenting a Model

You can invoke the commands using the Display Services Menu, or you can call them directly by using the appropriate CA 2E command. You can use the documentation commands on individual objects or groups of objects such as all functions for a file.

Each of the documentation commands has one or more selection parameters with which you can specify the following:

- Selection criteria to identify the CA 2E objects in the model you want to document
- Print options that allow you to specify what information you want listed for the selected design objects

## Documenting an Application

The application systems generated by CA 2E follow the CA 2E standards for producing self-documenting systems.

It is possible to run the Toolkit documentation utilities on the generated systems to produce documentation. Objects, members, formats, and fields all have descriptive text specified with the TEXT keyword. Generated source code automatically contains Toolkit source directives to specify a title, compiler overrides, and a synopsis.

With the Toolkit utilities you can create summary and cross-reference documentation directly from HLL source and i OS objects.

## National Language Support

National Language Support (NLS) allows you to generate applications in different national languages. NLS is available in two options:

- Generated application
- Advanced

## Generated Application Language Support

Generated application language support allows you to generate application modules or components in a number of languages. CA 2E provides national language versions of the default components of your generated application such as help text and function keys.



## Advanced National Language Support

Advanced NLS allows CA 2E generated applications to be language independent by removing hard-coded constants from panels and reports and placing them in message files. These message files can then be translated into several languages with the appropriate language resolved at execution. This allows a single set of generated programs and panels to support multiple languages at execution time.

**Note:** Executing DBCS and ideographic versions requires a DBCS i OS system.

## Integrating with Existing Databases and Applications

Descriptions of existing i OS physical files can be retrieved into a CA 2E data model using a process called assimilation. You can add extra relations and define access paths and functions for assimilated files in the same way that you would with other model files.

The three possible approaches to assimilating an existing database include:

- **Assimilation as is**—the existing database is preserved exactly as it is.
- **Assimilation with limited modifications**—the data model implied by the database is preserved, but is normalized or corrected where appropriate.
- **Full assimilation**—the data model implied by the existing database is used as a starting point for a new, normalized model.

Assimilating database files allows you to develop new functions and access paths over an existing database defined outside of CA 2E. It also saves you from having to enter all of the field names and attributes into your model again or having to identify individual fields as you use them on a program-by-program basis.

For more information on assimilation, see the section Data Modeling in the chapter “CA 2E Quick Tour” of this guide.

## Getting Help

For online technical assistance and a complete list of locations and phone numbers, contact Customer Support at <http://ca.com/supportconnect>. Customer support is available 24 hours a day, 7 days a week.

For telephone assistance, call:

U.S. and Canada 1-800-645-3042

International (1) 631-342-4683

## CA Technology Services: Delivering Business Value On Your Terms

CA Technology Services is a global organization of highly trained, experienced professionals who are determined to provide you with the technical expertise you need, when and how you need it. From implementing a CA solution to helping you get the most out of the CA technology that you have, CA Technology Services is committed to delivering business value to you on your terms.

Our professionals understand your unique business needs and work closely with you to assess which technology is right for your business. Whether the assignment is large or small or you need a custom, stand-alone, or packaged solution, we tailor our efforts to meet your business demands.

By offering a broad range of flexible services, we help you maximize your investment in our technology, achieve more efficient IT performance, and better manage your infrastructure, security, storage, applications, and data. Such flexibility ensures that you reach your time-to-market goals while improving your business performance.

Why not ask your CA representative for more information about how a CA Technology Services professional can help your organization get the most out of your CA business solutions?

## CA Education Services: Ready When You Are

The goal of CA Education Services is to help you realize the full potential of your CA software investment. To meet this goal, our high-quality instructors strive to understand your specific training requirements, and then deliver the right kind of training when, where, and how you need it.

All CA instructors are fully certified and offer a wealth of hands-on enterprise management experience gained in working with today's largest and most complex businesses. Whether your training is web-based, self-paced, or in the traditional classroom, you always receive the most up-to-date instruction and expertise that is available. The knowledge you gain through training prepares you to successfully leverage the capabilities of your CA software.

Why not ask your CA representative how our training and education programs can help you get more out of your CA business solutions?

## CA: Commitment, Quality, Innovation

For more than a quarter century, CA has been developing and supporting software solutions that are currently used by more than 99 percent of the Fortune 500 companies in more than 100 countries. CA is committed to offering leading technologies in flexible partnerships to help you derive full value from your software investments.

At CA, we are committed to offering simple and meaningful solutions to your complex problems, and to delivering management solutions that offer security, reliability, availability, and performance. We work hard to achieve the highest levels of quality in our solutions to help you meet your changing business needs.

To meet these needs, CA's world-class solutions address all aspects of process management, information management, and infrastructure management with six focus areas:

- Enterprise management
- Security
- Storage
- Portal and business intelligence
- Database management
- Application life cycle management and application development

In addition, our innovative approach to technology is carried over into our innovative business solutions. From a revolutionary new business model to a dedicated customer relationship organization, CA is responding to your changing business needs.

We know what it takes to deliver and support valuable solutions 24 hours a day, 7 days a week, 365 days a year while maintaining the highest standards for quality and innovation:

- We are the first global enterprise software company to meet the exacting standards for worldwide ISO 9002 certification.
- We have earned over 150 patents for innovative software solutions.
- We have the highest caliber software developers and consultants in the industry.

We also know you expect us to stand by our commitments. And we do.

## For More Information

After reading this *Getting Started* guide, you can refer to the numerous resources for additional information.

Your product CD contains instructional documents that showcase the software and provide detailed explanations about the product's comprehensive, feature-rich components.

In addition, you can obtain procedural information and answers to any questions you may encounter by accessing the CA web site at [ca.com](http://ca.com).

# Chapter 2: Components and Installation

---

This chapter describes the libraries that comprise CA 2E and the development environment. This chapter also outlines the installation and upgrade process and introduces the Toolkit prototyping facilities.

## Product Libraries

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

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

The Y2SYSRC library contains the supplied source for CA 2E. The source for the 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 Y2SYMDL library contains the CA 2E null model.

**Note:** At many sites, only Y1SY, Y2SY, Y2SYSRC and Y2SYMDL are listed on the machine. The other libraries are merged into the base product libraries during installation.

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

**Note:** *nll* refers to the national language, for example, Spanish (ESP) or English (ENG).

Ship Name	Library or Folder	Restore Library or Folder
Y1SY	Toolkit base	Y1SY
Y1SYVnll	Toolkit LDOs	Y1SY
Y2SY	CA 2E base	Y2SY
Y2SYVnll	CA 2E LDOs	Y2SY
Y2SYMDL	CA 2E null model	Y2SYMDL
Y2SYCBL	CA 2E COBOL generators	Y2SYCBL
Y2SYRPG	CA 2E RPG generators	Y2SY
Y2SYSRC	CA 2E source	Y2SYSRC
YLUSLIB0	CA 2E Security Library	YLUSLIB0

**Note:** The CA 2E Security Library (YLUSLIB0) must be restored to YLUSLIB0. The YCRTLIB command that runs during product installation automatically updates or creates the YLUSLIB library. For more information on this command, see the Product Licensing steps in the *Installation Guide*.

## Installing CA 2E

For new users, the *Installation Guide* provides detailed information about the entire installation process. These instructions provide the following information:

- Prerequisites that must be fulfilled before you install
- Step-by-step installation procedures
- Description of required post-load activities necessary to customize the product for your system

Before you begin the installation, be sure to read the instructions in this guide to familiarize yourself with the process.

## Upgrading CA 2E

For existing users, new releases may include modifications that must be applied to each design model before you can use new features to work with a model. The Apply Model Change (YAPYMDLCHG) command updates one or all named CA 2E design models with those changes.

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

The *Installation Guide* provides detailed information about upgrading to a new version of CA 2E. Before you begin the upgrade, be sure to read the instructions in this guide to familiarize yourself with the process.

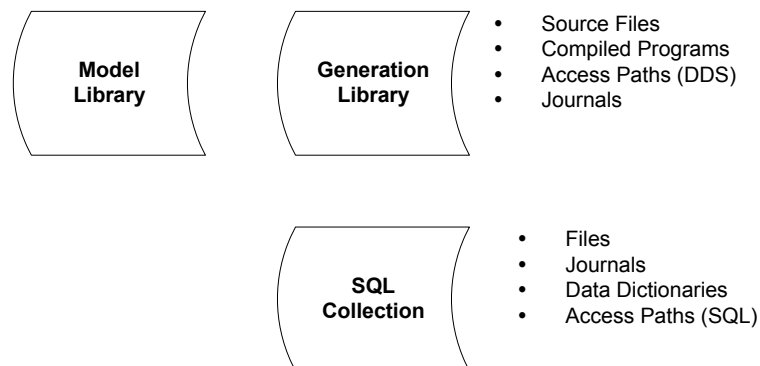
## Development Environment

The CA 2E development environment is the model in which the development team creates the application objects and where initial generation takes place. The development environment is also where objects are tested individually to determine whether they function correctly and efficiently.

### Development Libraries

Each CA 2E model must reside in a single library called the model library. Each model must also have a generation library associated with it. Models and generation libraries may also be associated with SQL collections. The collection is a separate library similar to the generation library.

The following diagram shows the relationship between the model library, the generation library, and the SQL collection:



### 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. The model library and the generation library should be regarded as a pair. The model library also contains other necessary i OS objects and a CA 2E job description. If the developer generates SQL, the model library also refers to SQL collections.

The Create Model Library (YCRTMDLLIB) command creates the libraries that compose a model. For more information about the YCRTMDLLIB command, see the *Command Reference Guide*.

## Generation Library

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

## SQL Collection

CA 2E lets you use SQL in place of or along with DDS in data definition statements. Using SQL, you define tables, views, rows, and columns, rather than IBM i server 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. When you implement 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 CA 2E applications that use SQL, you must have IBM's SQL/400 program products installed and the QSQL library loaded on your IBM i.

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

You can create SQL collections 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.

See the following references for more information:

- For more information on the YCRTSQLLIB and YCHGMDLVAL commands, see the *Command Reference Guide*.
- For more information on the differences between earlier versions of CA 2E SQL implementations and current implementations, see the SQL Implementation appendix in the *Administrator Guide*.



- For more information about IBM's Structured Query Language, see your IBM documentation.



# Chapter 3: CA 2E Quick Tour

---

This chapter provides a quick tour of CA 2E and highlights a number of the features while navigating through typical functions. As you proceed through this chapter you will preview many of the steps in sequence that you perform when you develop your applications with CA 2E.

Use this chapter as an introduction to other detailed CA 2E guides such as the:

- *Installation Guide*
- *Tutorial*
- *Administrator Guide*
- *Building Applications*

## Before You Begin

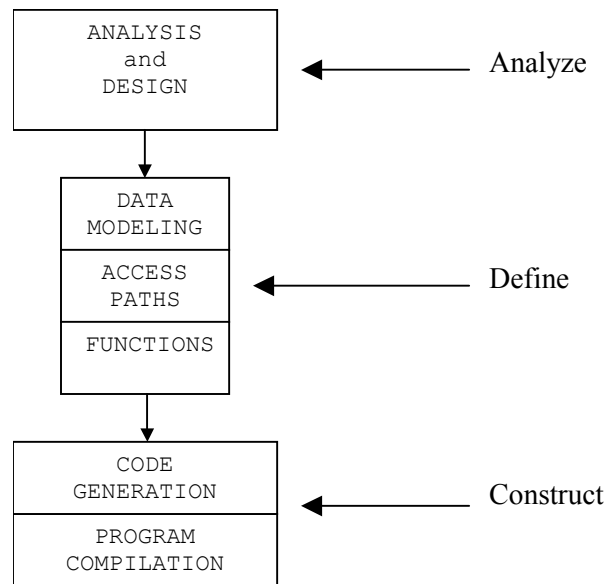
The following conditions must be met before beginning CA 2E development:

- **Installation**—CA 2E must be installed on your IBM i server.
- **Authorization**—you must be signed on an IBM i server with a user profile that is authorized to use CA 2E and perform programming operations.
- **Library List**—your library list must include certain libraries. See the *Installation Guide* for specific details.
- **Design Model**—a design model must be created. You can use the Create Model Library (YCRTMDLLIB) command as follows:

```
YCRTMDLLIB MDLLIB(MYMDL) OBJPFX(MY)+  
SYSTEXT('My Model') DSNSTD(*CUATEXT)
```

## Development Task Quick Tour

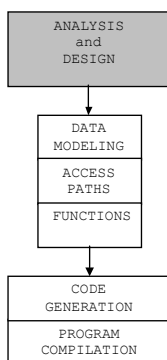
In the development task quick tour of CA 2E, you are introduced to the tasks in the three major phases of development: analyze, define, and construct. The following illustration displays a high level map of the various phases:



## Analysis and Design

Analysis and Design is the first major phase of development in CA 2E. Some of the tasks include:

- Making a design model
- Creating prototype panels
- Using Toolkit panel designs
- Entering sample data



The following three components must be established before you begin:

- Business requirements based on business operations
- Data elements based on the identified business requirements
- Entity relationship diagram (ERD)

Most of this information is obtained through discussion and analysis of the business you are working with.

The following lists show examples of the type of results you might obtain for a horse racing business. See the *Tutorial* for further information.

Business requirements may include:

- A record of the Sire (father) and Dam (mother) for each horse to ensure that they are older than the horse

- A record of horses entered in each race
- A list of all horses including a count and total value, and a list of the races each horse has entered

Data elements may include:

- Course name
- Course location
- Course seating capacity
- Horse name
- Horse age
- Horse gender
- Race name
- Race date

These lists are examined and studied to develop an entity relationship diagram. The entity relationship diagram associates data elements with business requirements so the data can be stored in the most logical and efficient manner.

An entity is a significant item with distinct characteristics. The following table contains an example of some possible entities and their relationships to each other:

Entity	Relationships
Course	Has one or more races
Race	Is at one course Has more than one horse
Horse	Is in one or more races

It is often a good idea to draw a picture of the ERD rather than just listing relationships in a table. See the *Tutorial* for more details on relationships.

## Entering the Design Model

Start the design model facility in one of the following ways:

- Go directly to the CA 2E Designer (\*DSNR) menu using the menu parameter on the Start Y2 (YSTRY2) command from any i OS command entry line as follows:

```
YSTRY2 LIBLST(MYMDL) MENU(DSNR)
```

- Bypass the Designer (\*DSNR) menu and directly enter your model for editing using the Toolkit Change Library List (YCHGLIBL) command and the Edit Model (YEDTMDL) command from any i OS command entry line as follows:

```
YCHGLIBL MYMDL  
YEDTMDL
```

The Y2 command is a short form of the YEDTMDL command.

```
YCHGLIBL MYMDL  
Y2
```

- Enter CA 2E using the Start Y2 (YSTRY2) command, specifying the name of your CA 2E design model as the LIBLST parameter from any i OS command entry line as shown in the following example:

```
YSTRY2 MYMDL
```

```
MAIN                                i OS Main Menu  
                                   System: 2EDV1  
Select one of the following:  
  
  1. User tasks  
  2. Office tasks  
  3. General system tasks  
  4. Files, libraries, and folders  
  5. Programming  
  6. Communications  
  7. Define or change the system  
  8. Problem handling  
  9. Display menu  
 10. Information Assistant options  
 11. PC Support tasks  
  
 90. Sign off  
  
Selection or command  
=> ystrv2 mymdl  
_____
```

---

```
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant F23=Set initial  
          menu
```

Enter **1** in the command entry line as shown in the following example to display the CA 2E Designer (\*DSNR) menu:

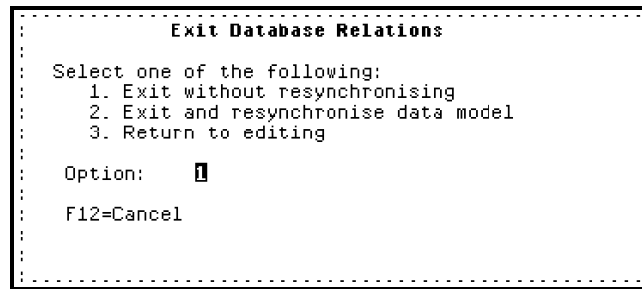
MAIN	2E Main Menu
Level . : 1	System: 2EDV1
Select one of the following:	
Design Model	1. Display Designer (*DSNR) menu 2. Display Programmer (*PGMR) menu 3. Display User (*USER) menu  8. Work with Model Object Lists 9. Change to work with another model
Commands	50. 2E commands in alphabetical order  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	
=> 1	
F3=Major Menu F6=Messages F8=Rev retrieve F9=Retrieve F10=Cmd Entry F24=More	

Enter **1** in the command entry line as shown in the following illustration to enter your design model for editing:

DSNR	2E Designer (*DSNR) Menu
Level . : 1	System: 2EDV1
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	
=> 1	
F3=Major Menu F6=Messages F8=Rev retrieve F9=Retrieve F10=Cmd Entry F24=More	







## Prototyping an Application

You can test the feasibility of your design by building a prototype. The Toolkit prototyping facilities let you interactively present a simulated system design. End-users can preview the prototyped system at a workstation.

Toolkit prototyping includes the following benefits:

- 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
- Link with CA 2E that lets you return to CA 2E after prototyping

The prototyping facilities use the following Toolkit commands:

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

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

- Go to Menu (YGO) displays menus.

The following references provide additional information:

- For Toolkit commands, see the *Toolkit Reference Guide*.
- For details about Toolkit prototyping, see the Design Aids chapter in the *Toolkit Concepts Guide*.

## Creating Prototype Panels

You can create prototype panels in several ways:

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

- Use the CA 2E animate options.

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

Each method converts CA 2E function device designs from a CA 2E design model into Toolkit panel designs. 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 components:

- 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.

## 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. The following provide examples of the naming prototype panels:

- 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 Device Design Animation

CA 2E animation provides a direct link between CA 2E and the Toolkit prototyping facilities. You can interactively simulate your CA 2E system design using the 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 Animate Function Panels opens. A sample panel is shown next.

Animate Function Panels	My Model
Convert Model Panel. : <u>Y</u> (Y-Yes, N-No)	Convert all panels : <u>Y</u> (Y-Yes, N-No)
Replace Navigation : <u>N</u> (Y-Yes, N-No)	
Replace Action Bar : <u>N</u> (Y-Yes, N-No)	
Clear Narrative. : <u>N</u> (Y-Yes, N-No)	
Clear Test Data. : <u>N</u> (Y-Yes, N-No)	
Panel Name(s) . . . . : *SRCMBR	*SRCMBR, *SELECT, *panel, name
File . . . . . : <u>YDSNP</u>	Name
Library. . . . . : *MDLLIB	*MDLLIB, *GENLIB, *LIBL, name
Member . . . . . : *FILE	*FILE, name
Display. . . . . : <u>Y</u> (Y-Yes, N-No)	
Display Option . . : <u>1</u> 1-DSPDTA, 2-DSPATR, 3-CHGDTA, 4-WRKPNL	
Return to this device design . . . . . : <u>N</u> (Y-Yes, N-No)	
Enter=Execute F3=Exit	

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

- Whether to convert the CA 2E device design to a Toolkit panel design
- Animation of Common User Access (CUA) panels, action bars, and window definitions
- Whether to keep or replace information 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 Toolkit commands.

## Converting a Device Design to Toolkit Panel Designs

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, which converts multiple CA 2E device designs to Toolkit panel designs in one step in batch.
- Enter **Y** for the Convert Model Panel option on the panel called Animate Function Panels.

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

Conversion is needed in the following cases:

- A panel design corresponding to your CA 2E device design does not exist in the Toolkit.
- You change the CA 2E device design and need to 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 Multiscreen Functions

By default, if you convert a multiscreen 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 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 entered for the Toolkit panel. The Replace and Clear options let you replace this information. If you are using the 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 the 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 F3 for exit; in other words, F3=\*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 *Command Reference Guide*.

## Transferring Control to the Toolkit

You transfer control to the Toolkit in the following ways:

- Enter **Y** for the Display option on the Animate Function Panels panel.
- If you just want to convert your CA 2E device design to the 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.
- Use the Toolkit YWRKPNL utility and 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 as shown in the following example:

```

YDSCTLR
                                Work with Panel Title Details
Panel. . . . . : UUAJEFR1__  Next panel. . . : _____
Title. . . . . : Edit Customer_____
Print Sequence. : _____
Fixed Header. . : _ (Y, blank)
Fixed Footer. . : _ (Y, blank)
Window. . . . . : _ (Y, blank)
Action Bar. . . : _ (Y, blank)
Option. . . . . : 1 1-Panel, 2-Narrative, 3-Command keys,
                  5-Window, 6-Action Bar
2E related program name . . . . . : UUAJEFR__
F3=Exit

```

You can use this panel to perform the following:

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

The CA 2E Related Program Name option is the source member name of the CA 2E 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, Release 5.0.

**Note:** More than one Toolkit panel may refer to the same CA 2E 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 as shown in the following example. Press 5 to edit or display the window definition.

```

Default Location . . : *CALC ('',*CALC)
      Row. . . . . : 1
      Column . . . . : 2
Window Size
      Height . . . . : 22
      Width. . . . . : 76

F2=Exit F12=Titles screen

```

You can specify the location of the window in one of the following ways:

- Enter specific values for Row and Column.
- Enter **\*CALC** for the Default Location. This causes the upper left 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 or 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 as shown in the following example:

```

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 are displayed. Press F9 to add new choices. Enter **A** to edit the actions for an existing menu choice. The Toolkit Edit Actions panel displays as shown in the following example:

```

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 window displays:

```

+-----+
|                                     |
|                               Edit Command String
|
| Panel name. . . . : UUAGEFR1
| Choice. . . . . : File
| Action. . . . . : Work with spool files
|
| Command string ...
| WRKSPLF _____
|
| F3=Exit  F11=Delete
|
+-----+

```

**Note:** When defining command key navigation, assign \*ABAR to the function key that is to activate the action bar. If you converted a CA 2E 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 the 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, and 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 the 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 window listing available colors and attributes displays:

```

m Blue      r Highlight
m Green     r Reverse Image
m Pink      r Underline
m Red       r Blink
m Turquoise r Column Separator
m White     r nondisplay
m Yellow

F3=Exit F12=Titles screen

```

On some terminals this window will appear as shown in the following example:

```

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 / 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, where: CS means Cursor Select, BL means Blink, UL means Underline, HI means Highlight, RI means Reverse Image, and ND means Non-Display.

X<sup>1</sup> indicates that the green highlight is white.

Color	CLR	CS	BL	UL	HI	RI	ND
Green	GRN					X	
					X <sup>1</sup>		
					X <sup>1</sup>	X	
				X			
				X		X	
				X	X <sup>1</sup>		
							X
White	WHT					X	
Red	RED					X	
			X				
			X			X	
				X			
				X		X	
			X	X			
Turquoise	TRQ	X					
		X				X	
		X		X			
		X		X		X	
Yellow	YLW	X					
		X				X	
		X		X			
							X

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

**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 to enter sample demonstration data, use one of the following options:

- Enter **3** for the Display Option on the Animate Function Panels panel. You can return to CA 2E by pressing the Home key from any Toolkit panel design.
- 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 Animate Function Panels panel. You can return to CA 2E by pressing the Home key from any Toolkit panel design.
- 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 go to the Toolkit, you can return directly to the CA 2E panel or device design from which you started the animation. You return from the Toolkit in one of the following ways:

- If you are editing your Toolkit panel design, press F3 to return to CA 2E. You are provided the 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 the Toolkit to work with your panel design, for example, viewing it, simulating it, or entering data, press the function key assigned for exit (F3 by default) or Home to return to CA 2E.

**Note:** See the documentation for your terminal or computer to learn which key 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 CA 2E.

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 the Toolkit:

- If the value was N, the default, you return to the CA 2E function corresponding to the last Toolkit panel design you accessed. As a result, the corresponding CA 2E function is loaded automatically into Open Functions.

The function loaded is the function whose implementation name appears in the CA 2E 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 CA 2E panel.

- If the value was Y, you return to the CA 2E function from which you invoked the animation.

**Note:** If you invoked the Toolkit using Toolkit commands, you cannot return directly to CA 2E.

Use the following references for more information:

- For information on Toolkit commands, see the Toolkit Reference Guide.
- For information about CA 2E device designs, see the Modifying Device Designs chapter in the Building Applications guide.
- For further details about Toolkit prototyping, see the Design Aids chapter in the Toolkit Concepts Guide.



- You can use the DUP key to reduce the amount of typing. When the cursor is in a field and you press the DUP key, values from the previous line will be duplicated on the line below when you press Enter.
- You only need to type the first letter of a relation; for example, you can enter **K** or **O** in place of Known by or Owned by, respectively.

When you press Enter after entering all the relations you will get a display like the following illustration:

EDIT DATABASE RELATIONS			My model		
=>	Rel lvl:		Seq	Typ	Referenced object
? Typ Object	Relation				
■ FIL Course	Known by			FLD	Course code
— FIL Course	Has			FLD	Course name
— FIL Race	Owned by			FIL	Course
— FIL Race	Known by			FLD	Race date
— FIL Race	Known by			FLD	Race time
— FIL Race	Has			FLD	Race name
— FIL Race	Has			FLD	Going conditions
— FIL Race	Has			FLD	Distance
— FIL Race	Has			FLD	Prize money
—					
—					
—					
—					
—					
—					
Bottom					
Z(n)=Details	F=Functions	E(n)=Entries	S(n)=Select	F23=More options	
F10=Define object	F17=Services			F24=More keys	

## Define Objects

The Define Objects panel shows each of the new objects you entered on the Edit Database Relations panel. Based on the relation statements you entered, each new object is automatically assigned a type and each field is assigned a usage, but you must enter the object attributes.

To display the Define Objects panel press F24 to display more function keys, then press F10 to display the Define Objects panel. A sample panel is shown next:

DEFINE OBJECTS		My model		
Object type	Object name	Object attr	Referenced field	Field Edit usage
FIL	Course			
FLD	Course code			CDE
FLD	Course name			ATR
FIL	Race			
FLD	Race time			CDE
FLD	Race date			CDE
FLD	Race name			ATR
FLD	Going conditions			ATR
FLD	Distance			ATR
FLD	Prize money			ATR
				+

F3=Exit

You can view a list of the available object attributes by typing **?** in the object attribute field.

## File Entries

CA 2E automatically resolves each relation you entered on the Edit Database Relations panel into one or more file entries. You can view how CA 2E has resolved your relations by displaying the entries for the files. Using the Edit File Entries panel from the Edit Database Relations panel, enter **E** next to any of the relations for the COURSE file. A sample panel is shown next:

EDIT DATABASE RELATIONS		My model		
=>	Rel lvl:	Seq	Typ	Referenced object
? Typ Object	Relation			
<b>E</b> FIL Course	Known by		FLD	Course code
FIL Course	Has		FLD	Course name
FIL Race	Owned by		FIL	Course
FIL Race	Known by		FLD	Race date
FIL Race	Known by		FLD	Race time
FIL Race	Has		FLD	Race name
FIL Race	Has		FLD	Going conditions
FIL Race	Has		FLD	Distance
FIL Race	Has		FLD	Prize money

Z(n)=Details   F=Functions   E(n)=Entries   S(n)=Select   Bottom  
 F10=Define object   F17=Services   F23=More options   F24=More keys

Press Enter to display the Edit File Entries panel. This panel shows the file entries that were automatically created for the file:

EDIT FILE ENTRIES		My model					
File . . . . . : Course							
? Field	Type	Ocr	Et	Ksq	GEN name	Length	Renamed
■ Course code	CDE	K	1	ABCD		6	
_ Course name	TXT	A		ABTX		25	

SEL: Z-Details, R-Replace field, U-Usage, M-Mapped field parameters, L-Locks.  
F3=Exit

For each file entry, this panel displays the field names, the field type, whether the field is a key field (K) or an attribute (A), the CA 2E generated implementation name, and the field length. Note that CA 2E assigned the implementation name and default length based on the data type you specified; for example, for fields of type CDE, CA 2E assigns a four-character implementation name ending in CD and a default length of 6.

## Documenting Relations

To obtain a listing of the relations that you have entered, use the Document Model Relations (YDOCMDLREL) command. First press F17 from the Edit Database Relations panel to display the Display Services Menu. Then do **one** of the following:

- Press F9 to display a command entry line, enter the YDOCMDLREL command, and press Enter.
- Select the Documentation menu option to display a list of documentation commands and select Document model relations.

## Field Details and Conditions

Having entered relations to define your data model, you must now add more detailed information about the fields. This includes defining simple validation rules and specifying any required overrides to the defaults CA 2E assigned for field properties such as field length.



Field conditions define both the values that a field may take and the meaning that the values represent.

For example, the Horse gender field of the HORSE file will be used to illustrate conditions. Enter **Z2** against the HORSE Has Horse gender relation as shown in the following example:

EDIT DATABASE RELATIONS		My model	
=>		Rel lvl:	
? Typ	Object	Relation	Seq Typ Referenced object
— FIL	Course	Known by	FLD Course code
— FIL	Course	Has	FLD Course name
— FIL	Horse	Known by	FLD Horse code
— FIL	Horse	Has	FLD Horse name
<b>Z2</b> FIL	Horse	Has	FLD Horse gender
— FIL	Horse	Has	FLD Horse value
— FIL	Horse	Has	FLD Date of birth
— FIL	Jockey	Known by	FLD Jockey code
— FIL	Jockey	Has	FLD Jockey name
— FIL	Jockey	Has	FLD Jockey gender
— FIL	Race	Owned by	FIL Course
— FIL	Race	Known by	FLD Race date
— FIL	Race	Known by	FLD Race time
— FIL	Race	Has	FLD Race name
— FIL	Race	Has	FLD Going conditions

Z(n)=Details F=Functions E(n)=Entries S(n)=Select More...  
 F10=Define object F17=Services F23=More options F24=More keys

## Edit Field Details

Press Enter to display the Edit Field Details panel for Horse gender as shown in the following example:

EDIT FIELD DETAILS		My model	
Field name . . .	: Horse gender	Document'n seq. . .	:
Type . . . . .	: STS	Field usage:	ATR
Internal length. :	1	GEN name:	A05T
	K'bd shift:	Lowercase :	
Headings. . . . . :-		Old DDS name:	
Text . . . . .	: Horse gender		
Left hand side text. :	Horse gender		
Right hand side text. :	Value		
Column headings. . .	: Horse gender		
Control . . . . . :-			
Default condition :	*NONE		
Check condition . .	*NONE		
Valid system name. . :	Mandatory fill . .	Translate cnd values:	
F3=Exit, no update      F8=Change name/type      F10=Appearance      F24=More keys			



The condition value (or Status value) for Stallion is to be represented by an M in the following example:

```

EDIT FIELD CONDITION DETAILS      My model
Field name . . . . . : Horse gender   Attr. : STS   Mode : *ADD
Length on file . . . : 1

Condition. . . . . : Stallion
Type . . . . . : VAL

Status value . . . :  M
File value . . . . :
Mnemonic . . . . . :

F3=Exit

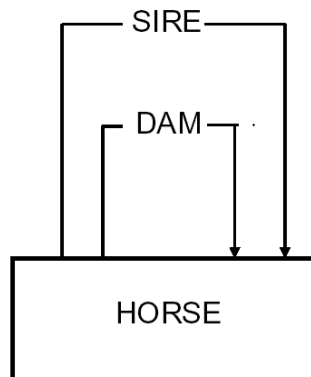
```

After pressing Enter, the panel is redisplayed, with the message "Condition 'Stallion' added" to indicate that the condition has been successfully added.

## Relationship Extension

One file can be associated with another by means of the Refers to relation. A file can also reference another file more than once or to itself. To distinguish among multiple references to the same file, CA 2E provides an extended form of the Refers to relation that lets you specify differences among the relations using For text.

In our example a horse's parents are also horses. To record the horse's parents, two HORSE Refers to HORSE relations—one for each parent—must be established as shown in the following illustration:



Because the HORSE file refers to itself more than once, you will need to use the extended form of the Refers to relation to distinguish between the two relations. For further details on how this is accomplished, see the *Tutorial*.

## Virtual Fields

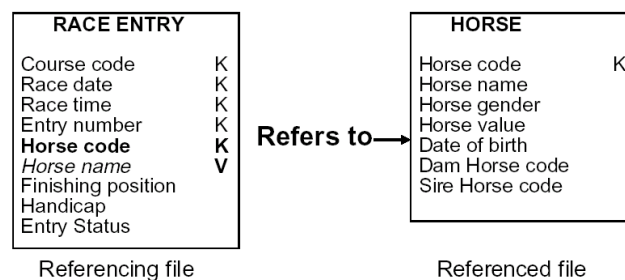
A good database design stores each item of data in only one place, rather than having multiple copies in different files. The i OS join logical file is a mechanism for building views that assemble data from different files.

In CA 2E you use virtual fields to design such views. A *virtual field* is a field that is logically, but not physically, present in a file. CA 2E virtual fields are implemented through i OS join logical files.

When a file is referenced by another file by means of a Refers to relation, entries are automatically created on the referencing file for the key fields of the referenced file. These are known as foreign keys.

To include a non-key field from the referenced file in the referencing file, you need to specify it as a virtual field. This makes the field available for use in the functions that operate upon the referencing file.

For example, to include the name of the horse when displaying the RACE ENTRY file, specify Horse name as a virtual field on the RACE ENTRY file. As a result, Horse name is physically present on the HORSE file, and is logically, but not physically, present on the RACE ENTRY file. This is shown in italics in the following figure:



**Note:** This same capability is available when two files are associated with each other by means of an Owned by relation. In other words, non-key fields from the owning file may be included in the owned file as virtual fields.

For further details on virtual fields, see the *Tutorial*.

## Defining Access Paths

The next stage in defining a design model in the defining stage is to define the required access paths. Access paths specify views of the files in the data model. The views define how data is to be presented to the functions. Functions specify the processes that operate on the data.

Access paths control three different aspects of how data is presented to a function. An access path determines the following:

- Which fields will be available to the function
- Which records will be selected or omitted from the file (select/omit)
- The order in which records are presented (key sequence) to the function

### Default Access Paths

CA 2E uses the relations in the data model to create a default set of access paths for each file. In many cases these will be sufficient. You may override the defaults or define additional access paths.

Three default access paths are automatically created for every file:

- A physical (PHY) access path (unkeyed). It contains the address of all data stored physically within a file and stored in the order in which the data was written to the file.
- An update (UPD) access path (logical view). It is used to update the file and contains all fields defined for the file. It cannot be altered.
- A retrieval (RTV) access path (logical view). It specifies a view of the data that CA 2E generated programs use to retrieve records from a file. Each file has at least one.

### View Default Access Paths

In our example you can enter **Course\*** on the selection line to display only relations for the COURSE file. This selection is shown in the following example:

```

EDIT DATABASE RELATIONS                               My model
=> Course# Rel lvl:
? Typ Object Relation Seq Typ Referenced object
FIL Race Owned by FIL Course
FIL Race Known by FLD Race date
FIL Race Known by FLD Race time
FIL Race Has FLD Race name
FIL Race Has FLD Going conditions
FIL Race Has FLD Distance
FIL Race Has FLD Prize money
FIL Race Entry Owned by FIL Race
FIL Race Entry Known by FLD Entry number
FIL Race Entry Refers to FIL Horse
FIL Race Entry Refers to FIL Jockey
FIL Race Entry Has FLD Finishing position
FIL Race Entry Has FLD Handicap
FIL Race Entry Has FLD Entry Status
Bottom
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 access paths for a file are recorded on the Edit File Details panel. To obtain this panel for the COURSE file, you will zoom against one of the relations for the file on the Edit Database Relations panel. Enter **Z** against one of the relations for the COURSE file as shown in the following illustration:

```

EDIT DATABASE RELATIONS
=> Course*
? Typ Object Rel lvl: Relation Seq Typ Referenced object
2 FIL Course Known by FLD Course code
FIL Course Has FLD Course name

```

Bottom

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

Press Enter to display the Edit File Details panel as shown in the following example:

EDIT FILE DETAILS		My model	
File name . . . . .	: Course		
Attribute . . . . .	: REF	Field reference file.:	*NONE
Documentation sequence. . .	:	Source library. . . . :	MYGEN
GEN format prefix . . . .	: AB	Distributed . . . . .	: N (Y,N)
Assimilated physical. . . .	:		
Record not found message. :	Course	NF	Msgid. : USR0001
Record exists message. . . :	Course	EX	Msgid. : USR0002
? Typ Access path	Source mbr Key	Index options	Auto add
PHY Physical file	MYABREP NONE		ATR ONLY

UPD Update index	MYABRELO	UNIQUE IMMED	ATR ONLY
RTV Retrieval index	MYABREL1	UNIQUE IMMED	ATR ONLY

SEL: Z-Details, G/J-Generate, E-STRSEU, D-Delete, L-Locks, O-Overrides  
H-Hold/Release, T-Trim, V-Virtualize, U-Usage, F-Func refs., N-Narrative  
F3=Exit F7=Funcs F8=Change name F17=Serv. F18=Triggers F20=Narr. F22=Locks

This panel shows the three default access paths that CA 2E automatically created for the COURSE file.

From this point you can establish access paths with various rules, restrictions and limitations to accomplish desirable results such as:

- You can define an access path to select only certain relations from the file or to include selection criteria based on conditions specified for certain fields.
- You can select or omit records from a view based on data values in specified fields. You can specify none, one, or many select/omit sets for a given access path.
- You can define an access path as either static or dynamic. Static access paths are permanently built into the access paths of the i OS logical files so that the logical files only contain the records that meet the selection criteria. Dynamic access paths are not stored in the access path, but are applied to each record as it is retrieved.

For implementation details, see the *Tutorial*.

## Defining Functions

A function defines a list of processes that will operate on the files and fields in your database. CA 2E provides a variety of standard functions based on your data model to which you can add your own unique processing.

A function can be implemented either as a separate program or as a subroutine within a program. An external function is implemented as a separate HLL program. Each external function is attached to an access path. In general, the standard device functions are external functions. An internal function is implemented as a subroutine of the program that calls it.

You can link functions together to create larger processes that become the building blocks of your applications.

In addition to the standard functions, CA 2E provides message functions, function fields, and built-in functions, that perform low-level processing such as arithmetic operations

This is the third task of the defining phase for development.

### Default Reference File Functions

CA 2E automatically creates five default functions for each reference file (REF) you declare: two external functions (Edit File and Select Record), and three internal functions (Change Object, Create Object, and Delete Object).

The two external functions CA 2E automatically creates are:

- Edit File (EDTFIL)—this type of function maintains multiple records on the file.
- Select Record (SELRCD)—this type of function provides a display of the records on the file, and allows the selection of one record for return to a calling program.

Both of these external functions are interactive device functions.

The three internal functions CA 2E automatically creates are:

- Change Object (CHGOBJ)
- Create Object (CRTOBJ)
- Delete Object (DLTOBJ)

These functions are used to update the database and are called by external functions as required. They are implemented as subroutines within the generated source code for an external function. Internal functions are specified separately to give you the capability to add additional processing for updating the database. This additional processing is made available to all the functions that call the internal functions.

The five default functions for a file are created the first time you enter **F** against the file on the Edit Database Relations panel. For example, to create the default functions for the COURSE file you would do the following:

First, display just those relations defined for the COURSE file by entering **Course\*** on the selection line of the Edit Database Relations panel.

Next, display the functions for a particular file by typing **F** against the file on the Edit Database Relations panel. Enter **F** next to any COURSE file relation as shown in the following example:



```

EDIT DATABASE RELATIONS
=>      Course*      Rel lvl:
? Typ Object      Relation      Seq Typ Referenced object
F FIL Course      Known by      FLD Course code
FIL Course      Has      FLD Course name

Bottom
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

```

If you press Enter you can view the five default functions for the COURSE file. CA 2E displays messages at the bottom of the panel to indicate this as shown in the following example:

```

EDIT FUNCTIONS
File name. . . : Course

My model

** 1ST LEVEL **

? Function      Function type      Access path
Change Course   Change object    Update index
Create Course    Create object    Update index
Delete Course    Delete object     Update index
Edit Course      Edit file        Retrieval index
Select Course    Select record     Retrieval index

More...
SEL: Z=Details  P=Parms  F=Action diagram  S=Device  D=Delete  O=Open
      T=Structure  A=Access path  G/J=Generate function  H=Generate HTML ...
F3=Exit  F5=Reload  F7=File details  F9=Add functions  F23=More options
F11=Next View  F17=Services  F21=Copy *Template function

```

## New Functions

To add a new function you must first create it, and then define additional aspects of the function.

You may choose to add functions for the HORSE file, such as Select Mares and Select Stallion. These functions would enable you to display a list of mares only or stallions only in response to an inquiry.

You would begin by viewing a list of possible function types and access paths for the HORSE file by typing **?** in the Function type column, and typing **?** in the Access path column.

Enter the function names, function types, and access paths as shown in the following illustration:

EDIT FUNCTIONS		My model	** 1ST LEVEL **
File name. . . : Horse			
? Function	Function type	Access path	
Change Horse	Change object	Update index	
Create Horse	Create object	Update index	
Delete Horse	Delete object	Update index	
	Retrieval index	Edit Horse	
Select Horse	Select record	Retrieval index	
Select Mares	Select record	Mares	
Select Stallions	Select record	Stallions	
More...			
SEL: Z=Details P=Parms F=Action diagram S=Device D=Delete O=Open			
T=Structure A=Access path G/J=Generate function H=Generate HTML ...			
F3=Exit F5=Reload F7=File details F9=Add functions F23=More options			
F11=Next View F17=Services F21=Copy *Template function			

Press Enter and the functions are added.

The two messages "Function 'Select Mares' is being created" and "Function 'Select Stallions' is being created" will appear briefly at the bottom of the panel. The functions will now be declared to your design model and are available for reference by other functions. At this stage no further definition is required but you will later need to define additional aspects of the new functions before they can be implemented.

Each interactive external function is implemented by:

- An HLL program
- A DDS display file
- A file containing Help text

The implementation language is controlled by the target HLL. You may generate in RPG, COBOL, UNIX, or Client Server Applications depending on your version of CA 2E.

The default implementation language for new functions is controlled by the YHLLGEN model value.

## Displaying Functions

To display the details of the Edit Horse function enter **Z** against the Edit Horse function as shown in the following example:

EDIT FUNCTIONS		My model	** 1ST LEVEL **
File name. . . : Horse			
? Function	Function type	Access path	

Change Horse	Change object	Update index
Create Horse	Create object	Update index
Delete Horse	Delete object	Update index
Z Edit Horse	Edit file	Retrieval index
Select Horse	Select record	Retrieval index
Select Mares	Select record	Mares
Select Stallions	Select record	Stallions

More...

SEL: Z=Details P=Parms F=Action diagram S=Device D=Delete O=Open  
T=Structure A=Access path G/J=Generate function H=Generate HTML ...  
F3=Exit F5=Reload F7=File details F9=Add functions F23=More options  
F11=Next View F17=Services F21=Copy \*Template function

Press Enter to view function details for the Edit Horse function. Note the program (RPG), display file (DDS), and the help text file (UIM) as shown in the following example:

EDIT FUNCTION DETAILS		My model	
Function name . . : Edit Horse		Type : Edit file	
Received by file. : Horse		Acpth: Retrieval index	
Workstation . . . : NPT			
Source library. . : MYGEN			
Object	Source	Target	
? Type	Name	HLL	Text
PGM	MYAEEFR	RPG	Edit Horse
DSP	MYAEEFRD	DDS	Edit Horse
HLP	MYAEEFRH	UIM	Edit Horse
			Edit file
			Edit file
			Edit file

SEL: E-STRSEU, O-Compiler Overrides, T-ILE Compilation Type (\*PGM/\*MODULE)  
F3=Exit F5=Action diagram F7=Options F8=Change name  
F9=Device design F17=HTML skeleton F20=Narrative

## Function Options

You can modify certain aspects of the behavior of a function. From the Edit Functions panel, enter **Z** against the Edit Horse function, as shown in the following example:

EDIT FUNCTIONS		My model	
File name. . . : Horse		** 1ST LEVEL **	
? Function	Function type	Access path	
Change Horse	Change object	Update index	
Create Horse	Create object	Update index	
Delete Horse	Delete object	Update index	
Z Edit Horse	Edit file	Retrieval index	
Select Horse	Select record	Retrieval index	
Select Mares	Select record	Mares	
Select Stallions	Select record	Stallions	

More...

```

SEL: Z=Details  P=Parms  F=Action diagram  S=Device  D=Delete  O=Open
      T=Structure  A=Access path  G/J=Generate function  H=Generate HTML ...
F3=Exit  F5=Reload  F7=File details  F9=Add functions  F23=More options
F11=Next View  F17=Services  F21=Copy  *Template function
Function 'Edit Horse' has been saved.

```

Press Enter to display the Edit Function Details panel as shown in the following example:

EDIT FUNCTION DETAILS				My model	
Function name . . : Edit Horse				Type : Edit file	
Received by file. : Horse				Acpth: Retrieval index	
Workstation . . . : NPT					
Source library. . : MYGEN					
Object	Source	Target			
? Type	Name	HLL	Text		
PGM	MYAEEFR	RPG	Edit Horse		Edit file
DSP	MYAEEFRD	DDS	Edit Horse		Edit file
HLP	MYAEEFRH	UIM	Edit Horse		Edit file
SEL: E-STRSEU, O-Compiler Overrides, T-ILE Compilation Type (*PGM/*MODULE)					
F3=Exit		F5=Action diagram		F7=Options	
F9=Device design		F17=HTML skeleton		F20=Narrative	

Press F7 to view the Edit Function Options panel. The actual function options available and their defaults depend on the function type.

In this example the following options can be modified:

- **Create record** —this option specifies whether you can create database records. The default is Y, which means you can add database records. If you change this value to N, the function code will not include logic to add database records.
- **Change record** — this option specifies whether you can change database records. The default is Y, which means you can add database records. If you change this value to N, the function code will not include logic to add database records.
- **Delete record**—this option specifies whether you can delete database records. The default is Y, which means you can delete database records. If you change this value to N, the function code will not include logic to delete database records.

- **Dynamic program mode**—this option specifies whether the program will dynamically change the program mode between Add and Change modes, depending upon whether data exists or not. If Dynamic program mode is set to Yes, then at runtime if there are records to be displayed the program mode will be Change and records will be shown. If no records exist, then program mode will be Add and blank lines for adding records will be shown. If Dynamic program mode is set to No, then at runtime the program mode will be Add and the user will have to use F9=Change to see any existing records.
- **Subfile selection**—this option specifies whether there will be subfile select options available within the program or not. If this value is Yes, then the screen will have a column for subfile selection options and allow for subfile option text. If Subfile selection is set to No, then the column for subfile selection options and the subfile option text will not display on the screen.

**Note:** If Delete record is allowed, then Subfile selection must be set to Yes.

- **Confirm prompt**—the Confirm prompt option specifies whether the Edit Horse function prompts for confirmation before updating the database files. The default is Y.

If the Confirm prompt option is Y, a related option, Initially Yes, specifies whether the default for the confirm prompt will be Y or N.

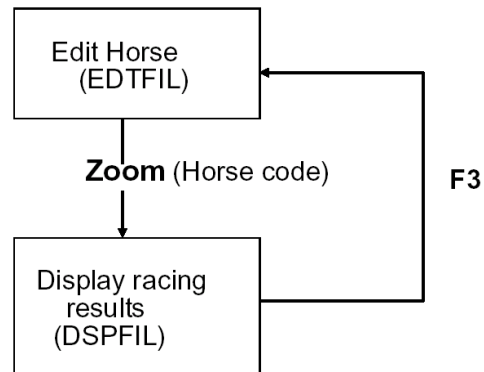
**Note:** To see full details on other possible function options, see the chapter “Modifying Function Options” in *Building Applications*.

## Linking Functions

To link two functions together to build a larger function you must define the second function and an access path on which it is based.

For example, you could modify the Edit Horse function so the user can zoom against the detail line for a particular horse on the Edit Horse panel to view a subsidiary display of the races in which the horse has raced.

The following illustration shows two functions linked together:



To achieve this, you need to define a new function that you might name Display Racing results. CA 2E lets you define the new function while you are in the process of modifying the action diagram of the existing function.

### Specifying Function Parameters

This section shows how to specify the parameters that will be passed to the Display Racing results function.

From the Edit Functions panel, enter **P** to edit the parameters as shown in the following example:

```

EDIT FUNCTIONS                               My model
File name. . . . : Race Entry                ** 2ND LEVEL **

? Function      Function type      Access path
Change Race Entry  Change object   Update index
Create Race Entry  Create object   Update index
Delete Race Entry  Delete object   Update index
P Display Racing results  Display file  Races for a Horse

SEL: Z=Details  P=Parms  F=Action diagram  S=Device  D=Delete  O=Open
      C=Copy  L=Locks  U=Usages  R=References  N=Narrative .. Y=Y2CALL
F3=Exit  F5=Reload  F7=File details  F9=Add functions  F23=More options
F11=Next View  F17=Services  F21=Copy *Template function
More...
  
```

Press Enter to display the Edit Function Parameters panel as shown in the following example:

```

EDIT FUNCTION PARAMETERS                     My model
Function name. . : Display Racing results   Type : Display file
Received by file : Race Entry               Acpth: Races for a horse
                                           Passed      Pgm      Par
? File/*FIELD      Access path/Field/Array  Seq      Ctx      Ctx
Race Entry          Races for a horse       KEY
  
```

```

                                |
                                Values
                                FLD: One parameter per field
                                RCD: One parameter for all fields
                                KEY: One parameter for key fields only

SEL: Z-Parameter details  X-Object details  D-Delete parameter  N-Narrative
F3=Exit    F5=Reload    F22=File locks

```

Enter the details as shown in the following example. In other words, enter **Race Entry** as the file name, enter **Races for a Horse** as the access path, enter **KEY** as the Passed as value, and enter **Z** in the Subfile selector to select the key fields on the access path to be used as parameters.

```

EDIT FUNCTION PARAMETERS                               My model
Function name. . : Display Racing results              Type : Display file
Received by file : Race Entry                         Acpth: Races for a horse
                                                    Passed          Pgm  Par
? File/*FIELD          Access path/Field/Array      Seq      Ctx  Ctx
Z Race Entry           Races for a horse             KEY
                                                    |
                                                    Values
                                                    FLD: One parameter per field
                                                    RCD: One parameter for all fields
                                                    KEY: One parameter for key fields only

SEL: Z-Parameter details  X-Object details  D-Delete parameter  N-Narrative
F3=Exit    F5=Reload

```

Press Enter to display the Edit Function Parameter Details panel.

The Horse code from the selected subfile record is to be passed as an input parameter to the Display Racing results function and will be restricted to show only race entries for this horse. This is where you specify that Horse code is an input only, restrictor parameter.

Enter **R** against Horse code as shown in the following example:

```

EDIT FUNCTION PARAMETER DETAILS      My model
Function name. . : Display Racing results  Type : Display file
Received by file : Race Entry             Acpth: Races for a Horse
Parameter (file) : Race Entry             Passed as: KEY
? Field                               Usage  Role  Flag error
R Horse code
■ Race date
- Race time

SEL: Usage: I-Input, O-Output, B-Both, N-Neither, D-Drop.
      Role: R-Restrict, M-Map, V-Vary length, P-Position. Error: E-Flag Error.
F3=Exit

```

Press Enter.

Note that the Usage for Horse code defaulted to I (Input only) and the parameter role changed to RST (restrictor).

```

EDIT FUNCTION PARAMETER DETAILS      My model
Function name. . : Display Racing results  Type : Display file
Received by file : Race Entry             Acpth: Races for a Horse
Parameter (file) : Race Entry             Passed as: KEY
? Field                               Usage  Role  Flag error
■ Horse code                           I      RST
- Race date
- Race time

SEL: Usage: I-Input, O-Output, B-Both, N-Neither, D-Drop.
      Role: R-Restrict, M-Map, V-Vary length, P-Position. Error: E-Flag Error.
F3=Exit

```

## Default Panels

CA 2E automatically creates a default interactive panel design for each interactive device function. You can modify the default designs to meet the requirements of your application.

A device design specifies:

- The selection of fields present on a device panel and the accompanying text for each field



- The display of both fields and text may be conditional
- The order in which the fields are displayed on the device panel and how they are edited
- The display attributes for the fields
- The use of each field

These are the defaults from the design model; however, you may override most of them.

You can view the default device design from the Edit Function Details panel by pressing F9 as shown in the following example:

EDIT FUNCTION DETAILS				My model	
Function name . . : Edit Horse				Type : Edit file	
Received by file. : Horse				Acph: Retrieval index	
Workstation . . . : NPT					
Source library. . : MYGEN					
Object	Source	Target			
? Type	Name	HLL	Text		
PGM	MYAEEFR	RPG	Edit Horse	Edit file	
DSP	MYAEEFRD	DDS	Edit Horse	Edit file	
HLP	MYAEEFRH	UIM	Edit Horse	Edit file	
SEL: E-STRSEU, 0-Compiler Overrides, T-ILE Compilation Type (*PGM/*MODULE)					
F3=Exit		F5=Action diagram		F7=Options F8=Change name	
F9=Device design		F17=HTML skeleton		F20=Narrative	

The default panel design for the Edit Horse function displays as shown in the following example:

File fUnction Selector Help						
*PROGRAM		*PGMMOD		DD/MM/YY HH:MM:SS		
Edit Horse						
Horse code . _____						
Select items, then select an action.						
Opt	Horse code	Horse name	Horse gender	Horse value	Date of birth	Dam Hors code
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
-	_____	_____	-	_____	_____	_____
F3=Exit F5=Reset F9=Open F10=Actions						
Design exceeds device size limits for function Edit Horse						

If you want to change the position of a field on the device design, position the cursor on the chosen field and press the F9 function key. For example, to begin the second line with Dam Horse code, you just place the cursor on this field and press F9.

The subfile record format splits into two lines and the device design is updated to reflect the change.

Position the cursor as shown in the following example:

```

File  fUnction  Selector  Help
-----
*PROGRAM  *PGMMOD                                DD/MM/YY HH:MM:SS
                                Edit Horse
Horse code . _____
Select items, then select an action.

Opt  Horse  Horse name                Horse  Horse value  Date of  Dam Hors
code  code                                     gender  birth      code
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -
-    -    -    -    -    -    -    -

F3=Exit  F5=Reset  F9=Open  F10=Actions
Design exceeds device size limits for function Edit Horse

```

Press F9. The subfile record format now displays only two HORSE records as shown in the following example:

```

File  fUnction  Selector  Help
-----
*PROGRAM  *PGMMOD                                DD/MM/YY HH:MM:SS
                                Edit Horse
Horse code . _____
Select items, then select an action.

Opt  Horse  Horse name                Horse  Horse value  Date of
code  code                                     gender  birth
-    -    -    -    -    -    -
-    -    -    -    -    -    -

Dam Horse  Dam name                Dam Date  Sire Horse  Sire name
code      code                                     code
-    -    -    -    -    -    -
-    -    000000000000000000000000  66/66/66  -    000000000000000000000000
-    -    -    -    -    -    -

Dam Horse  Dam name                Dam Date  Sire Horse  Sire name
code      code                                     code
-    -    -    -    -    -    -
-    -    000000000000000000000000  66/66/66  -    000000000000000000000000

F3=Exit  F5=Reset  F9=Open  F10=Actions
Design exceeds device size limits for function Edit Horse

```

You can repeat the process to move the fields relevant to the Sire to a separate line from those for the Dam by placing the cursor on the Sire Horse code and pressing F9.

For more details on modifying the device design, see the *Tutorial*.

## Default Action Diagram

CA 2E supplies a default action diagram for each function. An action diagram contains the processing steps that make up a function. Each action diagram consists of a list of actions, where each action may be either a call to another function or one of a number of low level built-in functions.

To view and edit the default action diagram for the Edit Horse function, from the Edit Function panel enter **F** in the Subfile selector for the Edit Horse function as shown in the following example:

```

EDIT FUNCTIONS                                     My model
File name. . . . : Race Entry                      ** 1ST LEVEL **

? Function      Function type      Access path
Change Race Entry  Change object    Update index
Create Race Entry  Create object     Update index
Delete Race Entry  Delete object     Update index
F Display Racing results  Display file  Races for a Horse

SEL: Z=Details  P=Parms  F=Action diagram  S=Device  D=Delete  O=Open
      C=Copy  L=Locks  U=Usages  R=References  N=Narrative ...Y=Y2CALL
F3=Exit  F5=Reload  F7=File details  F9=Add functions  F23=More options
F11=Next View  F17=Services  F21=Copy *Template function
More...

```

Press Enter to view the action diagram for the Edit Horse function. An example is shown next:

```

EDIT ACTION DIAGRAM      Edit      My model      Horse
FIND=>                   Edit Horse

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 Horse
---
. ...Initialize
. =REPEAT WHILE                  <--
. -*ALWAYS
. ...Load first subfile page      <--
. PGM.*Reload subfile = CND.*NO
. > Conduct screen conversation
. =REPEAT 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

The Action Diagram Editor's Subfile selector provides options for editing an action diagram, including options for inserting and manipulating user-defined actions. You can enter ? in the Subfile selector to view a list of all allowed values.

## Code Generation

Generating code is the first part of the third phase of development. You will learn how to generate source to implement the access paths and functions you have created.

You can generate source either interactively or in batch. CA 2E automatically keeps a list of the members to be generated and compiled that have not yet been compiled. We will take a look at how to use batch processing to generate source code and compile executable objects for all of your access paths and functions. These options are available on the Display Services Menu.

For more details on functions, function types, function options, parameter definitions, device designs, and action diagrams, see the *Tutorial* or *Building Applications* guides.

### Generate Access Paths

From the Edit Database Relations panel press F17 to go to the Display Services Menu. A sample menu is shown next:

EDIT DATABASE RELATIONS				My model			
=>				Rel lvl:		Seq	Typ
? Typ	Object			Relation			Referenced object
■ FIL	Course			Known by			FLD Course code
— FIL	Course			Has			FLD Course name
— FIL	Horse			Known by	10		FLD Horse code
— FIL	Horse			Has	20		FLD Horse name
— FIL	Horse			Has	30		FLD Horse gender
— FIL	Horse			Has	40		FLD Horse value
— FIL	Horse			Has	50		FLD Date of birth
— FIL	Horse			Refers to	60		FIL Horse
	For: Dam					Sharing: *ALL	
— FIL	Horse			Refers to	70		FIL Horse
	For: Sire					Sharing: *ALL	
— FIL	Jockey			Known by			FLD Jockey code
— FIL	Jockey			Has			FLD Jockey name
— FIL	Jockey			Has			FLD Jockey gender
— FIL	Race			Owned by			FIL Course

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

Select the Display all access paths option as shown in the following example:

DISPLAY SERVICES MENU		My model
Generation	1. Submit model create request (YSBMDLCRT) 2. convert model data menu 3. Job list 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 profile (YEDTMDLPRF) 12. Work with model lists (YWRKMDLLST) 13. Edit model list (YEDTMDLLST *SESSION) 14. Impact analysis menu	
Change Control	21. Go to Management Option (CM) menu	
Option: 8 (press F4 to prompt commands)		
F3=Exit F6=Messages F8=Submitted jobs F9=Command line F10=Display job log		

Press Enter.

**Note:** The Display All Access Paths panel also lets you perform many design, control, and generation tasks for one or more access paths.

Select all of the existing access path designs for batch generation and compilation by typing a **J** in the Subfile selector next to each access path in the list as shown in the following example:

DISPLAY ALL ACCESS PATHS				
Application area. . : <input type="checkbox"/>		My model		
? File		Source library. . : MYGEN		Prefix
Access path				
J Course	Physical file	PHY MYABREP	AB	
J Course	Retrieval index	RTV MYABREL1	AB	
J Course	Update index	UPD MYABREL0	AB	
J Horse	Mares	RTV MYAEREL2	AE	
J Horse	Physical file	PHY MYAEREP	AE	
J Horse	Retrieval index	RTV MYAEREL1	AE	
J Horse	Stallions	RTV MYAEREL3	AE	
J Horse	Update index	UPD MYAEREL0	AE	
J Jockey	Physical file	PHY MYAFREP	AF	
J Jockey	Retrieval index	RTV MYAFREL1	AF	
J Jockey	Update index	UPD MYAFREL0	AF	
J Race	Physical file	PHY MYACREP	AC	
J Race	Retrieval index	RTV MYACREL1	AC	
J Race	Update index	UPD MYACREL0	AC	
J Race Entry	Physical file	PHY MYADCPP	AD	+
SEL: Z-Details, G/J-Generate, E-STRSEU, N-Narrative, D-Delete, U-Usage, F-Function refs., L-Locks. F3=Exit F5=Reload				

Press Roll Up to display the next screen of access paths so you can select all remaining access paths with a J as shown in the following example:

DISPLAY ALL ACCESS PATHS				
Application area. . : <input type="checkbox"/>		My model		
? File		Source library. . : MYGEN		Prefix
Access path				
J Race Entry	Races for a Horse	RSQ MYADCPL2	AD	
J Race Entry	Retrieval index	RTV MYADCPL1	AD	
J Race Entry	Update index	UPD MYADCPL0	AD	
SEL: Z-Details, G/J-Generate, E-STRSEU, N-Narrative, D-Delete, U-Usage, F-Function refs., L-Locks. F3=Exit F5=Reload				

Press Enter.

When the process is complete, the Display All Access Paths panel will be redisplayed with messages at the bottom of the panel. The messages will state that the source generation requests have been accepted. A panel with messages is shown next:

```

DISPLAY ALL ACCESS PATHS
Application area. :  My model
Source library.  :  MYGEN
? File           Access path      Prefix
-----
- Race Entry     Races for a Horse    RSQ MYADCPL2    AD
- Race Entry     Retrieval index     RTV MYADCPL1    AD
- Race Entry     Update index        UPD MYADCPL0    AD

SEL: Z=Details, G/J=Generate, E=STRSEU, H=Narrative,
      D=Delete, U=Usage, F=Function refs., L=Locks.
F3=Exit F5=Reload
Source generation request for MYABREP accepted.
+
```

Press F3 to return to the Display Services Menu.

Generate External Functions Source Code

Only external functions need to be generated and compiled. External functions can be identified by the source member name found in the GEN name column; internal functions show \*N/A in this column.

List just the external functions in your design model by typing \*EXT in the Type column on the selection line as shown in the following example:

```

DISPLAY ALL FUNCTIONS
Application area. :  My model
Source library:    MYGEN
? File           Function      Type      GEN name
-----
- Course         Change Course  CHGOBJ    *N/A _ Course
- Course         Create Course  CRTOBJ    *N/A
- Course         Delete Course  DLT OBJ   *N/A
- Course         Edit Course   EDTFIL    MYACEFR
- Course         Select Course SELRCD     MYABSRR
- Horse          Change Horse  CHGOBJ    *N/A
- Horse          Create Horse  CRTOBJ    *N/A
- Horse          Delete Horse  DLT OBJ   *N/A
- Horse          Edit Horse   EDTFIL    MYAEFR
- Horse          Select Horse SELRCD     MYAISRR
- Horse          Select Mares SELRCD     MYAJSRR
- Horse          Select Stallions SELRCD     MYAKSRR
- Jockey         Change Jockey CHGOBJ    *N/A
- Jockey         Create Jockey CRTOBJ    *N/A
- Jockey         Delete Jockey DLT OBJ   *N/A
SEL: Z=Dtls P=Parms N=Narr F=Action diagram S=Device Design T=Structure
      A=Acp G/J=Gen E=STRSEU L=Locks D=Delete U=Usages 3=Doc H=Gen HTML
      F3=Exit F5=Reload
+
```

Proceed by selecting all functions and initiating the batch processing as you did for the access paths.

## Program Compilation

Program compilation is the second part of the third phase of development. You will learn how to call your compiled applications (programs) and test them by entering data.

When you submit a request for generation, compilation, or both, the CA 2E generator automatically generates a job list. The same job list controls generation and compilation. You can review this job list during generation to monitor the process and edit the job list.

Source is produced and placed in the appropriate source file in the generation library associated with your CA 2E design model, and then the generated source is compiled.

To begin, select the Submit model create request (YSBMMDLCRT) option as shown in the following example, and press Enter.

DISPLAY SERVICES MENU		My model
Generation	1. Submit model create request (YSBMMDLCRT) 2. convert model data menu 3. Job list 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 profile (YEDTMDLPRF) 12. Work with model lists (YWRKMDLLST) 13. Edit model list (YEDTMDLLST *SESSION) 14. Impact analysis menu	
Change Control	21. Go to Change Management Option (CM) menu	
Option: <u>1</u> (press F4 to prompt commands)		
F3=Exit   F6=Messages   F8=Submitted jobs   F9=Command line   F10=Display job log		

**Note:** CA 2E supplies default parameter values for the YSBMMDLCRT command based on your model profile and options you specified when you created your model. You can override these defaults by pressing F4 instead of Enter to prompt the command.

CA 2E displays a list of the source members to be generated and compiled. Each member has either GEN or CRT next to it to indicate whether the member has been submitted for generation or compilation. At this point all the functions and access paths should show GEN as shown in the following example:



SUBMIT MODEL GENERATIONS & CREATES. My model					
					GENLIB: MYGEN
? Member	Type	Act	Status	Text	
■ MYABREP	PF	GEN		Course	Physical file
— MYACREP	PF	GEN		Race	Physical file
— MYADCPP	PF	GEN		Race Entry	Physical file
— MYAEREP	PF	GEN		Horse	Physical file
— MYAFREP	PF	GEN		Jockey	Physical file
— MYABREL0	LF	GEN		Course	Update index
— MYACREL0	LF	GEN		Race	Update index
— MYADCPL0	LF	GEN		Race Entry	Update index
— MYAEREL0	LF	GEN		Horse	Update index
— MYAFREL0	LF	GEN		Jockey	Update index
— MYABREL1	LF	GEN		Course	Retrieval index
— MYACREL1	LF	GEN		Race	Retrieval index
— MYADCPL1	LF	GEN		Race Entry	Retrieval index +
SEL: G-Rqs GEN, C-Rqs CRT, E-STRSEU, D-Drop, JOB(1-DSP, 4-HLD, 6-RLS, 9-CNL)					
F3=Exit F5=Reload F6=Msgs F8=Submitted jobs F9=Command line ENTER=Submit					

Press Roll Up until you see the bottom of the list, and then Press Enter to submit the job list.

After confirming the list of objects, you will see the following series of messages at the bottom of the panel:

- Job YGENSRC is being prepared.
- Existing objects are being deleted.
- Joblist successfully processed.

In batch processing, the generations, compilations, or both take place in the background. You may continue to specify new objects (functions, access paths) while the batch is processing.

Before calling your program, you must use the Convert Condition Values (YCVTCNDVAL) command to convert the values that are entered into status fields. The values you defined for status fields are moved from the model library to the condition values list database file in the generation library.

Horse gender is an example of a status field. We specified the valid values for this field in our model library and those values are placed in the generation library database when we run the Convert Condition Values (YCVTCNDVAL) command. In this case, the end user running the application will then be able to select either M or F as valid values for Horse gender.

When you exit this session, if you have added or changed any files, fields, or relations in the current session, you may want to resynchronize the model. This causes all the relations to be resolved into entries. Only users of enter \*DSNR can resynchronize a model.

Select the desired option, as shown in the following example, and press Enter:

EDIT DATABASE RELATIONS				My model			
=>	Rel lvl:	Seq	Typ	Referenced object			
? Typ Object	Relation						
— FIL Course	Known by		FLD	Course code			
— FIL Course	Has		FLD	Course name			
— FIL Horse							
— FIL Horse							
Exit Database Relations							
— FIL Horse :	Select one of the following:						
— FIL Horse :	1. Exit without resynchronising						
— FIL Horse :	2. Exit and resynchronise data model						
— FIL Horse :	3. Return to editing						
— FIL Horse :	Option:	2					
— FIL Jocke :	** Model is not synchronised **						
— FIL Jocke :	F12=Cancel						
— FIL Jocke :							
— FIL Race :							

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

## Testing Your Programs

You can call your compiled programs from any command entry line by using the i OS CALL command to invoke your program. From within your model, you can display a command entry line by pressing F9 from the Display Services menu.

**Note:** If you do not know the name of your program, you can obtain it from the Edit Function Details panel or the Display all functions option on the Display Services Menu.

When you call your program you need to specify a dummy \*Return code parameter. All CA 2E programs require a dummy parameter to be passed to the program ('`' or ``'). This return code can be used to communicate between programs in more complex applications.

Enter the program name and return code, and press Enter. The following example calls the program MYAEEFR.

MAIN		2E Main Menu	
Level . :	1	System:	2EDV1
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. 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		

Selection or command	More. . .
==> <u>call myaeefr **</u>	
F3=Exit F6=Messages F8=Rev retrieve F9=Retrieve F10=Cmd Entry F14=Sbm jobs	

**Note:** To test complex programs, you can use the CA 2E Call a Program (Y2CALL) command. This command loads your model and determines the parameters required by an external function directly from details contained in the model.

You can provide values for all input-capable fields and you can reuse these values for subsequent calls. You can also retrieve and display output parameters when the called program ends.