CA 2E

Building Access Paths

Release 8.7

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Documentation Changes

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

- Allow SQL Record Level Access
  - YSQLFMT (see page 46)
- Refresh Action Diagram Statements
  - YRFSACT (see page 43)
- Allow RLA Access over DDL Database
  - Access Path Auxiliaries (see page 24)
  - Understanding Generator Types (see page 25)
  - Model Values (see page 25)
  - Creating an SQL Environment (see page 30)
  - Specifying Generation Mode (see page 30)
  - Changing the Generation Mode at the Access Path Level (see page 31)
  - YDBFGEN (see page 37)
  - Modifying Access Path Details (see page 67)
  - Specifying Generation Mode (see page 73)
  - Copying an Access Path Generated with SQL or DDL (see page 74)
  - For DDL Access Paths with *IMMED Maintenance (see page 96)
  - Implementing (see page 109)
  - Changing Compiler Overrides from DDS to SQL or DDL (see page 110)
  - Identifying the Implementation Attributes (see page 110)
  - For DDS Query (QRY) Access Paths (see page 95)
- Allow SQL/DDL generation without hard-coded schema name
  - Model Values (see page 25)
  - YSQLCOL (see page 45)
- Allow LVLCHK(*YES) for SQL/DDL indexes having RCDFMT keyword
  - YLVLCHK (see page 40)
  - YSQLFMT (see page 46)
- YSQLFMT override
  - YSQLFMT (see page 46)

- Select/Omit criteria in DDL Index
  - Select/Omit in DDL Index (see page 89)
  - Using Select/Omit Maintenance (see page 124)
- Meaningful Names for SQL/DDL
  - Model Values (see page 25)
  - YSQLVNM (see page 47)
- Suppress *DDL generation for SPN/QRY
  - Identifying the Implementation Attributes (see page 110)
- Option to generate RLA against DDL
  - Model Values (see page 25)
  - YSQLVNM (see page 47)
  - YDDLDBA (see page 37)
- F7=Auxiliaries in the DDL Implementation
  - Access Path Auxiliaries (see page 24)
  - Editing Access Path Auxiliaries (see page 97)
- IBM Limitation - File name is valid system name
  - Model Values (see page 25)
  - YSQLVNM (see page 47)
  - Edit File Details (see page 54)
- Meaningful Names for SQL/DDL - Control Table vs Fields
  - YSQLVNM (see page 47)
  - YDDLDBA (see page 37)
- DDL Limitation
  - Generating an Access Path (see page 112)
  - Identifying the Implementation Attributes (see page 110)

Contents

Chapter 1: Introduction to Access Paths 13

Purpose 13
Organization 13
Contents 14
Related Information 15
Acronyms and Terms Used in this Guide 15
    Acronyms 15
    Values 16
Understanding Access Paths 17
Recognizing the Basic Properties of Access Paths 17
    Identifying Access Path Types 17
    Physical (PHY) Access Path 18
    Examples 18
    Update (UPD) Access Path 18
    Examples 19
    Retrieval (RTV) Access Path 19
    Examples 20
    Resequence (RSQ) Access Path 20
    New Topic 20
    Query (QRY) Access Path 21
    Span (SPN) Access Path 22
    Example 22
    Characteristics of Access Paths 22
    Naming Access Paths 23
Recognizing Access Path Components 23
    Access Path Details 23
    Access Path Format Entries 23
    Access Path Relations 23
    Access Path Auxiliaries 24
Narrative Text 24
Understanding Generator Types 25
Model Values 25
Changing Values 28
    Allocating Names 28
    Allocating a Source Member Name for an Access Path 28
    Controlling Auxiliary Names 29
    Creating an SQL Environment 30

Specifying Generation Mode....30
Changing the Generation Mode at the Access Path Level....31
Changing Compiler Overrides....32

Chapter 2: Setting Default Options for Your Functions 33

Model Values Used in Building Functions....33
YABRNPT....34
YACTCND....34
YACTFUN....34
YACTSYM....34
YACTUPD....35
YALCVNM....35
YBNDDIR....35
YCNFVAL....35
YCPYMSG....35
YCRTENV....36
YCUAEXT....36
YCUAPMT....36
YCUTOFF....37
YDATFMT....37
YDATGEN....37
YDBFGEN....37
YDDLDBA....37
YDFTCTX....38
YDSTFIO....38
YERRRTN....38
YEXCENV....38
YGENCMT....39
YGENHLP....39
YGENRDB....39
YHLLGEN....39
YHLLVNM....39
YHLPCSR....40
YLHSFLL....40
YLVLCHK....40
YNPTHLP....41
YNLLUPD....41
YOBJPFX....41
YPMTGEN....42
YPMTMSF....42
YPUTOVR....42

YRP4HSP 43
YRFSACT 43
YRP4HS2 43
YRP4SGN 43
YSAAFMT 44
YSFLEND 44
YSHRSBR 44
YSNDMSG 45
YSQLCOL 45
YSQLFMT 46
YSQLLCK 46
YSQLVNM 47
YSQLWHR 48
YWSNGEN 48
User Interface Manager (UIM) 48
Window Borders 49
Changing Model Values 49
Function Level 49
Model Level 50
Changing a Function Name 50
Function Key Defaults 51

Chapter 3: Adding Access Paths 53

Before Adding 53
Edit File Details 54
Adding an Access Path 55
Adding a Physical (PHY) Access Path 56
Adding a Resequence (RSQ) Access Path 56
Adding a Query (QRY) Access Path 57
Adding a Span (SPN) Access Path 58

Chapter 4: Modifying Access Paths 61

Before You Begin 61
Before Modifying 62
Navigational Techniques and Aids 62
Automatic Add Options 62
Changing the Auto Add Setting 63
Trimming an Access Path 64
Virtualizing an Access Path 64
Locking an Access Path 64
Temporary Locks 65

Permanent Locks 65
Displaying Usages for Access Paths 65
Building Distributed Relational Database Applications 66
Specifying Distributed Files 66
Modifying Access Path Details 67
Editing Access Path Details 69
Specifying Unique/Duplicate Key Retrieval Sequence 70
Specifying Access Path Maintenance 71
Specifying Alternate Collating Sequence 72
Specifying Select/Omit Criteria 72
Specifying Generation Mode 73
SQL and DDS Joins 74
Copying an Access Path Generated with SQL or DDL 74
Changing Source Member Text and Names 75
Modifying Access Path Format Entries 75
Identifying Access Path Format Text 75
Identifying Access Path Format Keys 75
Changing the Key Sequence 76
Editing Access Path Format Entries 77
Editing Physical File Format Entries 77
Altering Field Sequence or Implementation Name 78
Modifying Access Path Relations 79
Understanding Required Relations 79
Adding Relations to a File 80
Editing Access Path Relations 81
Modifying Virtual Field Entries 82
Understanding Access Path Virtual Field Entries 82
Identifying Relations with Virtual Fields 83
Specifying File and Access Path Relations 84
Editing Virtual Field Entries 85
Tailoring Virtual Fields for Access Paths 86
Choosing Select/Omit Criteria 86
Understanding Select/Omit 87
Specifying Selection 88
Specifying Conditions 89
Select/Omit in DDL Index 89
Changing a Referenced Access Path 91
F4 Prompt Function Assignment 94
Modifying Access Path Auxiliaries 94
Understanding Access Path Auxiliaries 95
For DDS Query (QRY) Access Paths 95
For SQL Access Paths with *IMMED Maintenance 95

For DDL Access Paths with *IMMED Maintenance 96
Editing Access Path Auxiliaries 97

Chapter 5: Deleting Access Paths 99

Deleting an Access Path 99
Determining the Usage of an Access Path 100

Chapter 6: Defining Arrays 101

Understanding Arrays 102
Structuring Field Data Using Arrays 102
Passing Parameters 103
Storing Data Between Calls 103
Defining an Array 104
Editing an Array 107
Viewing Function References 107
Changing the Name of an Array 108
Deleting an Array 108

Chapter 7: Generating and Compiling 109

Implementing 109
i OS Index Versus CA 2E Index 109
Setting Your Options 109
Changing Compiler Overrides from DDS to SQL or DDL 110
Identifying the Implementation Attributes 110
Generating an Access Path 112

Chapter 8: Documenting Access Paths 115

Documenting an Access Path 115
Creating the Documentation 116

Chapter 9: Tailoring for Performance 117

Considering the Types of Data in the Physical File 118
Minimizing the Number of Active Indexes 118
The Active Index 119
Sharing Active Indexes 119
Access Path Maintenance (Immediate, Delay, or Rebuild) 121
Maintenance for Query (QRY) Access Paths 122
Using Select/Omit Maintenance 124
Using Join Logicals 125

Using Multi-Format Access Paths....126
Using Open Data Paths....126
Creating Default Retrieval Access Paths....127

Index **129**

Chapter 1: Introduction to Access Paths

Building Access Paths is part of a set of guides that provide instructions on how to use the CA 2E product.

This section contains the following topics:

Purpose (see page 13)
Organization (see page 13)
Contents (see page 14)
Related Information (see page 15)
Acronyms and Terms Used in this Guide (see page 15)
Understanding Access Paths (see page 17)
Recognizing the Basic Properties of Access Paths (see page 17)
Recognizing Access Path Components (see page 23)
Narrative Text (see page 24)
Understanding Generator Types (see page 25)
Model Values (see page 25)
Changing Values (see page 28)
Changing Compiler Overrides (see page 32)

Purpose

This guide describes how to build access paths and arrays in CA 2E. It explains how to set up your CA 2E system and model values and how to add, modify, delete, and document access paths and arrays. Each chapter is designed to provide the complete information needed to perform the tasks identified in the chapter. Review the entire guide or see the chapter that relates to the specific task you want to perform.

Organization

This guide contains a preface, an introductory chapter, and eight task-oriented chapters.

The introduction provides a high level overview of the CA 2E concepts for building access paths. Seven of the remaining eight chapters contain conceptual material and instructions on the specific tasks required to add, modify, delete, and generate access paths. One chapter deals specifically with building arrays.

Where necessary, these chapters also contain references to other topics and chapters in this guide and other guides or reference manuals with related information.

Contents

The chapters in this guide are as follows:

Chapter	Description
1. Access Paths: An Introduction	This chapter contains an introduction to the types of access paths and a high level overview of the CA 2E concepts for building access paths.
2. Setting Default Options for Your Access Paths	This chapter contains conceptual material and instructions on setting CA 2E model values for allocating prefixes, file names and SQL libraries. It also includes details on generating database file values and instructions on changing compiler overrides.
3. Adding Access Paths	This chapter contains conceptual material and instructions for editing file details and adding the following six access paths: physical, update, retrieval, resequence, query, and span.
4. Modifying Access Paths	This chapter contains conceptual material and instructions on how to modify existing access paths, including the details, format entries, relations, and auxiliaries. It also contains information on modifying virtual field entries and select/omit criteria.
5. Deleting Access Paths	This chapter contains conceptual material and instructions on how to delete existing access paths.
6. Defining Arrays	This chapter contains conceptual material and instructions on how to add, edit, and delete an array.
7. Generating and Compiling	This chapter contains conceptual material and instructions on how to set up your generation options and how to generate and compile your access paths.
8. Documenting Access Paths	This chapter contains conceptual material and instructions on how to document the access paths created in CA 2E.

Chapter	Description
9. Tailoring For Performance	This chapter contains material that can help you tailor your access paths to obtain the best system performance.

Related Information

Before you build your access paths, you should read or review the material in the following guides:

- *Readme*
- *Getting Started*
- *Defining a Data Model*

The following guides contain additional information relating to the generation of access paths and associated functions.

- *Building Applications*
- *Generating and Implementing Applications*

You may want to see the following IBM documentation in the context of using this guide.

- IBM i DDS Reference Manual

Acronyms and Terms Used in this Guide

Descriptions of the acronyms and values used in this guide are defined once, in this chapter. Thereafter, only the acronym or value is used.

Acronyms

The following acronyms appear in this guide:

ANSI	American National Standards Institute
CBL	COBOL
CL	Control Language
DDL	Data Definition Language
DDS	Data Description Specifications

DML	Data Manipulation Language
DRDA	Distributed Relational Database Architecture
ESF	External Source Format
FCFO	First Changed, First Out
FIFO	First In, First Out
HLL	High level language
IBM	International Business Machines Corporation
I/O	Input/Output
LIFO	Last In, First Out
ODP	Open Data Path
RPG	Report Program Generator
SAA	Systems Application Architecture
SQL	Structured Query Language

Values

The following values appear in this guide:

CPT	Capture file
PHY	Physical access path
QRY	Query access path
REF	Reference file
RSQ	Resequence access path
RTV	Retrieval access path
SPN	Span access path
UPD	Update access path

This chapter provides an overview of how to build *access paths*. Its purpose is to help you understand the CA 2E concepts for using access paths in your design model.

In this guide, the term access path refers to the CA 2E definition exclusively unless identified as an i OS access path.

Understanding Access Paths

A CA 2E access path can be implemented as one or more i OS objects. Access paths can be created over files that have been defined, but before the functions associated with the access path are created. The application uses access paths to retrieve, sequence, or update data from the physical file. CA 2E creates default access paths for you when you define a file in your model. However, you can create additional access paths for your file.

An access path defines the physical file and/or the logical views of that file. When you build one, you specify the following:

- The order in which you want to retrieve records from a file
- Which fields will be present
- Your select/omit criteria for deciding which records from the file will be retrieved by the access path

Recognizing the Basic Properties of Access Paths

There are six different types of access paths, each with a different purpose. These types are defined in this topic. Access paths are allowed for both Reference (REF) and Capture (CPT) files. In addition, each access path must have a valid CA 2E name.

For more information about REF and CPT files, see Understanding Your Data Model in *Defining a Data Model*.

Identifying Access Path Types

The following sections present a description of the six types of access paths.

Physical (PHY) Access Path

A PHY access path is a single-format file containing the fields derived from the resolution of all the relations on a file. This access path type:

- Is unkeyed
- Has no virtual fields
- Is created automatically by CA 2E for every defined REF or CPT file
- Is not referenced directly by functions
- Allows no additional PHY access path to be created for a given CA 2E file
- Is created in a model if an existing physical file is retrieved into the model through assimilation

For more information about:

- Assimilation, see the "Assimilation" chapter *Defining a Data Model.*
- Editing physical file format entries for assimilated files, see the topic Modifying Access Paths (see page 61).

Examples

Every CA 2E file has one access path of type PHY, called Physical file by default. For example:

- Physical file for the Company file
- Physical file for the Product file
- Physical file for the Order file
- Physical file for the Order detail file

Update (UPD) Access Path

A UPD access path specifies a uniquely keyed, single-format access path that describes a view to the function for updating the file. This access path type:

- Is always keyed on the fields that identify the file. These entries arise from the resolution of the key relations.
- Has no virtual fields
- Is created automatically by CA 2E for every defined REF or CPT file

You can create additional UPD access paths, with the same keys as specified on the file, but which have a subset of the fields defined by the relations. Additional UPD access paths are seldom required.

Examples

Every CA 2E file has a default CA 2E UPD access path that will be called Update index by default. For example:

- Update index for the Company file
- Update index for the Product file
- Update index for the Order file
- Update index for the Order detail file

You can create other CA 2E UPD access paths for use in functions that update only some fields from a file, for example:

- Company address update only
- Batch status only

Retrieval (RTV) Access Path

An RTV access path specifies a uniquely keyed, single- format access path that functions can use to retrieve records from a file. This access path type:

- Is always keyed in exactly the same way as the UPD access path using the relations of the based-on file.
- Allows virtual fields on the access path.
- Is automatically created by CA 2E for every defined REF or CPT file.
- Defaults to the virtual fields of the based-on file's relations. These are then present on the access path's relations.
- Is associated with an UPD access path; CA 2E automatically makes this association.
- Can be edited or trimmed to drop some or all non-key fields from the record layout.
- Can define select/omit logic to select or omit records from the access path.
- Can be set not to pick up virtual fields.

You can create many RTV access paths for a given file. Each can contain a different combination of fields and/or virtual fields and a different set of selection criteria, but all have the same key fields.

Examples

Every CA 2E file has a default RTV access path, created for it automatically by CA 2E, called Retrieval index by default. For example:

- Retrieval index for the Company file

You can define other CA 2E RTV access paths for the same CA 2E file. For example:

- Company active index (selecting active records only)
- Company summary index (with a subset of the file relations)

Resequence (RSQ) Access Path

A RSQ access path specifies a uniquely or non-uniquely keyed, single-format access path you can use to describe to CA 2E functions how records are to be retrieved from a file. This access path type:

- Must be created explicitly
- Defaults to those keys defined by the key relations for the based-on file, but allows them to be overridden to an alternative key sequence that does not need to be unique
- Allows virtual fields to be specified on the access path
- Is associated with a RTV access path that points to an associated UPD access path
- Defaults to the virtual fields of the based-on file's relations. These are then present on the access path's relations

You can create many RSQ access paths for a given file. Each can contain a different combination of data fields and/or virtual fields, a different set of selection criteria, or an alternative key order.

New Topic

- Company Known by Company code; RSQ by Company name
- Order Known by Order no.; RSQ by Order date
- Person Known by Person code; RSQ by Height

Query (QRY) Access Path

A QRY access path specifies a keyed, single-format access path you can use to describe to functions how records are to be retrieved from a file. This access path type:

- Allows virtual fields to be specified as key and non-key fields on the access path
- Can use virtual fields as key fields
- Is available for use with the following function types: Display File, Select Record, Retrieve Object, Print Object, and Print File
- Defaults to those keys defined by the key relations for the file, but allows them to be overridden to an alternative key sequence
- Must be created explicitly
- Is associated with a RTV access path that in turn points to an associated UPD access path
- Defaults to the virtual fields of the based-on file's relations. These are then present on the access path's relations

You can create many QRY access paths for a given file. Each can contain a different combination of data fields and virtual fields, a different set of selection criteria, and/or an alternative key sequence.

- Customer known by Customer code; Order refers to Customer and Customer name is a virtual field on Order. Using a QRY access path, you can retrieve Order records in Customer name order.
- Product known by product code; Order line refers to Product and Product name is a virtual field on Order line. Using a QRY access path, you can retrieve Order line records in Product name order.
- Company known by Company code; Employee owned by Company and Company name is a virtual field on Employee. Using a QRY access path, you can retrieve Employee records in Company name order.

Span (SPN) Access Path

A SPN access path specifies a keyed multi-format access path. It can be used to describe to the edit and display transaction functions how records are to be retrieved from a pair of related files. These files possess a common foreign key. These files must be related by an Owned by or Refers to relation. The SPN access path must be created over the owning or referred to file. This access path type:

- Initially defaults to those keys defined by the key relations of the based-on files but can be overridden to an alternative key sequence
- Allows virtual fields to be specified on the access path relations
- Must be created explicitly
- Is associated with a RTV access path that points to an associated UPD access path used to carry out any updates to the based-on file
- Allows explicit selection of multiple access path formats
- Defaults to the virtual fields of the based-on file's relations; these are then present on the access path's relations.

Example

Characteristics of Access Paths

The following table shows characteristics of CA 2E access paths:

Access Path Type	Real Fields	Key Fields	Virtual Fields	Virtual Keys
PHY (Physical)	Yes	No	No	No
UPD (Update)	Yes	Relation	No	No
RTV (Retrieval)	Yes	Relation	Yes	No
RSQ (Resequence)	Yes	User	Yes	No
QRY (Query)	Yes	User	Yes	Yes
SPN (Span)	Yes	User	Yes	No

Naming Access Paths

A name for an access path can be free format and up to 25 characters. Within a given file, the access path names must be unique. Since each access path is implemented as a separate i OS object, each access path also will be given a unique source member name (in the model) before source code can be generated for it. The member name becomes the name of the object used to implement the access path.

CA 2E supplies a default name if the Allocate Name (YALCVNM) model value is set to *YES or *MNC.

For more information about:

- Changing a default name refer to the "Setting Default Options for Your Access Paths" chapter.
- Naming implementation objects see the topic Setting Up the User Environment in the "Using Your Development Environment" chapter of the *Administration Guide.*
- The YCHGMDLVAL command, see the *CA 2E Command Reference Guide*.

Recognizing Access Path Components

This section explains access path details, format entries, and path relations.

Access Path Details

Access Path Details are the various implementation options specified for access paths. These options include changing source names and text, allowing selection criteria, and specifying generation mode, unique key sequence, access path maintenance, and alternate collating sequence.

Access Path Format Entries

Access Path Format Entries show which fields are present on the access path, which of those fields are key fields for that access path, and the order of those keys. SPN access paths have at least two formats. Other access path types can have only one format.

Access Path Relations

Access Path Relations are the set or subset of a file's relations that apply to a particular access path. The compulsory relations for an access path are the key level relations. They must be present on all access paths for the file. Each file-to-file relation on the access path can be associated with a different set of virtual fields.

Access Path Auxiliaries

Access Path Auxiliaries refer to:

- The three different i OS objects used to implement a query access path for DDS objects. They include a logical file, a physical file, and a control language (CL) program.
- The SQL index created for an SQL-implemented access path with *IMMED index maintenance.
- The DDL index created for a DDL- implemented access path with *IMMED index maintenance.
- Auxiliaries are not applicable for DDL type file as Views are not created for DDL type file and *only* Index with the same name as the source member name is created.

For more information about auxiliaries, see:

- The topic Adding a Query (QRY) Access Path in the "Adding Access Paths" chapter
- The topic Modifying Access Paths in the "Modifying Access Paths" chapter

Narrative Text

Narrative text is user-added text associated with any CA 2E object. You can add narrative text to any access path you create. After you create the access paths, add the narrative text to describe its definition and function. It is used in the following places:

- Documentation of the model
- Interactive explanation of the model
- Generation of help text for functions
- Generation of program synopses

For more information about how to use narrative text, see:

- The topic Using Narrative Text in the "Using Your Model" chapter of the *Administration Guide*
- The "Documenting Access Paths" chapter of this guide

Understanding Generator Types

Within a CA 2E design model, you can use either of DDS, SQL or DDL to implement data definitions for all types of access paths.

For more information about generator types, see the following:

- The "Setting Default Options for Your Access Paths" chapter and the "Generating and Compiling" chapter in this guide.
- The "Setting Default Options for Your Functions" chapter in the *CA 2E Building Applications* guide

The "Using Your Development Environment" chapter in the *Administration Guide*

This chapter explains how to set up options for the model values assigned to the access paths that you build and how to change compiler overrides.

Model Values

Model-specific values control particular features of the interactive use of CA 2E, code generation, and implementation.

For more information about what a model value is, see the "Using Your Development Environment" chapter in the *Administration Guide.*

The model values you need when building access paths are:

YALCVNM	The Allocate Valid Name (YALCVNM) model value specifies whether DDS and SQL object names are to be allocated automatically by CA 2E or by a specific standard you establish for the CA 2E model.
YOBJPFX	The Object Prefix (YOBJPFX) model value specifies the prefix to be used when generating system objects.
YFILPFX	The Last Used File Prefix (YFILPFX) model value contains the last 2-character identifying mnemonic CA 2E used when creating a new file. These two characters occupy positions three and four of the new file name, following the model object prefix.

For more information about object name prefixes, see the following:

- The "Creating and Managing Your Model" chapter in the *Administration Guide*
- The Setting Up the User Environment section in the "Using Your Development Environment" chapter of the *Administration Guide*

YDBFGEN — The Database Generation (YDBFGEN) model value specifies the default method of source generation (DDS, SQL or DDL) for database definition.

You can set your source generation type by doing the following:

- Setting the model value YDBFGEN at the time that you create your CA 2E model.
- Changing the model value YDBFGEN after creating the model.
- Changing the generation mode on a specific access path.

For more information about setting the source generation type for your model, see the "Creating and Managing Your Model" chapter in the *Administration Guide.*

YSQLLIB — The SQL Library (YSQLLIB) model value specifies the library (collection) in which the SQL objects needed to implement an SQL database should be placed.

Note: Earlier, when CA 2E used to generate SQL/DDL artifacts, the SQL collection mentioned in YSQLLIB was hard-coded into the generated source. Upon compilation the objects were placed in that SQL collection. The SQL/DDL generation has now been modified to do the following

- The YSQLLIB model value can now hold a normal library (non-SQL collection).
- The user is given the option to decide whether to generate the hard-coded value present in the YSQLLIB model value, through the YSQLCOL model value.
- If the library/SQL collection is not generated into the source during generation, when a compile is submitted to create the SQL/DDL objects, the objects are created into the library/SQL collection specified for the YSQLLIB model value.

YSQLVNM The SQL Naming (YSQLVNM) model value specifies whether to use the extended SQL naming capability. You can specify one of the DDS names (the shipped default), the names of CA 2E objects in the model (extend SQL naming), or the long names of the CA 2E objects in the model along with the DDS or implementation names.

Note: If a table that has a valid system name (less than or equal to 10 bytes in length), is generated with YSQLVNM model value set as *LNG or *LNT, and when you set the Enhance SQL Naming option on the Edit File Details panel to **Y** and then generate the source, the table is created with (underscores) "_"s and "TABLE" as suffix, so that the name of the table becomes more than 10 char long.

Examples:

- CUSTOMER is generated as CUSTOMER_TABLE along with its 2E implementation name.
- CUST is generated as CUST__TABLE along with its 2E implementation name.
- C is generated as C_____TABLE along with its 2E implementation name.
- To generate or regenerate a function with RLA code for DDL database, set the YSQLVNM model value to *DDS or *LNG or *LNT, or *LNF and set the YDDLDBA model value to *RLA.

YSQLLEN The SQL Naming Length (YSQLLEN) model value is a numeric value that controls the length of the extended SQL name. Its maximum value is 25. This model value is used only when YSQLVNM is *SQL.

For more information about extended SQL naming, see the "SQL Implementation" appendix in the *Administration Guide*.

YDBFACC The Database Access Method (YDBFACC) model value lets you specify whether to access data from a table or from a view when an access path and the table over which it is based contain the same fields.

For more information about direct table access, see the "SQL Implementation" appendix in the *Administration Guide.*

You can set your model values when you create your model or change the model value with the YCHGMDLVAL command. However, once you have set your model values, you can then override many of these defaults for a specific access path using the access path detail options.

For more information about model values and how to set model values, see YCHGMDLVAL in the *CA 2E Command Reference Guide*.

Changing Values

Use the following information to change the model values for your access paths.

Allocating Names

This topic tells you how to control the names assigned to CA 2E objects by changing the names given to the source member names when the access paths are built.

Allocating a Source Member Name for an Access Path

1. Zoom into the file.

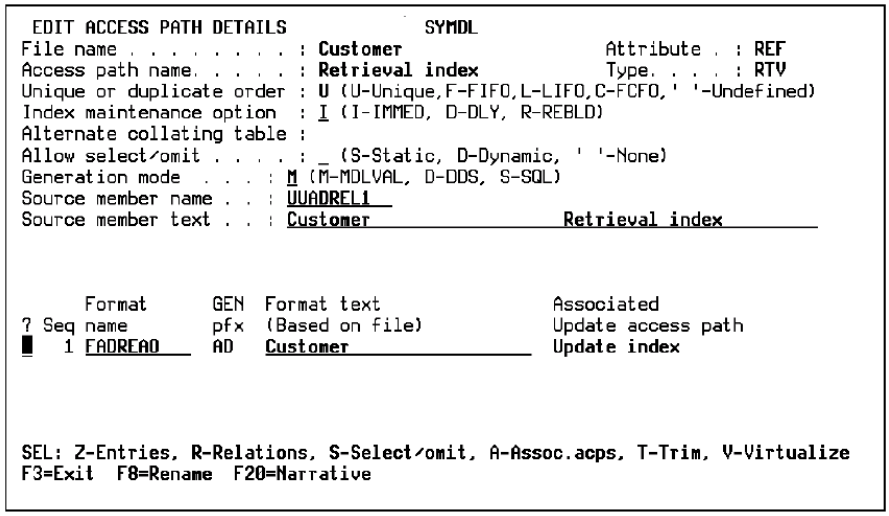
   At the Edit Database Relation panel, type **Z** next to any relation for the file and press Enter.

   The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** next to the one you want to rename and press Enter.

   The Edit Access Path Details panel displays:



3. Enter the new source member name in the source member name option field and press Enter.

Controlling Auxiliary Names

CA 2E generates default values for access path auxiliaries for Query (QRY) access paths and SQL tables or views with *IMMED maintenance capability.

For more information about:

- SQL generation, see Specifying Generation Mode later in this section.
- SQL naming and separate view and index creation, see the "SQL Implementation" appendix in the *Administration Guide*.

Change the name of auxiliaries using the following procedure:

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to the relation for the file and press Enter.

   The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** next to the selected QRY (or SQL) access path and press Enter.

   The Edit Access Path Details panel displays with the details for the selected access path.

3. View the auxiliaries.

   Press F7 to view the access path auxiliaries.

   The Edit Access Path Auxiliaries panel displays.

4. Type the new source member names and press Enter.

Creating an SQL Environment

Using SQL facilitates the portability of generated applications and is the only means of database access across machines in Distributed Relational Database Architecture (DRDA).

For more information:

- On SQL, see the appendix "SQL Implementation" in the *Administration Guide*.
- On DRDA, see the chapter "Modifying Access Paths" in this guide, and the chapter "Distributed Relational Database Architecture" in *Generating and Implementing Applications*.

To create SQL tables and views or DDL indexes for your model, you need an SQL library known as a collection. This collection contains the SQL objects, including the catalog, a data dictionary, a journal, and two journal receivers.

When creating an SQL environment, use one of the following:

- SQLLIB parameter on the YCRTMDLLIB command creates a collection library name with MDL replaced by SQL or a name of your choosing.
- Create SQL Library (YCRTSQLLIB) command creates a collection and links it to a model.

Specifying Generation Mode

The choice of which generation mode to use is controlled by the Database File Generation (YDBFGEN) model value that acts as an implementation flag. The default value is DDS. You can use the following procedure to assign a value to a specific access path when it is built. The options are DDS, SQL and DDL.

Changing the Generation Mode at the Access Path Level

To change the generation mode for a specific access path at the access path level:

1. **Zoom into the file.** At the Edit Database Relations panel, type **Z** next to the relations for the file and press Enter.

   The Edit File Details panel displays.

2. **Zoom into the access path**. Type **Z** next to the access path whose generation mode you want to change and press Enter.

   The Edit Access Path Details panel displays.

3. **Change the generation mode.** Type the character that represents the new generation model value.

   Options are:

   - D for DDS
   - S for SQL
   - M for MDLVAL
   - L for DDL

   Note: If an access path specifies M for MDLVAL when it is generated, it will use the current value for the YDBFGEN model value. If you want to override this value, enter D, S or L. The default is M.

4. Press Enter**.**

Changing Compiler Overrides

Within CA 2E, there are various properties of the i OS database files that you can modify by specifying compiler overrides. The overrides are the parameters on the compiler commands.

CA 2E allows you to prompt for and store these overrides that are then automatically applied by the compile pre-processor when you compile your programs.

Some of the overrides you can specify are:

- Physical files: i OS Create Physical File (CRTPF) command
- MAXMBRS, SIZE

Note: MAXMBRS is a parameter that specifies the maximum number of members the file can hold.

- Logical files: i OS Create Logical File (CRTLF) command
- MAXMBRS, DTAMBRS

Note: Some of the compile parameters (MAINT, TEXT) are specified by the access path details. Override values should not be specified for these values.

For more information:

- On how to prompt for and store overrides, see the Changing Compiler Overrides section in the "Generating and Compiling" chapter of this guide.
- On the i OS commands, see the IBM i CL Command Reference.

Chapter 2: Setting Default Options for Your Functions

This chapter identifies the model values specific to functions and shows you how to change them, how to change the default names that assigns to functions, and function key defaults.

This section contains the following topics:

Model Values Used in Building Functions (see page 33)
Changing Model Values (see page 49)
Changing a Function Name (see page 50)
Function Key Defaults (see page 51)

Model Values Used in Building Functions

This topic covers the model values used by functions. Function options can affect the device design and processing defaults. Model values are shipped as defaults for the Create Model Library (YCRTMDLLIB) command.

Many function options are derived from model values. If you find that you often change these options at the function level, you may want to review the settings in your model and change them at the model level.

For more information about:

- Understanding model values, see *Getting Started,* Setting Up the Model Environment in the chapter "Using Your Development Environment"
- Model values you can change at the function level, see Changing Model Values later in this chapter
- Descriptions of each model value, YCHGMDLVAL, see the *Command Reference*

YABRNPT

The YABRNPT value is only for NPT generation, and enables you to choose between creations of Action Bars or DDS Menu Bars for a given function. The default is DDS Menu Bars for models created as of r5.0 of COOL:2E. For existing models upgraded to r5.0, the default is Action Bars.

We recommend that you migrate to DDS Menu Bars over time since DDS Menu Bars make use of the new OS/400 ENPTUI features, which allow the menu bars to be coded in the DDS for the display file. The Action Bars require that an external program be called to process the action bar. As a result, the DDS Menu Bars are faster, have more functionality, and create more efficient functions.

For more information about NPT user interface options, see ENPTUI in the chapter "Modifying Device Designs."

YACTCND

The Action Diagram Compound Symbols (YACTCND) model value defines the symbols used in editing and displaying compound condition expressions.

The format for modifying this design option is:

```
YCHGMDLVAL  MDLVAL(YACTCND)  VALUE('& AND | OR ^ NOT  ( ( ) ) c c')
```

For more information about compound conditions, see Entering and Editing Compound Conditions in the chapter "Modifying Action Diagrams."

YACTFUN

The Action Diagram Compute Symbols (YACTFUN) model value defines the symbols used in editing compute expressions, which include + - * / \ ( ) x. You are only likely to change these defaults if you have national language requirements. The binary code values for these symbols can map to different values, depending on the code page in use. For example, a forward slash (/) on the US code page would map to a cedilla in a French National code page.

For more information on compute expressions, see Entering and Editing Compound Conditions in the chapter "Modifying Action Diagrams."

YACTSYM

The Action Diagram Structure Symbols (YACTSYM) model value defines the symbols used in action diagrams. The shipped default is *SAA. The Action Diagram Editor and the Document Model Functions (YDOCMDLFUN) command use this design option.

YACTUPD

The Action Diagram Update (YACTUPD) model value defines the default value for the Change/create function option on the Exit Function Definition panel. The shipped default is *YES. The value *CALC sets the Change/create function option to Y only when a change to the function's action diagram or panel design is detected.

YALCVNM

The Automatic Name Allocation (YALCVNM) model value indicates whether should automatically allocate DDS and object names. The shipped default is *YES.

For more information on name allocation, see *Getting Started*— Setting Up the User Environment topic, Naming Control in the chapter "Using Your Development Environment."

YBNDDIR

Specifies a binding directory that can resolve the location of any previously compiled RPGIV modules. Use this model value while compiling RPGIV programs with the CRTBNDRPG command.

Note: For more information, see the section *The YBNDDIR Model Value* in the Chapter *ILE Programming*.

YCNFVAL

The Confirm Value (YCNFVAL) model value determines the initial value for the confirm prompt. The shipped default is *NO.

For more information on function options, see the chapter, "Modifying Function Options."

YCPYMSG

The Copy Back Messages (YCPYMSG) model value specifies whether, at program termination, outstanding messages on the program message queue are copied to the message queue of the calling program. The shipped default is *NO.

For more information on function options, see the chapter, "Modifying Function Options."

YCRTENV

The Creation Environment (YCRTENV) model value determines the environment in which you intend to compile source is the iSeries. The shipped default is the iSeries.

For more information about:

- Controlling design, Setting Up the Model Environment in the chapter "Using Your Development Environment," in *Getting Started*
- Environments, see Managing Your Work Environment—Generating and Implementing Applications in the chapter "Preparing for Generation and Compilation"

YCUAEXT

The CUA Device Extension (YCUAEXT) model value determines whether the text on the right side text is used for device designs. The shipped default is *DEFAULT, which results in no right text and no padding or dot leaders.

The YCUAEXT value, *****C89EXT (for CUA Text), provides CUA design features on top of those which the model value YSAAFMT provides, such as defaulting and alignment of right side text, padding or dot leaders to connect fields with field text, and prompt instruction lines on all device function types.

For more information on field attributes and right side text defaults, see the chapter, "Modifying Device Designs," Device Design Conventions and Styles.

YCUAPMT

The CUA Prompt (YCUAPMT) model value controls the CUA prompt (F4). If enabled, this design option enables end users to request a list display of allowed values by pressing F4. The value *CALC provides additional F4 functionality by processing the CALC: user points in the function where F4 is pressed—for example, to provide Retrieve Condition functionality.

The default value for YCUAPMT is *****MDL. This value directs to enable the CUA prompt at the model level if the YSAAFMT model value is *CUATEXT or *CUAENTRY.

For more information about:

- Setting display defaults, see the chapter, "Modifying Device Designs"
- On the *CALC value, see the *Command Reference*, the YCHGMDLVAL command

YCUTOFF

The Year Cutoff (YCUTOFF) model value specifies the first of the hundred consecutive years that can be entered using two digits. It is specified as 19YY, which represents the hundred years: 19YY to 20YY-1. Values between YY and 99 are assumed to be in the 20th century; namely, 19YY to 1999; values between 00 and YY-1 are assumed to be in the 21st century; namely 2000 to 20YY-1. The default is 1940. The YCUTOFF value is retrieved at run time and applies to all date field types: DTE, D8#, and DT#.

YDATFMT

The Date Format (YDATFMT) model value works in conjunction with the model value YDATGEN. If YDATGEN is *VRY. The setting for YDATFMT determines the order of the date components at run time; for example, MMDDYY or DDMMYY.

YDATGEN

The Date Validation Generation (YDATGEN) model value determines the type of date editing source code generates. With YDATGEN set to *VRY, you can change the date format for an application with the YDATFMT model value. No recompilation of functions is necessary.

YDBFGEN

The Database Implementation (YDBFGEN) model value defines the method for database file generation and implementation: DDS, SQL or DDL.

YDDLDBA

The Database Access Method (YDDLDBA) model value specifies a method of accessing the database (RLA or SQL) when a function's Generation Mode option is set to A(ACPVAL) or M(MDLVAL), which resolves to DDL type.

***RLA**

Specifies that the external function generates with RLA access.

***SQL**

Specifies that the external function generates with SQL access.

Note: To generate or regenerate a function with RLA code for DDL database, set the YSQLVNM model value to *DDS or *LNG or *LNT, or *LNF and set the YDDLDBA model value to *RLA.

YDFTCTX

The Parameter Default Context (YDFTCTX) model value specifies the default context to use for a given function call in the action diagram editor when no context is supplied: LCL or WRK. The shipped default is *WRK.

YDSTFIO

The Distributed File I/O Control (YDSTFIO) model value, together with model value YGENRDB, provides DRDA support. The shipped default value is *****NONE, indicating that will not generate distributed functionality.

For more information on DRDA, see Generating and Implementing Applications in the chapter "Distributed Relational Database Architecture."

YERRRTN

For RPG-generated functions, the Error Routine (YERRRTN) indicates whether will generate an error handling routine (*PSSR) in the program that implements the function. The shipped default value is *NO.

Note: For EXCUSRPGM functions, this value specifies whether an error-handling routine should be generated in the calling program to check the value of the *Return code on return from the EXCUSRPGM (if the EXCUSRPGM does not have the *Return code as a parameter, this check will not be generated).

YEXCENV

The call to a CL program that implements an EXCMSG function uses an OS/400 program. The Execution Environment (YEXCENV) model value determines the default environment, QCMD (OS/400), in which Execute Message (EXCMSG) functions execute.

For more information about:

- EXCMSG functions, see Function Types, Message Types, and Function Fields in the chapter, "Defining Functions"
- QCMD and QCL, see Generating and Implementing Application*s*— Managing Your Work Environment in the chapter "Preparing for Generation and Compilation"

YGENCMT

The time required to generate a function can be significantly improved if comments are not required for the generated source code. The YGENCMT model value lets you specify whether or not comments are placed in the resulting generated source code. You can specify that all comments (*ALL), only header comments (*HDR), or no comments (*NO) be generated. The shipped default is *ALL.

YGENHLP

The Generate Help Text (YGENHLP) model value allows you to specify whether help text is generated for a particular function. You can specify generation of the function only (*NO), help text only (*ONLY), or both the function and help text (*YES). This value can be overridden at the function level. The shipped default is *YES.

YGENRDB

The Generation RDB Name (YGENRDB) model value provides the DRDA support for specifying a default database. When you execute the CRTSQLxxx command, this database is used in creation of the SQL package. The default value for YGENRDB is *NONE, which means that DRDA compilation is not enabled.

For more information about DRDA, see Generating and Implementing Applications in the chapter "Distributed Relational Database Architecture."

YHLLGEN

The HLL to Generate (YHLLGEN) model value identifies the default HLL type for new functions. The HLLGEN parameter on YCRTMDLLIB sets this model value.

Note: To default to the value for model value YSYSHLL, select *SYSHLL for the parameter HLLGEN.

YHLLVNM

The HLL Naming Convention (YHLLVNM) model value determines the HLL conventions for new function names. The HLLVNM parameter on YCRTMDLLIB sets this model value. The default is *RPGCBL, allocation of names that both RPG and COBOL compilers support.

For more information about converting HLLs, see Generating and Implementing Applications—Converting a Model from One HLL to Another, in the chapter "Preparing for Generation and Compilation."

YHLPCSR

The Generate Cursor Sensitive Text (YHLPCSR) model value gives you the option of generating your function with cursor-sensitive help. That is help- specific to the context (cursor position) from which the end user requests it. The shipped default is Y (Yes).

YLHSFLL

The Leaders for Device Design (YLHSFLL) model value refers to the symbols to usedas leaders between text and input or output fields on panels. The shipped default value is *SAA, for SAA default left-hand filler characters. You can change any of these characters using the YCHGMDLVAL command.

YLVLCHK

The Generate IDX with LVLCHK(*YES) (YLVLCHK) model value specifies whether an Index (SQL or DDL), when generated with the RCDFMT keyword in it, is created with LVLCHK(*YES). The possible values are *NO and *YES. The shipped default is *NO.

If YLVLCHK is specified as *NO, then existing defaults around LVLCHK are retained when an SQL or DDL index is generated and created. The existing defaults for the LVLCHK attribute in the case of SQL and DDL are as follows.

- When a table or view is generated and created, it is generated with LVLCHK(*YES), irrespective of the existence of the RCDFMT keyword.
- When an index (SQL or DDL) is generated and created without the RCDFMT keyword, it is created with LVLCHK(*YES).
- When an index (SQL or DDL) is generated and created with the RCDFMT keyword, it is created with LVLCHK(*NO).

If YLVLCHK is specified as *YES (in addition to YSQLFMT set as *YES), upon subsequent generation and creation of an index (SQL or DDL), the index is created with LVLCHK(*YES).

Note: If YLVLCHK is set to *YES (along with YSQLFMT set to *YES), upon re-generation of the access path, an additional line "Y* CHGLF LVLCHK(*YES)" is generated into the header portion. This informs YEXCSQL to create the corresponding index with LVLCHK(*YES). For any other combination of YLVLCHK and YSQLFMT, there is no change to the existing processing.

YNPTHLP

The NPT Help Default Generation Type (YNPTHLP) model value determines the type of help text to generate for NPT functions. All functions are NPT unless the functions are being generated for a GUI product. The types are UIM or TM. The shipped default for YNPTHLP is *UIM.

For more information about UIM support, see Objects from UIM Generation in the chapter "Implementing Your Application."

YNLLUPD

The Null Update Suppression (YNLLUPD) model value sets the default for whether CHGOBJ functions update or release the database record if the record was not changed. This can be overridden with a matching function option. The shipped default is *NO.

- *NO

  CHGOBJ functions do not check whether the record has changed before updating the database. In other words, null update suppression logic is not generated in CHGOBJ functions.

- *AFTREAD

  CHGOBJ checks whether the record changed between the After Data Read and Data Update user points.

- *YES

  CHGOBJ checks whether the record changed both after the Data Read and after the Data Update' user points.

For more information about:

- CHGOBJ database function, refer to the chapter, "Defining Functions"
- Suppressing null updates, see Understanding Contexts, PGM in the chapter "Modifying Action Diagrams"

YOBJPFX

The Member Name Prefix (YOBJPFX) model value specifies the prefix (up to two characters) uses to generate object names. The shipped default is UU. If you change this prefix, do not use Q, #, and Y because they are reserved characters for .

For more information about naming prefixes, see Creating an Model in the chapter "Creating and Managing Your Model" in *Getting Started*.

YPMTGEN

The Prompt Implementation (YPMTGEN) model value specifies whether the text on your device designs is generated, implemented, and stored in a message file, making it available for national language translation. The shipped default value is *OFF. The parameter PMTGEN on the YCRTMDLLIB command initially sets the YPMTGEN model value.

For more information about:

- National Language Support, see Generating and Implementing Applications in the chapter "National Language Support"
- YCRTMDLLIB, see the *Command Reference*

YPMTMSF

The Prompt Message File (YPMTMSF) model value specifies the message file into which device text message IDs are stored. retrieves the messages from this message file at execution time.

For more information about National Language Support, see Generating and Implementing Applications in the chapter "National Language Support."

YPUTOVR

The DDS Put With Override (YPUTOVR) model value is a function generation option. It enables you to specify use of the DDS PUTOVR keyword in the generated DDS. This keyword, in effect, reduces the amount of data that needs transmission between the system and its workstations. Its use can improve performance, particularly on remote lines.

For more information about system performance, see the *AS/400 Programming: Data Description Specifications Reference*.

YRP4HSP

Used by the RPGIV Generator for the contents of the Control (H) specification for objects of type *PGM. The allowed values are any RPGIV H-specification keywords, for example:

- DATEDIT(*YMD) DEBUG(*YES)
- DATFMT(*YMD)

Note: If you need to enter a value that is longer than 80 characters, you should use the command YEDTDTAARA DTAARA(YRP4HSPRFA)

YRFSACT

The Refresh Action Diagram on Entry (YRFSACT) model value specifies whether the YCHKFUNACT processor must be called during Action Diagram load to refresh the Action Diagram. The possible values are *NO and *YES. The shipped default is *NO.

Note: For more information about what is changed when an Action Diagram is refreshed, see the details of the YCHKFUNACT command in the *Command Reference Guide*.

YRP4HS2

Used by the RPGIV Generator for the contents of the Control (H) specification for objects of type *MODULE. The allowed values are any RPGIV H-specification keywords, for example:

- H DATFMT(*YMD)
- DATEDIT(*YMD) DEBUG(*YES)
- Note: If you need to enter a value that is longer than 80 characters, you should use the command YEDTDTAARA DTAARA(YRP4HS2RFA)

YRP4SGN

The RPGIV generator includes some source generation options that you can set at a model level. These options are in the model value YRP4SGN in a data area called YRP4SGNRFA (RPGIV source generation options). YRP4SGNRFA is a 16-character data area.

Note: For more information, see the section *Model Value YRP4SGN* in the Chapter *ILE Programming*.

YSAAFMT

The SAA Format (YSAAFMT) model value controls the design standard for panel layout. This standard can be CUA. *CUAENTRY is the shipped default.

The DSNSTD parameter on the YCRTMDLLIB command controls the initial YSAAFMT value. You can override the header or footer for a function from the Edit Function Options panel. You can also change the value of YSAAFMT using the YCHGMDLVAL command.

For more information about:

- Using YSAAFMT options, see Device Design Conventions and Styles in the chapter "Modifying Device Designs"
- YSAAFMT values, see YCHGMDLVAL in the *Command Reference*

YSFLEND

The Subfile End (YSFLEND) model value controls whether the + sign or More. . . is displayed in the lower right location of the subfile to indicate that the subfile contains more records. This feature is available for all subfile functions. The shipped default is *PLUS. To change to *TEXT everywhere, change the model value and regenerate your subfile functions.

The setting of YSFLEND is resolved in the following areas:

- Generated applications
- Device designs
- Animated functions
- Function documentation (YDOCMDLFUN)

YSHRSBR

The Share Subroutine (YSHRSBR) model value specifies whether generated source code for subroutines are shared and whether the subroutine's interface is internal or external. This model value and its associated function option are available on the CHGOBJ, CRTOBJ, DLTOBJ, RTVOBJ, and EXCINTFUN function types.

YSNDMSG

For new functions, the Send Error Message (YSNDMSG) model value specifies whether to send an error message for only the first error found or for every error. In either case, outstanding messages clear when the end user presses Enter. The shipped default value is *NO, do not send all error messages; send only the first error message.

YSQLCOL

The Generate SQL Collection/Library Name (YSQLCOL) model value specifies whether a hard coded SQL Collection/Library name should be generated for tables, indexes and views. The possible values are *YES and *NO. The shipped default is *YES.

If YSQLCOL is specified as *YES, the SQL Collection/Library specified for the YSQLLIB model value is generated into the tables, indexes and views, by default, as is the case now. Subsequently when YEXCSQL is executed to create tables, indexes and views, they are created into the hard coded SQL Collection/Library. If YSQLCOL is specified as *NO, the SQL Collection/Library specified for the YSQLLIB model value is not generated into the tables, indexes and views. However, when YEXCSQL is executed subsequently, the tables, indexes and views are generated into the SQL Collection/Library specified for the YSQLLIB model value.

Note: If YSQLCOL is set to *NO and the access paths are generated, another change that can seen in the source, apart from the absence of hard coded SQL collection/Library name is, the previously generated Z* line "Z* YEXCSQL NAMING(*SQL)" is now generated as "Z* YEXCSQL NAMING(*SYS)".

YSQLFMT

The Generate SQL RCDFMT clause (YSQLFMT) model value specifies whether the RCDFMT keyword must be generated for SQL tables, views, and indexes. The possible values are *NO, and *YES. The shipped default is *NO.

If YSQLFMT is specified as *NO, the record format is the same as the table, index, or view name (if YSQLVNM (*DDS) is specified) or will be generated by the system (if YSQLVNM(*SQL) is specified). If YSQLFMT is specified as *YES, the RCDFMT value is calculated using the same rules as are used when DDS files are generated.

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

Important!

If YSQLFMT is set to *YES or *NO and a DDL index is generated and created, the index is created with LVLCHK(*NO).

If YSQLFMT is set to *YES and an SQL index is generated and created, the index is created with LVLCHK(*NO).

If YSQLFMT is set to *NO and an SQL index is generated and created, the index is created with LVLCHK(*YES).

If you want to change the LVLCHK attribute of the index to LVLCHK(*YES), the model value YLVLCHK must be set to *YES, and the corresponding index-related access path must be regenerated and re-created. Upon regeneration of the access path, an additional line "Y* CHGLF LVLCHK(*YES)" is generated in the header portion, which informs YEXCSQL to create the corresponding index with LVLCHK(*YES).

Note: If YSQLFMT is set to *YES, YLVLCHK is set to *YES and RUNSQLSTM is used to create an index (SQL or DDL), the index would still be created with LVLCHK(*NO). The current functionality does not cater to the RUNSQLSTM command. Tables and Views (SQL) and Tables (DDL) are created with LVLCHK(*YES) by default, irrespective of the YSQLFMT model value. Therefore, YSQLFMT and YLVLCHK model values have no effect on tables and views regarding the LVLCHK attribute.

YSQLLCK

The SQL Locking (YSQLLCK) model value specifies whether a row to be updated is locked at the time it is read or at the time it is updated. The default is *UPD, lock rows at time of update.

YSQLVNM

The SQL Naming (YSQLVNM) model value specifies whether to use the extended SQL naming capability. The valid values are:

***DDS**

Use DDS names. The shipped default.

***SQL**

Use the names of the CA 2E objects in the model.

***LNG**

Use the long names of the CA 2E objects in the model along with the DDS or implementation names.

***LNF**

Use the long field names of the CA 2E objects in the model along with the DDS or implementation names.

***LNT**

Use the long table names of the CA 2E objects in the model along with the DDS or implementation names.

Note:

If a table that has a valid system name (less than or equal to 10 bytes in length), is generated with YSQLVNM model value set as *LNG or *LNT, and when you set the Enhance SQL Naming option on the Edit File Details panel to **Y** and then generate the source, the table is created with (underscores) "_"s and "TABLE" as suffix, so that the name of the table becomes more than 10 char long.

Examples:

- CUSTOMER is generated as CUSTOMER_TABLE along with its 2E implementation name.
- CUST is generated as CUST__TABLE along with its 2E implementation name.
- C is generated as C_____TABLE along with its 2E implementation name.
- To generate or regenerate a function with RLA code for DDL database, set the YSQLVNM model value to *DDS or *LNG or *LNT, or *LNF and set the YDDLDBA model value to *RLA.

YSQLWHR

The SQL Where Clause (YSQLWHR) model value specifies whether to use OR or NOT logic when generating SQL WHERE clauses. The default is *OR.

For more information about the YSQLLCK and YSQLWHR model values, see the appendix "SQL Implementation" in *Getting Started*.

YWSNGEN

The Workstation Generation (YWSNGEN) model value defines whether interactive functions operate on non-programmable terminals (NPT) or on programmable workstations (PWS) communicating with an iSeries host. For programmable workstations, you also specify the PC runtime environment. YWSNGEN can be overridden by a function option. The possible values are:

- *NPT

  Generates functions for non-programmable terminals (NPT) communicating with an iSeries host system.

- *GUI

  Generates functions for non-programmable terminals together with a Windows executable running in a Windows environment under emulation to the host.

- *JVA

  Generates functions for non-programmable terminals together with a Windows executable running in a Windows environment under emulation to the host and a Java executable running in a Windows environment using a Web browser with emulation to the host.

- *VB

  Generates functions for non-programmable terminals together with a Visual Basic executable running in a Windows environment under emulation to the host.

User Interface Manager (UIM)

Three model values provide options for UIM help text generation:

- The Bidirectional UIM Help Text (YUIMBID) model value provides national language support of languages with both left-to-right and right-to-left orientations
- The Default UIM Format (YUIMFMT) model value provides paragraph or line tags
- The UIM Search Index (YUIMIDX) model value provides search for the index name derived from Values List prefix

Window Borders

Three model values provide design options for the appearance of the border on windows:

- The Window Border Attribute (YWBDATR) model value provides shadow or no shadow
- The Window Border Characters (YWBDCHR) model value provides dot/colon formation
- The Window Border Color (YWBDCLR) model value provides CUA default (Blue) or another color

For more information on Modifying Windows, see Editing Device Designs in the chapter "Modifying Device Designs."

Changing Model Values

This topic summarizes changing model values for a function of your model.

Function Level

You can override model value settings that determine function options at the function level from the Edit Function Options panel. You can reach this panel by zooming into the function from the Edit Functions panel, then pressing F7 (Options) from the Edit Function Details panel.

The model values that have corresponding fields on the Edit Function Options panel are:

Values	Meaning
YABRNPT	Create Action Bars or DDS Menu Bars for NPT generation
YCNFVAL	Initial value for the confirm prompt
YCPYMSG	Copy back messages
YDBFGEN	Generation mode
YDSTFIO	Distributed file I/O control
YERRRTN	Generate error routine
YGENHLP	Generate help text
YNPTHLP	Type of help text to be generated
YPMTGEN	Screen text implementation

Values	Meaning
YSNDMSG	Send all error msgs (messages)
YSFLEND	Subfile end
YWSNGEN	Type of workstation

For more information about:

- Options applicable to each function see Function Types, Message Types, and Function Fields in the chapter, "Defining Functions"
- On step-by-step procedures, see Specifying Function Option in the chapter, "Modifying Function Options"

Model Level

You can change the setting of a model value for your model by executing the Change Model Value (YCHGMDLVAL) command. Be sure to use YCHGMDLVAL, rather than the OS/400 command, Change Data Area (CHGDTAARA). Changing model values involves more than changing data areas; many internal model changes are made by YCHGMDLVAL.

You should always exit from your model entirely when changing model values. Although the command can appear to run successfully while you are in the model, there is no guarantee that a full update has taken place.

For more information on using the YCHGMDLVAL command, see the *Command Reference*.

Changing a Function Name

To change a function name

1. Select the file. From the Edit Database Relations panel, type **F** next to the desired file and press Enter.

   The Edit Functions panel appears, listing the functions for that file.

2. Zoom into the function details. Type **Z** next to the desired function and press Enter.

   The Edit Function Details panel appears, showing the function name at the top.

3. Request to change the function name. Press F8 (Change name).

   The function whose name you want to change appears underlined on the panel.

4. Change the function name. Type the desired name. If you want, you can change any other underlined names to better correspond to the new function name . Press Enter, then F3 to exit.

Function Key Defaults

assigns the standard function key usage of your design standard. You can specify additional function keys in action diagrams or modify existing function key default values.

For more information about function keys, see the chapter "Modifying Device Designs."

The following table shows the shipped device design defaults for the iSeries.

Meaning	iSeries default
*Help	F01/HELP
Prompt	F04
Reset	F05
*Change mode request	F09
*Change mode to Add	F09
*Change mode to Change	F09
*Delete request	F11
*Cancel	F12
*Exit	F03
*Exit request	F03
*Key panel request/*Cancel	F12
*IGC support	F18
Change RDB	F22
*Previous page request	F07/ROLLDOWN
*Next page request	F08/ROLLUP

The default is determined by the design standard selected. The iSeries default is used if the YSAAFMT model value is set to *CUATEXT or *CUAENTY.

Chapter 3: Adding Access Paths

This chapter describes how to build an access path. A description of the types of access paths is provided in the Recognizing the Basic Properties of Access Paths section of the "Access Paths: An Introduction" chapter.

This section contains the following topics:

Before Adding (see page 53)
Edit File Details (see page 54)
Adding an Access Path (see page 55)

Before Adding

When you create and define your file, CA 2E automatically creates the following three default access paths for the file: physical, update, and retrieval. These default access paths have default values equal to those of the model values. Generally, the access path options are set when you create your model. However, you can change the values for the access paths at other times.

For more information:

- On how to change the model values or values for a specific access path, see the instructions in this guide's "Setting Default Options for Your Access Paths" and "Modifying Access Paths" chapters.
- On model values, see the YCHGMDLVAL command in *CA 2E Command Reference Guide*.

Adding access paths to your design model allows CA 2E to provide you with specific views of the data in the physical files for your application. These views allow you to retrieve data in a format that most suits your needs with less system overhead.

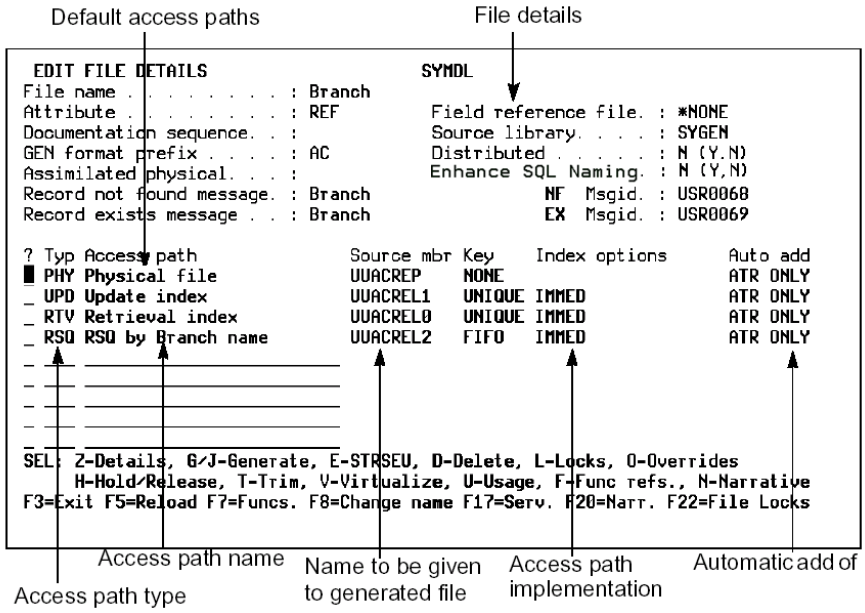
For more information on tailoring your access paths, see the "Tailoring for Performance" chapter in this guide.

Edit File Details

The Edit File Details panel is where you add new access paths to your model, modify existing access paths, or view the list of the existing access paths for a selected file.

1. **Zoom into the file.** Type **Z** next to any relation for the selected file on the Edit Database Relations panel and press Enter. Alternatively, select option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays with a list of the access paths built over that file, the source member names, key sequence, and implementation attributes:



2. **Review the details.** In the top half of this panel, view the details for the file.
3. **Review existing access paths.** In the bottom half of the panel, view the list of all access paths already defined for the file, including the three default access paths (PHY, RTV, and UPD).

Adding an Access Path

A list of the available types of access paths follows:

- Physical (PHY)
- Update (UPD)
- Retrieval (RTV)
- Resequence (RSQ)
- Query (QRY)
- Span (SPN)

In addition to the three default access path types created when the file is defined, you can create additional update and retrieval access paths or resequence, query, or span access paths using the instructions that follow.

To add any access path, except a PHY access path, use the following steps:

1. **Zoom into the file.** Type **Z** next to any relation for the selected file on the Edit Database Relations panel and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays with a list of any existing (default) access paths for the file.

2. **Add the new access path.** At the next available line on this panel (use the Roll Up key, if necessary), type the access path value type in the Typ column and the name of the access path in the Access Path column and press Enter.

   Your options for the Typ column are UPD**,** RTV**,** RSQ**,** QRY, and SPN.

   For more information on each access path type and when to use it, see the "Access Paths: An Introduction" chapter in this guide.

   CA 2E displays the following information for the new access path:

   - Source member name in the Source Mbr field
   - Unique and non-unique key selection in the Key field
   - Maintenance selection (immediate, delay, and rebuild) in the Index field

3. **View the access path details.** Type **Z** next to the new access path and press Enter.

   The Edit Access Path Details panel displays.

At this point, you have defined an access path to the model.

For more information on making changes to the access paths, such as to the keys, or to add virtuals, see the instructions in the "Modifying Access Paths" chapter of this guide.

For additional instructions that are commonly used when adding RSQ, QRY, and SPN type access paths, refer to that material in the following sections.

Adding a Physical (PHY) Access Path

Unlike other access path types, you cannot add a physical access path to an existing file. The PHY access path corresponds to an arrival sequence i OS physical file or an SQL table. When you create and define your file and then zoom into the file from the Edit Database Relations panel, CA 2E automatically creates a PHY access path as one of the three defaults.

For more information on creating and defining a file and instructions on assimilating a physical file, see the "Creating/Defining Your Data Model" and "Assimilation" chapters in *Defining a Data Model*.

Adding a Resequence (RSQ) Access Path

Once you add the RSQ access path, follow the instructions in the preceding section, Adding an Access Path. At the Edit Access Paths Format Entries panel, you can also identify the new key sequence order:

```
 EDIT ACCESS PATH FORMAT ENTRIES      SYMDL
File name . . . . . : Customer                Attribute . : REF
Access path name. . . . . : RSQ by customer name   Type. . . . : RSQ
Format text . . . . . : Customer
Based on. . . . . : Customer                  Format No . :   1

                                    GEN                Key    Altcol Ref
? Field                             Name       Type    no. Dsc seq   cnt
_ Customer code             CDE     AECD         A     ___  _          1
_ Customer name             TXT     AFTX         K     __1  _          1
_ Customer address          TXT     AGTX         A     ___  _          1
_ Customer city             TXT     AHTX         A     ___  _          1
_ Customer country          TXT     AITX         A     ___  _          1
_ Customer postal code      CDE     AFCD         A     ___  _          1
_ Customer phone number     NBR     ACNB         A     ___  _          1
_ Customer status           STS     ACST         A     ___  _          1
_ Customer credit limit     NBR     ADNB         A     ___  _          1
_ Customer state            TXT     AOTX         A     ___  _          1
_ Customer Allow Credit     STS     AGST         A     ___  _          1

SEL: 2-Field details, L-Locks.
F3=Exit  F7=Relations
```

Changing the Key Sequence

1. Remove the old key sequence order from the Key no. column.
2. Add the new key sequence order in the Key no. column. Ascending or descending sequence can also be specified for each field in the key sequence.

Note: Lower key numbers indicate a higher key order or a major key. The key sequence numbering should be unique.

Adding a Query (QRY) Access Path

Once you add the QRY access path, following the instructions in the preceding section, Adding an Access Path, CA 2E creates and generates default values for three auxiliary objects for DDS only. Each object type has its own source, either DDS or CL, that is held in the appropriate source file in the generation library. These objects include:

- Logical file, which is based on the physical file whose data is being referenced.
- Physical file, which never contains data and is used to define a record format and keys to any HLL program generated for a function based on the QRY access path.
- CL program, which executes the Open Query File **(**OPNQRYF**)** command. This command is called at execution by any program generated for a function based on the QRY access path.

For more information on how the auxiliaries are implemented, see the Implementation table in the "Generating and Compiling" chapter in this guide.

Adding a Span (SPN) Access Path

The SPN access path is a multi-format view that allows views of two or more formats. The SPN access path can only be specified over files with Owned by or Refers to relationships. The SPN access path must be created over an owning or referred to file.

Once you add the SPN access path, following the instructions in the preceding section, Adding an Access Path, you can also add the new format entries:

1. **View the SPN access path.** Press F9 from the Edit Access Path Details panel to select formats.

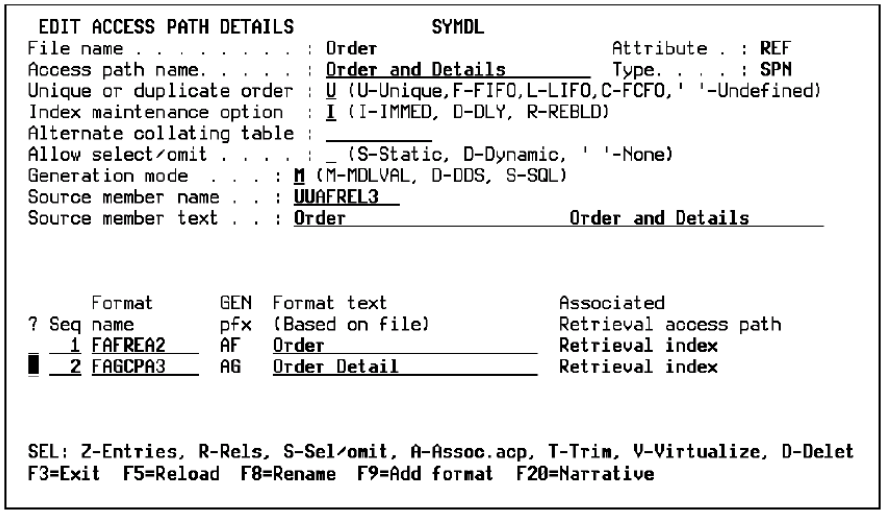
   The Display Access Path Formats panel displays where you select the format.

2. **Select the primary format.** Type an **X** next to the primary format for the specified file and press Enter.

   Note: You always select the Refers to or Owned by file first.

   The Edit Access Paths Details panel redisplays which shows the format selection indicated by the number in the Seq column next to the format.

3. **Repeat the above process for each format.** Follow Steps 1 and 2 to select the secondary format.



Note: The keys of the second format must include all of the keys of the first format, in the same order. Any additional keys on the second format must be sequenced after the first format keys.

Once you have completed the preceding steps, you have added the SPN access path. To view the entries on the format and change the key order, at the Edit Access Path Details panel, perform the following.

4. **Zoom into the format.** Type **Z** next to the format name of the format.

   The Edit Access Path Format Entries panel displays with a list of the details (fields), field type, source name, type, key number, alternate collating sequence, and reference count for your format.

5. **Change the key order number for the format**. At the Edit Access Path Format Entries panel, type the different sequence for the key order numbers and press Enter.

   CA 2E stores the key sequence.

Chapter 4: Modifying Access Paths

This chapter explains how to modify an existing access path; it contains nine topics that identify where you can modify the access paths.

Once you add access paths to your model, you can modify the details, values, or options you selected. CA 2E provides sensible defaults for the selections and protects the values that should not be changed. However, in creating your own application, you may need to change some of the defaults based on your own organization's application design conventions. These procedures walk you through the process.

This section contains the following topics:

Before You Begin (see page 61)
Before Modifying (see page 62)
Building Distributed Relational Database Applications (see page 66)
Modifying Access Path Details (see page 67)
Modifying Access Path Format Entries (see page 75)
Modifying Access Path Relations (see page 79)
Modifying Virtual Field Entries (see page 82)
Choosing Select/Omit Criteria (see page 86)
Changing a Referenced Access Path (see page 91)
Modifying Access Path Auxiliaries (see page 94)
Understanding Access Path Auxiliaries (see page 95)

Before You Begin

When an access path is created, it is created with defaults based on your model values. Some of the model values are specifically used with access paths. By changing the options for one of these model values, it is possible to modify the access paths if your application design warrants the changes.

For more information about:

- The types of access paths available in CA 2E, see the chapter "Access Paths: An Introduction"
- The default values for access paths and instructions on how to change them, see the chapter "Setting Default Options for Your Access Paths"
- How to add access paths to your model, see the chapter "Adding Access Paths"
- How to generate and compile an access path, see the chapter "Generating and Compiling"

Before Modifying

This topic describes navigational techniques and aids used in the subfile selection area of the left margin of the panel and the selection options found in the command text area at the bottom of the panels. This topic also identifies procedures for adding and removing virtual fields, holding and locking access paths, and displaying access path references.

Navigational Techniques and Aids

CA 2E provides you with ways to navigate to different panels other than by using function keys. CA 2E identifies a number of standard line selection values usually found in the command text area at the bottom of the panel.

For example, you can use Z to zoom into a file or D to delete. When you place one of these selections next to a file in the subfile selection area of your Edit Database Relations panel and press Enter, CA 2E executes the action.

For more information about navigation, see the following:

- The Navigation Facilities section of the chapter "Using Your Model" in the *Administrators Guide*
- The Editing Model Object Lists section of the chapter "Managing Model Objects" in *Generating and Implementing Applications*

Automatic Add Options

Each access path initially contains all of the relations for the file on which it is based, but none of the virtuals. If you add a new relation to a file, the effect on the access path is controlled by the Automatic Add setting. Following are the three Auto Add settings:

Setting	Description
*Attr Only	Only attributes are added (physical file changes).
*All	Both virtuals and attributes are added.
*Held	No changes are made to the access path. This prevents level checking.

Changing the Auto Add Setting

Do the following to change the Auto Add setting:

1. Zoom into the file.

   Type **Z** next to any relation for the selected file on the Edit Database Relations panel and press Enter.

   The Edit File Details panel displays.

2. Toggle the Auto Add setting.

   At the Edit File Details panel, type **H** next to the selected access path and press Enter**.**

   The allowable Auto Add settings and defaults are as follows.

Access Path	Atr Only	Held	All
PHY	Default	No	No
UPD	Default	Yes	No
RTV	Default	Yes	Yes
RSQ	Default	Yes	Yes
QRY	Default	Yes	Yes

   A refreshed panel displays with the indicator ALL, ATR ONLY, or HELD showing that the selected access path has changed its Auto Add setting.

Note: The key relations (Known by**,** Owned by, and Qualified by) are added to the access path regardless of the Auto Add setting. These relations must always be present on all access paths for the file.

If a relation is added to an access path and functions that use the access path already exist, the entries that result from resolving the new relations are added to any device designs (reports and panels) used by the functions. For example, if the entries are not key fields on the access path, they will be added as hidden fields to the device designs.

If they are key fields on the access path, they are added as input fields to the device designs. The device designs may therefore require readjustment before they can be successfully regenerated.

For more information about readjusting the device designs, see the "Modifying Device Designs" chapter of *Building Applications*.

Trimming an Access Path

You can remove all virtuals from an access path using the Trim option.

To trim an Access Path

1. Zoom into the file.

   Type **Z** next to any relation for the selected file on the Edit Database Relations panel and press Enter.

   The Edit File Details panel displays.

2. Trim the Access Path.

   At the Edit File Details panel, enter T next to the selected access path and press Enter. Repeat the action to confirm.

   Note: You can also trim a format using the Edit Access Path Details panel.

   If you trim an access path that has Auto Add set to ALL, Auto Add is reset to ATR ONLY.

Virtualizing an Access Path

You can add all virtual fields to an access path using the Virtualize option.

To virtualize an access path

1. Zoom into the file.

   Type **Z** next to any relation for the selected file on the Edit Database Relations panel and press Enter.

   The Edit File Details panel displays.

2. Virtualize the Access Path.

   At the Edit File Details panel enter V next to the selected access path and press Enter. You need to repeat the action to confirm.

   Note: You can also virtualize a format of an access path using the Edit Access Path Details panel.

   If you virtualize an access path that has Auto Add set to Held, it will be reset to All.

Locking an Access Path

CA 2E supports two types of object locks: temporary and permanent.

Temporary Locks

Temporary locks are imposed automatically by CA 2E to prevent two users from working on the same object at the same time. These locks are normally cleared by CA 2E when the object is no longer in use. Temporary locks hold the access paths so that they can be changed only by one user at a time. This lock is automatically placed on each CA 2E object while it is in use.

If you are using an object and leave the model abnormally, such as with a subsystem termination or power failure, a temporary lock can be left on the object. This lock now prevents you from accessing the object. To remove this inactive temporary lock, select L from the bottom of the Edit File Details panel to view the locks in your model. CA 2E automatically removes the locks no longer required.

Permanent Locks

Permanent locks can be placed by the designer on any object to prevent any modification or generation of that object. These locks stay in effect, even if the object and model are not in use, until they are removed by the designer. Permanent locks can be placed on CA 2E objects. A permanent lock prevents users from changing a CA 2E object. For designers to add and remove permanent locks, they must have *OBJOWN rights to the YMDLLIBRFA data area.

For more information about locks, see the section Locking Objects in the "Using Your Model" chapter of the *Administration Guide*.

Displaying Usages for Access Paths

To view a list of where each access path is used in the model view the usages, as follows:

1. Access the Edit File Details panel.
2. Enter U or F next to the selected access path and press Enter.

The Display Model Usages panel displays with a list of all model objects that use the selected access path.

For more information about usages, see the Impact Analysis section of the "Managing Model Objects" chapter in the *Generating and Implementing Applications* guide.

Building Distributed Relational Database Applications

This topic discusses Distributed Relational Database Architecture (DRDA). DRDA is IBM's architecture that provides access to data distributed across various machines. The objective of DRDA is to provide the user, via a high-level programming language, access to relational databases and files that reside on multiple machines.

Specifying Distributed Files

CA 2E lets you flag any files you create for DRDA on the Edit File Details panel. The distributed flag indicates whether the file is distributed or local. The field has two values; Y means it is distributed and N means it is local. The default is N. YGENRDB is the model value that specifies the relational database used when distributed functions are used. If it is set to *NONE, the flag is ignored regardless of its content.

AY implies that any access paths that are based on this file can conceivably exist on a remote machine. Files initially designed and created as Distributed N can be changed to Distributed Y**,** just as files created as Distributed Y can be changed to Distributed N.

Note: Any functions built over this file must be regenerated and recompiled to contain the distributed functionality.

For more information about Distributed Relational Database Architecture see the "Distributed Relational Database Architecture" chapter of the *Generating and Implementing Applications* guide.

Modifying Access Path Details

This topic discusses use of the Edit Access Path Details panel including specifying unique/duplicate key retrieval sequence, access path maintenance, alternate collating sequence, select/omit criteria, generation mode, and changing source member text and names.

Access paths are implemented as separate i OS objects. You can specify various implementation options for each access path such as the i OS object name for the logical file and whether the access path maintenance will be Rebuild, Delay, or Immediate. CA 2E provides defaults for those options and protects the values that should not be changed.

For instance, i OS requires that immediate access path maintenance be specified if you specify the DDS UNIQUE keyword. The values allowed for the implementation details depend on the access path. A table of i OS access path implementation attributes follows.

Access Path Type	(1) (2) Unique or Dup Key Sequence (DDS only)	(3) Access Path Maint.	(4) Alt Col (DDS only)	(5) Selection
PHY Physical	-	-	-	-
UPD Update (default)	U	I	-	-
UPD Update	U/L/F/ /C	I,D,R	-	-
RTV Retrieval (default)	U	I	-	S/D
RTV Retrieval	U/L/F/ /C	I,D,R	-	S/D
RSQ Resequence	U/L/F/ /C	I,D,R	Yes	S/D
QRY Query	F	I,D,R	-	D
SPN Span	U/L/F/ /C	I,D,R	Yes	S/D

The following legend applies for the access path types:

(1) Unique key status (DDS unique keyword or SQL unique keyword with create index statement). U indicates unique; if not unique, see note (2). The default UPD and RTV access paths must be unique.

(2) Duplicate key sequence for DDS only (L=LIFO, F=FIFO,' '=undefined, C=FCFO)

(3) i OS access path maintenance (I=*IMMED, R=*REBLD, D=*DLY);

for QRY access paths (I=*FIRSTIO, D=*MINWAIT, R=*ALLIO). See the following table.

(4) Alternative collating sequence table for DDS only (DDS ALTCOL keyword)

(5) Selection type (S=static, D=dynamic) (DDS DYNSLT keyword)

The following table shows the effect of each of the Access Path Maintenance options depending on whether this option is implemented in DDS, CL, SQL or DDL.

Edit Access Path Details Panel Maintenance Option	**Method Used to Implement the Access Path Maintenance Option**			
	DDS (Non-QRY Access Paths)	(1) OPNQRYF command (QRY Access Paths)	SQL	DDL
I (Immediate)	*IMMED	*FIRSTIO	Create Index	Create Index
D (Delay)	*DLY	*MINWAIT	No Index	No Index
R (Rebuild)	*REBLD	*ALLIO	No Index	No Index

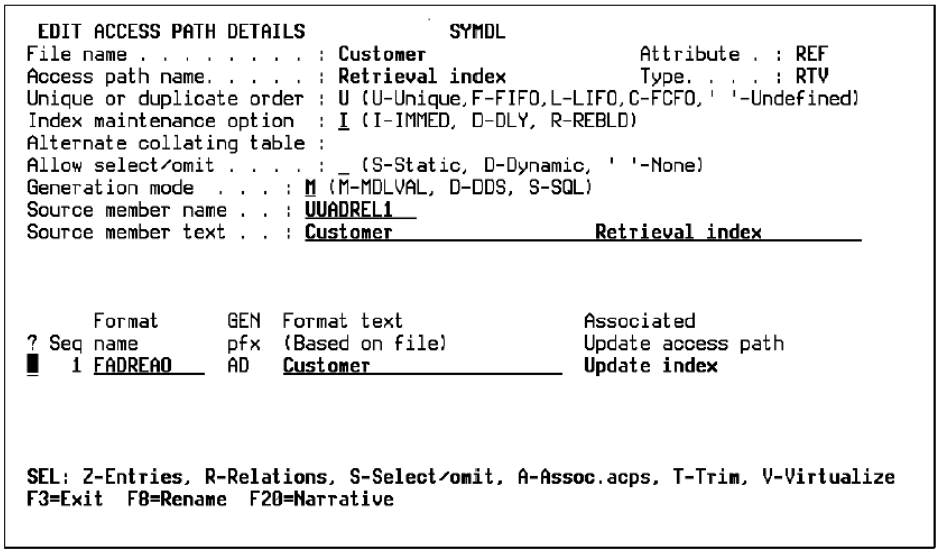
(1) For QRY access paths the access path maintenance options are implemented using OPTIMIZE parameter values on the i OS

OPNQRYF command.

For more information about parameters for the OPNQRYF command, see *IBM i Control Language Reference.*

Editing Access Path Details

You can display and change the details for a CA 2E access path using the Edit Access Path Details panel.



You can display and edit the following details for an access path using the Edit Access Path Details panel.

- Unique/Duplicate Key sequence
- Access Path Maintenance
- Alternate Collating Sequence
- Select/Omit Criteria
- Generation Mode
- Source Member Name and Text

Specifying Unique/Duplicate Key Retrieval Sequence

CA 2E lets you determine if the key values of the access path are unique or duplicate and, if they are duplicate, what the sequence would be.

1. Zoom into the file.

   From the Edit Database Relations panel, type **Z** next to the selected file and press Enter. The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** next to the selected access path and press Enter. The Edit Access Path Details panel displays.

3. Specify the key sequence.

   Change the key sequence by entering the selected option in the Duplicate Sequence field and press Enter.

   The possible options are:

   - U for unique (this requires Immediate maintenance).
   - F for first in, first out (FIFO).
   - L for last in, first out (LIFO).
   - C for first changed, first out (FCFO).
   - Blank means an unspecified sequence is used.

For more information about the unique/duplicate key sequence, see *IBM i DDS Reference Manual* and *IBM i Database Guide*.

Specifying Access Path Maintenance

CA 2E lets you select the type of maintenance for your i OS access paths. i OS maintains all access paths immediately, while they are open, regardless of the maintenance option. However, when the file is closed, the access path maintenance option specifies to i OS how the access path should be maintained.

The type of access path maintenance you specify depends on the number of records, the frequency of additions, deletions, and updates to a file, and the frequency of opens. If you do not specify the type of maintenance, the default is immediate maintenance.

Specify access path maintenance. At the Edit Access Path Details panel, enter or change the maintenance selection for updating the records in the Maintenance field and press Enter.

The access path maintenance options are:

Option	Description
I=*IMMED	Immediate Maintenance. The i OS access path is maintained as changes are made to its associated data, regardless of whether the i OS file is open. For QRY access paths, the OPNQRYF command's OPTIMIZE parameter is set to *FIRSTIO, which minimizes time required to open the file and to retrieve the first buffer of records from the file.
D=*DLY	Delay maintenance. Any maintenance for the i OS access path is done the next time the associated file is opened. For QRY access paths, the OPNQRYF command's OPTIMIZE parameter is set to *MINWAIT, which minimizes delays while reading the file.
R=*REBLD	Rebuild maintenance. The i OS access path is completely rebuilt each time the file is opened. For QRY access paths, the OPNQRYF command's OPTIMIZE parameter is set to *ALLIO, which attempts to minimize total processing time.

Note: Specify I (immediate maintenance) for all files that require unique keys in order to ensure uniqueness for inserts and updates.

For more information:

- About access path maintenance, see the "Tailoring for Performance" chapter and the *IBM i DDS Reference Manual*.
- On the i OS OPNQRYF command, see *IBM i Control Language Reference.*

Specifying Alternate Collating Sequence

CA 2E lets you specify a keyword to direct the i OS program to use an alternative collating sequence table when sequencing the records.

A typical example is to use the i OS-supplied translate table, QCASE256, to make the collating sequence for both upper and lower case the same. This creates an access path that suppresses unwanted upper/lower case discrepancies in the collating sequence while preserving the upper/lower case differences in the data. Any field used as a key for this access path should be similarly translated (uppercase only).

Specify Alternate Collating Sequence. At the Edit Access Path Details panel, enter or change the keyword name of the alternative collating sequence table in the Alternating Collating Table field and press Enter.

Use the i OS Create Table (CRTTBL) command to create the table or use an existing i OS table, such as QSYSTRNTBL or QCASE256.

Specifying Select/Omit Criteria

CA 2E allows you to specify select/omit criteria that filter your view of the records for the RTV, RSQ, SPN, and QRY type access paths.

For more information about select/omit criteria, see the section Choosing Select/Omit Criteria in this chapter.

Specify select/omit criteria. At the Edit Access Path Details panel, enter or change the select/omit criteria at the Allow Select/Omit field and press Enter.

Options are:

- **S**—*Static* applies the selection and omission criteria as the records are added (stored)
- **D**—*Dynamic* specifies the selection and omission of logical file records performed during processing, instead of when the access path (if any) is maintained
- **Blank**—No selection/omission criteria

Note: Dynamic must be specified if there are any virtual fields on the access paths. For QRY access paths, dynamic will be defaulted if required.

Specifying Generation Mode

CA 2E lets you specify the mode in which you generate the source (Data Definition Language).

Specify Generation Mode. At the Edit Access Path Details panel enter or change the generation mode at the Generation Mode field and press Enter.

Options are:

- **D**—for DDS
- **S**—for SQL
- **M**—for Model value
- **L**—for DDL

In CA 2E, some combinations of files and access paths that use different Data Definition Languages are permitted, while some are not. For example, you cannot have a CA 2E SQL logical file over a CA 2E DDS physical file. The following table outlines these rules:

	CA 2E Logical Access Path			
CA 2E Physical Access Path		DDS	SQL	DDL
	DDS	Yes	No	No
	SQL	Yes	Yes	Yes

SQL and DDS Joins

To join information from tables/files, SQL uses inner joins and DDS uses outer joins. Outer joins are not part of the American National Standards Institute (ANSI) standard for SQL. If you switch an access path from DDS to SQL or vice versa, be aware that the same records might not be included.

The following examples illustrate an SQL inner join and a DDS outer join, where the joins resolve to a different set of records:

Order File
(Primary File)

Order #	Cust. #	Order Date
001 002 003	1 2 3	4/1/92 4/2/92 4/3/92

Customer File
(Secondary File)

Cust. #	Cust. Name
1 3 4	JONES SMITH BROWN

Resulting Joined file–SQL

Inner Join

Order #	Cust. #	Cust. Name	Order Date
001 003	1 3	JONES SMITH	4/1/92 4/2/92

Resulting Joined file–DDS

Outer Join

Order #	Cust. #	Cust. Name	Order Date
001 002 003	1 2 3	JONES * SMITH	4/1/92 4/2/92 4/2/92

* = spaces

In the preceding example, the customer record for Order # 002, Cust. # 2, does not exist in the Customer file. In the SQL join file, the record for Order # 002 is dropped from the file. In the DDS join file, the record for Order # 002 is included in the file. Note that the virtual field Cust. Name is filled with blanks in the DDS join file.

Copying an Access Path Generated with SQL or DDL

If you use the Copy Model Objects (YCPYMDLOBJ) command to copy an SQL or DDL generated access path or function to a model that does not have an SQL environment, YCPYMDLOBJ runs successfully, but you need to create an SQL collection for the receiving access path before you can generate source.

Changing Source Member Text and Names

CA 2E lets you change the source member names and text created.

Change the text or name. At the Edit Access Path Details panel, change the name in the Source Member Name field or the text in the Source Member Text field and press Enter.

Modifying Access Path Format Entries

This topic provides information on identifying access path format text and keys and instructions on changing the key sequence and editing access path format entries.

An access path format shows which fields are present in the access path. It also indicates which of those fields are key fields for the access path, and the order in which the key fields appear.

SPN type access paths can have more than one format. Other types of access paths have only one format.

Identifying Access Path Format Text

An access path format can have up to fifty characters of descriptive text. The default text is a concatenation of the file name and the access path name. The text appears in the Format Text field on the Edit Access Path Format Entries panel.

Identifying Access Path Format Keys

The keys of UPD and RTV access paths come from the key relations for the based-on file and cannot be changed.

The keys of the RSQ, SPN, and QRY type access paths are initially defaulted to the entries defined by the key relations but can be changed. Keys can be sequenced in ascending or descending order.

For more information about the i OS limit on the number of keys that can be specified, see *IBM i Database Guide*.

If an alternative collating table is specified for the access path, you can specify whether to use it to collate particular key fields. You can use this panel to flag those keys that are to be alternately collated.

Changing the Key Sequence

Do the following to change the key sequence

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** next to the selected access path and press Enter.

   The Edit Access Path Details panel displays.

3. Zoom into the format.

   Type **Z** next to the selected format and press Enter.

   The Edit Access Path Format Entries panel displays:

```
 EDIT ACCESS PATH FORMAT ENTRIES      SYMDL
File name . . . . . : Customer                Attribute . : REF
Access path name. . . . : RSQ by customer name  Type. . . . : RSQ
Format text . . . . . : Customer
Based on. . . . . : Customer                  Format No . :   1

                                  GEN                  Key    Altcol Ref
? Field                           Name       Type      no. Dsc seq   ont
_ Customer code             CDE   AECD         A       ___ _          1
_ Customer name             TXT   AFTX         K       __1 _          1
_ Customer address          TXT   AGTX         A       ___ _          1
_ Customer city             TXT   AHTX         A       ___ _          1
_ Customer country          TXT   AITX         A       ___ _          1
_ Customer postal code      CDE   AFCD         A       ___ _          1
_ Customer phone number     NBR   ACNB         A       ___ _          1
_ Customer status           STS   ACST         A       ___ _          1
_ Customer credit limit     NBR   ADNB         A       ___ _          1
_ Customer state            TXT   AOTX         A       ___ _          1
_ Customer Allow Credit     STS   AGST         A       ___ _          1

SEL: Z-Field details, L-Locks.
F3=Exit  F7=Relations
```

4. Change the key sequence.

   Change the key order as appropriate by changing the numbers in the Key no. column.

   Numbers represent the order of the fields that make up a composite key. Ensure that the key sequence numbers are unique. Low key order indicates the sequence of the major keys.

5. Generate the access path.

   For the new specification to take effect, you must regenerate the access path.

For more information about how to generate an access path see the "Generating and Compiling" chapter.

Editing Access Path Format Entries

The presence of a field on an access path format (an access path entry) is controlled by the relations specified for the access path. By default, all the relations specified for the based-on file are declared to be present on an access path so that all the fields from the file are initially present. By dropping particular relations from an access path, you can omit fields from the access path's format.

To display and change the relations for an access path, use the Edit Access Path Relations panel. To display the fields that are present on the access path format, use the Edit Access Path Format Entries panel.

You can use the Edit Access Path Format Entries panel to specify an alternative key order for RSQ, QRY, and SPN type access paths. CA 2E lets you edit access path format entries using the Edit Access Path Details panel. Access path formats are created automatically for all access path types except SPN. For SPN access paths you need to add formats explicitly.

For more information on adding SPN access path format entries, see this guide's "Adding Access Paths" chapter.

Note: Relations are present on an access path if they were added to the based-on file after a hold was specified on the access path.

Editing Physical File Format Entries

The physical file format entries for an assimilated CA 2E physical access path can be edited using the Edit Physical File Format Entries panel. You use the panel to specify override values to be used when generating source.

This panel allows you to specify:

- That the fields in a given database file have different implementation names from those used in the logical files based over them
- That the fields occur in a different order from that shown on the Edit Access Path Entries panel

Altering Field Sequence or Implementation Name

Do the following to alter the field sequence or implementation name:

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can select option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. Zoom into the physical access path of the assimilated file. Type **Z** next to the PHY access path and press Enter.

   The Edit Access Path Details panel displays.

3. Zoom into the format.

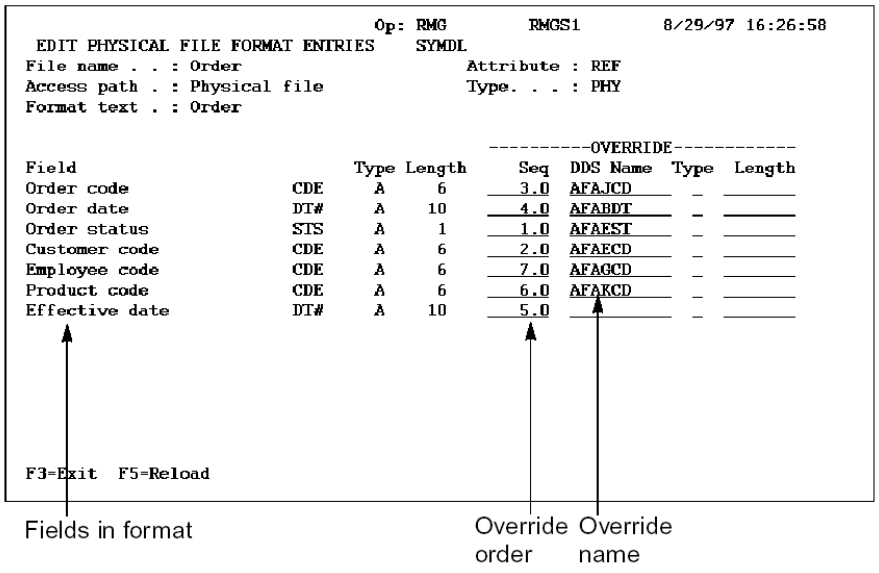
   Type **Z** next to the format and press Enter.

   The Edit Access Path Format Entries panel displays.

4. Access the Edit Physical File Format Entries panel.

   Press F8.

   The Edit Physical File Format Entries panel displays:



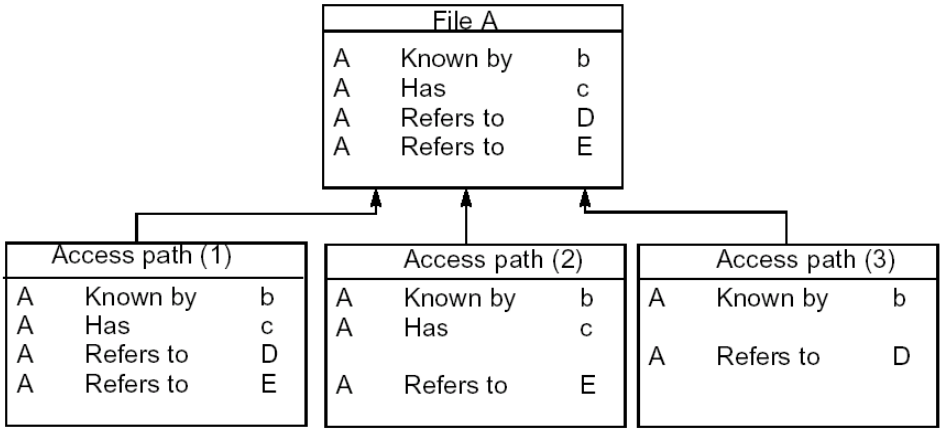
5. Change the DDS name or the file sequence as appropriate and press Enter.

Modifying Access Path Relations

This topic discusses required relations and provides instructions on editing access path relations.

The relations for an access path are composed of the set or subset of a file's relations that apply to a particular access path. For a specific access path belonging to a given file, only some of the relations specified for the file need to apply.

In other words, a particular access path may not require all of the fields and relations from the file to be present on the access path. Each access path may have its own subset of the file's attributes and relations. When you drop file-to-file relations, such as Refers to, those key fields associated with the relation are dropped as are all associated virtual fields. The following is an illustration of access path and file relations.



For more information about relations, see the Using Relations section of the "Understanding Your Data Model" chapter in *Defining a Data Model*.

Understanding Required Relations

The key level relations for a file (Known by**,** Owned by**,** and Qualified by) must be present on all access paths for the file.

A PHY type access path always contains all of the relations for the file on which it is based. On UPD and RTV type access paths, define the entries resulting from the resolution of the relations as the keys of the access path. On other access path types, the relations merely cause the fields to be present on the file, not necessarily as key fields.

For more information about how to define specific required relations, see the next section Adding Relations to a File.

Adding Relations to a File

Each access path initially contains all the relations for the file on which it is based. If a new relation is added to a file, it will be added to the list of access path relations for each of the file's access paths provided that the access paths are not held. Held access paths remain unchanged, as do any functions attached to them.

If a relation is added to an access path and functions already exist that use that access path, the entries that result from resolving the new relations will be added to the function's device designs as follows:

- If they are not key fields on the access path, they will be added as hidden fields to the device design.
- If they are key fields, they will be added as input fields to the device designs.

Editing Access Path Relations

CA 2E lets you display and edit access path relations with the Edit Access Path Relations panel. On this panel, you can reinstate or drop relations from an access path.

1. Zoom into the file.

   From the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. Zoom into the access path.

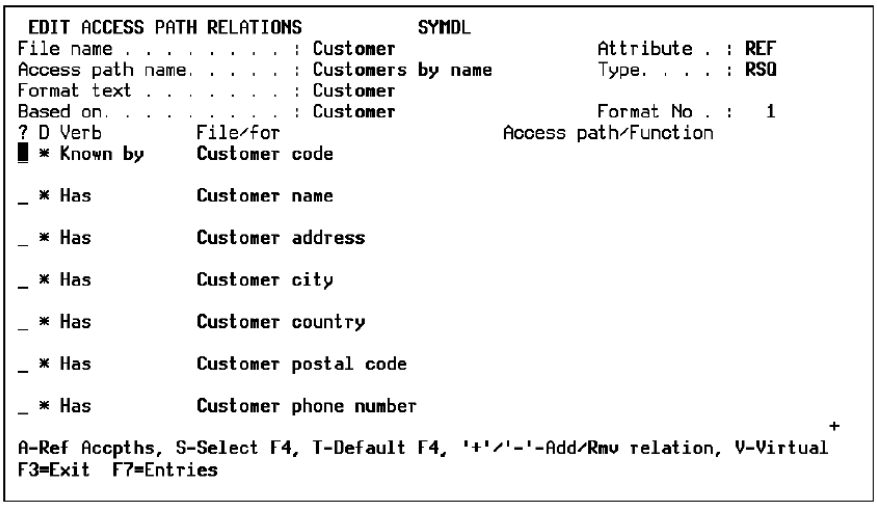
   Type **Z** next to the selected access path and press Enter.

   The Edit Access Paths Details panel displays.

3. Select the access path relation.

   Type **R** next to the selected access path and press Enter.

   The Edit Access Path Relations panel displays:



4. Specify the access path relations.

   Type **+** (plus) or **-** (minus) next to the relations you want to reinstate or remove from the access path and press Enter.

Each access path initially contains all of the relations for the file on which it is based. If a new relation is added to a file, it is added to the list of access path relations for each of the file's access paths except those access paths specified to be held. Held access paths remain unchanged as do any functions attached to them.

Modifying Virtual Field Entries

This topic explains how to identify relations with virtual fields, specify file and access path relations, and tailor virtual fields for access paths. This topic also contains instructions on editing virtual field entries.

A virtual field specified on a relation of a file is added, by default, to each instance of that relation on a particular access path of the file.

Understanding Access Path Virtual Field Entries

An access path virtual field is a field that is logically, rather than physically, present on an access path. Although the field does not reside on the based-on physical file, a view of it is available through the relations that exist for the access path.

It is possible to omit particular virtual fields from a particular access path. The virtual fields for an access path can be selected only from among fields that have been specified as virtual fields for the file.

For example, if Customer file has five fields (Customer no., Branch name, Branch no., Customer name, and Customer address), and if Customer file Refers To Branch file with only Branch name field as a virtual field, only the Branch name field is available as a virtual field on all access paths of Customer file.

Branch	Known by	Branch no.
Branch	Has	Branch name
Branch	Has	Branch address
Customer	Known by	Customer no.
Customer	Has	Customer name
Customer	Has	Customer address
Customer	Refers to	Branch

The Customer Refers to relation adds the field Branch No. to the Customer file. Virtualization allows you to add the field Branch name as a virtual field to the Customer file. You can include or drop this virtual field from any access paths over the Customer file.

Identifying Relations with Virtual Fields

CA 2E lets you specify only virtual fields on the Owned by, Refers to, and Extended by relation types. Virtual fields can be used only as read-only fields in standard functions. This read-only field is implemented as a join logical file or SQL view. It checks on the generation mode and access path type.

Different access paths for a file can contain different combinations of relations, and each file-to-file relation on the access path can have a different set of virtual fields associated with it.

For more information about files and relations, see the chapters "Understanding Your Data Model" and "Creating/Defining Your Data Model" in *Defining a Data Model*.

Specifying File and Access Path Relations

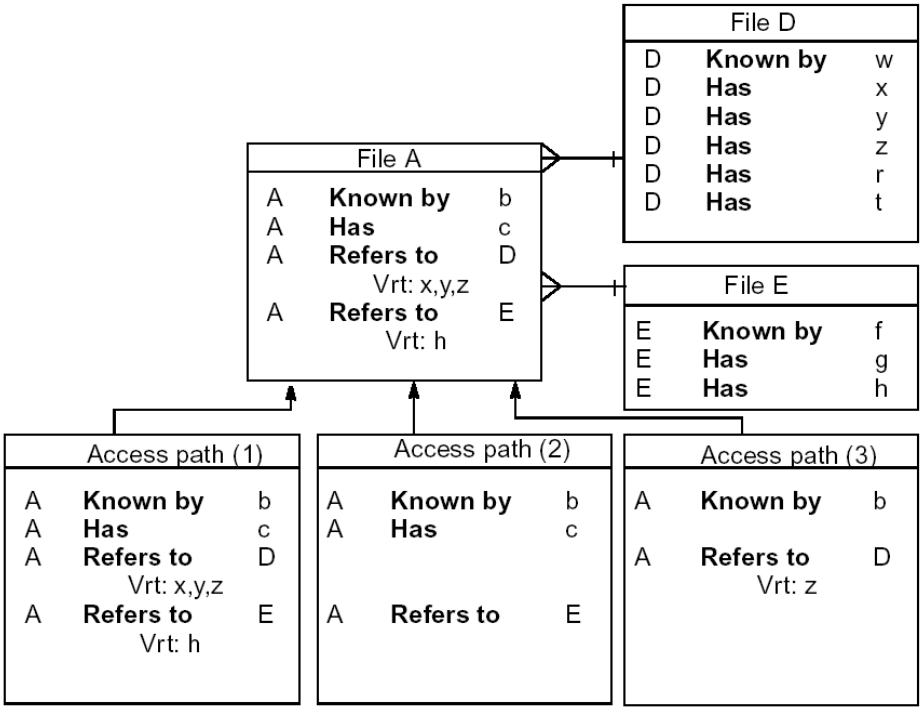
Specifying virtual fields is a two level process: For a field to be used as a virtual field, you must specify it as a virtual field on both the file relations and the access path relations.

In the previous example, the field Branch address is not available as a virtual field on any access path of the Customer file, as it is not specified as a field on the relations for the file.

For more information about:

- Specifying virtual fields on a file, see the Adding/Modifying Virtual Fields section in the "Maintaining Your Data Model" chapter of *Defining a Data Model*.
- Specifying virtual fields on an access path, see the Tailoring Virtual Fields for Access Paths section later in this chapter.

The following example shows how access paths can have subsets of the relations on the file:



Editing Virtual Field Entries

CA 2E lets you edit the virtual field entries on both the file and the access path. Field level virtual fields are specified using the Edit Virtual Field Entries panel. Access path virtual fields are specified using the Edit Access Path Relation Virtual Fields panel.

The file level panel declares that a field is available for use as a virtual field on any of the access paths for the file. The access path level panel specifies that a virtual field is present on a particular access path.

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Detail panel displays.

2. Zoom into the access path.

   Type **Z** next to the selected access path and press Enter.

   The Edit Access Path Details panel displays.

3. Zoom into the format.

   Type **Z** next to the selected format and press Enter.

   The Edit Access Path Format Entry panel displays.

4. Specify virtual fields.

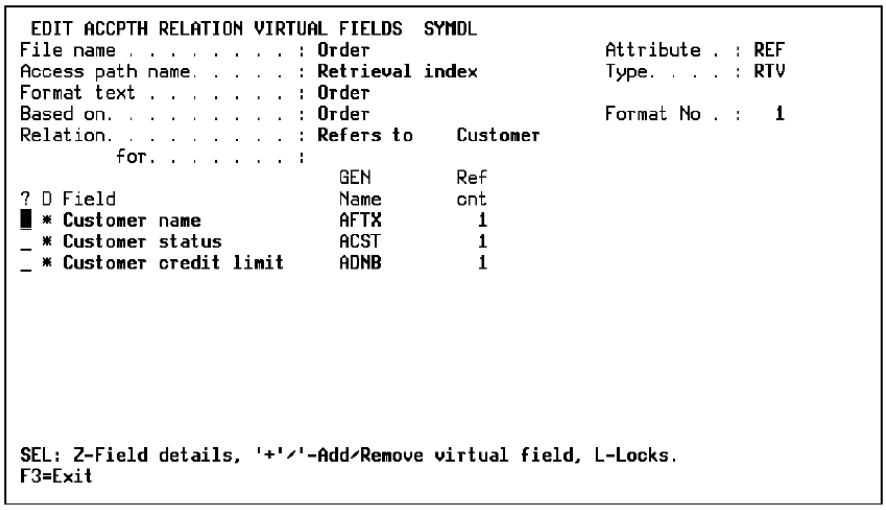
   Press F7 and press Enter.

   The Edit Access Path Relations panel displays.

5. Select the virtual field entry.

   Type **V** next to the selected relation and press Enter.

   The Edit Access Path Relation Virtual Field panel displays:



Note: The selected relation must be an Owned by, Refers to, or Extended by relation or you will not be able to virtualize the field.

6. Edit the virtual field entry.

   Type **+** (plus) or **–** (minus) next to the field(s) you want to virtualize and press Enter.

   A refreshed panel displays with your choices highlighted.

Tailoring Virtual Fields for Access Paths

It is important to consider the design of your application when specifying virtual fields. Ensure that the join logicals you set up when you virtualize provide you with the data you need. This saves the operating system from having to do unnecessary Input/Output (I/O) to other files.

For more information about tailoring virtuals, see the Using Join Logicals section in the "Tailoring for Performance" chapter.

Choosing Select/Omit Criteria

This topic provides instructions about how to specify selection and conditions.

Access path selection lets you specify that a particular access path retrieves only the records from the file that meet specified select/omit criteria, such as those that contain specified values for particular fields.

For example, you have a personnel file containing records for both full-time and part-time employees. If you want to obtain a view of only the part-time employees, you can define an access path with selection on an employee type field that identifies part-time employees.

Understanding Select/Omit

CA 2E lets you specify selection using a select/omit set. A given access path may have none, one, or many select/omit sets specified. If there is more than one set, the sets are ORed together.

Once a record satisfies a select or omit set, it is either selected or omitted and further sets are not relevant. If a record does not satisfy a select or omit set, it will be tested against subsequent sets. If a record does not satisfy any select or omit set, it is omitted if the last set was a select set and selected if the last set was an omit set.

Each set is made up of one or more conditions that specify the actual values that a field may take. If there is more than one condition, the conditions within a set are ANDed together.

For example, if there are two conditions, both must be valid for the entire set to be valid.

In the following example, the access path suspended order is made of two select/omit sets: Awaiting Confirmation and Passed Credit Check. Each of the select/omit sets are defined by one or more conditions.

Accpth		Select/Omit set		Field	Condition	Op	Value
Suspended order							
Select	001	Passed credit **check**	001	Order status	Open	*EQ	H
		AND	002	Credit limit	Not exceeded	*GT	5000
Select	002	Awaiting confirmation					
			001	Invoice status	Unconfirmed	*EQ	U
		AND	002	Invoice value	not zero	*GT	0

Specifying Selection

Use the Edit Access Path Select/Omit panel to specify selection and the names of the select/omit sets that make up the selection criteria. The following steps tell you how to specify selection.

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** next to the selected access path and press Enter.

   The Edit Access Path Details panel displays.

3. Select the format.

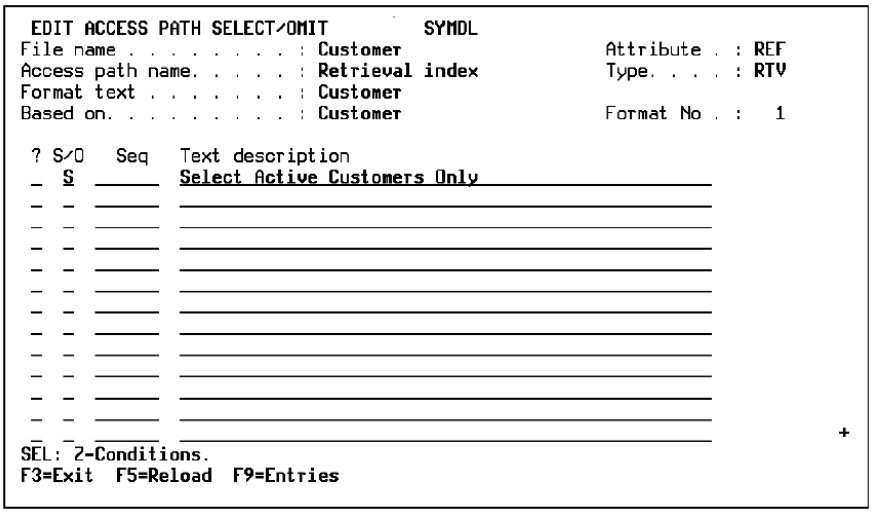
   Enter S next to the selected format and press Enter.

   The Edit Access Path Select/Omit panel displays.

   Note: The Allow Select/Omit access path option must be set to either S or D.

4. Specify selection.

   Type **S** for select or **O** for omit in the S/O column and the text description to indicate the name for the selection criteria set.



5. Zoom into the selection.

   Type **Z** next to the selection to specify the conditions and press Enter.

   The Edit Access Path Conditions panel displays.

Specifying Conditions

Use the Edit Access Path Conditions panel to specify conditions (which are the actual field conditions that make up a given select/omit set).

1. Specify the condition.

   Enter the field and condition name for the selection in the appropriate column. If you do not know the field or condition name, place a question mark in the appropriate column and press Enter.

   The Display Access Path Format Entries panel or the Edit Field Conditions panel displays.

2. View the fields or conditions.

   Type **X** next to the selected field or condition and press Enter.

   The Edit Access Path Conditions panel displays with the selections.

For more information about how to add conditions, see the Adding/Modifying Conditions section in the "Maintaining Your Data Model" chapter of *Defining a Data Model*.

Select/Omit in DDL Index

In CA 2E r8.7, a new generation mode *DDL is introduced. Using this generation mode, the existing DDS database can be regenerated to have an SQL type database. You can still access the existing RLA functions that were built using the DDS database without regenerating and recompiling them. Prior to CA 2E r8.7, the select/omit criteria was applicable to *DDS and *SQL generation modes only. Henceforth, the select/omit criteria is applicable to the *DDL generation mode too.

Considering all the three generation modes, *DDS, *SQL, and *DDL, the following table displays the location where the Select/Omit criteria get generated.

Generation Mode	Allow Select/Omit criteria Settings		YDBFACC=T(TABLE)	YDBFACC=G(DBFGEN)
DDS	Static	Select/Omit criteria in the logical file		
	Dynamic	Select/Omit criteria in the logical file		
SQL	Static		Select/Omit criteria in the Program code	Select/Omit criteria in the SQL View
	Dynamic		Select/Omit criteria in the Program code	Select/Omit criteria in the Program code

Generation Mode	Allow Select/Omit criteria Settings	YDBFACC=T(TABLE)	YDBFACC=G(DBFGEN)
DDL	Static	Select/Omit criteria in the DDL index	
	Dynamic	Select/Omit criteria in the DDL index	

As shown, when you regenerate the database as a DDL database, the select/omit criteria gets generated into the DDL index as a WHERE clause. Any RLA function that uses the DDL database, can now access the same records that the RLA function has accessed over the DDS database.

In DDL generation mode, irrespective of the "Allow select/omit criteria" settings being static or dynamic, the select/omit criteria gets embedded in the DDL index. You do not have to regenerate and recompile the function. The select/omit criteria are generated for all the access path types that DDL supports.

Note: The select/omit criteria, over DDL implementation supports RTV and RSQ access paths.

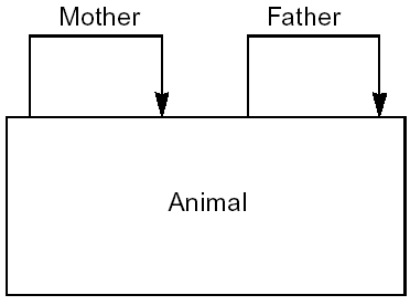
Changing a Referenced Access Path

When an access path includes a relation that refers to another file, the relation always references an access path. This access path is used by functions for validation and, by default, this is the access path RTV automatically created by CA 2E. You can, however, alter the relation so that a different access path is used. This enables you to specify selection criteria for the relationship.

Using the F4 prompt function assignment, you can change the prompt function assigned to a file-to-file relation. If you change the access path, then the prompt function will default to the SELRCD function for that access path.

For more information about function assignment, see the information at the end of this section.

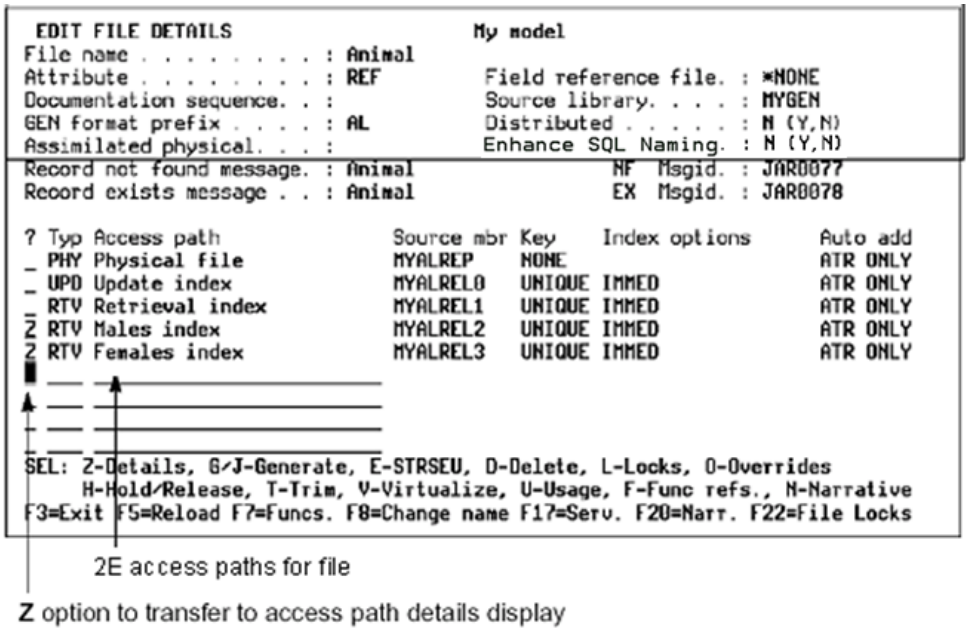
In a database recording pedigrees, you could specify that Mothers be only female, and Fathers be only male by specifying the access path selection on the relation.



The relations you would need to specify the pedigree are as follows:

FIL	Animal	REF	Known by	FLD	Animal code	CDE
FIL	Animal	REF	Has	FLD	Animal name	TXT
FIL	Animal	REF	Has	FLD	Gender	STS
FIL	Animal	REF	Refers to	FIL	Animal	REF
FIL	For	Mother		Sharing:		
FIL	Animal	REF	Refers to	FIL	Animal	REF
FIL	For	Father		Sharing:		

Then, having attached two conditions to gender, Male and Female, you can define two additional retrieval access paths on the Animal file that select on each gender respectively.



Having added the new access paths, you can return to the relations for the original retrieval CA 2E access path and specify that the Refers to relations are to use the additional CA 2E access paths with gender-specific selection. Thus, fathers must be male, and mothers must be female.

Follow these steps:

1. Zoom into the access path details.

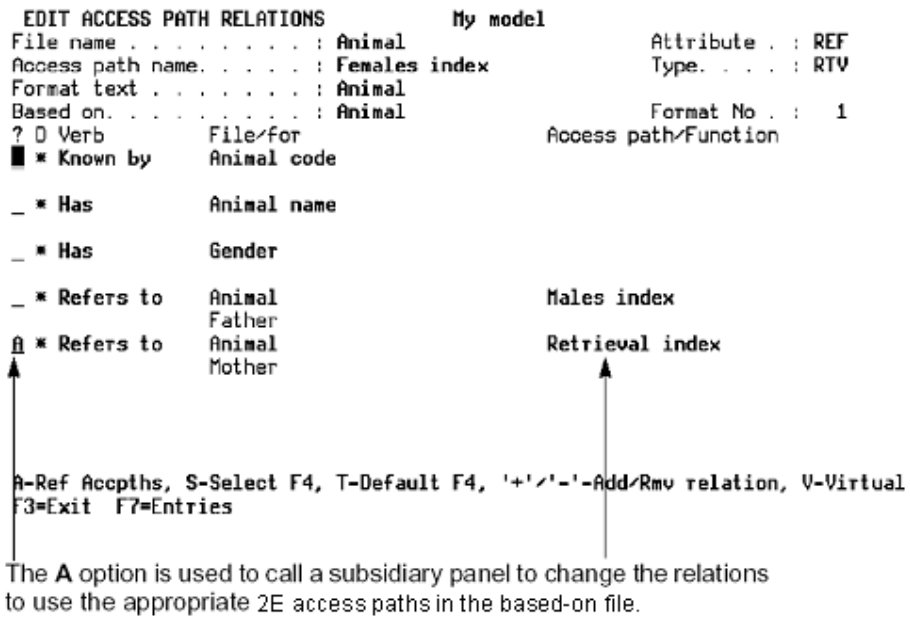
   Type **Z** next to the selected access path on the Edit File Details panel.

   The Edit Access Path Details panel displays.

2. Go to the access path relations panel.

   Type **R** next to the formats and press Enter.

   The Edit Access Path Relations panel displays:



The **A** option is used to call a subsidiary panel to change the relations to use the appropriate 2E access paths in the based-on file.

3. Select the Referenced Access Paths option.

   Type **A** next to the relation and press Enter.

   The Display File Access Paths panel displays.

4. Select the access path.

   Type **X** next to the appropriate access path and press Enter.

F4 Prompt Function Assignment

CA 2E allows you to change the access path assigned to the file-to-file relation and to assign a new prompt function over the access path. This can be done at the access path or function level. Function level overrides take precedence over access path level overrides.

For more information about F4 prompt function assignment at the function level, see the SELRCD section of the "Defining Functions" chapter in *Building Applications*.

To prompt for a new function assignment:

1. Use the instructions on the previous page to get to the Edit Access Path Relations panel.
2. Type **S** next to the selected file-to-file relation you want to assign to the access path and press Enter.

   The Edit Function panel displays.
3. Type **X** next to the selected function and press Enter.

   You can select any external function other than Print File (PRTFIL) and the function can be based over any access path that is valid for the function type you select.

To cancel the function assignment and return to the default function:

1. Type **T** next to the selected relation.
2. Press Enter**.**

Modifying Access Path Auxiliaries

This topic provides instructions on editing access path auxiliaries.

To implement QRY type access paths for DDS, use the i OS Open Query File (OPNQRYF) command. For SQL, use dynamic SQL. In order to do this, CA 2E holds some additional information for QRY access paths and SQL tables and views with *IMMED maintenance capability, which is shown on the Edit Access Path Auxiliaries panel. CA 2E generates default values for the access path auxiliary display when the access path is created.

Understanding Access Path Auxiliaries

For DDS Query (QRY) Access Paths

Use the following three types of i OS objects to implement a QRY access path in DDS:

- An i OS logical file based on the real physical file whose data is being referenced.
- An i OS physical file, which does not contain any data, but is used to define a record format and keys to any HLL program generated for a function based on the QRY access path.
- A CL program that executes the OPNQRYF command. It is called at execution by any program generated for a function based on the QRY access path.

Each object type has its own source, either DDS or CL, which is held in the appropriate source file in the generation library.

All three of the auxiliary objects must be generated and compiled.

You can control the implementation names given to the auxiliary objects by controlling the names given to the source members generated for them. If the YALCVNM model value is set to YES, CA 2E will automatically supply source member names.

Note: The current implementation of DDL generation mode uses only *TABLE as data access method. Therefore, QRY access path is not generated in DDL mode as it uses only SQL views.

For more information about access path auxiliaries and QRY access paths, see the "Access Paths: An Introduction" chapter.

For SQL Access Paths with *IMMED Maintenance

When an access path that is implemented in SQL is created with *IMMED index maintenance, CA 2E also creates an SQL index as an access path auxiliary. You can suppress generation of the SQL index and also retain *IMMED maintenance capability.

For more information on SQL, see the "SQL Implementation" appendix in the *Administration Guide*.

For DDL Access Paths with *IMMED Maintenance

When an access path that is implemented in DDL is created with *IMMED index maintenance, CA 2E creates only an SQL index as an access path auxiliary.

For more information on SQL, see the "SQL/DDL Implementation" appendix in the *Administration Guide*.

Editing Access Path Auxiliaries

Note: Auxiliaries are not applicable for DDL type file as Views are not created for DDL type file and *only* Index with the same name as the source member name is created. F7=Auxiliaries is not applicable for the DDL implementation.

Use the Edit Access Path Auxiliaries panel to display and change the auxiliary details for a QRY (or SQL) access path including changing source member names.

1. Zoom into the file.

   At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can, select option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. Zoom into the access path.

   Type **Z** on the selected QRY (or SQL) access path and press Enter.

   The Edit Access Path Details panel displays.

3. View the auxiliaries.

   Press F7.

   The Edit Access Path Auxiliaries panel displays.

Note: In order to get the F7 option for auxiliaries, you must zoom into the access path by typing Z next to the access path.

1. Edit the auxiliaries.

   Change any of the following details as appropriate:

   - Source Member Name
   - Source Member Text
   - Index Name for SQL only. Enter *NONE to suppress generation of the SQL index.

2. Generate the access path.

   For the new specifications to take effect, you need to regenerate the access path.

For more information about how to generate the access path, see the instructions in the chapter "Generating and Compiling."

Chapter 5: Deleting Access Paths

This chapter contains instructions on how to delete access paths from your application. You may want to delete access paths created in error or those no longer needed.

This section contains the following topics:

Deleting an Access Path (see page 99)
Determining the Usage of an Access Path (see page 100)

Deleting an Access Path

Access paths can be deleted only if they are not referenced by any other function or access path. A cross-reference panel, Display Access Path Reference, is available to show you where a given access path is used.

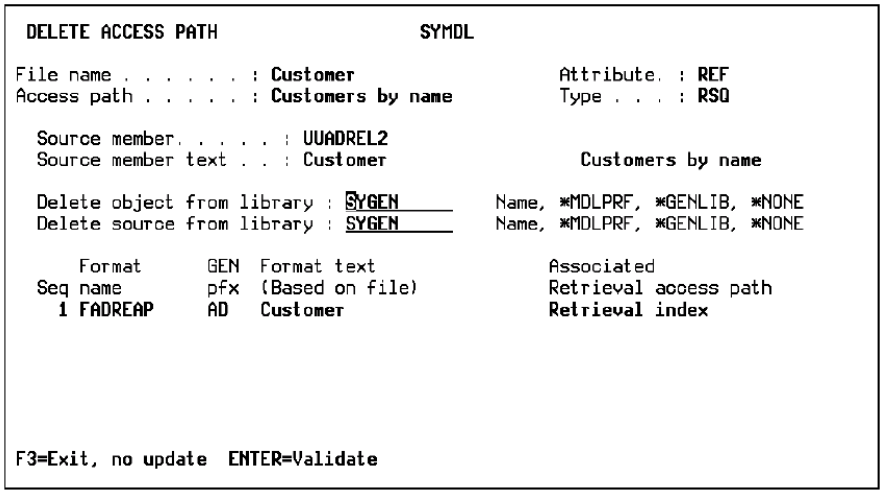
To delete an access path

1. **Zoom into the file.** At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can use selection option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays with a list of the access paths for the file.

2. **Delete the access path.** Type **D** next to the access path you want to delete and press Enter.

   The Delete Access Path panel displays:



3. **Validate your actions**. Press Enter to validate the deletion.

   An additional confirmation prompt displays before the access path is deleted. This prompt allows you to specify that the source and object are to be deleted.

   CA 2E does not allow you to delete the access path if it is referenced by any other function or access path.

   You must eliminate the references before you can delete the access path. You can do so by determining the usage of an access path.

4. **Confirm the deletion.** At prompt, press Enter to confirm the deletion.

Determining the Usage of an Access Path

1. **View the access path usages.** Type **U** next to the selected access path on the Edit File Details panel and press Enter.

   The Display Model Usages panel displays with a list of functions and other access paths that reference the selected access path.

2. **View the functions that use the access path.** Type **F** next to the selected access path on the Edit File Details panel and press Enter.

   The Display Model Usages panel appears with a list of the functions that reference the access path. A function built over an access path is an example of a function using an access path.

For more information on usages, see the Impact Analysis section in the "Managing Model Objects" chapter of Generating and Implementing Applications, and the Advantage 2E Command Reference, YDSPMDLUSG command.

Chapter 6: Defining Arrays

This chapter contains procedures for defining, editing, renaming, and deleting an array. An array is a structure that stores sets of data within a function. The contents of an array are available only as long as the function is active. The structure has a defined layout and each entry (element) of the structure has the same layout. Define this layout with any set of fields from the model, such as attribute, code, and function fields.

Use arrays to:

- Improve performance where a function requires repeated access and use of a finite number of entries, such as table lookups.
- Move between a field and a data structure.
- Sequence a finite set of data by a set of unrelated key fields.

Programmers (*PGMR) and designers (*DSNR) can define arrays.

This section contains the following topics:

Understanding Arrays (see page 102)
Structuring Field Data Using Arrays (see page 102)
Passing Parameters (see page 103)
Defining an Array (see page 104)
Editing an Array (see page 107)
Deleting an Array (see page 108)

Understanding Arrays

The array has a defined layout and each entry (element) of the structure has the same layout. Define this layout with any set of fields from the model, such as attribute, code, and function fields. The array must have a defined key that acts as an index into the array. The key is any subset of the fields in the array structure, up to a maximum composite key length of 990. Each key field in the composite key list acts as a different dimension to the array. The composite key is defined as being either unique or non-unique.

CA 2E does not guarantee the sequence of equally keyed elements in a non-unique array. The key also is defined as ascending or descending. This definition applies to the complete composite key. When determining the key sequence, CA 2E ignores the signs of any numerical fields in the key of the array.

Arrays can be used only by the *CVTVAR built-in function and the following four standard data function types:

- Create Object (CRTOBJ) to add an element
- Change Object (CHGOBJ) to change an element
- Delete Object (DLTOBJ) to delete an element
- Retrieve Object (RTVOBJ) to read an element or set of elements

For more information about how to use or clear the data from an array, refer to the Array Processing section in the "Defining Functions" chapter of the *Building Applications*.

Structuring Field Data Using Arrays

Since a single-element array is equivalent to a data structure, you can use the *CVTVAR built-in function and the ELM context to apply a data structure to a field. This gives you a simple way to decompose a field into a structure and to (re)compose a set of fields into a single field in a single operation.

Structure files (STR) provide a similar capability, but you need to be a designer (*DSNR) to define a structure file. Both *DSNR and *PGMR can define arrays.

For more information on structuring field data and examples, see the sections Understanding Built-In Functions (*CVTVAR), and Understanding Contexts (ELM) in the chapter "Modifying Action Diagrams," in *Building Applications*.

Passing Parameters

You can also use arrays to define a set of fields that are then used as a parameter entry to any function. This process allows you to define a large number of parameters of any field type to a function without creating a structure file (a *DSNR function).

Storing Data Between Calls

For programs that do not close down, arrays are initialized on the first call to the program. Subsequent calls do not clear the array. The first call loads the array and subsequent calls retrieve that data. In addition, you can define an array to store and restore any fields in the program between calls. The PGM context variable Initial Call can be used to distinguish between first and subsequent calls. This variable is always set to *YES if the program closes down.

For more information on defining arrays, see *Building Applications*.

Defining an Array

This topic tells you how to create and fully define an array. This process includes selecting the fields for the array and defining the relative order of the fields by assigning sequential numbers to key fields.

1. **View the list of files.** At the Edit Database Relations panel, type ***a** or ***ARRAYS** in the positioner field at the top of the panel and press Enter.

   The list of the CA 2E reserved files displays.

2. **Zoom into the array file.** Type **Z** next to the *Arrays file and press Enter.

   The Edit Array panel displays.

3. **Define the array.** Type the name of the new array under the Arrays heading in the first blank line and press Enter.

   The name must be a unique array name. This field becomes an output-only field.

   The default line data for the new array displays, including the following information:

   - Sequence—Sequence of the key or composite key of the array in either ascending or descending order. The default for this field is ASCEND.
   - Unique—Defines if the keys in the array must be unique or non-unique. Unique is the default.
   - Elements—Defines the maximum number (9999) of elements that this array can hold. An element is a full array entry and not a field within that entry. The default is 100.

The following array sample could be used to accumulate Order Totals for each State/Branch combination:

	Customer	Customer no. Customer name State code
	Order Header	Order no. Customer no. Branch no. Order value

Order Value by State/Branch

File (1)	Access Path (1)	Fields (2)	Key (3)
Customer	Retrieval Index	State code	1
Order	Retrieval Index	Branch no.	2
		Order value	

(1) This definition comes form the Edit Array Details panel
(2) This definition comes form the Edit Array Entries panel
(3) This definition comes form the Edit Array Key Entries panel

Selecting Field and Key Details for Your Array

1. **Zoom into the array.** Type **Z** next to the selected array and press Enter.

   The Edit Array Details panel displays. This panel is where you specify the files and fields that are used to define the layout of the array entry.

2. **Specify the layout.** Specify the file or field that defines this part of the array entry layout.

3. **Define source.** Define the source of the field definitions. This source may specify file or *field.

4. **Specify the field.** If the source of the field is *FIELD, specify the actual field in the next column (or select by entering ?).

5. **Specify the access path.** If the source of the field is a file, specify the access path of the file (or *NONE for all fields for the file) in the second column.

   a. Type **Z** next to the line and press Enter to select the required fields from the access path.

      The Edit Array Entries panel displays with a list of fields for the selected access path or file.

   b. Type **+** (plus) next to the fields you want to add to the array and press Enter.

   c. Type **–** (minus) next to selected fields you want to drop from the array and press Enter.

6. **Exit.** Press F3 to exit panel.

   The Edit Array Details panel redisplays with a list of selected fields.

7. **Select keys.** Press F7 to select the keys for an array.

   The Edit Array Key Entries panel displays.

   Note that you must define a key for an array even if the array holds a single element.

8. **Select the key order for the fields.** Type a number in the Key no. column that defines the relative order of the field in the composite key list. The lowest value (1) is the major key or first dimension of an array.

You have created and defined your array.

Note: 2E does not support numeric keys that can have negative values.

Editing an Array

To edit an array, use the steps in the preceding Defining An Array topic to determine where to make the following types of changes to your array.

Array Details:

- Sequence status
- Unique key status
- Elements

Fields Definitions:

- Selected files
- Selected access paths
- Selected fields

Key Order:

- Key and composite key

Type **–** (minus) next to any selected + (plus) field you want to remove from the array on the Display All Fields panel.

In addition, use the following instructions to examine array usage and change the array name.

Viewing Function References

At the Edit Array panel, type **F** next to the selected array to examine the usage.

The Display Model Usages panel displays the functions that use the selected array.

Changing the Name of an Array

Follow these steps to change the name of an array.

1. **View the list of files.** At the Edit Database Relations panel, type ***a** or ***ARRAY** in the positioner field on the top line of the panel and press Enter.

   The list of reserved files displays.

2. **Zoom into the array file**. Type **v** next to the Arrays file.

   The Edit Arrays panel displays with a list of the arrays in your model.

3. **Zoom into the array.** Type **Z** next to the array you want to rename.

   The Edit Array Details panel displays with the name of the array and the array details.

4. Change output field to input field. Press F8.

   The output-only field, Array Name, is changed to an input-capable field.

5. **Rename the array.** Type the new name of the array and press Enter to accept the change.

   The panel redisplays with the new array name.

Note: The array name must be unique; CA 2E does not accept duplicate names.

Deleting an Array

The CA 2E Delete Array feature allows you to delete an existing array.

1. **View the list of files.** At the Edit Database Relations panel, type ***a** or ***ARRAY** in the positioner field on the top line of the panel and press Enter.

   The list of the reserved files displays.

2. **Zoom into the array file.** Type **Z** next to the Arrays file.

   The Edit Arrays panel displays with a list of the arrays in your model.

3. **Delete the array.** Type **D** next to the array you want to delete and press Enter.

   A Delete Array window displays at the top of the panel that confirms the delete array request.

Note: You cannot delete an array if it is used by any function.

For more information on how to determine array usage, see the Editing an Array and Viewing Function References sections in this chapter.

Chapter 7: Generating and Compiling

This chapter contains information on the results of generating a specific type of access path, depending on the generation mode you selected, and provides instructions on how to generate and compile an access path.

You must first generate the HLL source code needed to implement access paths you created. Instructions for this are provided earlier in this module. After you generate the access paths, you compile the source to produce executable i OS objects, files, and tables.

This section contains the following topics:

Implementing (see page 109)

Implementing

An access path closely corresponds to the i OS use of the term. You can specify an access path to various generation modes, including values such as DDS, SQL or DDL. DDS generates physical and logical file and, SQL or DDL generate indexes and table.

You must first generate the source members for the database files to implement the access paths. The generated source needs to be compiled. You can generate source either interactively or in batch.

i OS Index Versus CA 2E Index

The type of index generated is determined by the source you select in the model value. CA 2E access paths are generated as i OS objects. i OS access paths are generated as indexes and views.

Setting Your Options

You can specify various implementation options for each access path such as the i OS object name used for the logical file and whether the access path maintenance is Rebuild, Delay or Immediate. CA 2E provides defaults for these options.

Changing Compiler Overrides from DDS to SQL or DDL

If you change your model from DDS to SQL or DDL, verify that the *MESSAGE shipped file contains the CA 2E shipped compiler options. Any DDS overrides in effect during generation for an SQL or DDL implementation, will cause the compile to fail.

Identifying the Implementation Attributes

The following table shows the i OS implementation attributes for each of the access paths:

Access Path Type	(1) (2) Unique or Dup Key Sequence (DDS only)	(3) Access Path Maintenance	(4) Alt Col (DDS only)	(5) Selection	SQL Implementation	DDS Implementation	DDL Implementation
PHY Physical	Generating and Compiling	-	-	-	Table	Physical File	Table
UPD Update (default)		I	-	-	View and Index	Logical File	Index
UPD Update		I/D/R	-	-	View and Index	Logical File	Index
RTV Retrieval (default)		I	-	S,D	View and Index	Logical File	Index
RTV Retrieval		I/D/R	-	S,D	View and Index	Logical File	Index
RSQ Resequence		I,D,R	Yes	S,D	View and Index	Logical File	Index
QRY Query		I/D/R (6)	-	D	View and Index	Physical File	-
SPN Span		I,D,R	Yes	S,D	Two Indexes and two Views	CL Program Logical File	-

The following legend applies for the access path types:

(1) Unique key status (DDS unique keyword or SQL unique keyword with create index statement). U indicates unique; if not unique, see Note (2). The default UPD and RTV access paths must be unique.

(2) Duplicate key sequence for DDS only (L=LIFO, F=FIFO,' '=undefined, C=FCFO).

(3) i OS access path maintenance (I=*IMMED,R=*REBLD, D=*DLY). For QRY access paths, see item (6).

(4) Alternative collating sequence table for DDS only (DDS ALTCOL keyword).

(5) Selection type (S=static, D=dynamic) (DDS DYNSLT keyword).

(6) Causes the i OS OPNQRYF command to be called with the OPTIMIZE option equal to *FIRSTIO, *MINWAIT, or *ALLIO, respectively.

For SQL, static selection is implemented through SQL Data Definition Language (DDL) statements. Dynamic selection is implemented by Data Manipulation Language (DML) statements.

Note: If an SQL-generated RSQ access path has select/omit criteria and is defined as Unique with Static maintenance, the key defined as unique must be unique over the entire file and not just with a subset of that file (as defined by the select/omit criteria).

The current implementation of the DDL generation mode is not valid for SPN access path, QRY access path, access paths that have virtual fields, and multi-member files.

In addition, the following table shows which of the SPN access path / QRY access path / access paths that have virtual fields / multi-member files are valid for SQL implementation.

YDBFGEN / YDBFACC	SPN Access Path	QRY Access Path	Access Path with Virtual Fields	Multi-Members
YDBFGEN = *SQL YDBFACC = *DBFGEN	Y	Y	Y	N
YDBFGEN = *SQL YDBFACC = *TABLE	Y	Y (Source does not contain SQL statements. Function based on access table)	Y	N

Generating an Access Path

You must generate and compile the source members for your access paths before you can run your application. The following steps provide you with instructions to generate your access path.

1. **Go to the Services Menu.** At the Edit Database Relations panel, press F17.

   The Display Services Menu appears.

2. **Select access paths.** Type **8** at the bottom of the screen and press Enter.

   The Display All Access Paths panel appears with a list of all of the access paths in your model.

3. **Generate the access path.** Type **J** next to each of the access paths you want to generate and press Enter.

   The Display All Access Paths panel reappears with messages at the bottom of the panel that say the source generation requests have been accepted.

   Note: Selecting either J for batch generation or G for interactive generation generates your access paths. However, selecting G has an impact on system performance. Generating interactively negatively impacts other interactive users.

4. Press F3 to return to the Services Menu panel.

Note: An alternative to this procedure is to use option 14 (generate in batch) or 15 (generate interactively) from the Edit Model Object List panel.

Limitation:

When you take the G / J option to generate the access paths from the Display All Access Paths panel or take the G / J option to generate the access paths from the Edit File Details panel or take option 14 / 15 from the Edit Model Object List panel against a *DDL-based access path and the access path has either of the four DDL limitations, the generation is prevented. The access path source is not generated and no entry is added to the job list.

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

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

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

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

For more information:

- On the Edit Model Object List panel, see the Editing Model Object Lists section in the "Managing Model Objects" chapter of *Generating and Implementing Applications*.
- On the following topics, see *Generating and Implementing Applications:*
  - Generating Request Panels/Displays
  - Generating Access Paths
  - Changing Generation Mode
  - Verifying Results
  - Checking Code Generation Errors
  - Identifying Common Errors

Chapter 8: Documenting Access Paths

This chapter contains procedures on how to document your access paths. CA 2E includes a number of commands to produce hard copy documentation of your design model. For access paths, this documentation identifies the access paths in your model and provides a complete list of their details. This documentation consists of CA 2E functional text. Functional text is entered by the designer to describe the purpose of the design object and any restrictions and notes on the reason for design decisions.

This section contains the following topics:

Documenting an Access Path (see page 115)

Documenting an Access Path

There are two types of narrative text allowed for each object entry: functional text, used to describe the purpose of the design object; and operational text, used to describe the function of an object to an end user. If no operational text is available, functional text is used in the generated help panels.

All the documentation commands have a PRTTEXT parameter that allows you to specify whether you want text to be included in the listing and, if so, which type of text.

A maximum of ten pages of text of each type can be associated with each CA 2E object to explain the purpose of the object within the design.

Creating the Documentation

Use the following method to document an access path:

1. **Go to the Display Services Menu.** At the Edit Database Relations panel, press F17.

   The Display Services Menu panel displays.

2. **Display Documentation Menu.** From the Model Documentation options on the Display Services Menu panel, select the option and press Enter.

   The Display Documentation Menu panel displays.

3. **Select access paths.** Select the option to document the access paths.

   The Document Model Access Paths (YDOCMDLACP) panel displays.

4. **Select type of documentation.** Choose the documentation option you want from the following list of choices.

   - Model files
   - Application area code
   - Print text
   - Print access path details
   - Access path type
   - Begin new page

CA 2E creates a print file that contains your documentation.

Chapter 9: Tailoring for Performance

When building access paths within your model, you should pay attention to certain aspects of system design that will enable you to obtain the best system performance. Some issues regarding access paths could affect the performance of your application. This chapter explains the issues for i OS logical files and relates them to the default values for access paths. Depending on your design, you might want to consider the following topics.

- Considering the Types of Data in the Physical File
- Minimizing the Number of Active Indexes
- Maintaining Access Paths (Immediate, Delay, Rebuild)
- Using Select/Omit Criteria
- Using Virtual Fields (Join Logical Files)
- Multi-format Access Paths
- Using Open Data Paths
- Creating Default Retrieval Access Paths

For more information about system performance considerations, refer to *CA Xtras Performance Expert User Guide* and *IBM i Database Guide*.

This section contains the following topics:

Considering the Types of Data in the Physical File (see page 118)
Minimizing the Number of Active Indexes (see page 118)
Access Path Maintenance (Immediate, Delay, or Rebuild) (see page 121)
Using Select/Omit Maintenance (see page 124)
Using Join Logicals (see page 125)
Using Multi-Format Access Paths (see page 126)
Using Open Data Paths (see page 126)
Creating Default Retrieval Access Paths (see page 127)

Considering the Types of Data in the Physical File

An access path is a view of the data in a physical file, in a given key sequence. In terms of performance, some of the overhead of access paths has to do with the amount of work that the operating system (i OS) does to maintain those views. Each time a record is added, deleted, or changed in the physical file, or when the data in the key of a record changes, the operating system may have to update each CA 2E access path belonging to the PHY file. Consequently, the more access paths you build, the more work the operating system may have to do for each record change.

For example, with non-volatile data files (master or table files of relatively unchanging data), the number of access paths built over them is not as important as it is for the number of logical files built over volatile files (transaction files of constantly changing data) because their access paths must be constantly updated. This implies that master files should not contain a mix of master and volatile data. For example, balance-related information (volatile) should not be held in an Item Master file (non-volatile).

For more information on the data modeling and normalization process, see the "Developing a Conceptual Model" chapter in *Defining a Data Model.*

You can show the difference between the non-volatile and volatile data files using REF and CPT file types respectively. The only difference between the REF and CPT file types is the automatic creation of an edit file and select record function for REF files.

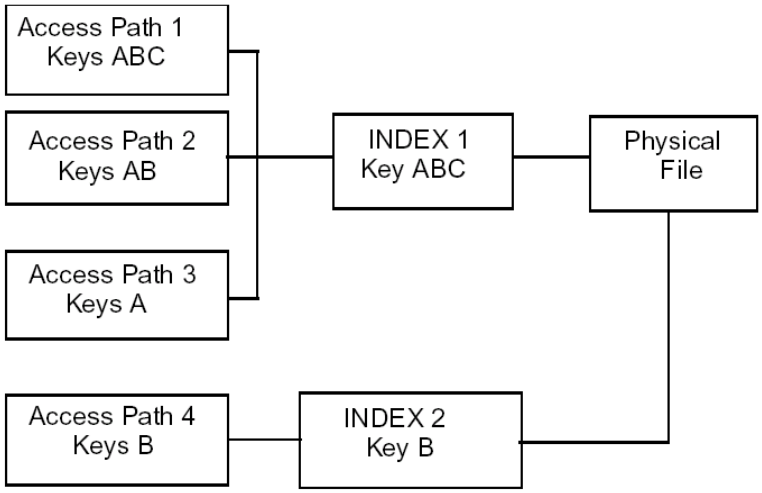
Minimizing the Number of Active Indexes

An access path is defined with key fields specified in a given order and/or with dynamic or static selection maintenance specified. Together the keys and selection maintenance specify which records on a file are implemented as a logical file. However, the operating system implements logical files by automatically creating what is called an active index. The active index is part of the operating system's implementation of logical files. There is a separate active index for each set of key and static selection maintenance.

Where possible, the operating system tries to share an active index between logical files. If logical files have the same key fields and static selection maintenance, they may share an active index. If two similar logical files (same keys) have dynamic select/omit maintenance, they may be able to share the same active index. However, if one (or both) has different static select/omit maintenance, they cannot share an active index, and the operating system always creates separate active indexes.

The Active Index

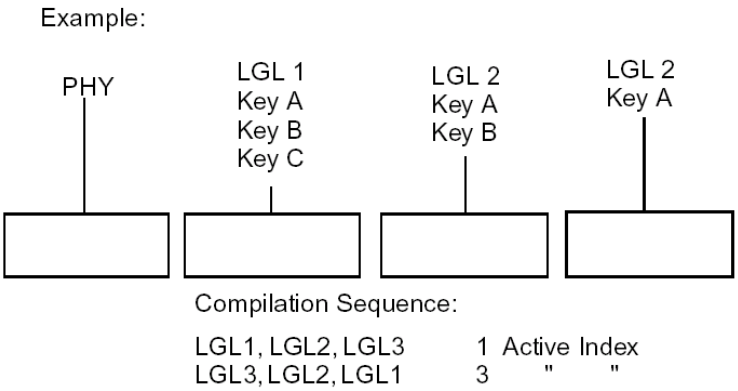
The following example shows separate active indexes:



It is important to minimize the number of active indexes because active indexes cause extra system overhead. Active indexes sharing multiple access paths in the design model do not always equate to multiple indexes. For example, the UPD and RTV access paths have the same active indexes.

Sharing Active Indexes

The following example shows a shared active index.



The compilation sequence can affect the number of active indexes required.

In the previous example, if the compilation sequence is LGL1, LGL2, LGL3, one active index is created. Since the key sequence for LGL2 and LGL3 are subsets of LGL1, they can share the same active index.

If the compilation sequence is LGL3, LGL2, LGL1, three active indexes are created because the key sequence for LGL2 and LGL1 are not subsets of LGL3.

Access Path Maintenance (Immediate, Delay, or Rebuild)

There are three types of access path maintenance options: immediate (IMMED), delay (DLY), and rebuild (REBLD). These options determine how the operating system applies changes to the access paths. While a file is open, the system maintains the access paths as changes are made to the data in the file. However, when the file is closed, the access path maintenance option specifies to i OS how the access path should be maintained. It is important to pay close attention to access paths in the design stage.

- IMMED maintenance means that the active index is maintained as changes are made to its associated data regardless of whether the file is open.
- DLY maintenance means that any maintenance for the active index is done after the file member is opened and while it remains open. Updates to the access path are collected from the time the access path is closed until it is opened again. When it is opened, only the collected changes are merged into the access path.
- REBLD maintenance means that the active index is maintained only when the file is open, not when the file is closed; the access path is rebuilt the next time the file is opened. When the file is opened again, the access path is totally rebuilt. If one or more programs has opened a specific file member with rebuild maintenance specified, the system maintains the access path for that member until the last user closes the file member.

The use of maintenance options applies only to RSQ and SPN access paths. The UPD and RTV access paths are defined with a UNIQUE key; as a result, maintenance has to be IMMED. For QRY access paths, maintenance does not apply because the access path is rebuilt at execution.

There are considerations with each type of maintenance option. When you change a file, indexes with IMMED maintenance are updated. However, when programs that open DLY and REBLD maintenance access paths are invoked, changes are applied that then make the programs slower to load.

Use DLY maintenance with caution. If more than approximately 10% of the number of entries in the access path are changed, the whole index will be rebuilt at the next open. This rebuild could occur during the use of a program that does not use that index.

If the file records are non-volatile, you can always take the default IMMED maintenance. If the data is likely to change for infrequently used access paths, you may use DLY or REBLD maintenance. However, keep in mind that for each different type of maintenance, you will get a separate active index. Specify a REBLD access path if you already have an IMMED maintenance access path with the same set of keys and select/omit sets.

The following is a comparison of IMMED, DLY, and REBLD maintenance as they affect opening and processing files:

Immediate	Delay	Rebuild
Fast open because the access path is current.	Moderately fast open because the access path does not have to be rebuilt, but it must still be changed. Slow open if extensive changes are needed.	Slow open because access path must be rebuilt.
Slower update/output operations when many access paths with immediate maintenance are built over changing data (the system must maintain the access paths).	Moderately fast update/output operations when many access paths with delayed maintenance are built over changing data and are not open (the system records the changes, but the access path itself is not maintained).	Faster update/output operations when many access paths with rebuild maintenance are built over changing data and are not open (the system does not have to maintain the access paths).

For more information:

- On the access path maintenance options, see Editing Access Path Details in the Modifying Access Paths" chapter and Identifying the Implementation Attributes in the "Generating and Compiling" chapter.
- On the effect of the access path maintenance options on performance, see *IBM i Database Guide*.

Maintenance for Query (QRY) Access Paths

For Query (QRY) access paths, the access path maintenance option is approximated using the OPTIMIZE parameter on the i OS Open Query File OPNQRYF command. The OPTIMIZE parameter indicates the optimization goal the system is to use when satisfying the QRY access path specifications.

The following table gives a brief description of each OPTIMIZE parameter value and shows the corresponding access path maintenance option:

OPNQRYF Command OPTIMIZE Parameter Values (for QRY Access Paths)		Corresponding Access Path Maintenance Option
*FIRSTIO	The system attempts to improve the time required to open the file and to retrieve the first buffer of records from the file.	*IMMED

*MINWAIT	The system attempts to minimize delays when reading records from the file.	*DLY
*ALLIO	The system attempts to improve the total time to process the whole query, assuming that all query records are read from the file.	*REBLD

For more information:

- On QRY access paths, see Recognizing the Basic Properties of Access Paths in the "Access Paths: An Introduction" chapter and Adding a Query (QRY) Access Path in the "Adding Access Paths" chapter.
- On the OPNQRYF command's OPTIMIZE parameter, see *IBM i Control Language Reference.*

Using Select/Omit Maintenance

Select/omit maintenance filters records that match the specified selection or omission maintenance. The two types of select/omit maintenance are: static and dynamic.

If the select/omit maintenance is dynamic, all records, regardless of the select/omit set, are included in the access path. The system filters the records as the program reads them. The program returns only the records that match the select/omit maintenance.

If the select/omit criteria is static, only those records that satisfy the select/omit maintenance are included in the access path. As each record is added or changed, the system determines if it should be included in the access path. As the data is read, no filtering is required since the access path maintenance has performed the filtering.

Note: In the DDL generation mode, irrespective of the select/omit criteria setting being static or dynamic, only those records that satisfy the select/omit criteria are included in the access path.

For more information about the relation between static/dynamic maintenance and the generation modes, refer to Select/Omit in DDL Index (see page 89).

Because of this difference, any access path with static select/omit criteria usually has a separate internal active index, whereas any access path with dynamic select/omit maintenance can share an active index with other similarly keyed access paths even if the select/omit criteria differ. The provision and use of dynamic select/omits increases the possibilities of sharing active indexes.

Note: For join logical files in CA 2E (files with virtual fields) with select/omit criteria, the static select/omit maintenance cannot be used. Use the dynamic select/omit maintenance.

The type of data file is again important. For non-volatile (master or table) data files the logical files should, if possible, have static select/omit criteria. The static select/omit maintenance requires an extra active index to be maintained. But the frequency of change to the data is low and therefore maintenance does not occur often. For volatile (transaction) data files, the type of select/omit criteria depends on the actual number of records being read and the frequency of use of the active index.

If the active index is used infrequently or in batch, the overhead of having dynamic select/omit criteria may might be acceptable.

Using Join Logicals

A join logical file is a logical view of the physical files upon which it is based. It also enables data from another physical file or a related (or joined-to) physical file to be read. The join is performed when the records in the based-on physical are read. One or more fields in the based-on physical are matched with fields in each joined-to physical and data from each matched record is read. The join logical is composed of only one record format, which contains fields from the based-on file and the joined-to physical files. The join fields are read-only capable and cannot be updated through the join logical.

An access path generates and compiles as a join logical if there are virtual field entries on the access path format entries. The physical file that contains the virtual fields is joined to the primary files using the keys of the file-to-file relations to provide the match. These relations can be Owned by, Refers to, or Extended by.

The operating system does the work. The join (matching records through common fields) is implemented using a machine interface (I/O) routine to read from the joined-to physical files. You can define join logicals that join through several physical files. The maximum number of physical files allowed in a join file definition (direct joins or chained joins) is 32. A field that is brought across more than six (or as few as three or four) chained joins actually loses the performance benefit because it requires many I/O routines to obtain the data. Consequently, obtaining data across several joins is not recommended.

The use of join logicals is efficient only when the data that is being read is actually needed. If the data is not required, the operating system is doing unnecessary I/O to other files. In terms of internal active indexes, the join logical shares the active indexes of the based-on file just as with access path types.

Maintenance can be specified as IMMED, DLY, or REBLD. Only dynamic select/omit maintenance can be specified on CA 2E join logical files. In terms of performance, this has some significant consequences:

- Where it may be better to have static select/omit maintenance, you can drop the virtual field so that it can be specified. The additional data can be retrieved using your own reads to the secondary file.
- The operating system reads the joined-to data before applying the selection because the selection itself could be on a joined-to virtual field. This compounds the performance problem particularly where there is I/O to several different physical files.

Using Multi-Format Access Paths

Multi-format access paths are access paths with more than one record format. They are based on more than one physical file and can read/write to several physical files.

The multi-format can be treated as though each format were a separate access path. The use of the physical files for each format should collectively determine the type of maintenance for the access path. A select/omit maintenance can be applied to each format that is best suited for the physical file, the effect on performance (hit-rate), and the number of records being read. In terms of active indexes, each multi-format access path can share an active index. However, in practice, this is unlikely. Therefore, treat each multi-format access path as if it creates an extra active index over each of the physical files. CA 2E supports the multi-format logicals for the SPN access path.

Using Open Data Paths

An open data path (ODP) is the path through which all input/output (I/O) operations for the file are performed. ODPs can be shared by more than one program in the same job but not between jobs.

The object is to be able to share ODPs and still have the applications work as before. Sharing the ODP reduces the amount of main memory the job needs and reduces the amount of time it takes to open and close the file.

The amount of work the operating system does can be reduced by sharing the ODP among all the programs in the jobs that use it. When a file is used more than once in the same job, it can be shared. Sharing an ODP should be done with care. With each access path the position of the current record (the file cursor) is held only once on the ODP. It is possible that one program accessing a file with a shared ODP can change the position in the file inadvertently, causing unpredictable results in other programs in the job.

Sharing an ODP is not defined as a default for access paths. You must specify share ODP on the compiler override for a given access path.

Creating Default Retrieval Access Paths

To improve the performance of your application, try dropping all of the virtual fields from the default RTV access path. You can use this standard with every file you implement in CA 2E that contains virtuals.

The idea is to create an access path that can be used for validation and that does not have any virtual fields. To do this use the following steps:

1. **Zoom into the file.** At the Edit Database Relations panel, type **Z** next to any relation for the selected file and press Enter. Alternatively, you can select option 2 from the Edit Model Object List panel.

   The Edit File Details panel displays.

2. **Rename the default retrieval.** Rename the default RTV access path for your file to the name Validation View.
3. **Create a new RTV access path** if you require one that contains virtuals.
4. **Rename.** Name the new RTV access path Retrieval Index.
5. **Use the new RTV.** Base any functions for the file over the new RTV access path that requires those virtual fields.
6. **Create the validation view** with a compiler override of SHARE (*YES).

For more information on compiler overrides, see the "Generating and Compiling Your Application" chapter in *Generating and Implementing Applications*.

CA 2E uses the default RTV Index Validation View for all file validations. This view also uses the index to determine which fields in the file can be virtualized. The fields available for virtualization will default to only those fields present in the file itself. Although it is not generally a good performance technique, you can virtualize fields that are themselves virtualized. You can do this by adjusting the referenced access path of the relation to point to an untrimmed access path using the Edit Access Path Relations panel.

Index

*

*ARRAYS • 104
*DSNR User type • 103

A

access path • 17, 18, 19, 20, 21, 22, 23, 24, 53, 55, 56, 57, 58, 61, 62, 64, 67, 69, 70, 71, 72, 75, 77, 79, 81, 82, 84, 86, 91, 94, 97, 99, 100, 110, 112, 115, 116, 117, 121, 122, 126, 127
  adding • 53
  auxiliaries • 24, 94, 97
  characteristics • 22
  default retrieval • 117, 127
  deleting • 99
  details • 23
  documenting • 115, 116
  editing • 67, 69
  entry • 77
  format • 77
  format entries • 75, 77
  format keys • 75
  format text • 75
  generating • 112
  holding • 62, 81
  maintenance • 69, 70, 71, 121
  modifying • 61
  multi-format • 117, 126
  naming • 23
  performance • 117
  physical (PHY) • 18, 55, 56
  query (QRY) • 21, 55, 57, 67, 71, 94, 110, 122
  referenced • 91
  relations • 79, 84
  required relations • 79
  resequence (RSQ) • 20, 55, 56
  retrieval (RTV) • 19, 55
  span (SPN) • 22, 55, 58
  tailoring • 86
  trimming • 64
  types • 17, 72
  update (UPD) • 18, 55
  usages • 100
  virtual field • 64, 82
action diagram • 34
  compute condition symbols (YACTCND) • 34
  compute expression symbols (YACTFUN) • 34
  structure symbols (YACTSYM) • 34
active indices • 117, 118, 119
  minimizing number • 118
  sharing • 119
adding • 56, 57, 58, 91
  access paths • 56
  based-on access paths • 91
  query access paths • 57
  resequence access paths • 56
  span access paths • 58
allocating names • 25, 28
alternate collating sequence • 69, 72
alternating collating table • 72
ANDed • 87
array • 101, 102, 103, 104, 107, 108
  *CVTVAR function • 102
  as data structure • 102
  changing names • 108
  Closedown Programs • 103
  COBOL arrays • 103
  deleting • 108
  details • 107
  editing • 107
  ELM context field • 102
  field definitions • 107
  field details • 104
  key details • 104
  key order • 107
  sample • 104
assimilated files • 77
assimilated physical files • 77
associated access path • 91
auxiliaries • 24

B

based-on access path • 91
basic properties • 17
Bi-directional support • 48
  help text • 48
built-in functions • 102
  convert variable • 102

C

changing • 76, 91, 108
  array name • 108
  key sequence • 76
  referenced access path • 91
CHGOBJ • 102
CL program • 57
COBOL arrays • 103
command key defaults • 51
comments • 39
  supressing in source code • 39
compilation • 109
  overview • 109
compiler overrides • 32, 110
  changing • 32
components • 23
condition values • 36
  CUA prompt (YCUAPMT) • 36
confirm prompt value (YCNFVAL) • 35
context • 102
  ELM (array element) • 102
Copy Model Objects (YCPYMDLOBJ) • 74
  SQL access path • 74
Create Logical File • 32
Create Object • 102
Create Physical File • 32
Create Table • 72
CRTLF • 32
CRTOBJ • 102
CRTPF i OS command • 32
CRTTBL • 72
CUA prompt • 36

D

data area • 65
data definition language • 73
database file generation • 30
database implementation • 37
date • 37
  cutoff (YCUTOFF) • 37
  format (YDATFMT) • 37
  validation (YDATGEN) • 37
DDS • 25, 30, 57, 67, 73, 74, 94, 109, 110
  *Unique keyword • 67
  implementation attributes • 110
  joins • 74
default • 25, 33, 38, 51, 53, 117, 127
  access paths • 53
  environment (YEXCENV) • 38
  function keys • 51
  model values • 25, 33
  options • 25
  retrieval access paths • 117, 127
defining • 101
  array • 101
delay • 121
  DLY • 121
delay maintenance • 67
delete • 99, 108
  access path • 99
  array • 108
  confirmation • 99
designer user type (*DSNR) • 101, 102, 103
  arrays • 101, 102
device design • 36, 40, 42, 44, 63
  external MSGIDs (YPMTGEN) • 42
  leaders (YLSHFLL) • 40
  panel layout (YSAAFMT) • 44
  right-hand text (YCUAEXT) • 36
Display Documentation Menu • 116
Display Services Menu • 112, 116
displaying references • 65
distributed flag • 66
DLTOBJ • 102
DLY • 121
documentation • 115, 116
  access paths • 116
  functional text • 115
  operational text • 115
  type • 116
Documentation Model Access Paths panel • 116
DRDA • 39, 66
  default RDB (YGENRDB) • 39
  distributed flag • 66
duplicate sequence • 70
  options • 70
dynamic selection • 124

E

Edit File Details panel • 54
Edit Function Options panel • 49
editing • 69, 77, 97, 107
  access path auxiliaries • 97
  access path details • 69
  arrays • 107
  format entries • 77

physical file format entries • 77
ELM context • 102
entry • 77
Extended by relation • 83

F

F4 prompt • 36, 91
function • 91
YCUAPMT model value • 36
FCFO • 70, 110
FIFO • 70, 110
file and access path relations • 84
file-to-file relation • 79
format • 77
format entries • 23, 77
format keys • 75
format text • 75
function • 50
changing name • 50
function key • 51
defaults • 51
function name • 50
changing • 50
function option • 42
DDS PUTOVR keyword (YPUTOVR) • 42
function references • 107
functional text • 115

G

generation • 25, 109, 112
objects • 25, 109
generation mode • 30, 31, 69, 73
changing • 31
options • 73
generation options • 109

H

held access paths • 80
Help text • 39, 40, 41, 48
cursor-sensitive (YHLPCSR) • 40
generation (YGENHLP) • 39
UIM generation (YNPTHLP) • 41
UIM model values • 48
hidden fields • 63, 80
high level language (HLL) • 39, 109
default (YHLLGEN) • 39
source code • 109
HLL • 39, 109
high level language (HLL) • 39, 109
holding • 62

I

i OS access path • 71
i OS index • 109
I/O • 86, 125, 126
IBMsarchitecturalframework' • 66
identifying • 17, 75
format keys • 75
format text • 75
immediate maintenance • 67
implementation attributes • 69, 110
SQL-generated RSQ • 110
table • 69
input fields • 63
input/output • 86, 126
item master file • 118

J

join logicals • 117, 125

K

key sequence • 56, 76

L

LIFO • 70, 110
locks • 65
permanent • 65
temporary • 65

M

maintenance • 117, 121, 124
dynamic • 124
select/omit • 124
static • 124
maintenance options • 71, 110, 121
D • 71
delay • 71
I • 71
immediate • 71
R • 71
rebuild • 71
master files • 118
MDLVAL • 31
message • 35, 42, 45
Copy Back (YCPYMSG) • 35

device prompt file (YPMTMSF) • 42
when to send (YSNDMSG) • 45
minimizing number • 118
active indices • 118
model values • 25, 28, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 48, 49, 50, 61, 73
changing • 28
changing function level • 49
changing model level • 50
comments in source code (YGENCMT) • 39
CUA prompt • 36
F4 prompting • 36
last used file prefix (YFILPFX) • 25
prompting (F4) • 36
YABRNPT • 34
YACTCND • 34
YACTFUN • 34
YACTSYM • 34
YACTUPD • 35
YALCVNM • 25, 35
YBNDDIR • 35
YCNFVAL • 35
YCPYMSG • 35
YCRTENV • 36
YCUAEXT • 36
YCUAPMT • 36
YCUTOFF • 37
YDATFMT • 37
YDATGEN • 37
YDBFGEN • 25, 37
YDFTCTX • 38
YERRRTN • 38
YEXCENV • 38
YFILPFX • 25
YGENCMT • 39
YGENHLP • 39
YGENRDB • 39
YHLLGEN • 39
YHLLVNM • 39
YHLPCSR • 40
YLSHFLL • 40
YNLLUPD • 41
YNPTHLP • 41
YOBJPFX • 25, 41
YPMTGEN • 42
YPMTMSF • 42
YPUTOVR • 42
YRP4HS2 • 43
YRP4HSP • 43
YRP4SGN • 43
YSAAFMT • 44
YSFLEND • 44
YSHRSBR • 44
YSNDMSG • 45
YSQLLIB • 25
YUIMBID • 48
YUIMFMT • 48
YUIMIDX • 48
YWBDATR • 49
YWBDCHR • 49
YWBDCLR • 49
modifying • 61, 67, 75, 79, 82, 94
access path auxiliaries • 94
access path details • 67
access path format entries • 75
access path relations • 79
access paths • 61
virtual field entries • 82
multi-format • 117, 126

N

naming • 23, 35, 39, 41
automatic (ALCVNM) • 35
new functions (YHLLVNM) • 39
prefix (YOBJPFX) • 41
narrative text • 24
non-volatile • 118
null update suppression • 41

O

objects • 25
prefixes • 25
ODP • 117, 126
sharing • 126
omit set • 87
open data path (ODP) • 117, 126
Open Query File (OPNQRYF) • 57, 67, 71, 94, 110, 122
operational text • 115
ORed • 87

P

parameters • 30, 32, 115
PRTTEXT • 115
SQLLIB • 30
performance considerations • 117
access paths • 117

permanent locks • 65
PHY (physical) access path • 18, 55, 56
physical file • 77, 118
  format entries • 77
  types of data • 118
programmer user type (*PGMR) • 101, 102
  arrays • 101, 102
prompting (F4) • 36

Q

QCASE256 • 72
QRY (query) access path • 21, 29, 55, 57, 67, 71, 94, 110, 122
  adding • 57
  auxiliaries • 29
QSYSTRNTBL • 72
query (QRY) access path • 29, 67, 71, 94, 110, 122

R

rebuild maintenance • 67, 121
referenced access path • 91
Refers to relation • 79
relation • 23, 79, 125
  file-to-file • 79
required relations • 79
retrieve condition • 36
right-hand side text • 36
  YCUAEXT model value • 36
RPG • 38
  error handling (*PSSR) • 38
RSQ (resequence) access path • 20, 55, 56, 110, 112
  adding • 56
RTV (retrieval) access path • 19, 55
RTVOBJ • 102

S

select/omit • 69, 72, 86, 87, 117, 124
  criteria • 69, 72, 86
  maintenance • 124
  sets • 87
  static • 72
Services Menu • 116
  Display Services Menu • 116
sharing active indices • 119
source generation type • 25
source member name • 28, 69
source member text • 69
SPN (span) access path • 22, 55, 58
  adding • 58
SQL • 25, 30, 73, 74, 109, 110, 112
  access path • 110
  copying • 74
  creating an environment • 30
  environment • 30
  implementation • 110
  joins • 74
  resequence (RSQ) access path • 110
  SQLLIB model value • 25
static selection • 124
Synon/2E index • 109

T

table files • 118
tailoring for performance • 86, 117
tailoring virtual fields • 86
temporary locks • 65
translate table • 72
type of maintenance • 71
types of access paths • 17, 18, 19, 20, 21, 22, 55
  physical • 18, 55
  query • 21, 55
  resequence • 20, 55
  retrieval • 19, 55
  span • 22, 55
  update • 18, 55
types of data • 117, 118

U

UIM • 48
unique/duplicate key sequence • 69, 70
  options • 70
UPD (update) access path • 18, 55
UPD (update) physical path • 18
update suppression • 41
  null update suppression • 41
upper/lower case discrepancies • 72
usages • 65, 100
  impact analysis • 100
User Interface Manager (UIM) • 48
user-added text • 24

V

virtual fields • 62, 64, 82, 86, 117
  auto add to access path • 62
  entries • 82
volatile • 118

W

windows • 49
    border model values • 49

Y

YALCVNM (Allocate Name) model value • 25
YCHGMDLVAL (Change Model Value) • 25
YCPYMDLOBJ (Copy Model Object) • 74
    SQL access path • 74
YCRTMDLLIB (Create Model Library) • 30
YCRTSQLLIB (Create SQL Library) • 30
YDBFGEN (Database Generation) model value • 25, 30
YFILPFX (last used file prefix) model value • 25
YGENCMT model value • 39
YOBJPFX (Object Prefix) • 25
YSAAFMT model value • 44
YSQLLIB (SQL Library) model value • 25