ASG Software Solutions

ASG-Encore™ Reference Guide

Version 7.0
Publication Number: ENX0400-70
Publication Date: February 2003

The information contained herein is the confidential and proprietary information of Allen Systems Group, Inc. Unauthorized use of this information and disclosure to third parties is expressly prohibited. This technical publication may not be reproduced in whole or in part, by any means, without the express written consent of Allen Systems Group, Inc.

© 2003 Allen Systems Group, Inc. All rights reserved.
All names and products contained herein are the trademarks or registered trademarks of their respective holders.

ASG Worldwide Headquarters Naples Florida USA | asg.com | info@asg.com
1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 800.932.5536 (USA only)

Contents

Preface . . . . . **xvii**

About this Publication . . . . . **xvii**

Related Publications . . . . . **xviii**

ASG-Existing Systems Workbench (ASG-ESW) . . . . . **xix**

Invoking ESW Products . . . . . **xxii**

ESW Product Integration . . . . . **xxiii**
Example 1 xxiv
Example 2 xxv

Publication Conventions . . . . . **xxvii**

ASG Customer Support . . . . . **xxvii**
Intelligent Support Portal (ISP) xxviii
Telephone Support xxviii

ASG Documentation/Product Enhancements . . . . . **xxx**

1 Introduction **1**

Encore Advantages **1**

Encore Overview **2**
Encore Problem Solving Features 2

Encore Components **3**
Re-engineering Analysis 3
Code Extraction 4
Migration 4
Target Generation 4

Operational Requirements **5**
Environment 5
COBOL Compilers 6
Preprocessor Support 6

2 Product Overview 7

Introduction 7

Encore User Interface 8
The Action Bar 9

Help Facility 15

Multiple Views 15

Wildcard Patterns 16

Understanding COBOL Terms 16
Sets 16
Subsets 16
COBOL Subsets 17
Targets 20

3 File Menu 25

File Pull-down 26

File - Open Program Pop-up 28

File - Program Name List Pop-up 30

File - Logic Segment Directory Pop-up 33

Logic Segment Description Pop-up 34

File - Save Logic Segments Pop-up 35

File - Save Segment Pop-up 37

File - Analyze Submit Pop-up 39

File - AKR Utility Pop-up 43

File - AKR Directory Pop-up 45

File - AKR Allocate/Expand Pop-up 49

Edit Interface 52
File - Edit Options Request Pop-up 52
Options - COPY/Include Libraries Pop-up 53
ISPF Edit - Entry Panel 55
Edit Screen 56
File - Display Logic Segment Entry Pop-up 57
Displayed Logic Segment 58

File - Execute Script Pop-up 59

Anomaly Facility **61**
File - Anomaly Facility Pop-up 61
Anomaly Facility - Batch Submit Pop-up 62
Anomaly Facility - Program Selection Pop-up 64
Anomaly Facility - User Parameters Pop-up 66
Anomaly Facility - Edit Screen 67

4 View Menu **69**

Introduction **69**

View Pull-down **71**

Source View Screen **73**
Source View Primary Commands 74
Source View Line Commands 75
View - Source Pop-up 76
View - Tree View Request Pop-up 77

Tree View Screen **80**
Tree View Primary Commands 82
Tree View Line Commands 83
View - Tree Pop-up 84
View - Structure View Request Pop-up 85

Structure View Screen **87**
Structure View Primary Commands 89
View - Perform Structure Pop-up 90
View - Select Logic Segment Pop-up 91
View - Program Summary Pop-up 92
Program Summary - Files Pop-up 94
Program Summary - Data Pop-up 96
Program Summary - Copy Members Pop-up 98
Program Summary - Sections/Paragraphs Pop-up 99
Program Summary - Control Flow Pop-up 101
Program Summary - Preprocessor Statements Pop-up 102
View - Paragraph Cross-Reference Request Pop-up 104
View - Paragraph Cross Reference Pop-up 106
View - Levels Request Pop-up 108
View - Reset Request Pop-up 109
View - Exclude Request Pop-up 111

5 Extract Menu **113**

Extract Overview **114**

Perform Range Extract **116**

Report Extract **116**

Computation Variable Extract 117

Transaction Extract 118

Statement Extract 118

Complement Extract 119

Common Code Extract 119

Test Driven Extract 120

CICS Server Extract 121

Extract Objective 122
Overview 122

Extract Pull-down 124

Extract - Name Logic Segment Pop-up 126

Perform Range Extract 127
Extract - Perform Name List Pop-up 127
Extract - Perform Range Filter Pop-up 132
Select View Type Pop-up 134
Extract - Sort PERFORM Range Pop-up 136

Report Extract 137
Extract - Report Name List Pop-up 138
Extract - Report Filter Pop-up 141
Select Extract-Within Logic Segment Pop-up 144
Extract - Sort Report Names Pop-up 145
Extract - Source Ordered View Pop-up 146

Computation Variable Extract 147
Extract - Data Name List Pop-up 148
Extract - Computation Variable Filter Pop-up 152
Extract - Sort Data Name Pop-up 154
Extract - Source Ordered View Pop-up 155

Transaction Extract 157
Extract - Transaction Paths List Pop-up 159
Extract - START/END Statements Pop-up 162
Extract - Data Name List Filter Pop-up 164
Extract - Transaction Variable Pop-up 167
Extract - Sort Data Name Pop-up 170

Statement Extract 171
Extract - Source Statement List Pop-up 171
Extract - Statement Filter Pop-up 174
Statement - And/Or Logic Segment Pop-up 177

Complement Extract 178
Extract - Complement Pop-up 179
Extract - Transaction Complement Paths Pop-up 180

Common Code Extract 182
Extract - Common Code Pop-up 183

Test Driven Extract 184
Extract - Test Driven Pop-up 184
Extract - Edit Segment Pop-up 187

CICS Server Extract 188
Extract - Server Pop-up - CICS Selection 188
Extract - CICS Server Endpoints Pop-up 189
Extract - CICS Server Startpoints Pop-up 190
Extract - CICS Server Paths Pop-up 191
Extract - Server Pop-up - Self-Directed Selection 192
Extract - Self Directed Server Endpoints Pop-up 193
Extract - Self Directed Server Filter Pop-up 194
Extract - Self Directed Server Data Names Pop-up 196
Extract - Self Directed Server Startpoints Pop-up 198
Extract - Self Directed Server Paths Pop-up 199
Extract - Self Directed Server Outputs Pop-up 201

6 Generate Menu 203

Introduction 203

Generate Pull-down 204

COBOL Modules 206
Generate - Select Logic Segment Pop-up 207
Generate - COBOL Module Pop-up 208
Select FD Names Pop-up 210
Generate - Specify Perform Range ENTRY Names Pop-up 211
Generate - Multi-module PERFORM Ranges Pop-up 214
Generate - Specify Perform Range CALL Names Pop-up 215
COBOL Module Contents 216
Callable Module USING Clause 218
Callable Module CALL Statement 219
DATA DIVISION COPY Member Processing 220
Subprogram and Program Contents 220
Perform Range Extract 221
Report Extract 222
Computation Variable Extract 224
Transaction Extract 226
Statement Extract 228

File IO Modules 230
Generate - FD Name List Pop-up 230
Generate - IO Module Pop-up 231

Output Files 233
Output File - Select Logic Segment Pop-up 234
Generate - Output File Pop-up 235

Batch Computation 236
Batch Computation - Select Logic Segments Pop-up 236
Batch Computation - Save Segment Pop-up 237
Logic Segment - Batch Submit Pop-up 239

CICS and Self-Directed Server Generation 241
Generate - Select Logic Segment Pop-up 241
Generate - HOST Copy Member Pop-up 243
Generate - HOST Server Module Pop-up 245

7 Search Menu 249

Introduction 250
Current search command options 250

Search Pull-down 251

Search - Data Name Pop-up 253

Search - Label Name Pop-up 256

Search - Paragraph Name Pop-up 258

Search - String Pop-up 260

Search - COBOL Subset Name Pop-up 262

Search - Perform Range Name Pop-up 265

Search - Program Name Pop-up 267

Search - Line Range Pop-up 269

Search - Any/Unknown Type Pop-up 270

Search - Branch Request Pop-up 272

Branch Previous Options Pop-up 275

Selection List Pop-ups 276

IN-Clause Option Pop-up 277

8 List Menu 281

List Pull-down 282

List - CALL Statements Pop-up 284

List - Equates Pop-up 285
List - Perform Range Names Pop-up 286
List - Program/Subprogram Names Pop-up 287
List - COBOL Subset Names Pop-up 288
List - Segments Pop-up 289
Copy Logic Segment Pop-up 291
Rename Logic Segment Pop-up 292

9 Options Menu 293
Options Pull-down 294
Options - Product Parameters Pop-up 296
Options - Product Allocations Pop-up 298
Options - Log/List/Punch Definition Pop-up 300
Options - Script File Allocations Pop-up 304
Options - PF Key Definition Pop-up 305
Options - Processing Modes Pop-up 308
Options - Scratchpad Equate Pop-up 310
Options - Generate Pop-up 312

10 Help Menu 315
Introduction 315
Help Pull-down 316
Help - Specific ASG Command Pop-up 318
Help - Specific ASG Message Number Pop-up 319
Help - About Pop-up 320

11 Commands 321
Introduction 324
Command Processing 324
Command Syntax Diagrams 325
Cursor Position 327
Cursor Substitution Character 327

& (Retain) Command 328
Function. 328
Operands 328

ALLOCDEF Command 329
Function. 329
Operands 329

ANALYZE Command 330
Function. 330
Operands 330

BRANCH Command 331
Function. 331
Operands 331

CALL Command 334
Function. 334
Operands 334

CANCEL Command 335
Function. 335
Operands 335

CURRENT Command 336
Function. 336
Operands 336

DISPLAY Command 337
Function. 337
Operands 337

END Command 338
Function. 338
Operands 338

EQUATE Command 339
Function. 339
Operands 339

EXCLUDE Command. 340
Function. 340
Operands 340

EXECUTE Command. 346
Function. 346
Operands 346
Examples 348

EXIT Command 350
Function. 350
Operands 350

EXPORT Command 351
Function. 351
Operands 351

FIND Command 353
Function. 353
Operands 353

FINDXTND Command 358
Function. 358
Operands 358
DBCS Strings 366

GOBACK Command 368
Function. 368
Operands 368

HELP Command 369
Function. 369
Operands 369

JUMP Command 371
Function. 371
Operands 371

KEYS Command 372
Function. 372
Operands 372

LEVELS Command. 374
Function. 374
Operands 374

LIST Command 375
Function. 375
Operands 375
List Pop-up 376

LOCATE Command 377
Function. 377
Operands 377

LPRINT Command 379
Function. 379
Operands 379

LPUNCH Command. 381
Function. 381
Operands 382

PARMDEF Command. 389
Function. 389
Operands 389

PREF Command **390**
Function 390
Operands 390

PRINTLOG Command **393**
Function 393
Operands 393

PRINTLST Command **394**
Function 394
Operands 394

PROCESS Command **395**
Function 395
Operands 395
Examples 397

PRODLVL Command **399**
Function 399
Operands 399

PROGRAM Command **400**
Function 400
Operands 400

QUALIFY Command **401**
Function 401
Operands 401

RECALL Command **402**
Function 402
Operands 402

REDO Command **404**
Function 404
Operands 404

REPEAT Command **405**
Function 405
Operands 405

RESET Command **406**
Function 406
Operands 406

RETURN Command **408**
Function 408
Operands 408

RFIND Command **409**
Function 409
Operands 409

RNSCOPY Command **410**
Function. 410
Operands 410

RSTRUCTURE-VIEW Command **411**
Functions 411
Operands 411

RTREEVW Command **412**
Function. 412
Operands 412

SAVE Command **413**
Function. 413
Operands 413

SCROLL Command **414**
Function. 414
Operands 414

SELECT Command **415**
Function. 415
Operands 415

SET Command **417**
Function. 417
Operands 417
LEARN Mode - Generated Command Pop-up 418

SORT Command **419**
Function. 419
Operands 419

SOURCE-VIEW Command **420**
Function. 420
Operands 420

STRUCTURE-VIEW Command **421**
Function. 421
Operands 421

TREEVIEW Command **423**
Function. 423
Operands 423

UNSELECT Command **425**
Function. 425
Operands 425

VIEW Command **427**
Function. 427
Operands 427

ZOOMIN/ZOOMOUT Commands 428
Function. 428
Operands 428

12 Analyze Functions 431

Introduction 432

Analyzing a COBOL Program 432

Analyze Input Descriptions 432
COBOL Source Program 432
Compile/Link JCL. 433
Analyze Options and Features 434

The Analyze Process 434

Analyze Using ISPF 435

Analyze Using ISPF/PDF Edit 436

File - Analyze Submit Screen 439

Automatic JCL Modifications 444

Analyze Summary Report. 449

Adding Analyze Facilities to a Standard Compile Mechanism 452
CLIST Compile Mechanism 452
ISPF Compile Mechanism 453

Analyze Options. 453
Buffers. 454
COBOL Level 455
DB2 Load Library 455
DB2 Application Plan 456
Dynamic CALLs 456
Flag Messages 457
Input 457
IO. 458
Language Level. 458
Line Count. 459
Main 459
Maximum Number of Errors. 459
Output 460
Program. 460
Recursion. 461
Return 461
Sequence 462
Source 462
Spacing 462
SQL Authorization ID. 463

DB2 Subsystem 463
Anomaly Reports 463
Anomaly Facility 464
Live Exit 464
Memory 464

13 AKR Utilities 465

Introduction to AKR Management 465

AKR Structure 466

Online AKR Utilities 466

Batch AKR Utilities 467
Job Control Statements 468
Control Cards 468

Command Format 469

Command Syntax 470
Comments 471
CONVERT Batch AKR Command 472
COPY Batch AKR Command 473
DELETE Batch AKR Command 474
EXPORT Batch AKR Command 475
HELP Batch AKR Command 476
INIT Batch AKR Command 477
MOVE Batch AKR Command 478
PRINT Batch AKR Command 479
PUNCH Batch AKR Command 480

Batch AKR Reports 481
Allocating and Expanding AKRs without ISPF 484

Glossary 489

Appendix A
Help Facility 503

Introduction 503

Help Navigational Commands 505

Screen and Pop-up Help 506

Field Help 507

Command Help 508

General Information 509

Specific Information **510**
Help Abends **511**
Help Messages **512**
Printing Messages 514

Appendix B
Command Reference **517**

Appendix C
Paragraph Control Transfer Methods **525**

Index **533**

Preface

This *ASG-Encore Reference Guide* provides detailed information about the pop-ups, screens, and commands used in ASG-Encore (herein called Encore). Encore is an integrated re-engineering environment for COBOL programs. The two functional components of Encore are re-engineering analysis and code extraction. Because Encore uses ISPF/PDF as its standard interface, you should be familiar with ISPF/PDF conventions and usage.

Allen Systems Group, Inc. (ASG) provides professional support to resolve any questions or concerns regarding the installation or use of any ASG product. Telephone technical support is available around the world, 24 hours a day, 7 days a week.

ASG welcomes your comments, as a preferred or prospective customer, on this publication or on any ASG product.

About this Publication

This publication consists of these chapters:

- Chapter 1, "Introduction," contains an overview of Encore's benefits, components, and operational requirements.
- Chapter 2, "Product Overview," contains information about using Encore's online help, graphical user interface (GUI) elements, and migration functions.
- Chapter 3, "File Menu," describes Encore's File pull-down menu functions.
- Chapter 4, "View Menu," describes Encore's View pull-down menu functions.
- Chapter 5, "Extract Menu," describes Encore's Extract pull-down menu functions.
- Chapter 6, "Generate Menu," describes Encore's Generate pull-down menu functions.
- Chapter 7, "Search Menu," describes Encore's Search pull-down menu functions.
- Chapter 8, "List Menu," describes Encore's List pull-down menu functions.
- Chapter 9, "Options Menu," describes Encore's Options pull-down menu functions.
- Chapter 10, "Help Menu,", describes Encore's Help pull-down menu functions.

- Chapter 11, "Commands," contains command syntax diagrams and descriptions for all of Encore's commands, which are used to enter easily-understood primary and line commands that are available to make your tasks easier.
- Chapter 12, "Analyze Functions," describes Encores' Analyze functions, which are used to gather information about a program, including program relationships, logic, data and execution paths, that are then stored in the Application Knowledge Repository (AKR). This chapter also includes a description of the analysis report parameters and the panel driven anomaly repair facility.
- Chapter 13, "AKR Utilities," describes Encore's online and batch AKR management utilities.

Related Publications

The documentation library for ASG-Encore consists of these publications (where *nn* represents the product version number):

- *ASG-Center Installation Guide* (CNX0300-*nn*) contains installation and maintenance information for ASG-Center, the common set of libraries shared by most ASG-Existing Systems Workbench mainframe products. ASG-Center must be installed before installing ASG-Encore.
- ASG-*Encore Installation Guide* (ENX0300-*nn*) contains installation and maintenance information for ASG-Encore.
- *ASG-Encore Reference Guide* (ENX0400-*nn*) provides detailed information about the pop-ups, screens, and commands used in ASG-Encore.
- *ASG-Encore User's Guide* (ENX0200-*nn*) provides user information about ASG-Encore.
- *ASG-ESW Enhancement Summary* (ESW1000-*nn*) highlights the new functionality for this release.

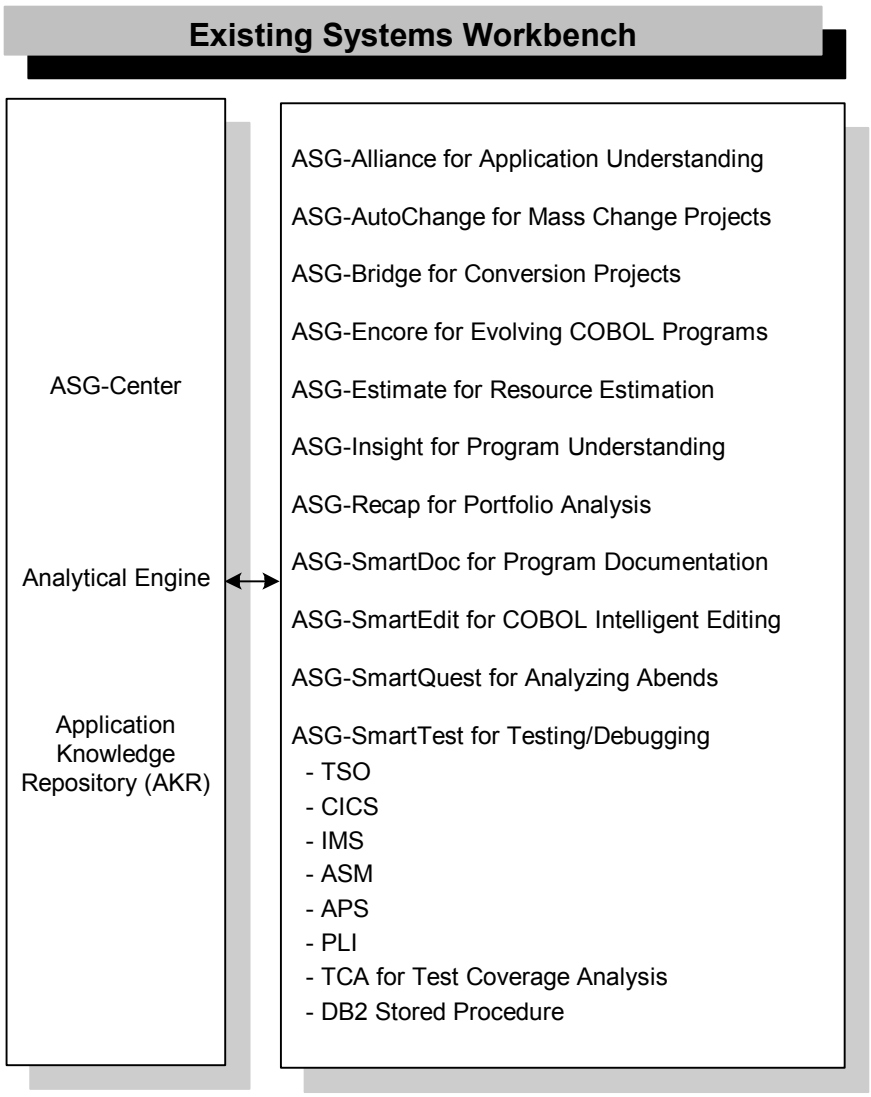
Note:

To obtain a specific version of a publication, contact ASG Customer Support.

ASG-Existing Systems Workbench (ASG-ESW)

ASG-ESW (herein called ESW) is an integrated suite of components designed to assist organizations in enhancing, redeveloping, or re-engineering their existing systems. ESW products use the Application Knowledge Repository (AKR) to store source program analysis information generated by the Analytical Engine. Figure 1 represents the components of ESW.

Figure 1 • ASG Existing Systems Workbench



This table contains the name and description of each ESW component:

ESW Product	Herein Called	Description
ASG-Alliance	Alliance	The application understanding component that is used by IT professionals to conduct an analysis of every application in their environment. Alliance supports the analysis and assessment of the impact of change requests upon an entire application. Alliance allows the programmer/ analyst to accurately perform application analysis tasks in a fraction of the time it would take to perform these tasks without an automated analysis tool. The impact analysis from Alliance provides application management with additional information for use in determining the resources required for application changes.
ASG-AutoChange	AutoChange	The COBOL code change tool that makes conversion teams more productive by enabling quick and safe changes to be made to large quantities of code. AutoChange is an interactive tool that guides the user through the process of making source code changes.
ASG-Bridge	Bridge	The bridging product that enables field expansion for program source code, without being required to simultaneously expand the fields in files or databases. Because programs are converted in smaller groups, or on a one-by-one basis, and do not require file conversion, testing during the conversion process is simpler and more thorough.
ASG-Center	Center	The common platform for all ESW products. Center provides the common Analytical Engine to analyze the source program and store this information in the AKR. This common platform provides a homogeneous environment for all ESW products to work synergistically.

ESW Product	Herein Called	Description
ASG-Encore	Encore	The program re-engineering component for COBOL programs. Encore includes analysis facilities and allows you to extract code based on the most frequently used re-engineering criteria. The code generation facilities allow you to use the results of the extract to generate a standalone program, a callable module, a complement module, and a CICS server. Prior to code generation, you can view and modify the extracted Logic Segment using the COBOL editor.
ASG-Estimate	Estimate	The resource estimation tool that enables the user to define the scope, determine the impact, and estimate the cost of code conversion for COBOL, Assembler, and PL/I programs. Estimate locates selected data items across an application and determines how they are used (moves, arithmetic operations, and compares). Time and cost factors are applied to these counts, generating cost and personnel resource estimates.
ASG-Insight	Insight	The program understanding component for COBOL programs. Insight allows programmers to expose program structure, identify data flow, find program anomalies, and trace logic paths. It also has automated procedures to assist in debugging program abends, changing a computation, and resolving incorrect program output values.
ASG-Recap	Recap	The portfolio analysis component that evaluates COBOL applications. Recap reports provide function point analysis and metrics information, program quality assessments, intra-application and inter-application comparisons and summaries, and historical reporting of function point and metrics information. The portfolio analysis information can also be viewed interactively or exported to a database, spreadsheet, or graphics package.
ASG-SmartDoc	SmartDoc	The program documentation component for COBOL programs. SmartDoc reports contain control and data flow information, an annotated source listing, structure charts, program summary reports, exception reports for program anomalies, and software metrics.

ESW Product	Herein Called	Description
ASG-SmartEdit	SmartEdit	The COBOL editing component that can be activated automatically when the ISPF/PDF Editor is invoked. SmartEdit provides comprehensive searching, inline copybook display, and syntax checking. SmartEdit allows you to include an additional preprocessor (for example, the APS generator) during syntax checking. SmartEdit supports all versions of IBM COBOL, CICS, SQL, and CA-IDMS.
ASG-SmartQuest	SmartQuest	The diagnostic tool for analyzing batch and CICS transaction abends. SmartQuest has been designed to make the maximum use of simple point-and-shoot techniques to enable fast and easy navigation through any data dump.
ASG-SmartTest	SmartTest	The testing/debugging component for COBOL, PL/I, Assembler, and APS programs in the TSO, MVS Batch, CICS (including file services), and IMS environments. SmartTest features include program analysis commands, execution control, intelligent breakpoints, test coverage, pseudo code with COBOL source update, batch connect, disassembled object code support, and full screen memory display.

Invoking ESW Products

The method you use to invoke an ESW product depends on your system setup. If you need assistance to activate a product, see your systems administrator. If your site starts a product directly, use the ISPF selection or CLIST as indicated by your systems administrator. If your site uses the ESW screen to start a product, initiate the ESW screen using the ISPF selection or CLIST as indicated by your systems administrator and then typing in the product command on the command line.

The product names can also vary depending on whether you access a product directly or through ESW. See "ESW Product Integration" on page xxiii for more information about using ESW.

To initialize ESW products from the main ESW screen, select the appropriate option on the action bar pull-downs or type the product shortcut on the command line.

Product Name (ESW Name)	Shortcut	ESW Pull-down Options
Alliance (Application Understanding)	AL	Understand ▸ Application
AutoChange (Conversion Set)	CC	Change ▸ Conversion Set
Bridge	BR	Change ▸ ASG-Bridge
Encore (Program Re-engineering)	EN	Re-engineer ▸ Program
Estimate	ES	Measure ▸ ASG-Estimate
Insight (Program Understanding)	IN	Understand ▸ Program
Recap (Portfolio Analysis)	RC	Measure ▸ Portfolio
SmartDoc (Program Documentation)	DC	Document ▸ Program
SmartEdit	SE	Change ▸ Program **Or** Change ▸ Program with Options
SmartQuest	SQV	Understand ▸ Abend/Dump
SmartTest (Testing/Debugging)	ST	Test ▸ Module/Transaction

ESW Product Integration

Because ESW is an integrated suite of products, you are able to access individual ESW products directly, or through the main ESW screen. As a result, different fields, values, action bar options, and pull-down options display on a screen or pop-up depending on how you accessed the screen or pop-up.

Certain ESW products also contain functionality that interfaces with other ESW products. Using SmartTest as an example, if Alliance is installed, SmartTest provides a dynamic link to Alliance that can be used to display program analysis information. If Insight is installed and specified during the analyze, the Insight program analysis functions are automatically available for viewing logic/data relationships and execution path. For example, the Scratchpad option is available on the Options pull-down if you have Insight installed.

Access to these integrated products requires only that they be installed and executed in the same libraries.

Example 1

Figure 2 shows the Encore Primary screen that displays when you access Encore directly.

The Encore Primary screen contains these eight action bar menu items: File, View, Extract, Generate, Search, List, Options, and Help.

Figure 2 • Encore Primary Screen

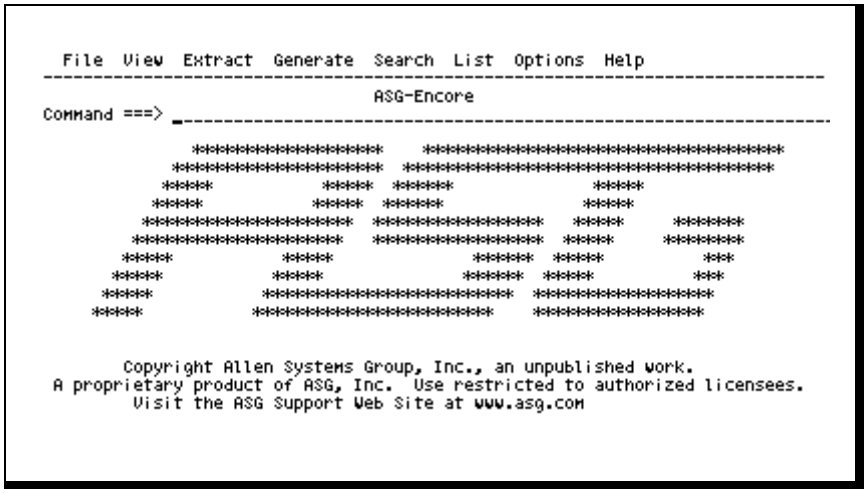


Figure 3 shows the Encore Primary screen that displays when you access Encore through ESW by selecting Re-engineer ▸ Program from the ESW action bar menu. Notice that the Primary screen name changes to ASG-ESW - Program Re-engineering when you enter Encore through ESW. Also, the Logic menu item displays if Insight is installed.

Figure 3 • ESW Encore Primary Screen

```
 File  View  Extract  Generate  Search  Logic  List  Options  Help
--------------------------------------------------------------------------------
                        ASG-ESW - Program Re-engineering
Command ===> ____________________________________________________________________

                 ******************    ***********************************
               ********************  ************************************
              *****            *****  ******               *****
             *****            *****  ******               *****
            ********************  *****************   *****     *******
           ********************   *****************  *****     *******
          *****            *****               ******  *****          ***
         *****            *****               ******  *****          ***
        *****            ************************  ******************
       *****            ************************    ****************


          Copyright Allen Systems Group, Inc., an unpublished work.
   A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
           Visit the ASG Support Web Site at www.asg.com
```

Example 2

Figure 4 shows the File - Analyze Submit pop-up that displays when you access SmartTest directly. Figure 5 on page xxvi shows the File - Analyze Submit pop-up that displays when you access SmartTest through ESW.

Figure 4 • File - Analyze Submit Screen

```
                              File - Analyze Submit
Command ===> ____________________________________________________________

          E - Edit JCL                                S - Submit JCL

Compile and link JCL (PDS or sequential):
   Data set name  ________________________________________________________

Analyze features (Y/N):
   ASG-SmartTest:  Y   Extended Analysis:  N

AKR data set name ____________________________________________
AKR program name  NEWDEMO              (if overriding PROGRAM-ID)

Analyze options:
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y
```

The actions shown on these screens can also vary. For example, the D - Doc Options action is only available on the File Prepare Program screen (or File - Analyze Submit screen) if SmartDoc is installed on your system. In Figure 4 on page xxv, the ASG-SmartDoc Options action does not display.

Figure 5 • ASG-ESW - Prepare Program Screen (accessed through ESW)

```
                         ASG-ESW - Prepare Program
Command ===> ____________________________________________________________

             E - Edit JCL     S - Submit JCL     D - Doc Options

Compile and link JCL (PDS or sequential):
   Data set name  ___________________________________________________________

Analyze features (Y/N):
   Understand:  N   Test:  Y   Extended Analysis:  N   Document:  N
   Re-engineer:  N   Abend/Dump:  N
AKR data set name ____________________________________________
AKR program name  NEWDEMO              (if overriding PROGRAM-ID)

Analyze options:
   ______________________________________________________________________
   ______________________________________________________________________
   ______________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y     (Test and Abend/Dump only)
```

Notice that the Analyze features field in Figure 5 lists additional ESW products than shown on Figure 4 on page xxv. This field is automatically customized to contain the ESW products you have installed on your system. These are the names of the analyze types:

Analyze Type	Analyze Type (ESW)
ASG-Encore	Re-engineer
ASG-Insight	Understand
ASG-SmartDoc	Document
ASG-SmartQuest	Abend/Dump
ASG-SmartTest	Test
Extended Analysis (ASG-SmartTest with Insight installed)	Extended Analysis

Publication Conventions

ASG uses these conventions in technical publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
`lowercase italic monospace`	Information that you provide according to your particular situation. For example, you would replace *`filename`* with the actual name of the file.
`Monospace`	Characters you must type exactly as they are shown. Code, JCL, file listings, or command/statement syntax. Also used for denoting brief examples in a paragraph.
Vertical Separator Bar (|) with underline	Options available with the default value underlined (e.g., Y|N).
Underline	Denotes a cursor-selectable field or line.

ASG Customer Support

ASG provides support throughout the world to resolve questions or problems regarding installation, operation, or use of our products. We provide all levels of support during normal business hours and emergency support during non-business hours.

ASG Third-party Support. ASG provides software products that run in a number of third-party vendor environments. Support for all non-ASG products is the responsibility of the respective vendor. In the event a vendor discontinues support for a hardware and/or software product, ASG cannot be held responsible for problems arising from the use of that unsupported version.

Intelligent Support Portal (ISP)

Online product support is available at: http://www.asg.com/support/support.asp via the ASG Intelligent Support Portal (ISP). Your logon information for ISP online support is:

Customer ID = *NNNNNNNNNN*
Password = *XXXXXXXXXX*

where:

NNNNNNNNNN is your customer ID supplied by ASG Product Distribution.
XXXXXXXXXX is your unique password supplied by ASG Product Distribution.

The *ASG-Intelligent Support Portal User's Guide* provides instructions on how to use the ISP and is located on the ASG Support web page.

Telephone Support

To expedite response time, please have this information ready:

- Product name, version number, and release number
- List of any fixes currently applied
- Any alphanumeric error codes or messages written precisely as displayed
- A description of the specific steps that immediately preceded the problem
- Verify whether you received an ASG Service Pack or cumulative service tape for this product. It may include information to help you resolve questions regarding installation of this ASG product. The Service Pack instructions are in a text file on the distribution media included with the Service Pack. You can access the latest software corrections and Service Packs via the ISP.
- The severity code (ASG Customer Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)

Severity Codes and Expected Support Response Times

Severity	Meaning	Expected Support Response Time
1	Production down, critical situation	Within 30 minutes
2	Major component of product disabled	Within 2 hours
3	Problem with the product, but customer has work-around solution	Within 4 hours
4	"How-to" questions and enhancement requests	Within 4 hours

The Americas

	Phone	Fax	E-mail
United States and Canada	800.354.3578	1.703.464.4901	support@asg.com

Europe, Middle East, and Africa (EMEA)

During normal business hours, we recommend that you call the Central Support number first (except in South Africa).

	Phone	Fax	E-mail
Central Support	00.800.3544.3578	44.1727.812018	support.emea@asg.com
English	44.1727.736305	44.1727.812018	support.uk@asg.com
French	33.141.028590	33.141.028589	support.fr@asg.com
German	49.89.45716.200	49.89.45716.400	support.de@asg.com
Italian	39.0290450025		support.it@asg.com
Dutch	31.30.241.6133		support.nl@asg.com
Spanish	34.913.523.800	34.917.156.961	support.es@asg.com
South Africa	800.201.423		support.sa@asg.com

Asia Pacific (APAC)

	Phone	Fax	E-mail
Central Support	61.3.9645.8500	61.3.9645.8077	support.au@asg.com
Australia	800.637.947	61.3.9645.8077	support.au@asg.com
Hong Kong	800.96.2800		support.hk@asg.com
Japan	81.3.5326.3684	81.3.5326.3001	support.au@asg.com
Singapore	65.224.3080	65.224.8516	support.sg@asg.com

All Other Countries (Also for any non-working numbers)

	Phone	Fax	E-mail
All other countries	1.239.435.2201		support@asg.com

If you receive a voice mail message, follow the instructions to report a production-down or critical problem. Leave a detailed message including your name and phone number. An ASG Customer Support representative will be paged and will return your call as soon as possible. Please have available the information described previously when the ASG Customer Support representative contacts you.

ASG Documentation/Product Enhancements

Submit all product and documentation suggestions to ASG's product management team at http://www.asg.com/asp/emailproductsuggestions.asp.

If you do not have access to the web, FAX your suggestions to product management at (239) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.

1 Introduction

This chapter contains an overview of Encore's benefits, components, and operational requirements, and contains these sections:

Section	Page
Encore Advantages	1
Encore Overview	2
Encore Components	3
Operational Requirements	5

Encore Advantages

Corporations are increasingly under pressure to provide these services:

- Integrate and web-enable their applications for better customer relations management.
- Open up their applications through standards-conforming interfaces for better supply-chain integration.
- Migrate their applications to newer platforms and technologies for better throughput, as well as to accommodate the capabilities of a changing labor force.
- Identify, consolidate, and extract processing logic from their applications to explore alternative implementations.
- Re-engineer their applications to improve the accuracy, repeatability, and efficiency of the maintenance process.

Encore provides automated support to help your IT staff achieve these objectives through the ability to analyze and report on the information in a program, including data usage, control flows, and PERFORM and CALL hierarchies. Encore extracts reusable, functional code fragments from a program and has the ability to package and generate the code fragments as separate program modules. Encore also regenerates the original program to take advantage of the new modules.

Encore Overview

Encore is the ESW re-engineering product for COBOL applications and includes these features:

- Analysis facilities that allow you to extract code based on the most frequently used re-engineering criteria.
- Code generation facilities that allow you to use the results of the analysis extract to generate a standalone program, a callable module, and a complement module. Prior to code generation, you can use the COBOL editor to view and modify the extracted Logic Segment.

Encore is designed to support these functions:

- Web enablement
- Open interface
- Platform and technology migration
- Re-implementation of business functions
- Component-based maintenance

Encore contains a Common User Access (CUA) interface with action bars, pull-down menus, and pop-up screens that make it easy to learn and use.

Encore Problem Solving Features

Encore assists you in these functional areas:

- Accelerating the transition to e-business by:
  - — Automatically creating CICS server applications from existing CICS COBOL programs by identifying and extracting server-side logic in CICS COBOL programs.
  - — Guiding users through the identification and extraction of business logic, user interface logic, and database logic using a variety of customer-specified options.

- Simplifying the integration of traditional applications and new technologies by:
  - Assisting in identifying key integration points in complex COBOL programs.
  - Facilitating the modernization of green screen user interfaces by automatically identifying and extracting BMS screen logic in CICS COBOL programs.
  - Facilitating the transition to new database technologies by automatically identifying and extracting database access logic in COBOL programs.
- Reducing the cost of maintaining complex applications by:
  - Automatically identifying and creating reusable components from COBOL programs regardless of size and complexity.
  - Preserving the functionality of the original COBOL program by creating calls to the newly created components.

Encore Components

Encore is comprised of these four functional components:

- Re-engineering analysis
- Code extraction
- Migration
- Target generation

Re-engineering Analysis

Encore re-engineering analysis displays the source code in various levels. The re-engineering analysis component consists of structure, tree, and source views.

The main purpose of re-engineering analysis is to gather enough information about the program to make the best decision regarding code isolation and extraction. Re-engineering analysis is available through the View facility.

Code Extraction

Encore's main focus is the Extract facility, which addresses the re-engineering aspects of code renewal, and provides the capability to isolate and extract logical code segments from existing programs. These are the eight methods used to isolate and extract code:

- Perform Range
- Transaction
- Computation Variable
- Report
- Statement
- Complement
- Common Code
- CICS Server

Migration

The most basic feature of Encore is to convert an existing MVS COBOL application to a component-based architecture. Encore provides these capabilities for migration:

- Clear separation of user interface and data access code from the business logic.
- Removal of redundant logic.

Target Generation

Target Generation addresses the disposition of the segment created in the Code Extraction phase. Encore provides these capabilities for Target Generation:

- Separate program generation.
- CALLable module generation.
- Complement module generation consisting of the original program with the extracted Logic Segment removed.
- IO module generation containing all input/output functions for an FD (File Description).
- Logical COBOL segment access with an editor such as ISPF Edit or SmartEdit for modification and customization.
- Knowledge retention that helps in everyday maintenance tasks or major system rewrites.
- Logic Segment extraction used for input to CASE tools.

The Target Generation environment is a powerful lever against change. When business needs call for enhancements to a specific function only one Logic Segment is impacted, dramatically reducing the time required for analysis and change.

Operational Requirements

This section describes the platforms on which Encore runs and the hardware and software requirements for running the application.

Environment

Encore products have these requirements:

Platform	Software
Host	• MVS/XA, MVS/ESA, or OS/390 • MVS ISPF Version 3.5 thru 4.8 (3.1 is the minimum supported release level, Estimate requires ISPF 3.3 (minimum supported release level)) • BATCH region size of 4096 KB or larger • TSO logon region size of 2048 KB or larger below 16 MB • Direct Access storage • 3270 type terminals; Models 2, 3, 4, or 5 • VSAM (required if any AKR is allocated as VSAM)
Clients	Windows 95/98/2000, Windows NT with access to MVS
Servers	OS/390

COBOL Compilers

ESW products support these COBOL compilers:

- COBOL D, E, and F
- COBOL II
- COBOL/370
- COBOL for MVS and VM
- COBOL for OS/390 and VM

Note:

Object Oriented extensions added to COBOL for MVS VM and COBOL for OS/390 and VM are not supported.

Preprocessor Support

Encore supports these preprocessor languages directly:

- Command-Level CICS
- Command-Level DL/I
- IDMS
- SQL

Note:

Other preprocessed languages are not recognized, but can be supported from the generated COBOL code.

2 Product Overview

This chapter discusses using Encore's online help, graphical user interface (GUI) elements, and migration functions, and contains these sections:

Section	Page
Introduction	7
Encore User Interface	8
Help Facility	15
Multiple Views	15
Wildcard Patterns	16
Understanding COBOL Terms	16

Introduction

Encore is an integrated re-engineering environment for COBOL programs. It is designed to isolate and extract logical COBOL segments from existing programs. The benefits of Encore enable you to:

- Extract the knowledge and investment made in COBOL systems over time.
- Manage each application by its business rule.
- Forward engineer new enhancements and systems quickly and with higher quality.

Encore operates interactively in ISPF, allowing you to identify, isolate, and extract logical business functions from COBOL programs. You can then save these business functions or business rules for later use (e.g., enhancement or code reusage).

Encore User Interface

The online component of Encore features Common User Access (CUA) screens, action bars, pull-down menus, and pop-up screens that are designed to provide easy access to all of the product features.

- An action bar contains a line of keywords, or actions, that display at the top of a screen. Each keyword represents a category of actions that can be performed on that screen using a pull-down menu.

  To select an action, follow this step:

  ▶ Move the cursor to the desired keyword and press Enter.

- A pull-down menu is the list that displays when an action is selected on the action bar. On a pull-down, actions followed by an ellipsis (...) display a pop-up when selected. Actions not followed by an ellipsis immediately activate internal commands.

  To select an item on a pull-down, follow this step:

  ▶ Move the cursor to the desired keyword and press Enter.

  Or

  Enter the number of the desired action in the input field and press Enter.

- Screens contain a full-width display of information containing a full action bar. Encore screens are modeled after TSO/ISPF screens.
- A pop-up screen is a screen that displays when an item is selected on a pull-down menu or another pop-up screen, or as the result of entering certain commands. It is superimposed on your screen to allow entry of information for the requested action. Enter the desired data or option and follow instructions on the pop-up to process the information.

To exit a pull-down or a pop-up without processing any actions, follow this step:

▶ Use the END command (default PF3 or PF15).

Note:

If the cursor is on the action bar when you press PF03/15, the cursor moves to the command input area of the screen.

The Action Bar

The action bar is contained on all Encore screens. Some pop-ups contain a shortened action bar that contains fewer actions and associated options than are available on the regular action bar.

Note:

If other ESW products are installed at your site, other actions may appear on the action bar. For more information about the features of other products, see the corresponding product User's Guide or online help.

To select an action, follow this step:

- Move the cursor with the tab, mouse, or arrow keys to the desired keyword and press Enter.

To move the cursor to the command line on the screen (Command = = =>), follow this step:

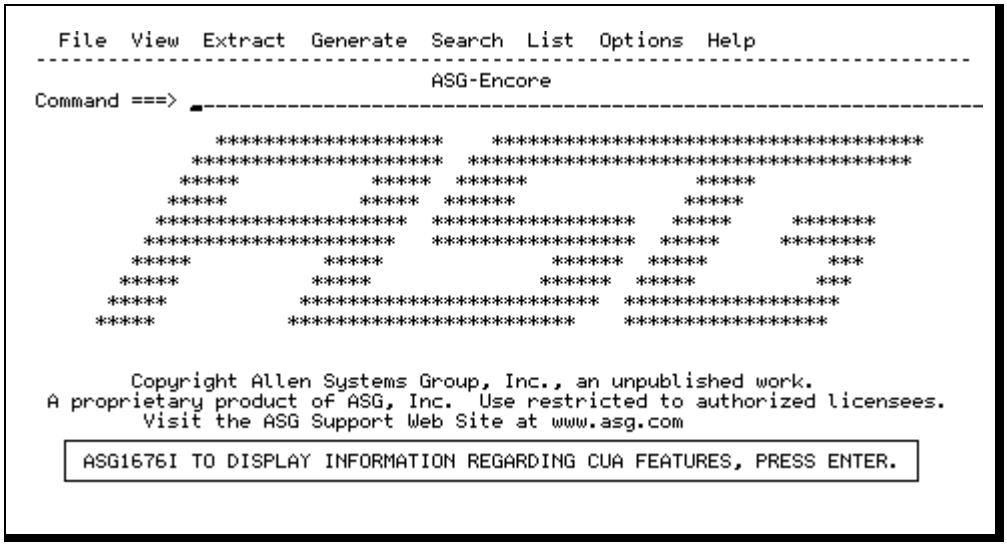
- Place the cursor on the action bar and press PF03/15 (END).

To move the cursor to the first action on the action bar when the cursor is on a full screen, follow this step:

- Press HOME.

Figure 6 shows the action bar on the Encore Primary screen.

Figure 6 • Action Bar Format (Primary Encore Screen)



Actions

Action	Description
File	Displays the File pull-down, used to open and close programs, save Logic Segments and Equates for the current program, analyze programs prior to use by Encore, manage the AKR, edit COBOL source programs, execute script files, fix AKR program anomalies, and to exit Encore.
View	Displays the View pull-down, used to view a program, Logic Segment, or complement, view program information, display or exclude source according to the program hierarchical structure; change the level of information viewed; reset the display; and exclude source from the display.
Extract	Displays the Extract pull-down, which presents various available task-oriented extract options.
Generate	Displays the Generate pull-down, used to create a program or callable module, build an IO module, perform a Batch Computation, generate a CICS Server, and to print or punch the Logic Segment.
Search	Displays the Search pull-down, used to conduct a COBOL intelligent search of the source code for one or all occurrences of a specified target.
Logic	Displays the Logic pull-down, used to follow the execution of a program, searching for a specific target. **Note:** The appearance of the Logic menu option on the action bar depends on whether you have also installed other ESW products, such as Insight. If other products have not been installed, Logic does not appear on the action bar.
List	Displays the List pull-down, used to access prompt list pop-ups that list information about calls, equates, performs, programs, subsets, and segments.
Options	Displays the Options pull-down, used to customize the Encore environment.
Help	Displays the Help pull-down, used to access the online help facility.

Pull-Down Menus

A pull-down menu displays in an existing window when you select an action on the action bar. These are the two types of actions on pull-downs:

- An action followed by an ellipsis (...) displays a pop-up.
- An action not followed by an ellipsis activates an internal command and usually redisplays the previous screen.

Pop-Up Screens

A pop-up screen is a window that displays when you select an item on a pull-down or pop-up, or when you enter certain commands (see Figure 7). A pop-up is superimposed over a screen, has a border, and usually does not have an action bar. Each pop-up contains one or more fields.

Follow the instructions on a pop-up to process the information.

To exit from a pop-up without processing the actions you have performed, follow this step:

▶ Type `END` (default PF03/15).

Figure 7 • Pop-Up Format (View - Tree View Request Pop-Up)

```
                        View - Tree View Request

Type request options.  Then press Enter.  For a name selection list
for a range (other than 1 or 2), type a pattern (e.g. ABC*) in the
name field.

Levels 2    of    4     of depth

Contents                                Range
1   1.  Labels                          1   1.  Procedure division
    2.  Labels and performs                 2.  All entry points
    3.  Labels, performs and GOTOs          3.  Perfrange name
    4.  Labels, performs, GOTOs and         4.  Entry point name
        conditionals                        5.  Nested program name

Name ____________________________________________________________

Options
_  Repetition
```

Note:

This pop-up displays after a valid program has been opened.

Fields

Field	Description
Text Entry Fields	Used to type textual information, such as a data name, program name, etc. The length of each text entry field is designated by an underline. Name is an example of a text entry field shown in Figure 7 on page 11.
Selection Fields	
Numbered Selection Fields	Allows selection of one item from a list. Type the number of the desired selection in the field input area to the left of the first numbered item. Contents is an example of a numbered selection field shown in Figure 7 on page 11.
Unnumbered Selection Fields	Selected by entering a forward slash (/) in the field input area, and unselected by entering a blank space to remove the slash. Repetition is an example of an unnumbered selection field shown in Figure 7 on page 11.
Field Input Area	A field input area is designated by an underline. When a pop-up first displays, each field contains its default value. On some pop-ups, if the default is changed the new value is retained for subsequent sessions.

Screens

A screen is a full-width display of information containing a full action bar. Encore screens are modeled after TSO/ISPF screens.

The various parts of the Source View screen are identified by the alphabetic letters shown in Figure 8.

Figure 8 • Screen Format (Source View Screen)

```
File  View  Extract  Generate  Search  List  Options  Help   (A)
-------------------------------------------------------------------------------
                                Source View  (B)        (C) Program: VIARDEMO
Command ===>        (D)                                  (E)  Scroll ===> CSR

****** **************************** TOP OF DATA ******************************
000001  IDENTIFICATION DIVISION.
000002  PROGRAM-ID. VIARDEMO.
000003  AUTHOR. VIASOFT DEMONSTRATION.
000004
000005 ******************************************************************
000006 *     THIS PROGRAM DOES THE MONTH END CALCULATIONS FOR EACH TYPE *
000007 *     OF CHECKING ACCOUNT CARRIED BY THE UNIVERSAL BANK.  IT     *
000008 *     HANDLES THE FOLLOWING TYPES OF ACCOUNTS.                   *
000009 *         CHECKING WITH INTEREST                                 *
000010 *         CHECKING WITH NO INTEREST                              *
000011 *         MONEY MARKET ACCOUNT (MMA)                             *
000012 *         SAVINGS ACCOUNT                                        *
000013 *     IT ALSO UPDATES THE ACCOUNT BALANCE, BRANCH BALANCE AND    *
000014 *     BANK BALANCE.                                              *
000015 ******************************************************************
000016  (F)
000017  ENVIRONMENT DIVISION.
000018  CONFIGURATION SECTION.
```

Fields

Field	Description
(A)	File View Extract - The action bar displays at the top of the screen.
(B)	Source View - The name of the screen.
(C)	Program: VIARDEMO - The name of the active program. A program must be open for this field to display. This area of the screen is also used to temporarily display short informational or error messages.
(D)	Command ===> - The command input area (command line), used to enter Encore primary commands.
(E)	SCROLL - Specifies the number of screen lines or columns to scroll. This field is omitted on screens that cannot be scrolled, such as the primary Encore screen. These are the scroll values: • 1-9999 A number from 1 to 9999 can be entered to specify the number of lines or columns to scroll. • CSR Specifies a scroll value of one page from the cursor location. • DATA Specifies a scroll value of one less than a page. • HALF Specifies a scroll value of one half page. • MAX Specifies a scroll value of the top, bottom, right or left margin. • PAGE Specifies a scroll value of one page.
(F)	The long message area contains descriptive, information, or error messages. Long messages display when there are no corresponding short messages, or when HELP is entered in the command input area while a short message displays.

See online help for more information about Encore commands.

Help Facility

Encore has extensive Help facilities. Help is available for screens and pop-ups, commands, general information, specific information, abends, messages, and fields on certain pop-ups.

The Help Tutorial is a topic-driven sequence of display-only screens that describe the features and capabilities of Encore. These descriptions include an overview of Encore and its main features.

Help for screens and pop-ups describes their purpose, why they display, a description of each field, and the required input.

Help for commands describes the purpose of each command, displays the command syntax diagram, and describes each operand and its values.

Help is also available for some input fields on certain pop-ups unique to Encore. Enter a question mark (?) in a field to view the help screen for that field. The field help screen describes the field and lists valid input values. You can enter the desired value on this help screen. When the product screen redisplays, the value you entered displays in the proper field.

Multiple Views

The View facility presents various displays of an existing COBOL source program. You can choose to have program information presented in a graphic structure, logical execution, or source code format. This facility is used to examine the program to become more familiar with the program structure and functions, or to increase understanding of the program prior to making a change or extracting a Logic Segment.

These are the available view methods:

- Structure view, which presents a graphical display of the PERFORMed structure of the program.
- Tree view, which presents the Perform Range hierarchy that shows a representation of the program in logical execution order.
- Source view, which presents the actual COBOL source code in source statement order.

Wildcard Patterns

On some Encore screens, there are fields that require you to enter names. If you leave these name fields blank and press Enter, a selection list displays. Encore allows you to enter wildcard patterns to reduce the selection list.

These are the wildcard pattern matching characters:

- A question mark (?) character matches any one character.
- An asterisk (*) character matches zero or more characters.

Performing a search using the wildcard format is only valid for those name fields that are explicitly noted.

Understanding COBOL Terms

Sets

A set is a grouping of source lines in a COBOL source program. Sets consist of subsets and line range sets. Many Encore commands make use of one or more of these sets or subsets.

You can concatenate sets by placing a plus sign (+) between them. For example, this set would highlight the set of lines containing IO statements and PARAGRAPH labels:

```
HIGH IO + PAR
```

Subsets

These are the three types of subsets:

- COBOL language subsets
- Screen subsets
- Tagged lines subsets

To get a description of a subset, follow this step:

▶ Use the LIST SUBSETS command.

Or

Select List ▶ Subsets from the action bar.

COBOL Subsets

Encore classifies COBOL statements into subsets by grouping together COBOL verbs of a similar nature. For example, you could refer to any lines that contain READ, WRITE, OPEN, or CLOSE verbs by using the COBOL language subset name IO.

The COBOL subsets and their corresponding entities are described in the Tagged Line Subsets section below.

Screen Subsets

Screen subsets are generally the result of an interactive search request.

To specify one of these subsets, follow this step:

- Type the entire name or the minimum abbreviation, as indicated by the upper case letters in this syntax:

```
HIghlighted = HI

NONHighlighted = NONH or NHI

EXcluded = EX or X

NONExcluded = NONE or NX
```

Tagged Line Subsets

Tagged line subsets display in columns 73-80 of the Source View screen. Search results show tags in Source View. Source View also marks data items that are never referenced, tags statements that contain dead code, tags program exits, and tells you if PERFORMed paragraphs fallthrough or return.

You can use this tagged line subset (which refers to all lines that have information tags on them) as a subset in commands that accept subsets as targets:

```
TAGged
```

This table contains COBOL subsets that accept tagged line subsets as targets.

COBOL Subsets	Description
ASSIGNMENT	Statements that assign a value to a data item (e.g., MOVE, ADD, or COMPUTE).
CALL	Statements related to subprogram calls such as CALL and CANCEL.
CICS	Any CICS (Customer Information Control System) or DL/I Command Level Commands.

COBOL Subsets	Description
COBOLII	COBOL II, including CONTINUE, END, and INITIALIZE verbs.
COBOL/370	Statements and clauses unique to COBOL/370, such as intrinsic function calls, procedure pointers, and calls to the LE/370 run-time environment.
COMMENT	Statements having no run-time effect, such as all lines with an asterisk (*) in column 7, the entire IDENTIFICATION DIVISION, and NOTE statements.
CONDITIONAL	Statements or parts of statements that conditionally change the flow of control in a program such as IF, ELSE, and WHEN.
COPY | INCLUDE	COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, and -INC statements.
DB2 | SQL	EXEC SQL statements.
DDL	SQL Data Definition Language statements, such as CREATE, ALTER, DECLARE, and DROP.
DEAD	Statements containing dead code and dead data.
DEADCODE	Statements containing code that cannot be executed under any conditions.
DEADDATA	DATA DIVISION statements containing data names and their aliases that are not referenced in the PROCEDURE DIVISION.
DEBUG	Statements containing a DEBUG, EXHIBIT, ON, READY, or RESET verb, as well as statements containing a D in column 7.
DEFINITION	Declaratives of data items including the SPECIAL-NAMES paragraph in the ENVIRONMENT DIVISION and the entire DATA DIVISION.
DIRECTIVE	Statements that direct the compiler to take specific actions during compilation such as BASIS, EJECT, and TITLE.
DL/I | DL/1	EXEC DL/I commands, ENTRY ’DLITCBL’, and CALL ’CBLTDLI’.
DML	SQL Data Manipulation Language statements, such as SELECT, UPDATE, INSERT, and COMMENT.
ENTRY	The PROCEDURE DIVISION statement and all ENTRY statements.

COBOL Subsets	Description
EXIT | PGMEXIT	Statements containing a STOP RUN, GOBACK, or EXIT PROGRAM verb, and CALL statements that are indicated as NORET (non-returning).
FALLTHROUGH	Statements of PERFORMed paragraphs or units that fall through to the next paragraph.
GOTO	Statements containing an ALTER or GOTO verb.
IDMS	IDMS statements.
INPUTOUTPUT IO INPUT OUTPUT	COBOL IO statements (IO, Input, or Output) including CALL statements that are indicated as containing IO, Input, or Output.
LABEL DIVISION PARAGRAPH SECTION	Statements containing DIVISION or SECTION headers or PARAGRAPH labels. LABEL refers to the PROCEDURE DIVISION line and all section and paragraph names in the PROCEDURE DIVISION.
MAINLINE	Mainline code statements that are reachable from the PROCEDURE DIVISION line to the program units by following FALLTHROUGHs and GO TOs, but not PERFORMs.
MATH	Statements containing arithmetic operators and verbs.
PERFORM	Statements containing the PERFORM, SORT, or MERGE verbs.
RETURN	Statements of a PERFORMed paragraph ranges that return control.
SORTMerge	SORT/MERGE statements and related IO procedures.
STRUCTURE	A group of COBOL subsets that, together, help show the general structure of the program. These COBOL subsets include CALL, PERFORM, DIVISION, SECTION, PARAGRAPH, EXIT, and GO TO.
TESTED	Identifies the lines of code that have been tested based on information created and updated with TCA reports. This is only applicable to SmartTest and Insight program views.
UNTESTED	Identifies the lines of code that have not been tested based on information created and updated with TCA reports. This is only applicable to SmartTest and Insight program views.

Targets

A target is the object of an Encore primary command. Targets are defined in these categories:

- Label name
- Perform range name
- Program name
- Subset name
- Line range
- Paragraph name
- Data name
- Pattern string

Label Name

A label name is any paragraph or section name of the PROCEDURE DIVISION, as well as the literals PROCEDURE and PROC. Label name specifies all transfers of control to a paragraph or section.

Perform Range Name

A Perform Range name is the name specified in a PERFORM statement. The Perform Range consists of the source code contained in the PERFORM statement, and includes all code that is or could be executed as a result of GO TOs, PERFORMs, etc., within that PERFORM. You can also specify the name of any section contained in the DECLARATIVES.

Program Name

A program name is the name of the main program or any nested program, and includes all the source code contained in the program. This includes all programs physically nested inside the specified program.

Subset Name

A subset name is one of the COBOL language subsets, screen subsets, or tag subsets previously described in "Screen Subsets" on page 17 .

Line Range

A line range can be a single line or a group of lines.

To specify line ranges, follow this step:

▶ Place a hyphen (-) between the first and last line numbers in the range (e.g., 214 - 376).

Line numbers display in the first six columns of the Source View screen. If the specified line number is greater than the last line in the program, the last line is assumed.

Paragraph Name

A paragraph name is any paragraph or section name of the PROCEDURE DIVISION, as well as the literals PROCEDURE and PROC. Paragraph name includes the entire paragraph or section.

Data Name

A data name can be any of these:

- Elementary data name
- Table name
- File name
- Table element name
- Group name
- Special name

You can specify any legal COBOL reference for a data element as a data name. If you redefine a variable to another name, Encore searches for the specified variable name and the redefined name. Any reference to an entry in a table is treated as a reference to the entire table.

When data items overlap so a name can refer to parts of multiple data items, searches are performed on each part and all references are reported. For example, if a group item is specified in a search command, references to the group item as well as the individual elements within the group are located. This is also true of modifications, uses, or references to the data item. Encore locates valid references to the variable item as opposed to simple pattern matching of the characters in the variable name. These references are called aliases.

To specify fully qualified data names follow this step:

- Follow them with a standard COBOL OF clause, then by the group level data name. For example:

```
DATA-NAME-ELEMENT OF DATA-NAME-GROUP
```

or for COBOL II Release 3 (minimum supported release level) programs:

```
DATA-NAME-ELEMENT OF SUBPROGRAM
```

To locate multiple data names at the same time, follow this step:

- Concatenate the data names with a plus sign (+) between them. For example:

```
DATA-NAME1 + DATA-NAME2
```

Data names can be specified with one of these subordinate operands:

Operand	Description
MODIFICATION	Occurrences of a data item where its value is being set or altered.
USE	Occurrences of a data item where its value is being tested or used.
DEFINITION	Definitions of a data item and its aliases as specified in the DATA DIVISION.
REFERENCE	All MODIFICATION and USE occurrences. REFERENCE also includes DEFINITION occurrences on some commands. This is the default usage for data names.

Note:

The FINDXTND primary command also offers ALIAS/NOALIAS operands. ALIAS includes all aliases for the specified data name and is the default.

Pattern String

A pattern string is a sequence of characters. Strings of non-alphanumeric characters can be specified by the HEX, TEXT, and PICTURE operands. The string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands.

String	Description
X'string'	A hexadecimal string, enclosed in single or double quotes.
T'string'	A text string, which disregards upper and lowercase, enclosed in single or double quotes.
P'string'	A picture string, enclosed in single or double quotes. These are the valid picture strings: • P'=' Any character • P'¬' Any nonblank character • P'.' Any nondisplay character • P'#' Any numeric character • P'- ' Any non-numeric character • P'@' Any alphabetic character (upper or lowercase) • P'<' Any lowercase alphabetic character • P'>' Any uppercase alphabetic character • P'$' Any special character (not alphabetic or numeric)
WORD	A string preceded and followed by any non-alphanumeric character (except a hyphen).
PREFIX	A word that begins with the specified string.
SUFFIX	A word that ends with the specified string.

3 File Menu

This chapter describes Encore's File pull-down menu functions and contains these sections:

Section	Page
File Pull-down	26
File - Open Program Pop-up	28
File - Program Name List Pop-up	30
File - Logic Segment Directory Pop-up	33
Logic Segment Description Pop-up	34
File - Save Logic Segments Pop-up	35
File - Save Segment Pop-up	37
File - Analyze Submit Pop-up	39
File - AKR Utility Pop-up	43
File - AKR Directory Pop-up	45
File - AKR Allocate/Expand Pop-up	49
Edit Interface	52
File - Execute Script Pop-up	59
Anomaly Facility	61

File Pull-down

The File pull-down, shown in Figure 9, is used to complete these tasks:

- Open and close programs
- Save Logic Segments and Equates for the active program
- Analyze programs prior to use by Encore
- Manage the AKR
- Edit COBOL source programs
- Execute script files
- Invoke the Anomaly facility
- Exit Encore.

Figure 9 • File Pull-Down

```
File  View  Extract  Generate  Search  List  Options  Help
                                          ---------------------------------------------
 _ 1. Open...                                ASG-Encore
   2. Close                               -----------------------------------------------
   3. Save segment...
   4. Save equates                        ****    ************************************
   5. Analyze...                          ****  ************************************
   6. AKR utility...                      ***  ******               *****
   7. Edit program...                     **  ******               *****
   8. Execute script...                   *  *****************   *****     *******
   9. Anomaly facility...                    *****************  *****     ********
  10. Exit                                              ******  *****          ***
                                                       ******  *****          ***
     *****           *************************  ******************
    *****           *************************    *****************

       Copyright Allen Systems Group, Inc., an unpublished work.
A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

To display the File pull-down, follow this step:

▶ Select File on the action bar and press Enter.

Actions

Action	Description
1. Open	Displays the File - Open Program pop-up (see Figure 10 on page 28) used to specify the AKR and program name to open for use.
2. Close	Closes the active program, but does not exit Encore. The active Logic Segment is closed, but not saved. The options set on the Options - Product Parameters pop-up (see the Figure 141 on page 296) are used to determine whether Equates are saved when the program is closed.
3. Save segment	Displays the File - Save Segment pop-up (see Figure 15 on page 37) used to specify a name and description for the Logic Segment being saved.
4. Save equates	Displays the saves equates for the active program in the AKR. Equates can be entered on the Command line of the Encore Primary screen using the EQ command, and also by using the Product Parameters option (see the "Options - Product Parameters Pop-up" on page 296) and the Equates option (see the "Options - Scratchpad Equate Pop-up" on page 310) from the Options pull-down (see the Figure 140 on page 294). You can see existing equates by using the Equates option (see the "List - Equates Pop-up" on page 285) on the List pull-down (see the Figure 131 on page 282).
5. Analyze	Displays the File - Analyze Submit pop-up (see Figure 16 on page 39) used to specify the program to be analyzed, and the AKR to store the analyze information.
6. AKR utility	Displays the File - AKR Utility pop-up (see Figure 17 on page 43) used to display the program directory, allocate or expand an AKR, rename a program, and delete a program.
7. Edit program	Displays the File - Edit Options Request pop-up (see Figure 21 on page 52) used to specify whether or not the Options - COPY/Include Libraries pop-up (see Figure 22 on page 53) should display.
8. Execute script	Displays the File - Execute Script pop-up (see Figure 27 on page 59) used to specify the dataset name and script option.

Action	Description
9. Anomaly Facility	Displays the File Anomaly Facility pop-up (see Figure 28 on page 61) used to select the Anomaly Facility - Batch Submit screen (see Figure 29 on page 62) or exit back to the Encore Primary screen.
10. Exit	Closes the active program and Logic Segment, and returns to the screen that was used to enter Encore. All Logic Segments created in the current session are discarded if they were not previously saved.

File - Open Program Pop-up

The File - Open Program pop-up, shown in Figure 10, is used to specify the AKR and program name to open for use.

Figure 10 • File - Open Program Pop-up

```
                              File - Open Program

      Type AKR information and program name. Then press Enter.


      Application Knowledge Repository (AKR):

       Data Set Name . . . 'USER.TEST.AKR'
       Program Name  . . . VIARDEMO     (blank for selection list)

       Volume Serial . . . ______       (if not cataloged)
       Password  . . . . . ________     (if password protected)
```

To display the File - Open Program pop-up, follow this step:

- Select File ▸ Open and press Enter.

Fields

Field	Description
Data Set Name	Required. Enter the dataset name of the AKR where the program resides.
Program Name	Optional. Enter the name of the program if it was not entered in the Data Set Name field. If the program name is not entered, the File - Program Name List pop-up (Figure 11 on page 30) displays so you can select the program name. If $$SEGMENTS is entered in this field, the File - Logic Segment Directory pop-up (see Figure 13 on page 34) displays. Wildcard characters are allowed in this field. The question mark (?) matches any one character. The asterisk (*) matches zero or more characters.
Volume Serial	Required if the dataset is not cataloged. Enter the volume serial number.
Password	Required if the dataset is protected. Enter the dataset password.

The AKR dataset name must already exist. An error message is produced if the dataset name is invalid or does not exist. The program must have been previously analyzed and placed in the specified AKR. The AKR and program name appearing on this screen are retained from the previous session. If the AKR and program name are correct for the current session, press Enter. Otherwise, enter a valid AKR and program name.

If a program name is not entered, the File - Program Name List pop-up (see Figure 11 on page 30) displays to allow selection of a valid program. Wildcard characters asterisk (*) and question mark (?) can also be entered in the Program Name field to reduce the size of the list displayed on the File - Program Name List pop-up. A question mark (?) character matches any one character and an asterisk (*) character matches zero or more characters.

If $$SEGMENTS is entered as the program name, the File - Logic Segment Directory pop-up (see Figure 12 on page 33) displays, which lists all the Logic Segments defined in all programs in the specified AKR.

File - Program Name List Pop-up

The File - Program Name List pop-up, shown in Figure 11, is used to select a program name.

Figure 11 • File - Program Name List Pop-up

```
                              File - Program Name List
Command ===> ____________________________________________ Scroll ===> CSR

Select desired Program Name. Then press Enter.

  AKR: USER.TEST.AKR

  Name        Alias of    Type      Lines   Date         Time      Jobname
- ----------  ----------  --------  ------  -----------  --------  --------
_ $$SEGMENTS              INTERNAL       0  DD-MMM-YYYY  HH:MM:SS  USERJOB
_ VIARDEMO                EN           434  DD-MMM-YYYY  HH:MM:SS  USERJOB2
******************************* BOTTOM OF DATA ******************************
```

To display the File - Program Name List pop-up, follow this step:

- Leave program name on the File - Open Program pop-up blank and press Enter.

  Or

  Enter a wildcard specification for the program name and press Enter.

This pop-up displays a list of all programs that exist on the specified AKR in alphabetical order, filtered by any wildcard characters. You can scroll the list up and down, or use the LOCATE primary command or the slash (/) line command to locate a particular program. Once the desired program is located, you can use the SELECT primary command or S line command to select the program.

Fields

Field	Description
AKR	The complete dataset name of the AKR specified on the File - Open Program pop-up (see Figure 10 on page 28).
Line command area	The line command area, to the left of the Name field, accepts commands for programs with a Type of EN, RN or $$SEGMENT. For programs with Types other than EN, RN or $$SEGMENT, the line command area is protected.
S	Selects the specified program.
/	Locates the specified program. The screen is scrolled until the specified program is at the top of the display.
Name	The name of the program in the AKR from the PROGRAM-ID statement. If the $$SEGMENTS program is selected, the File - Logic Segment Directory pop-up displays (see Figure 12 on page 33). • If the analyzed program contains an ENTRY point, Name is the ENTRY point name. • If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Name is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39).
Alias of	• If the analyzed program contains an ENTRY point, Alias of is the name of the program that contains the ENTRY point. • If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39).

Field	Description
Type	Indicates the type(s) of analysis that was performed on the program. The type of analysis is specified on the File - Analyze Submit pop-up (see Figure 16 on page 39) at the time the Analyze job is submitted. Only programs with the $$SEGMENT, EN or RN analysis type can be selected for use by Encore. These are the valid analysis types: • EN, R, RN - Indicates a Encore analysis was performed. • I, IN - Indicates a Insight analysis was performed. • SQ - Indicates a SmartQuest analysis was performed. • ST - Indicates a SmartTest analysis was performed. • STX - Indicates that an Extended SmartTest analysis is to be performed. When the INS and ST parameters are specified, an Extended SmartTest analysis is automatically performed. • DS - Indicates a SmartDoc analysis was performed. • DC - Indicates a SmartDoc analysis with a COBOL compile was performed. • DX - Indicates an Extended SmartDoc analysis was performed. • DA - Indicates an Extended SmartDoc analysis with a COBOL compile was performed. • RC - Indicates a Recap analysis was performed. • ASM - Indicates the program is an Assembler source program. • PROFILE - Indicates that the member contains SmartTest profile information. • INTERNAL - Indicates that the member $$SEGMENTS contains Logic Segment criteria information. If this member is selected, the File - Logic Segment Directory pop-up displays (see Figure 12 on page 33). • METRICS - Indicates that the member contains SmartDoc metric information. This member cannot be selected in Encore.
Lines	The number of lines contained in the program.
Date	The date the program was analyzed.
Time	The time the program was analyzed.
Jobname	The job name that analyzed the program.

File - Logic Segment Directory Pop-up

The File - Logic Segment Directory pop-up, shown in Figure 12, is used to view the names of all the Logic Segments in the specified AKR. The Logic Segments are listed in alphabetical order by program name and then by Logic Segment name.

Figure 12 • File - Logic Segment Directory Pop-up

```
                        File - Logic Segment Directory
Command ===> ________________________________________ Scroll ===> CSR

  D - Delete Logic Segment     V - View Segment Description

  Program    Logic Segment Name                Author   Description
- ---------- ------------------------------- -------- --------------------
_ VIARDEMO   GET-NUM-OF-DAYS                  VIASHG   CALCULATE NUMBER OF
  VIARDEMO   INTEREST-CALCULATION             VIASHG   INTEREST-CALCULATION
*************************** BOTTOM OF DATA ***************************
```

To display the File - Logic Segment Directory pop-up, follow this step:

- Specify the $$SEGMENTS program name on the File - Open Program pop-up (see Figure 10 on page 28) or the File - Program Name List pop-up (see Figure 11 on page 30) and press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the Program field, accepts these line commands: • D - Delete the specified Logic Segment definition. • V - Display the Logic Segment Description pop-up (see Figure 13 on page 34) to view information pertaining to the Logic Segment.
Program	The name of the program containing the Logic Segment definition.
Logic Segment Name	The name of the Logic Segment you entered or modified.

Field	Description
Author	The user ID of the person who created the Logic Segment.
Description	The short description of the Logic Segment entered on the File - Save Segment pop-up (see Figure 15 on page 37), the Copy Logic Segment pop-up, or the Rename Logic Segment pop-up. (See Chapter 8, "List Menu," on page 281 for more information about the Copy Logic Segment and Rename Logic Segment pop-ups.) The description can be truncated on the screen due to column limitations.

Logic Segment Description Pop-up

The Logic Segment Description pop-up, shown in Figure 13, is used to view the description of a Logic Segment that has been saved in the AKR.

Figure 13 • Logic Segment Description Pop-up

```
                          Logic Segment Description
 Command ===> ____________________________________________ Scroll ===> CSR

 Objective  . . . . : Perform Range
 Author . . . . . . : USERID
 Created  . . . . . : DD-MMM-YYYY HH:MM:SS

 Name  . . . GET-NUM-OF-DAYS
 Description CALCULATE NUMBER OF DAYS

 Long Description:
THIS SEGMENT CONTAINS CODE TO DETERMINE THE NUMBER OF DAYS IN THE INTEREST
PERIOD. IT WILL BE USED TO GENERATE A CALLABLE SUB-MODULE.
 ****************************** BOTTOM OF DATA ****************************
```

To display the Logic Segment Description pop-up, follow this step:

- Enter the V line command on the File - Logic Segment Directory pop-up (see Figure 12 on page 33) or the List - Segments pop-up (see to Chapter 8, "List Menu," on page 281) and press Enter.

Fields

Field	Description
Objective	The objective type that was used to create the Logic Segment.
Author	TSO USERID that created the Logic Segment.
Created	Date and time the Logic Segment was created. If the Logic Segment was copied from another definition, the date is the COPY date.
Name	The name of the Logic Segment, entered or modified on the File - Save Segment (see Figure 15 on page 37), Copy Logic Segment, or Rename Logic Segment pop-up. (See Chapter 8, "List Menu," on page 281 for more information about the Copy Logic Segment and Rename Logic Segment pop-ups.)
Description	The short description of the Logic Segment, entered or modified on either the File - Save Segment (see Figure 15 on page 37), Copy Logic Segment (see Figure 138 on page 291), or Rename Logic Segment pop-up (see Figure 139 on page 292). (See Chapter 8, "List Menu," on page 281 for more information about the Copy Logic Segment and Rename Logic Segment pop-ups.)
Long Description	The long description text of the Logic Segment, entered or modified on the File - Save Segment pop-up (see Figure 15 on page 37).

File - Save Logic Segments Pop-up

The File - Save Logic Segments pop-up, shown in Figure 14, is used to select the desired Logic Segment to save. This pop-up lists all Logic Segments extracted in the current session but not saved in the AKR.

Figure 14 • File - Save Logic Segments Pop-up

```
                          File - Save Logic Segments       1 LOGIC SEGMENTS
Command ===> _                                            Scroll ===> CSR

Select segment to be Saved or marked for Batch computation and press Enter.
When selections are complete, press PF3/15(END).

   S - Save      B - Batch computation

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
s  VIARDEMO   PERFORM-RANGE-0001             Perform Range
****************************** BOTTOM OF DATA ******************************
```

To display the File - Save Logic Segments pop-up, follow this step:

▶ Select File ▶ Save segment and press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts these line commands: • S - Save the Logic Segment. • B - Submit the Logic Segment for generation using a batch job. After the command is entered, the File - Save Segment pop-up displays (see Figure 15 on page 37).
Program	The name of the program containing the Logic Segment definition.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up. (See Chapter 5, "Extract Menu," on page 113 for more information.)
Description	The short description of the extract that was used to create the Logic Segment.

Note:

If you exit the program without first saving the Logic Segment, the Exit pop-up displays giving you a final option to save the Logic Segment by entering `1` (Save segments) on the Options line. This displays the Close - Unsaved Logic Segments pop-up, which allows you to save the member.

File - Save Segment Pop-up

The File - Save Segment pop-up, shown in Figure 15, is used to save a Logic Segment or change the description of a previously-saved Logic Segment.

To display the File - Save Segment pop-up, follow this step:

- Press Enter on the File - Save Logic Segment pop-up.

Figure 15 • File - Save Segment Pop-up

```
                              File - Save Segment
Command ===> ____________________________________________ Scroll ===> CSR

Type segment definition information. Then press Enter.

Objective  . . . . : Perform Range
Author . . . . . . : USERID
Created  . . . . . : DD-MMM-YYYY HH:MM:SS

Name  . . . PERFORM-RANGE-0001
Description ____________________________________________________

Long Description:
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
________________________________________________________________
```

For new Logic Segments, the initial display contains the author (from the TSO USERID) and current date and time already filled in. If the current Logic Segment was previously saved, the File - Save Segment pop-up can be used to change the short or long description.

Note:

If the segment was selected for batch generation, the Logic Segment - Batch Submit pop-up (see Figure 112 on page 239) displays when you press Enter on the File - Save Segment pop-up. For additional information about submitting a segment for batch generation, see Chapter 6, "Generate Menu," on page 203

Fields

Field	Description
Objective	The objective type that was used to create the Logic Segment.
Author	TSO USERID that created the Logic Segment.
Created	Date and time that the Logic Segment was created. If the Logic Segment was copied from another definition, the date is the COPY date.
Name	Required if the current Logic Segment was not saved previously. Enter a unique name of up to 32 characters to identify the extract criteria. The name must be unique within the program and cannot contain embedded blanks. **Note:** This field automatically displays the name entered during the extract or an earlier extract name that has been changed.
Description	Optional. Enter a short description of up to 60 characters.
Long Description	Optional. Enter a description of the Logic Segment. The long description can be any number of characters. As Enter is pressed, additional lines for text entry become available.

Usage Notes

The bottom portion of the pop-up is available for entering freeform text. Text is entered as 75-character lines. As text is entered, and Enter is pressed, the last entered line is repositioned to the top of the text entry area, to allow the maximum amount of text to be entered. UP and DOWN scrolling commands are available to review or change previously-entered text.

If the Logic Segment has been previously saved, changes to the short and long descriptions made on this pop-up are made to the previously-saved Logic Segment description. The Logic Segment name and short description can be changed on the Copy Logic Segment pop-up (Figure 138 on page 291) or Rename Logic Segment pop-up (Figure 139 on page 292).

The Logic Segment description can be printed on the List - Segments pop-up (see Figure 137 on page 289).

See Chapter 8, "List Menu," on page 281 for information about the List - Segments pop-up, the Copy Logic Segment pop-up, and the Rename Logic Segment pop-up.

File - Analyze Submit Pop-up

The File - Analyze Submit pop-up, shown in Figure 16, is used to analyze a program and place it in the AKR for use by Encore.

Figure 16 • File - Analyze Submit Pop-up

```
                          File - Analyze Submit
Command ===> _____________________________________________________________

           E - Edit JCL                              S - Submit JCL

Compile and link JCL (PDS or sequential):
   Data set name  'USER.TEST.CNTL(MEMBER)'

Analyze features (Y/N):
   ASG-Encore:  Y

AKR data set name 'USER.TEST.AKR'
AKR program name  __________          (if overriding PROGRAM-ID)

Analyze options:
   ____________________________________________________________________
   ____________________________________________________________________
   ____________________________________________________________________

Compile? (Y/N) . . . . . . . . . N     (Y if needed by features)
Link load module reusable? (Y/N) N     (ASG-SmartTest only)
```

To display the File - Analyze Submit pop-up, follow this step:

- Select File ▸ Analyze (see Figure 9 on page 26) and press Enter.

  Or

  Enter the ANALYZE command on any screen and press Enter.

Programs can also be analyzed using other methods. See Chapter 12, "Analyze Functions," on page 431 for more information.

Note:

If you started your Encore session using the ESW Primary screen (see Chapter 1, "Introduction," on page 1), the product names listed under the Analyze features (Y/N) field may differ from the names shown in this example.

Options

Field	Description
E - Edit JCL	Select option E to review or change the compile/analyze JCL, if necessary. When the E option is selected, the JCL to be edited is generated from the JCL member specified in the Data set name field, applying the rules outlined in Chapter 12, "Analyze Functions," on page 431. Then, the generated JCL displays on the Edit screen (see Figure 24 on page 56). When editing is complete, the ISPF SUBMIT command must be entered to submit the edited JCL for execution. Optionally, the edited JCL can be saved in a partitioned dataset by using the CREATE command. Otherwise, any changes made at this time are not saved.
S - Submit JCL	Select option S to submit the JCL to compile/analyze the specified program. The JCL submitted is generated from the JCL member specified in the Data Set Name field, applying the rules outlined in Chapter 12, "Analyze Functions," on page 431.

Fields

Field	Description
Data set name	Required. Enter the PDS member or sequential dataset containing the JCL to compile and link the program. If the JCL resides in a source manager such as Librarian or Panvalet, use the VIASUB edit macro to submit the compile/analyze job.

Analyze Features

Field	Description
Encore	Optional. Yes specifies that an Encore compile/analysis is to be performed. This type of analysis provides the logic and program execution flow capabilities of Encore. The default is Y.
AKR dataset name	Required. Enter the Application Knowledge Repository (AKR) that is to contain the information for the analyzed program.

Field	Description
AKR program name	Optional. Enter an alias name to be used by the analyze process to save its results in the AKR. If a value is not entered in this field, the analyze job uses the program name from the PROGRAM-ID statement in the COBOL source as the name under which to save results in the AKR. If an AKR program name is entered, the analyzed program is saved in the AKR as the entered name, and also as an alias of the PROGRAM-ID. If a program contains ENTRY points, the analyze job also saves a member in the AKR for each ENTRY point, with an alias of the PROGRAM-ID. **Note:** This field is only used for the AKR program name and does not change the COBOL program name in the source.
Analyze options	Optional. Enter analyze options that are to be overridden. Default options for the analyze job are established at installation time. You can enter report parameters used to produce analysis reports that help you understand and diagnose analysis problems and anomalies. You can also enter additional analyze parameters to utilize the Anomaly Repair Facility to correct certain anomalies that may exist within your code.
Analyze options - Analysis/Anomaly Report Parameters	• VIARERPT - Specifies that the analysis process stops after an analysis report is generated. • VIARERPTAN - Specifies that the analysis process continues after an analysis report is generated and the information is saved in the AKR. • NUMPRM(##) - Determines the number of parameters you allow before a CALL is listed on the Excessive Parameter List report. • RPTCEN - Centers the generated report (the default format for all reports is left justified).

Field	Description
Analyze options - Anomaly Repair Facility Parameters	VIAANOMF - Instructs the analyzer to run the Anomaly Repair Facility. This is the batch method for anomaly repair. You can also select the online method by using the Anomaly Facility - Batch Submit screen (see "Anomaly Facility - Batch Submit Pop-up" on page 62) accessed from the File pull-down menu. This parameter can be followed by these additional anomaly report parameters, described below and shown in the associated example: • PARM(user parm) - Any valid user parameter (e.g., DEBUG). • ANOMALY - Required for anomaly fixes and is used to specify these type of anomalies: — OOPGT - for out of perform GOTO — GTD - for GOTO depending — IPR - for inconsistent performance ranges • COMMENT(YES or NO) - Indicates whether or not you want to include the original commented out code in the generated output that includes the new repaired code. The default is YES. • DSN - Indicates the name of the output dataset where the repaired code should be written. If this parameter is not entered, the output is written to DD VIAANOUT. If DSN is a sequential file type, do not code the MEM parameter. • MEM - Indicates the member name of the output dataset if DSN is specified. Do not code this parameter if DSN is a sequential file. • HEADER(YES or NO) - Indicates whether or not you want an ASG generated header at the beginning of the repaired code. The default is YES. For example: `VIAANOMF,PARM(DEBUG1),ANOMALY(OOPGT,GTD,IPR),COMMENT(YES),DSN('ASG.ANOMALY.REPAIRED.CODE'),MEM(MEM1),HEADER(YES)` The reports that can be generated from these parameters are described in the online help and in “File - Analyze Submit Screen” on page 439 in Chapter 12, "Analyze Functions" on page 431. See "Anomaly Facility" on page 61 for more information about correcting specific anomaly types.

Field	Description
Compile?	Optional. A program need not be compiled if Insight, Encore, or SmartDoc are the only features specified. The compile step can be suppressed by entering N in this field. This field is forced to a value of Y if SmartTest is selected.
Link load module reusable	Optional. This field is used to test a program under SmartTest that is dynamically loaded and is tested with RUN MONITOR. It is necessary to mark the load module as reusable so that the Breakpoints are retained across CALLs. The default is Y.

File - AKR Utility Pop-up

The File - AKR Utility pop-up, shown in Figure 17, is used to display the program directory, allocate or expand an AKR, rename a program, and to delete a program.

Figure 17 • File - AKR Utility Pop-up

```
                          File - AKR Utility
Command ===> ____________________________________________________________

       Blank - Display member list       D - Delete member
       A     - Allocate/expand AKR       R - Rename member

Application Knowledge Repository (AKR):

   Data set name . . 'USER.TEST.AKR'
   Member  . . . . . __________  (if "R" or "D" selected)
   New name  . . . . __________  (if "R" selected)

   Volume serial . . ______      (if not cataloged)
   Password  . . . .             (if password protected)
```

To display the File - AKR Utility pop-up, follow this step:

- Select File ▸ AKR utility (see Figure 9 on page 26) and press Enter.

Options

Field	Description
Blank - Display member list	Displays the File - AKR Directory pop-up (see Figure 18 on page 45) that lists the program directory. Programs can also be deleted or renamed on the File - AKR Directory pop-up.
A - Allocate / expand AKR	Select option A to allocate a new AKR or to expand an existing AKR. Enter the AKR name in the Data set name field. Then, the File - AKR Allocate/Expand pop-up displays (see Figure 19 on page 49). If expanding an existing AKR, enter YES in the Expand existing AKR field on that pop-up.
D - Delete member	Select option D to delete a program. Prior to selecting option D, enter the AKR name in the Data set name field, and the program name in the Program field. Programs can also be deleted on the File - AKR Directory pop-up (see Figure 18 on page 45).
R - Rename member	Select option R to rename a program. Prior to selecting option R, enter the AKR name in the Data set name field, the program name in the Program field, and the new name in the New name field. Programs can also be renamed on the File - AKR Directory pop-up (see Figure 18 on page 45).

Fields

Field	Description
Data set name	Required. Enter the name of the AKR directory. If the TSO ID qualifier is the same as the user ID, enter the group, type, and program without quotes. If the TSO ID qualifier is different than the user ID, enter the project, group, type, and program within quotes. Alternately, the program name can be entered in the Program field.
Member	Required when deleting or renaming a program; otherwise, the program name can be entered in the Program field or in the Data set name field. Enter the name of the program in the AKR.
New name	Required when renaming a program. Enter the new name of the program. The name must be 1-10 alphanumeric characters.
Volume serial	Required if the dataset specified in the Data set name field is not cataloged. Enter the volume serial number. This field is optional if the dataset is cataloged.
Password	Required if the dataset is protected. Enter the dataset password.

File - AKR Directory Pop-up

The File - AKR Directory pop-up, shown in Figure 18, lists all members in the specified AKR.

Figure 18 • File - AKR Directory Pop-up

```
                              File - AKR Directory        Program:
Command ===> ____________________________________________ Scroll ===> CSR

AKR: USER.TEST.AKR                                              Row:      1
                                 Total members:      5   Total entries:      5
D - Delete   R - Rename          Total records:   1500      Free space:  87.5%

     Name      New name   Alias of    Type       Date     Time  Jobname  Space
_ ---------- ---------- ---------- --------- --------- ----- -------- -----
_ $$SEGMENTS __________             INTERNAL  15JUN2000 11:35 USERID    0.1%
_ VIADDDMO   __________             DC        09OCT2000 10:47 USERIDX   5.7%
_ VIARDEMO   __________             EN        19OCT2000 10:05 USERID8   6.3%
****************************** BOTTOM OF DATA ******************************
```

To display the File - AKR Directory pop-up, follow this step:

- Press Enter on the File - AKR Utility pop-up (see Figure 17 on page 43) with no entry in the command input area.

This pop-up can be scrolled to view members that are not visible or the L (Locate) command can be entered with a character string to view a specific program. The pop-up is scrolled to the program that most closely matches the character string.

Options

Field	Description
D - Delete	Type D to the left of each member to be deleted from the AKR. Alias programs cannot be deleted. Members can also be deleted on the File - AKR Utility pop-up (see Figure 17 on page 43).
R - Rename	Type R to the left of the member to be renamed, and enter the new name in the New name field. Note that alias members cannot be renamed. When a primary member name is changed, the alias name is automatically changed. Members can also be renamed on the File - AKR Utility pop-up (see Figure 17 on page 43).

Fields

Field	Description
AKR	The complete dataset name of the requested AKR entered on the File - AKR Utility pop-up (see Figure 17 on page 43).
Row	The relative number in the AKR of the first member displayed on this pop-up.
Total members	Indicates the total number of members in this AKR, excluding aliases.
Total entries	The total number of members in this AKR, including aliases.
Total records	The number of records allocated to this AKR, entered in the Space Amount field on the File - AKR Allocate/Expand pop-up (see Figure 19 on page 49) when the AKR was allocated or expanded. If the Space Units was entered as Tracks or Cylinders instead of Records on that pop-up, the amount is converted to Records for display in the total records field.
Free space	Indicates the amount of available space in this AKR, rounded to the nearest (.1) percent.
Line command area	The line command area, to the left of the Name field, accepts these line commands: • D - Delete - Type `D` to the left of each program to be deleted from the AKR. Alias programs cannot be deleted. Programs can also be deleted on the File - AKR Utility pop-up (see Figure 17 on page 43). • R - Rename - Type `R` to the left of the program to be renamed, and enter the new name in the New name field. Note that the alias programs cannot be renamed. When a primary program name is changed, the alias name is automatically changed. Programs can also be renamed on the File - AKR Utility pop-up (see Figure 17 on page 43).

Field	Description
Name	The names of the members that reside in this AKR. Program names are taken from the PROGRAM-ID statement. If the analyzed program contains an ENTRY point, Name is the ENTRY point name. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Name is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39).
New name	Required when renaming a member. Enter the new member name. The member name must be 1 - 10 alphanumeric characters.
Alias of	If an analyzed program contains an ENTRY point, Alias of is the name of the program that contains the ENTRY point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39). An alias program cannot be deleted or renamed. The alias name is automatically changed when the primary program is deleted or renamed. **Note:** See the File - Analyze Submit pop-up (Figure 16 on page 39) for more information about alias names in the AKR.

Field	Description
Type	Indicates the type of analysis that was performed on the member. These are the types of analysis specified on the File - Analyze Submit pop-up (see Figure 16 on page 39): • AL - Indicates a Alliance analysis was performed. • SQ - Indicates a SmartQuest analysis was performed. • ST - Indicates a SmartTest analysis was performed. • I, IN - Indicates a Insight analysis was performed. • DS - Indicates a SmartDoc analysis was performed. • DC - Indicates a SmartDoc analysis with a COBOL compile was performed. • DX - Indicates an Extended SmartDoc analysis was performed. • DA - Indicates an Extended SmartDoc analysis with a COBOL compile was performed. • RC - Indicates a Recap analysis was performed. • ASM - Indicates the program is an Assembler source program. • EN, R, RN - Indicates a Encore analysis was performed. • PROFILE - Indicates that the member contains SmartTest profile information. • INTERNAL - Indicates that the member ($$SEGMENTS contains Logic Segment criteria information. If this member is selected, the File - Logic Segment Directory pop-up (see Figure 12 on page 33) displays. • METRICS - Indicates that the member contains SmartDoc metric information. This member cannot be selected in Encore.
Date	The date on which the program was analyzed.
Time	The time at which the program was analyzed.
Jobname	The JOB NAME used to analyze the program.
Space	The percentage of space the program is using on this AKR, rounded to the nearest (.1) percent.

File - AKR Allocate/Expand Pop-up

The File - AKR Allocate/Expand pop-up, shown in Figure 19, is used to allocate a new AKR or to expand an existing AKR.

Figure 19 • File - AKR Allocate/Expand Pop-up

```
                          File - AKR Allocate/Expand
Command ===> S

        S - Submit JCL       E - Edit JCL       C - Specify Catalog

Expand existing AKR . . . NO              (Yes or No)

AKR data set name . . . . 'USER.TEST.AKR'
  Volume  . . . . . . . . ______
  Unit  . . . . . . . . . SYSDA           (Generic unit name)
  Space units . . . . . . RECORDS         (Records, Tracks or Cylinders)
  Primary space . . . . . 4000            (Primary amount in above units)
  Secondary space . . . . 0               (Secondary amount in above units)

Job statement information:
  //NAME     JOB (ACCOUNT),NAME,
  //          MSGCLASS=A
  //*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
  //*
```

To display the File - AKR Allocate/Expand pop-up, follow this step:

- Select option A on the File - AKR Utility pop-up (see Figure 17 on page 43).

Note:

The File - AKR Allocate Expand pop-up that you see may be different, depending on the options set when Encore was installed at your site.

To expand an existing AKR, follow this step:

- Enter YES in the Expand existing AKR field.

To allocate a new AKR, follow this step:

- Enter NO in the Expand existing AKR field.

The space needed for the AKR depends on the size and the number of COBOL programs that are analyzed and placed in it. See the Primary and Secondary space field descriptions for more information.

Options

Field	Description
S - Submit JCL	Type `S` on the command line and press Enter to submit the JCL to allocate or expand the AKR shown in the AKR dataset name field.
E - Edit JCL	Type `E` on the command line and press Enter to edit the JCL to allocate or expand the AKR shown in the AKR dataset name field.
C - Specify Catalog	Type `C` on the command line and press Enter to display the AKR Catalog Information pop-up (Figure 20 on page 51) that is used to specify the dataset name and password of a catalog, if required.

Fields

Field	Description
Expand existing AKR	Required. Enter YES to expand the AKR shown in the AKR dataset name field. Enter NO to allocate a new AKR. The default is NO.
AKR dataset name	The AKR name specified on the File - AKR Utility pop-up (see Figure 17 on page 43).
Management Class	If your site uses SMS, enter the management class for the AKR. The default is set during installation.
Storage Class	If your site uses SMS, enter the storage class for the AKR. The default is set during installation.
Data Class	If your site uses SMS, enter the data class for the AKR. The default is set during installation.
Volume	Leave this field blank if Unit is a generic type (e.g., SYSDA). Enter the volume serial number if Unit equals a non-generic type.
Unit	Required. Enter the type of units to allocate (e.g., SYSDA).
Space units	Required. Enter the type of units to allocate. The units must be records, tracks, or cylinders. The default is records.

Field	Description
Primary space / Secondary space	Required. Enter the amount of space to allocate for the type of units specified in the Units field. Enter the amount of secondary space. The space needed for the AKR depends on the size and the number of COBOL programs that are analyzed and placed in it. See the ASG-*Center Installation Guide* for space estimates.
Job statement information	Enter appropriate JOB statement information for your site.
Catalog DSN	Optional. Enter the catalog dataset name if the AKR dataset name is to be added to a catalog.
Password	Required if the CATALOG DSN is protected. Enter the password for the catalog dataset name specified in the CATALOG DSN field.

To specify a catalog or the password for your catalog

1. Type `C` on the command line of the File - AKR Allocate Expand pop-up and press Enter to display the AKR Catalog Information pop-up.

2. Specify the Catalog DSN and Password, as shown in Figure 20.

Figure 20 • AKR Catalog Information Pop-up

```
                    AKR Catalog Information
Command ===> _____________________________________________________

  Catalog DSN  . . .
  Password . . . . .              (Catalog password, if required)
```

Usage Notes

Note:

If the AKRVSAM install parameter is set to NO, copying an existing VSAM AKR converts it to a BDAM AKR.

Edit Interface

The Edit facility is used to edit a source COBOL program and have Logic Segment knowledge available.

To initiate the Edit facility, follow this step:

- Select File ▸ Edit program.

Note:
A program cannot be opened when initiating the Edit facility if no current Logic Segment exists.

Selecting Edit program provides a direct interface to the Edit facility, and provides you with a seamless interface to the site-defined ISPF editor so that modifications can be made to the original source program.

File - Edit Options Request Pop-up

The File - Edit Options Request pop-up, shown in Figure 21, is used to specify the source manager.

Figure 21 • File - Edit Options Request Pop-up

```
File  View  Extract  Generate  Search  List  Options  Help
.----------------------------------------. ---------------------------------
|       File - Edit Options Request      |
|                                        |  ________________________________
| If desired, select the Edit Options    |
| screen display. Then press Enter.      |  ******************************
|                                        |  ******************************
|   Options                              |           *****
|   /  Display options screen            |          *****
|                                        | *****   *****     *******
'----------------------------------------' *****  *****     ********
        *****           *****             ******  *****          ***
       *****           *****             ******  *****          ***
      *****          ************************  ******************
     *****          ************************    *****************


        Copyright Allen Systems Group, Inc., an unpublished work.
A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

To display the File - Edit Options Request pop-up, follow this step:

- Select File ▸ Edit program and press Enter. This pop-up displays so that you can specify whether to display the Options - COPY/Include Libraries pop-up (see Figure 22 on page 53).

Field

Field	Description
Display options screen	Optional. If specified, the Options - COPY/Include Libraries pop-up (see Figure 22 on page 53) displays so you can identify the source manager type, COBOL version, and COPY/Include libraries. If left blank, the Options - COPY/Include Libraries pop-up is bypassed, and the ISPF EDIT - Entry Panel (see Figure 23 on page 55) displays so you can specify the dataset and member name. The default is to display the Options - COPY/Include Libraries pop-up.

Options - COPY/Include Libraries Pop-up

The Options - COPY/Include Libraries pop-up, shown in Figure 22, is used to specify the COBOL version and source manager and define the COPY/Include libraries for a SmartEdit session (if applicable). You can also specify other product parameters by selecting additional actions from the Options pull-down (see Figure 140 on page 294).

Figure 22 • Options - COPY/Include Libraries Pop-up

```
File  COBOL  SourceManager  Options
--------------------------------------------------------------------------------
                        Options - COPY/Include Libraries
Command ===> ____________________________________________ Scroll ===> CSR

Specify Source Manager, COBOL Version and COPY/Include Libraries.

Source Manager . . : User             COBOL Version . : COBOL/370

   COPY List:
  Data Set Name  . :
  Member . . . . . :
  Description  . . :

        Copy/Include Data Set Name(s)             Type VolSer   Unit   Password
- ---------------------------------------------- ---- ------ -------- --------
/ 'ASG.VIACENxx.CNTL'                             PDS  ______ ________
****************************** BOTTOM OF DATA ********************************
```

To display the Options - COPY/Include Libraries pop-up, follow this step:

- Specify the Display options panel on the File -Edit Options Request pop-up (see Figure 21 on page 52) and press Enter.

Fields

Field	Description
Source Manager	Specifies the source manager for the edit session. To change the source manager, select Source Manager from the action bar.
COBOL Version	Specifies the COBOL version for the edit session. To change the COBOL version, select COBOL from the action bar.
COPY List	• Data Set Name - If a copy list is specified, the name of the dataset containing the copy list member displays. To load a copy list, select File ▸ Load Copy list on the Options - COPY/Include Libraries pop-up. • Member - If a copy list is specified, the name of the member containing the copy list displays. • Description - Displays the description of the copy list specified, if a description was entered when the copy list was saved. **Note:** A copy list defines COPY/Include libraries for an edit session. To create a copy list, enter COPY/Include library information on the Options - COPY/Include Libraries pop-up, and then select File ▸ Save COPY List on the Options - COPY/Include Libraries pop-up. Copy lists can also be created from compile JCL using the File ▸ Import COPY List.
Copy/Include Data Set Name(s)	Displays the COPY/Include datasets specified for the edit session. Use line commands to add or delete datasets. • To add datasets from a COPY list, select File ▸ Load COPY list on the Options - COPY/Include Libraries pop-up. • To disable a copy library for the current edit session, replace the / in the line command area with a blank.
Type	The COPY or INCLUDE dataset source type. These are the valid entry types: PDS - Partitioned Data Set or Sequential Data Set, LIB - Librarian, or PAN - Panvalet.
VOLSER	The volume serial number containing the COPY or INCLUDE dataset. This field is required if the dataset is not catalogued.

Field	Description
Unit	The generic storage unit used by the COPY or INCLUDE dataset.
Password	The password for the COPY or INCLUDE dataset, if applicable.

See the Encore online help for more information about using the Options - COPY/Include Libraries pop-up.

ISPF Edit - Entry Panel

The Edit - Entry Panel, shown in Figure 23, is used to identify the dataset and member name. Upon successful completion of the Options - COPY/Include Libraries pop-up, an Edit - Entry Panel specific to the selected source manager displays. The name of the desired COBOL program to be edited should be entered.

This example shows the format of the PDS Source Manager option.

Figure 23 • ISPF Edit Entry Panel

```
  Menu  RefList  RefMode  Utilities  LMF  Workstation  Help
--------------------------------------------------------------------------------
                                Edit Entry Panel
Command ===> _

ISPF Library:
   Project . . . ASG
   Group . . . . VIACENxx . . .          . . .          . . .
   Type  . . . . CNTL
   Member  . . .                 (Blank or pattern for member selection list)

Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Workstation File:
   File Name . . . . .
                                         Options
Initial Macro  . . . .                   /  Confirm Cancel/Move/Replace
Profile Name . . . . .                      Mixed Mode
Format Name  . . . . .                      Edit on Workstation
Data Set Password  . .                      Preserve VB record length
```

Edit Screen

Once a member name is identified, the Edit session is started, as shown in Figure 24. If the member name did not previously exist, an empty edit screen displays. Otherwise the contents of the identified member display. All normally available edit functions and commands are valid.

If the Edit session was requested from the Generate - COBOL Module pop-up (see Figure 98 on page 208), use the RNSCOPY command to copy the generated COBOL module into the current member.

Figure 24 • Edit Screen

```
ASG-Encore EDIT - ASG.ENCXXXX.COBOL                    COLUMNS 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001       ********************************************************************
000002       *                        ASG, INC.                                 *
000003       *                                                                  *
000004       * GENERATED BY ASG-ENCORE        RELEASE 6.0(000)                 *
000005       *                                                                  *
000006       * PROGRAM:        VIARDEMO                                         *
000007       *                                                                  *
000008       * DATE:           DD-MMM-YYYY HH:MM:SS                             *
000009       * REQUESTOR:      USERID                                           *
000010       *                                                                  *
000011       * AKR INFORMATION                                                  *
000012       *     DS NAME:    USERID.TEST.AKR                                  *
000013       *     MEMBER:     VIARDEMO                ALIAS: (NONE)            *
000014       *                                                                  *
000015       * SEGMENT NAME:   PERFORM-RANGE-0001                               *
000016       *                                                                  *
000017       * OBJECTIVE TYPE: PERFORM RANGE                                    *
000018       * PERFORM NAME(S):                                                 *
000019       *                 GET-NUM-OF-DAYS                                  *
000020       *                                                                  *
000021       ********************************************************************
```

File - Display Logic Segment Entry Pop-up

If the DISPLAY command is entered during an Edit session and there is no current Logic Segment, the File - Display Logic Segment Entry pop-up displays, as shown in Figure 25.

You must identify the AKR dataset and program names.

Figure 25 • File - Display Logic Segment Entry Pop-up

```
                    File - Display Logic Segment Entry

Type AKR information and program name. Then press Enter.


Application Knowledge Repository (AKR):

   Data Set Name . . 'USER.TEST.AKR'
   Program Name  . . __________  (blank for selection list)

   Volume Serial . . ______      (if not cataloged)
   Password  . . . .             (if password protected)
```

Fields

Field	Description

Data Set Name	Required. Enter the AKR dataset name. The program name can also be entered in this field. If the TSO ID qualifier is the same as the user ID, enter the library and type without quotes. For example:
	`TEST.AKR or`
	`TEST.AKR(`*`pgmname`*`)`
	If the TSO ID qualifier is different than the user ID, enter the project, group, and type in quotes. For example:
	`'TSOID.TEST.AKR'or`
	`'TSOID.TEST.AKR(`*`pgmname`*`)'`

Program Name	Optional. Enter the program name contained in the AKR. If left blank, the File - Program Name List pop-up displays (see Figure 11 on page 30) so you can select the program name.
Volume Serial	Required if the dataset is not cataloged. If the dataset is cataloged, this field is optional. Enter the volume serial number.
Password	Required if the dataset is protected. Enter the dataset password.

Displayed Logic Segment

After the Logic Segment name has been identified, the Edit screen redisplays with all statements in the program that are not contained in the current Logic Segment excluded from the screen, as shown in Figure 26. The Logic Segment statements are highlighted.

This example shows the Edit screen after the DISPLAY command has been issued and a Logic Segment selected.

Figure 26 • Edit Screen After Using DISPLAY Command

```
 ASG-ESW - COBOL EDIT - ASG.VIACENXX.CNTL(VIARDEMO) - 5 COLUMNS 00001 00072
 Command ===>                                                Scroll ===> CSR
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 37 Line(s) not Displayed
 HI===> 003800 01  ACCT-RECORD.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
 HI===> 004000     05  ACCT-KEY.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
 HI===> 004200         10  ACCT-TYPE                 PIC X(3).
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  2 Line(s) not Displayed
 HI===> 004500     05  ACCT-MIN-BALANCE              PIC S9(6)V99.
 HI===> 004600     05  ACCT-AVG-BALANCE              PIC S9(6)V99.
 HI===> 004700     05  ACCT-MULTI-ACCT               PIC X(3).
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
 HI===> 004900     05  ACCT-INTEREST                 PIC S9(6)V99.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 15 Line(s) not Displayed
 HI===> 005600 01  TRAN-RECORD.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
 HI===> 005800     05  TRAN-ACCT-KEY.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - 25 Line(s) not Displayed
 HI===> 008400 77  TRAN-FILE-FLAG                    PIC X    VALUE ' '.
 HI===> 008500     88  TRAN-FILE-ENDS                VALUE 'X'.
 - - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
 HI===> 008700 77  CHK-INTEREST-RATE                 PIC S9(4)V9(4) COMP-3

 ASG1604I ALL LOGIC SEGMENT LINES WERE DISPLAYED.
```

File - Execute Script Pop-up

The File - Execute Script pop-up, shown in Figure 27, is used to begin processing a script file.

Figure 27 • File - Execute Script Pop-up

```
                          File - Execute Script

Type a data set name, a data set name with a member name, or just a member
name in parentheses. Select the "Single step" option if you wish to
execute the script one command at a time.

If you wish to "resume" normal execution of a "single step" script, select
the "Resume automatic mode" option. If you wish to cancel a "single step"
script, select the "Cancel execution" option. The "resume" and "cancel"
options should be entered with a blank data set name field.

Data set name _____________________________________________________________

Options
1   1.  Automatic
    2.  Single step
    3.  Resume automatic mode
    4.  Cancel execution
```

To display the File - Execute Script pop-up, follow this step:

- Select File ▸ Execute script and press Enter.

Fields

Field	Description
Data set name	Required. Enter the name of the script file to be processed.
Options	Specify the processing mode for the script file. The default is 1. Automatic. 1. Automatic - Each command in the script file is executed consecutively. 2. Single step - Each command in the script file displays in the command input area. The displayed command can be changed or erased. Enter must then be pressed to actually execute the displayed command. Processing of the script file continues in this manner until all commands have displayed. 3. Resume automatic mode - After a script file has begun executing in Single step mode, selecting Resume executes the remaining script file commands without stepping through each. 4. Cancel execution - After a script file has begun executing in Single step mode, selecting Cancel stops execution of the remaining script file commands. **Note:** Script files can also be executed with the EXECUTE command, (see "EXECUTE Command" on page 346 for more information.)

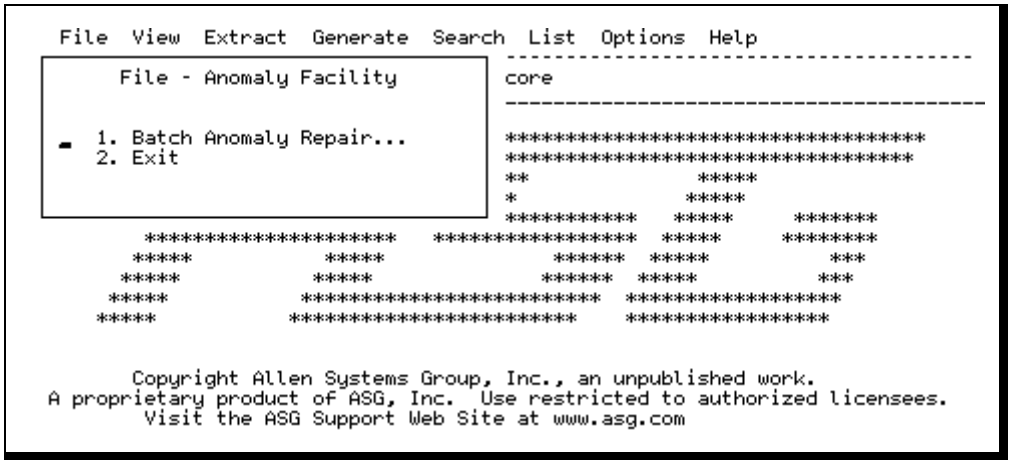
Anomaly Facility

The Anomaly facility is used to repair programs already analyzed in an AKR that contain certain anomalies. This is accomplished by allowing you to submit the batch job for the member that generated the anomaly condition using the pop-ups described in this section.

File - Anomaly Facility Pop-up

The File - Anomaly Facility pop-up, shown in Figure 28, is used to select the Anomaly Facility - Batch Submit screen or to exit back to the Encore Primary screen.

Figure 28 • File - Anomaly Facility Pop-up



To display the File - Anomaly Facility pop-up, follow this step:

- Select File ▸ Anomaly facility and press Enter.

Fields

Field	Description
1. Batch Anomaly Repair	Select this option to display the Anomaly Facility - Batch Submit screen (see Figure 29 on page 62).
2. Exit	Select this option to return to the Encore Primary screen.

Anomaly Facility - Batch Submit Pop-up

The Anomaly Facility - Batch Submit pop-up, shown in Figure 29, is used to fix anomalies found during an Encore program analyze.

Figure 29 • Anomaly Facility - Batch Submit Pop-up

```
                        Anomaly Facility - Batch Submit   Program: VIARDEMO
Command ===> ____________________________________________________________________

Specify required information.  Then press PF key for action.

   AKR data set name . . . 'USER.TEST.AKR'
   AKR program name  . . . __________  (blank for selection list)
   Output dataset name . . ____________________________________________
                                       (blank for VIAANOUT DD SYSOUT=*)
   Output member name  . . ________    (blank if sequential file)

Anomalies to be repaired:
/  Out of perform goto
_  Goto depending
_  Inconsistent perf ranges

Job statement information:
  //USERID  JOB (ACCOUNT),'DDDYYB',
  //    MSGCLASS=X,CLASS=A,NOTIFY=USERID,PRTY=6
  //*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
  //*
            PF4=User Parameters  PF5=Edit JCL  PF6=Submit JCL
```

To display the Anomaly Facility - Batch Submit pop-up, follow this step:

- Select Batch Anomaly Repair or press Enter on the File - Anomaly Facility pop-up.

Fields

Field	Description
AKR dataset name	Required. Type the name of the AKR that contains the analyzed program member that contains the anomaly types listed in the Anomalies to be repaired section.
AKR program name	Required. Type the name of the analyzed member that contains the anomalies. You can also leave this field blank and press Enter to display the Anomaly Facility - Program Selection List pop-up (see Figure 31 on page 66), which allows you to select AKR members that contain anomalies.
Output dataset name	Type the name of a PDS or sequential file that is already allocated. You can also leave this field blank to send the output to DD VIAANOUT.
Output member name	If the output dataset name is a PDS, type the name you want to give the output member or leave blank if the output dataset is a sequential file.

Field	Description
Anomalies to be repaired	Required. Lists the types of anomalies that can be repaired using this facility. Select one or more anomaly types to be repaired. 1. Out of perform goto - Program contains perform ranges that should be re-engineered due to out-of-perform jumps or multiple entry points. 2. Goto depending - Program contains GOTO DEPENDING statements that should be re-engineered. 3. Inconsistent perf ranges - Program contains perform ranges that have common starting points and different ending points that should be re-engineered.
Job statement information	Required. Enter standard JOB card routing information.
PF KEYS	• PF4 = User Parameters - Displays the Anomaly Facility - User Parameters Program pop-up (see Figure 31 on page 66), which is used to select the type of anomaly user parameters you want to include in the anomaly repair run. It also contains settings for the Comment and Header options, which are described in "File - Analyze Submit Pop-up" on page 39 . • PF5 = Edit JCL - Displays the Edit screen (see Figure 32 on page 67), which is used to edit your analyze submit JCL. • PF6 = Submit JCL - Submits your member for a new analysis so the anomalies can be repaired.

Anomaly Facility - Program Selection Pop-up

The Anomaly Facility - Program Selection List pop-up, shown in Figure 30, is used to select the AKR member that contains anomalies.

Figure 30 • Anomaly Facility - Program Selection List Pop-up

```
                    Anomaly Facility - Program Selection List
Command ===> ____________________________________________ Scroll ===> CSR

Select desired Program Name. Then press Enter.

  AKR: 'USER.PART.AKR'

  Name        Alias of    Type      Lines   Date         Time      Jobname
- ----------  ----------  --------  ------  -----------  --------  --------
  MEMBER1                 IN,EN        423  DD-MMM-YYYY  HH:MM:SS  JOBEN
_ MEMBER2                 EN           740  DD-MMM-YYYY  HH:MM:SS  JOBB
****************************** BOTTOM OF DATA ******************************
```

To display the Anomaly Facility - Program Selection List pop-up, follow this step:

- Leave the AKR program name field blank on the Anomaly Facility- Batch Submit pop-up (see Figure 29 on page 62) and press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the Name field, accepts commands for programs with a Type of EN or RN, and that contain anomalies that can be repaired. For programs with Types other than EN or RN, the line command area is protected.
AKR	The complete dataset name of the AKR specified on the Anomaly Facility - Batch Submit pop-up (see Figure 29 on page 62). **Note:** You can only select a member that contains anomalies that can be repaired.

Field	Description
Name	The name of all the programs in the AKR, from the PROGRAM-ID statement. If the analyzed program contains an ENTRY point, Name is the ENTRY point name. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Name is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39). These are the valid name selection entries: • S - Selects the specified program that contains anomalies that can be repaired. • / - Locates the specified program. The screen is scrolled until the specified program is at the top of the display.
Alias of	If the analyzed program contains an ENTRY point, Alias of is the name of the program that contains the ENTRY point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up (see Figure 16 on page 39).
Type	Indicates the type(s) of analysis that was performed on the program. The type of analysis is specified on the File - Analyze Submit pop-up (see Figure 16 on page 39) at the time the Analyze job is submitted. Only programs with the EN or RN analysis type that contain anomalies that can be repaired can be selected for use by the Batch Anomaly Repair Facility. These are the valid analysis types: • EN, R, RN - Indicates a Encore analysis was performed. • I, IN - Indicates a Insight analysis was performed. • SQ - Indicates a SmartQuest analysis was performed. • ST - Indicates a SmartTest analysis was performed. • DS - Indicates a SmartDoc analysis was performed. • DC - Indicates a SmartDoc analysis with a COBOL compile was performed. • DX - Indicates an Extended SmartDoc analysis was performed.

Field	Description
	• DA - Indicates an Extended SmartDoc analysis with a COBOL compile was performed. • RC - Indicates a Recap analysis was performed. • ASM - Indicates the program is an Assembler source program. • PROFILE - Indicates that the member contains SmartTest profile information. • INTERNAL - Indicates that the member ($$SEGMENTS contains Logic Segment criteria information. • METRICS - Indicates that the member contains SmartDoc metric information. This member cannot be selected in Encore.
Lines	The number of lines contained in the program.
Date	The date the program was analyzed.
Time	The time the program was analyzed.
Jobname	The job name that analyzed the program.

Anomaly Facility - User Parameters Pop-up

The Anomaly Facility - User Parameters pop-up, shown in Figure 31, is used to select options for source generation and to enter any valid user parameters.

Figure 31 • Anomaly Facility - User Parameters Pop-up

```
                    Anomaly Facility - User Parameters
Command ===> _

Type in anomaly user parameters.  Then press Enter.

Options:
   Comment out changed lines?
   Generate ASG header?

User parameters:
```

Fields

Field	Description
Options	This field is used to specify one or both of these options at source generation: • Comment out changed lines? - Specifies that you want to see all changed lines as commented out lines in the generated repaired code. • Generate ASG header? - Specifies that you want to include a header in the generated source.
User parameters	This field is used to enter any valid user parameters (e.g., DEBUG).

Anomaly Facility - Edit Screen

The Edit screen, shown in Figure 32, is used to edit your analyze JCL, if required. All normally available edit functions and commands are valid.

Figure 32 • Anomaly Facility - Edit Screen

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------
EDIT       USERID.T######.VIAJCL                          Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ****************************** Top of Data ******************************
000001 //USERID JOB (ACCOUNT),'JOBNAMR',
000002 //     MSGCLASS=A,
000003 //*    INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
000004 //*
000005 //*******************************************************************
000006 //* THIS JCL HAS BEEN CREATED BY THE ASG ANOMALY SUBMIT FACILITY *
000007 //*******************************************************************
000008 //*
000009 //VIAIN     EXEC PGM=IEBGENER
000010 //SYSIN     DD DUMMY
000011 //SYSPRINT DD DUMMY
000012 //SYSUT2    DD DSN=&&VIAIN,DISP=(,PASS),UNIT=SYSDA,
000013 //             SPACE=(TRK,(1,1),RLSE),
000014 //             DCB=(RECFM=FB,LRECL=80,BLKSIZE=7440)
000015 //SYSUT1    DD *
000016 * AKR DSN NAME:
000017  AKR('USER.TEST.AKR')
000018 * PROGRAM NAME IN AKR
000019  PGM(VIARBRWS)
000020 * ANOMALIES TO BE FIXED
000021  ANOM(OOPGT,GTD,IPR)
000022 * SHOULD REMOVED LINES BE COMMENTED OUT AND ADDED
000023  COMMENT(YES)
000024 * SHOULD A ASG GENERATED HEADER BE ADDED
000025  HEADER(YES)
```

To display the Edit screen, follow this step:

- Press PF5 on the Anomaly Facility - Batch Submit pop-up (see Figure 29 on page 62).

4 View Menu

This chapter describes Encore's View pull-down menu functions and contains these sections:

Section	Page
Introduction	69
View Pull-down	71
Source View Screen	73
Tree View Screen	80
Structure View Screen	87

Introduction

The View pull-down (see Figure 33 on page 71) presents various displays of an existing COBOL source program. Program information can be presented in a graphic structure, logical execution, or source code format. This facility is used to examine the program to become more familiar with the program structure and functions, or to increase understanding of the program prior to making a change or extracting a Logic Segment.

These are three different view methods available to assist you in isolating, understanding, and identifying important sections of the source program or Logic Segment.

- The Source view method presents the actual COBOL source code in source statement order, in a format familiar to most programmers. See "Source View Screen" on page 73 for more information.

- The Tree view method presents the paragraph hierarchy, which shows a structural representation of the program in logical execution order. See "Tree View Screen" on page 80 for more information.
- The Structure view method presents a graphical display of the PERFORMed structure of the program. See "Structure View Screen" on page 87 for more information.

In all view methods the magnification can be changed, (i.e., you can zoom in and zoom out of logical program units). Each of these three views is created from the same original COBOL source program, but presents a different perspective, and is concurrently available and synchronized with the other views. You can switch back and forth between views and retain the results of certain commands. Each view displays the result of some command or commands issued against the COBOL source code. Then, these results display in a format appropriate to the specific view.

The Structure and Tree view methods, which display the logical program units that compose the program, can be tailored to display varying levels of the program structure. These displays are configured on a separate pop-up before the View screen displays.

When a program is first opened, you can have one of the view methods initially display by default. You can specify the default view method on the Options - Product Parameters pop-up (see Figure 141 on page 296 for more information).

In addition, Program Profile pop-ups are provided to display summary information about the program files, data variables, Copybooks, sections and paragraphs, and control flow.

View Pull-down

The View pull-down, as shown in Figure 33, is used to select the method to be used to view the program or Logic Segment, view program information, display or exclude source according to the program hierarchical structure, change the level of information viewed, reset the display, and exclude source from the display.

To display the View pull-down, follow this step:

- Select View on the action bar and press Enter.

Figure 33 • View Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
-----
Comma      1. Source                  G-Encore
           2. Tree...
           3. Structure...
           4. Logic segment
           5. Program summary...
           6. Paragraph X-ref...
           7. Zoom in
           8. Zoom out
           9. Levels...
          10. Reset...
          11. Exclude...

     Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

Note:

Actions on the View pull-down cannot be selected unless there is an active program, (i.e., a program has been opened using the File ▸ Open option). (See Chapter 3, "File Menu," on page 25 for more information.)

Actions

Action	Description
1. Source	Displays the Source View screen (see Figure 34 on page 73), which shows the program in COBOL source statement order.
2. Tree	Displays the program in logical execution order, showing logical program units and the relationships between related units. Selecting this action displays the View - Tree View Request pop-up (see Figure 36 on page 77) used to specify the format of the Tree View display.

Action	Description
3. Structure	Displays the program in a graphical format, with each box on the screen representing a logical program unit. Selecting this action displays the View - Structure View Request pop-up (see Figure 40 on page 85) used to specify the format of the Structure View display.
4. Logic segment	Displays the View - Select Logic Segment pop-up (see Figure 43 on page 91) used to select the Logic Segment to be viewed.
5. Program summary	Displays the View - Program Summary pop-up (see Figure 44 on page 92) used to select the type of summary information that displays.
6. Paragraph X-ref	Displays the View - Paragraph Cross Reference Request pop-up (see Figure 51 on page 104) used to customize the View - Paragraph Cross Reference pop-up (see Figure 52 on page 106).
7. Zoom in	Displays source code according to the hierarchical levels of the program. When this action is selected, a message displays as a reminder to position the cursor and then press Enter. When this action is selected on the Structure View screen (see Figure 41 on page 87), the View - Source pop-up (see Figure 35 on page 76) displays with a shortened action bar, showing the source lines of the perform range.
8. Zoom out	Excludes source code according to the hierarchical levels of the program. When this action is selected, a message displays as a reminder to position the cursor and then press Enter. This action is not valid on the Structure View screen (see Figure 41 on page 87).
9. Levels	Displays the View - Levels Request pop-up (see Figure 53 on page 108) used to specify the new level number. This action is valid only on the Tree View (see Figure 38 on page 80) and Structure View screens (see Figure 41 on page 87).
10. Reset	Displays the View - Reset Request pop-up (see Figure 54 on page 109) used to specify which display features to reset, including highlighting, tags, excluded lines, pending line commands, and messages.
11. Exclude	Displays the View - Exclude Request pop-up (see Figure 55 on page 111) used to specify the exclude options and string.

Source View Screen

The Source View screen is a textual display of the COBOL source program code. This screen displays each source line and functions similar to the ISPF Browse option.

To display the Source View screen, follow this step:

Select View ▸ Source and press Enter.

Figure 34 shows the Source View screen as it displays when entered from the View pull-down.

Figure 34 • Source View Screen

```
File  View  Extract  Generate  Search  List  Options  Help
--------------------------------------------------------------------------------
                                Source View                 Program: VIARDEMO
Command ===> ____________________________________________ Scroll ===> CSR

000225  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,    PATTERN
000226                           BANK-BALANCE.
000227
000228      OPEN OUTPUT EXCEPTION-FILE.
000229
000230      PERFORM INITIALIZE-PGM.
000231
000232      PERFORM ACCT-MAINTENANCE
000233         THRU ACCT-MAINTENANCE-EXIT
000234        UNTIL TRAN-FILE-ENDS.
000235
000236      PERFORM REPORT-FINAL-CNTRS.
000237
000238      PERFORM CLOSE-PGM.
000239
000240      GOBACK.                                                  PGM EXIT
000241
```

The Source View screen is also available in the Extract facility. When Source View is entered from the Extract facility, it displays as the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar and tags are placed at the right margin of specified lines. (See Chapter 5, "Extract Menu," on page 113 for more information.)

The Source View screen is also used to display individual source lines, with copy members properly expanded.

After a Report or Computation Variable extract has been completed, you can enter the I (Include) line command on the Source View screen to include specific additional statements in the extracted Logic Segment.

When a program is first opened, you can have it initially display in Source View by specifying Source View as the default view on the Options - Product Parameters pop-up. (See Chapter 9, "Options Menu," on page 293 for more information.)

Source View Primary Commands

These are the Source View primary commands used to format the display, view additional data, search or locate strings, search data items, or search lines. You enter primary commands in the command input area of the screen (see Chapter 11, "Commands," on page 321 for more information).

Command	Results
BRANCH	Positions the cursor at the specified target.
FIND	Highlights and tags all lines found and positions the cursor at the first target in the specified direction. The short message indicates the number of occurrences found.
FINDXTND	Positions the cursor at the first target in the specified direction. All targets are highlighted and tagged, and the long or short message indicates the number of occurrences found.
LOCATE	Scrolls to the specified string or line label.
RESET	Turns off highlighting, cancels pending line commands, terminates message lines, or redisplays excluded lines.
ZOOMIN	Displays additional source lines according to the hierarchical levels of the program
ZOOMOUT	Removes source lines according to the hierarchical levels of the program.

Source View Line Commands

These are the valid Source View Line commands that can be used on the Source View screen (see Figure 34 on page 73). Line commands are typed over the source line number on the appropriate line (see Chapter 11, "Commands," on page 321 for more information).

Command	Results
Fn	First. Redisplays the specified number of excluded lines, starting with the first line in the block of excluded lines. The default is 1.
Hn	Highlight. Highlights the specified number of lines. The default is 1.
HH	Highlight Block.Highlights a block of lines.
I	Include. Incorporates the specified line into the current Logic Segment. This line command is valid only when the current Logic Segment type is Report or Computation Variable.
.label	Equate. Assigns a name to a particular source line. This command is entered as a period followed by 1 to 4 alphabetic characters. This command can be used in conjunction with the LOCATE primary command.
Ln	Last. Redisplays the specified number of excluded lines, starting with the last line in the block of excluded lines. The default is 1.
Sn	Show. Redisplays the specified number of excluded lines, starting with the first line in the block of excluded lines. The default is 1.
SS	Show block. Redisplays a block of exclude lines.
Xn	Exclude. Excludes the specified number of lines from displaying. The default is 1. Excluded lines are replaced with a row of dashes specifying the number of lines excluded.
XX	Exclude Block. Excludes a block of lines from displaying.

View - Source Pop-up

The View - Source pop-up, shown in Figure 35, contains a textual display of the COBOL source program code. This pop-up is the same as the Source View screen (see Figure 34 on page 73), but has a shortened action bar.

Figure 35 • View - Source Pop-up

```
File  View  Extract  Generate  Search  List  Options  Help
--------------------------------------------------------------------------------
                              Source View                 Program: VIARDEMO
Command ===> ____________________________________________ Scroll ===> CSR

000225  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,    PATTERN
000226                           BANK-BALANCE.
000227
000228      OPEN OUTPUT EXCEPTION-FILE.
000229
000230      PERFORM INITIALIZE-PGM.
000231
000232      PERFORM ACCT-MAINTENANCE
000233         THRU ACCT-MAINTENANCE-EXIT
000234        UNTIL TRAN-FILE-ENDS.
000235
000236      PERFORM REPORT-FINAL-CNTRS.
000237
000238      PERFORM CLOSE-PGM.
000239
000240      GOBACK.                                                  PGM EXIT
000241
```

When a View screen displays by any means other than the View pull-down (Figure 33 on page 71), (e.g., entering a command or specifying a Selection Screen on an Extract pop-up), the View screen presents as a pop-up with a shortened action bar.

View - Tree View Request Pop-up

The View - Tree View Request pop-up, shown in Figure 36, is used to format the display of the program to be shown on the Tree View screen.

To display the View - Tree View Request pop-up, follow this step:

- Select View ▸ Tree and press Enter.

Figure 36 • View - Tree View Request Pop-up

```
                          View - Tree View Request

        Type request options.  Then press Enter.  For a name selection list
        for a range (other than 1 or 2), type a pattern (e.g. ABC*) in the
        name field.

        Levels 2    of    4     of depth

        Contents                                Range
        1   1.  Labels                          1   1.  Procedure division
            2.  Labels and performs                 2.  All entry points
            3.  Labels, performs and GOTOs          3.  Perfrange name
            4.  Labels, performs, GOTOs and         4.  Entry point name
                conditionals                        5.  Nested program name

        Name ____________________________________________________________

        Options
        _  Repetition
```

When the View - Tree View Request pop-up initially displays, each field contains default values. If you change these defaults the new values are retained for subsequent sessions.

Note:

Tree View is also available in the Extract facility. (See Chapter 5, "Extract Menu," on page 113 for more information.)

To change the level depth

1. Enter the desired configuration parameters on View - Tree View Request pop-up and press Enter. The Tree View screen displays.

2. Use the LEVELS primary command to change the level depth.

Fields

Field	Description
Levels	Optional. Specify the number of hierarchical levels of COBOL source statements display. Change the level depth value to increase or decrease the amount of detail displayed. The amount of detail increases as the level depth increases. The total number of levels in the program display to the right of this field. Enter a number equal to or less than this number, or enter the value MAX. The default is 2. When Levels is set to MAX, the actual number of levels displayed depends on the Repetition Option (see "Usage Notes" on page 80 for more information).
Contents	Optional. Enter the number of the statement type(s) to display. Each statement type is a subset of the COBOL statements contained in the program. Encore classifies COBOL statements into subsets by grouping together COBOL verbs of a similar nature. These are the subsets available on this pop-up: 1. Labels - The PROCEDURE DIVISION statement, and all section and paragraph names in the PROCEDURE DIVISION. This is the default. 2. Performs - Statements containing the PERFORM, SORT or MERGE verbs. 3. GOTOs - Statements containing an ALTER or GO TO verb. 4. Conditionals - Statements or parts of statements that conditionally change the flow of control in a program, such as IF, ELSE, and WHEN.
Range	Optional. Specify how much of the program to display. The default is 1, Procedure Division. These are the available ranges: 1. Procedure division - The entire PROCEDURE DIVISION displays. This is the default. 2. All entry points - The entire PROCEDURE DIVISION and all ENTRY points display. 3. Perfrange name - One perform range displays. If this range is selected, the perform range name must be entered in the Name field. 4. Entry point name - One ENTRY point displays. If this range is selected, the entry point name must be entered in the Name field. 5. Nested program name - One nested program displays. If this range is selected, the program name must be entered in the Name field.

Field	Description
Name	This field is used if Range 3, 4, or 5 is selected, and ignored if Range 1 or 2 is selected. Enter the name of the perform range, entry point, or nested program to display. If this field is left blank, a Selection List - Perform Range Names pop-up (see Figure 37) displays to allow selection of the desired target name. Wildcard characters (a question mark (?) for 1 character; an asterisk (*) for zero or more characters) can be entered in this field to reduce the size of the selection list.
Repetition	Optional. Determines whether perform range structures entirely display wherever they appear in the program. If not specified, the complete perform range structure displays the first time it displays in the program. Subsequent references to the perform range do not display, and are tagged as REPEAT to the right of the PERFORM statement. If specified, the perform range fully displays wherever it occurs. The default is to display the perform range only the first time it displays.

Figure 37 • Selection List - Perform Range Names Pop-up

```
                    Selection List - Perform Range Names       LINE 1 OF 2
Command ===> _                                          Scroll ===> CSR

S    Perform range name
-  ---------------------------------------------------------------------
_  READ-NEXT
   READ-PREV
 ***************************** BOTTOM OF DATA   ************************
```

To specify the Name, follow this step:

- Type `S` on the line command next to the desired name on the Selection List - Perform Range Names pop-up and press Enter. The View - Tree View Request pop-up (see Figure 36 on page 77) redisplays with the selected name in the Name field.

Usage Notes

When you request MAX level depth, the actual number of levels displayed may be different, depending on the value in the Repetition field. MAX displays all levels in the program, however, if the maximum level consists of only repeated perform ranges, then the level display is truncated to the next higher level that does not consist exclusively of repeated perform ranges.

Tree View Screen

The Tree View screen (see Figure 38) contains a textual representation of the execution sequence of a COBOL program.

To display the Tree View screen, follow this step:

- Press Enter on the View - Tree View Request pop-up (see Figure 36 on page 77).

Figure 38 shows the Tree View screen as it displays when entered from the View pull-down (see Figure 33 on page 71).

Figure 38 • Tree View Screen

```
  File  View  Extract  Generate  Search  List  Options  Help
--------------------------------------------------------------------------
                              Tree View                        LINE 1 OF 21
Command ===> ___________________________________________________ Scroll ===> CSR

****** *************************** TOP OF DATA ********************************
000000  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,
000002        INITIALIZE-PGM.
000003           OPEN-FILES.
000003           GET-NUM-OF-DAYS.
000004              SUM-TOTAL-DAYS.
000002        ACCT-MAINTENANCE.
000003           ACCT-MAINTENANCE-EXIT.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 LINE NOT DISPLAYED
000003           ACCT-MAINTENANCE-EXIT.                          REPEATED
000003           UPDATE-ACCT.                                    DATA MOD
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  5 LINES NOT DISPLAYED
000003           UPDATE-ACCT-EXIT.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 LINE NOT DISPLAYED
000002        ACCT-MAINTENANCE-EXIT.
- - -  -  -                                                  NOT DISPLAYED
000002          ASG1582I 4 OF 4 LEVELS DISPLAYED IN THE VIEW.
000003
****** ************************* BOTTOM OF DATA ******************************
```

The Tree View screen is also available in the Extract facility. When Tree View is entered from the Extract facility it displays as the View - Tree pop-up with a shortened action bar (see Figure 39 on page 84).

The Tree View screen displays logical program units and the relationships between related units. A logical program unit is defined as a PERFORMed range of code including GO TO code, or a CALLed program. Logical program units display as COBOL source statements, and the relationship between each unit is shown by indentation.

When this screen first displays, a long message indicates the number of levels displayed and the total number of levels in the program.

To change the number of levels displayed

1. Enter with the LEVELS command.

   Or

   Select View ▸ Levels from the View pull-down (see Figure 33 on page 71) and press Enter.

2. Enter the appropriate information.

To have a program initially display in Tree View when first opened, follow this step:

- Specify Tree View as the default view on the Options - Product Parameters pop-up. (See Chapter 9, "Options Menu," on page 293 for more information.)

Usage Notes

The left-most column of the Tree View screen shows the nesting level of each statement.

Statement types preceded by a plus sign (+) indicate further levels of source exist for each statement, meaning that the label/statement contains nested levels of source code not currently displayed.

To display the source code, follow this step:

- Choose one of these actions:
  - Enter the ZOOMIN primary command.
  - Enter the ZI line command.
  - Select View ▸ Zoom in.

Tree View Primary Commands

These are the Tree View primary commands that are used to format the display, view additional data, search or locate strings, search data items, or search lines. You enter primary commands in the command input area of the screen (see Figure 38 on page 80).

Command	Results
FIND	Highlights and tags all lines found, and positions the cursor at the first perform range in a top/down manner which contains the target. The short message indicates the number of occurrences found.
FINDXTND	Performs a COBOL intelligent search of the source code for one or all occurrences of the specified target. Data names are tagged with USE, MOD or REF. These operands are not applicable on the Tree View screen: FIRST, LAST, PREV, and NEXT.
JUMP	Displays the View - Source pop-up positioned to the line where the JUMP command was entered. Enter `JUMP` in the command input area, move the cursor to the desired line and press Enter. JUMP also functions as the J line command (see "Tree View Line Commands" on page 83).
LEVELS `n`	Redisplays the Tree View screen to the specified level. If there are levels less than `n`, they are expanded to the specified level. If there are levels greater than `n`, they are removed. The LEVELS command works globally throughout the displayed program.
RESET	Turns off highlighting, cancels pending line commands, terminates message lines, or redisplays excluded lines.
SCROLL	Finds highlighted or nonhighlighted lines.
ZOOMIN	Expands the paragraph to the next lower hierarchical level. The paragraph name is identified by cursor position. If the cursor is left in the command input area, the entire PROCEDURE DIVISION is expanded. Entering ZOOMIN on the label expands one level of PERFORMs contained by the paragraph. If there are no more hierarchical levels, or if they have already been expanded, the source displays for that label. If all levels displayed, entering ZOOMIN results in an error message. ZOOMIN cannot be entered on a source line.
ZOOMOUT	Removes levels. The first level removed is the source. If no source displays, lower levels are removed. If both source and lower levels have been removed, no more levels are removed.

Tree View Line Commands

These are valid Line commands that can be used on the Tree View screen (see Figure 38 on page 80). Line commands are typed over the nesting level number on the appropriate line.

Command	Results
Fn	First. Redisplays the specified number of excluded lines, starting with the first line in the block of excluded lines. The default is 1.
J	Jump. Displays the Source view screen positioned to the line where the J line command is entered.
Ln	Last. Redisplays the specified number of excluded lines, starting with the last line in the block of excluded lines. The default is 1.
Sn	Show. Redisplays the specified number of excluded lines, starting with the first line in the block of excluded lines. The default is 1.
SS	Show block. Redisplays a block of exclude lines.
Xn	Exclude. Excludes the specified number of lines from displaying. The default is 1. Excluded lines are replaced with a row of dashes specifying the number of lines excluded.
XX	Exclude Block. Excludes a block of lines from displaying.
ZI	Zoom In. Performs the ZOOMIN primary command for the paragraph label on which it is entered.
ZO	Zoom Out. Performs the ZOOMOUT primary command for the paragraph label on which it is entered.

View - Tree Pop-up

The View - Tree pop-up, shown in Figure 39, is a textual representation of the execution sequence of a COBOL program. This pop-up is the same as the Tree View screen (see Figure 38 on page 80), but has a shortened action bar.

Figure 39 • View - Tree Pop-up

```
  View  Search  List  Options  Help
-------------------------------------------------------------------------------
                                View - Tree                   LINE 1 OF 21
Command ===> _____________________________________________ Scroll ===> CSR

****** *************************** TOP OF DATA ******************************
000000  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALAN
000002        INITIALIZE-PGM.
000003           OPEN-FILES.
000003           GET-NUM-OF-DAYS.
000004              SUM-TOTAL-DAYS.
000002        ACCT-MAINTENANCE.
000003           ACCT-MAINTENANCE-EXIT.
000003           INITIATE-ACTION.
000003           ACCT-MAINTENANCE-EXIT.                          REPEATED
000003           UPDATE-ACCT.
000004              INIT-ACTION-RECORD.
000004              INITIATE-ACTION.                             REPEATED
000004              INITIATE-ACTION.                             REPEATED
000004              INITIATE-ACTION.                             REPEATED
                                                                 REPEATED
 ASG1582I 4 OF 4 LEVELS DISPLAYED IN THE VIEW.
```

When a View screen displays by any means other than the View pull-down (e.g., entering a command, or specifying a Selection Screen on an Extract pop-up), the View screen presents as a pop-up with a shortened action bar. (See Chapter 5, "Extract Menu," on page 113 for more information.)

View - Structure View Request Pop-up

The View - Structure View Request pop-up, shown in Figure 40, is used to format the display of the program to be shown on the Structure View screen (see Figure 41 on page 87).

To display the View - Structure View Request pop-up, follow this step:

- Select View ▸ Structure.

Note:

Structure View is also available in the Extract facility, (see Chapter 5, "Extract Menu," on page 113 for more information).

Figure 40 • View - Structure View Request Pop-up

```
                    View - Structure View Request

Type request options.  Then press Enter.  For a name selection list
for a range (other than 1 or 2), type a pattern (e.g. ABC*) in the
name field.

Levels 2    of    3    of depth

Range
1   1.  Procedure division
    2.  All entry points
    3.  Perform range name
    4.  Entry point name
    5.  Nested program name

Name ____________________________________________________________

Options
_  Repetition
```

When the View - Structure View Request pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

To change the level depth

1. Enter the configuration parameters one the View - Structure View Request pop-up and press Enter to display the Structure View screen.

2. Enter the LEVELS primary command.

   Or

   Select View ▸ Levels.

Fields

Field	Description
Levels	Optional. Specifies the number of hierarchical levels of COBOL source statements that display. Change the level depth value to increase or decrease the amount of detail displayed. The amount of detail increases as the level depth increases. The total number of levels in the program displays to the right of this field. Enter a number equal to or less than this number, or enter the value MAX. The default is 2. When Levels is MAX, the actual number of levels displayed depends on the Repetition Option (see Figure 40 on page 85). See the "Usage Notes" on page 87 for more information.
Range	Optional. Specifies how much of the program displays. These are the available ranges: 1. Procedure Division - The entire PROCEDURE DIVISION displays. This is the default. 2. All entry points - The entire PROCEDURE DIVISION and all entry points display. 3. Perform range name - One perform range displays. If this range is selected, the perform range name must be entered in the Name field. 4. Entry point name - One entry point displays. If this range is selected, the entry point name must be entered in the Name field. 5. Nested program name - One subprogram displays. If this range is selected, the subprogram name must be entered in the Name field.
Name	This field is used if Range 3, 4 or 5 is selected. It is ignored if Range 1 or 2 is selected. Enter the name of the perform range, entry point, or nested subprogram to display. If this field is left blank, a selection list pop-up displays to allow selection of the desired target name. Wildcard characters (a question mark (?) for 1 character; an asterisk (*) for zero or more characters) can be entered in this field to reduce the size of the selection list. Specify the Name by entering the S line command next to the desired name on the selection list pop-up.
Repetition	Optional. Specifies whether perform range structures display entirely wherever they appear in the program. If not specified, the complete perform range structure displays the first time it displays in the program. Subsequent references to the perform range are represented by a box containing the notation REPEAT. If specified, the perform range structure displays fully whenever it occurs. The default is display the perform range only the first time it displays.

Usage Notes

When MAX level depth is requested, the actual number of levels displayed may be different, depending on the value in the Repetition field. MAX displays all levels in the program; however, if the maximum level consists of only repeated perform ranges, then the level display is truncated to the next higher level which does not consist exclusively of repeated perform ranges.

Structure View Screen

The Structure View screen, as shown in Figure 41, is a graphical representation of the PERFORM structure of a COBOL program.

To display the Structure View screen, follow this step:

- Press Enter on the View - Structure View Request pop-up (see Figure 40 on page 85).

Figure 41 • Structure View Screen

```
File  View  Extract  Generate  Search  Logic  List  Options  Help
-------------------------------------------------------------------------------
                              Structure View                Program: VIARDEMO
Command ===> ____________________________________________ Scroll ===> CSR
ASG1582I 3 OF 3 LEVELS DISPLAYED IN THE VIEW.
                                                              More:   +   >
+---- 001+
|VIARDEMO|
|        |
|        |
+--------+
    |---------------------|-----------------------------------------------------
+---- 002+            +---- 003+
|INITIALI|            |ACCT-MAI|
|ZE-PGM  |            |NTENANCE|
|        |            | THRU A*|
+--------+            +--------+
    |----------|          |----------|----------------------------------------
+---- 006+ +---- 007+ +---- 008+ +---- 009+
|OPEN-FIL| |GET-NUM-| |INITIATE| |UPDATE-A|
|ES      | |OF-DAYS | |-ACTION | |CCT THRU|
|        | |        | |        | | UPDATE*|
+--------+ +--------+ +--------+ +--------+
               |                    |----------|----------|----------|--------
```

The Structure View screen is also available in the Extract facility. When you enter the Structure View screen from the Extract facility it displays as the View - Perform Structure pop-up (see Figure 42 on page 90) with a shortened action bar. (See Chapter 5, "Extract Menu," on page 113 for more information.)

The Structure View screen displays the logical program units and all CALLing and CALLed relationships between the units. A logical program unit is defined as a perform range, including GO TO code, or a CALLed program.

Logical program units are represented by boxes, and the relationships by lines between the boxes. Encore assigns a unique number to each box, which displays in the upper right corner of the box. The name of the unit displays in the box. If the name does not fit inside the box, the name is truncated and remaining characters are represented by an asterisk (*).

If the box contains `REPEATED`, the box number of the first occurrence, which displays the entire perform range structure, is contained inside the box in the lower right corner.

When this screen first displays, a long message indicates the number of levels displayed, as well as the total number of levels in the program.

When a program is first opened, it can initially display in Structure View by specifying Structure View as the default view on the Options - Product Parameters pop-up. (See Figure 141 on page 296 for more information.)

Structure View produces a virtual page chart. The content and shape of the Structure View are dependent on the COBOL program being viewed.

Usage Notes

Scrolling commands (UP, DOWN, LEFT, and RIGHT) allow you to view the whole structure chart picture. When the screen is scrollable, the More field displays in the upper right corner of the screen below the SCROLL field. The More field lists the available scrolling directions for the current window. That is, they indicate that there is additional data that can be viewed by scrolling in the direction(s) indicated. These are the indicators:

- + view may be scrolled DOWN
- - view may be scrolled UP
- > view may be scrolled RIGHT
- < view may be scrolled LEFT

To view the COBOL source code statements of a particular logical program unit box, follow this step:

▶ On the Structure View screen enter the ZOOMIN primary command.

Or

Select View ▶ Zoom in and press Enter.

The Source View screen (see Figure 34 on page 73) displays as a pop-up with a shortened action bar. The COBOL source statements in the requested perform range are highlighted, and all other lines are excluded from the display.

See Chapter 11, "Commands," on page 321 for more information about the ZOOMIN primary command. See the View pull-down (Figure 33 on page 71) for more information about the Zoom in action on the View pull-down.

Structure View Primary Commands

These are the Structure View primary commands that are used to format the display, view additional data, search or locate strings, search data items, or search lines. Primary commands are entered in the command input area on the fourth line of the screen (see Figure 41 on page 87).

Command	Results
FIND	Positions the cursor at the first box containing an occurrence of the target and highlights all boxes containing occurrences of the target. The short message indicates the number of occurrences found.
FINDXTND	Positions the cursor at the first box containing an occurrence of the target and highlights all boxes containing occurrences of the target. The short or long message indicates the number of occurrences found.
LEVELS `n`	Redisplays the Structure View screen to the specified level. If there are levels less than `n`, they are expanded to the specified level. If there are levels greater than `n`, they are removed. The LEVELS command works globally throughout the displayed program.
RESET	Turns off highlighting or terminates message lines.
SCROLL	Positions to the next results occurrence, to a highlighted or nonhighlighted box, or to a specific box number.
ZOOMIN	The ZOOMIN command can be used to view the source code statements that are represented by the perform unit box. Enter the ZOOMIN command in the command input area, place the cursor anywhere in the appropriate box, and press Enter. The COBOL source code of the program displays on the View - Source pop-up. The perform ranges specified to be viewed are highlighted and tagged to the right with PERF RNG. Use the END command to return to the Structure view screen.

View - Perform Structure Pop-up

The View - Perform Structure pop-up, shown in Figure 42, is a graphical representation of the PERFORM structure of a COBOL program. This pop-up is the same as the Structure View screen (see Figure 41 on page 87), but has a shortened action bar.

Figure 42 • View - Perform Structure Pop-up

```
  View  Search  List  Options  Help
--------------------------------------------------------------------------
                         View - Perform Structure      Program: VIARDEMO
Command ===> _____________________________________________ Scroll ===> CSR

                                                            More:   +   >
+---- 001+
|VIARDEMO|
|        |
|        |
+--------+
    |---------------------|-----------------------------------------------
+---- 002+            +---- 003+
|INITIALI|            |ACCT-MAI|
|ZE-PGM  |            |NTENANCE|
|        |            | THRU A*|
+--------+            +--------+
    |----------|          |----------|-------------------------------------
+---- 006+ +---- 007+ +---- 008+ +---- 009+
|OPEN-FIL| |GET-NUM-| |INITIATE| |UPDATE-A|
|ES      | |OF-DAYS | |-ACTION | |CCT THRU|
|        | |        | |        | | UPDATE*|
--------------------------------------------------------------------------
```

When a View screen displays by any means other than the View pull-down (see Figure 33 on page 71), (e.g., entering a command, or specifying a Selection Screen on an Extract pop-up), the View screen is presented as a pop-up with a shortened action bar, rather than a full screen View. (See Chapter 5, "Extract Menu," on page 113 for more information.)

See the Structure View screen (Figure 41 on page 87) for more information.

View - Select Logic Segment Pop-up

The View - Select Logic Segment pop-up, shown in Figure 43, is used to select the Logic Segment to view.

To display the View - Select Logic Segment pop-up, follow this step:

- Select View ▸ Logic segment and press Enter.

Figure 43 • View - Select Logic Segment Pop-up

```
                          View - Select Logic Segment       4 LOGIC SEGMENTS
Command ===> _                                             Scroll ===> CSR

   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
   VIARDEMO   PERFORM-RANGE-0001             Perform Range
_  VIARDEMO   PERFORM-RANGE-0002             Perform Range
_  VIARDEMO   PERFORM-RANGE-0005             Perform Range
S  VIARDEMO   PERFORM-RANGE-0006             Perform Range
****************************** BOTTOM OF DATA ******************************
```

After the Logic Segment is selected, the display refreshes to contain only the statements defined for the Logic Segment. If you select Logic Segment while on another view, the display transfers to the Source View screen (see Figure 34 on page 73).

To include the specified line into the active Logic Segment, follow this step:

- On the Source View screen, use the I (Include) line command.

Note:

This line command is valid only when the Logic Segment type is Report or Computation Variable.

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select the Logic Segment to view.
Program	The name of the program containing the Logic Segment definition.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up. (See Chapter 5, "Extract Menu," on page 113 for more information.)
Description	The description of the extract that was used to create the Logic Segment.

View - Program Summary Pop-up

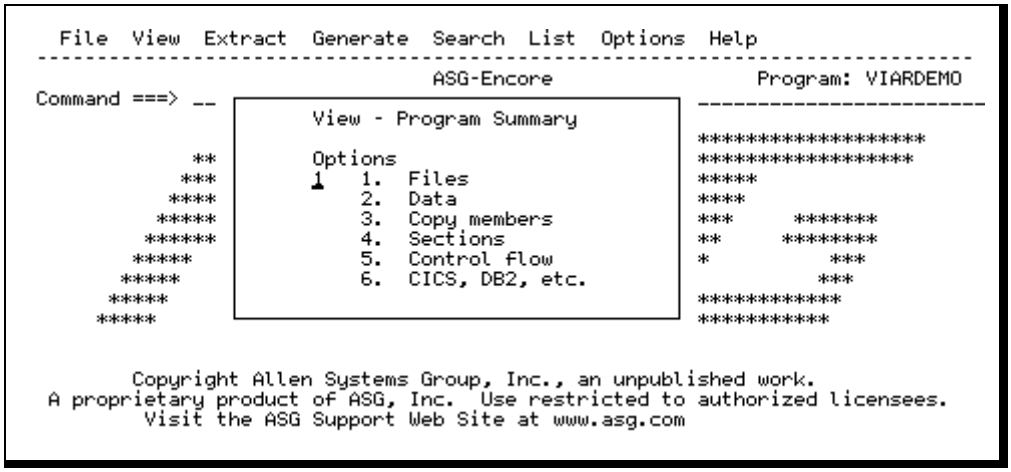
The View - Program Summary pop-up, shown in Figure 44, is used to select the type of summary information to view.

To display the View - Program Summary pop-up, follow this step:

- Select View ▸ Program summary and press Enter.

Each pop-up selected on this pop-up is a collection of facts about a particular type of entity in a program, such as a file, data item, perform range, or CALLed subprogram. Viewing the program summary information can save time, facilitate decision making, and increase the quality of re-engineered programs.

Figure 44 • View - Program Summary Pop-up



Fields

Field	Description
Options	Optional. Enter the number corresponding to the type of summary information to be viewed. The default is 1, Files. 1. Files - Displays the Program Summary - Files pop-up (see Figure 45 on page 94), which shows counts of file usage and statements. 2. Data - Displays the Program Summary - Data pop-up (see Figure 46 on page 96), which shows the number of statements that include the various types of uses of data items. 3. Copy members - Displays the Program Summary - Copy Members pop-up (see Figure 47 on page 98), indicating the number of lines copied for all copy members. 4. Sections - Displays the Program Summary - Sections/Paragraphs pop-up (see Figure 48 on page 99), which shows the number of lines in the various program sections and paragraphs. 5. Control flow - Displays the Program Summary - Control Flow pop-up (see Figure 49 on page 101), indicating the number of each type of control flow statement. 6. CICS, DB2, etc. - Displays the Program Summary - Preprocessor Statements pop-up (see Figure 50 on page 102) indicating the number of data base statements, including SQL, IDMS, CICS and DL/I.

Program Summary - Files Pop-up

The Program Summary - Files pop-up, shown in Figure 45, displays counts of file usage and statements.

To display the Program Summary - Files pop-up, follow this step:

- Select 1. File on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Figure 45 • Program Summary - Files Pop-up

```
-              Program Summary - Files

                          FD     FD     FD     SD
                          Seq    Rel    Indx   (Seq)
                         -----  -----  -----  -----
 Usage:
   Input only . . . :      1      0      0      0
   Output only  . . :      3      0      0      0
   Input/Output . . :      0      0      1      0
 Total File Count . :      4      0      1      0

 Input:
   Statements . . . :      1      0      1      0
   Paragraphs . . . :      1      0      1      0
 Output:
   Statements . . . :      8      0      1      0
   Paragraphs . . . :      3      0      1      0
```

Fields

Field	Description
FD Seq	File Descriptions (FDs) defined with ORGANIZATION IS SEQUENTIAL in the FILE-CONTROL paragraph.
FD Rel	FDs defined with ORGANIZATION IS RELATIVE in the FILE-CONTROL paragraph.
FD Indx	FDs defined with ORGANIZATION IS INDEXED in the FILE-CONTROL paragraph.
SD (Seq)	Sort File Descriptions (SDs), which are categorized by Encore as sequential files.
Usage	• Input only - The number of FD and SD definitions that are referenced only in input (e.g., READ) statements. • Output only - The number of FD and SD definitions that are referenced only in output (e.g., WRITE) statements. • Input/Output - The number of FD and SD definitions that are referenced in both input (e.g., READ) and output (e.g., WRITE, REWRITE) statements.

Field	Description
Total File Count	The number of FD and SD definitions in the FILE SECTION.
Input	• Statements - The number of input statements that reference an FD or SD definition. • Paragraphs - The number of paragraphs containing input statements that reference an FD or SD definition.
Output	• Statements - The number of output statements that reference an FD or SD definition. • Paragraphs - The number of paragraphs containing output statements that reference an FD or SD definition.

Program Summary - Data Pop-up

The Program Summary - Data pop-up, shown in Figure 46, displays the number of statements that include the various types of data items contained in the program.

To display the Program Summary - Data pop-up, follow this step:

- Select 2. Data on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Figure 46 • Program Summary - Data Pop-up

```
-                    Program Summary - Data

                        01/77 Subitem      88    COMP DISPLAY
                       ------ ------  ------  ------ ------
  Total Count . . . :     36     91       6      16     94
Usage:
  Redefines . . . . :      1      0       0       0      0
  Modified, not used       4     49       0       1     48
  Used, not modified       5     14       1       0     13
  Indirect ref. only       4     17       5       0     17
  Dead  . . . . . . :      0      2       0       0      2
  Input statements  :      3      1       0       0      0
  Output statements :     10      1       0       0      3
Role:
  SQL host variable :      0      0       0       0      0
  Linkage . . . . . :      3      2       0       4      0
  CALL ... Using  . :      0      0       0       0      0
  in FD/RD/SD . . . :      4     32       0       1     28
```

Fields

Field	Description
01/77	Data variables defined as 01 or 77 level data items.
Subitem	Data variables defined as subordinate to 01 level data items, with the exception of 88 level data items that are counted separately.
88	Data variables defined as 88 level data items.
COMP	Data items defined with USAGE IS COMP.
DISPLAY	Data items defined with USAGE IS DISPLAY.
Total Count	The total number of data items defined as 01 or 77 level, Subitems, 88 level, COMP usage and DISPLAY usage.

Field	Description
Usage	• Redefines - The number of data items defined with a REDEFINES clause. • Modified, not used - The number of data items that are modified but not used. • Used, not modified - The number of data items that are used but not modified. • Indirect ref. only - The number of data items that are referenced (i.e, used or modified) through an alias, (e.g., a RENAME or REDEFINE). • Dead - The number of data items that are not used or modified, directly or indirectly, in the program. • Input statements - The number of data items referenced in input statements. • Output statements - The number of data items referenced in output statements.
Role	• SQL host variable - The number of data items referenced in EXEC SQL statements as host COBOL variables. • Linkage - The number of data items appearing in the LINKAGE SECTION. • CALL ... Using - The number of data items referenced in the USING clause of CALL statements. • in FD/RD/SD - The number of data items associated with an FD, RD, or SD definition, (i.e., the number of defining input/output buffers).

Program Summary - Copy Members Pop-up

The Program Summary - Copy Members pop-up, shown in Figure 47, displays information about all COPY statements in the program, including Librarian -INC members, Panvalet ++INCLUDE members, and SQL INCLUDEs.

To display the Program Summary - Copy Members pop-up, follow this step:

- Select 3. Copy members on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Figure 47 • Program Summary - Copy Members Pop-up

```
                        Program Summary - Copy Members
Command ===> _____________________________________________ Scroll ===> CSR

Member Name             Lines   Contents
----------------------  ------  ----------------------------------------
VIARCPY1                    27  01(2) OTHER-DATA(17)
****************************** BOTTOM OF DATA ******************************
```

Fields

Field	Description
Member Name	The member name as it displays in the copy statement.
Lines	The number of source lines copied for that member.
Contents	This field describes the overall contents of the copied member. These are some of the descriptions: • IDENTIFICATION DIVISION lines • ENVIRONMENT DIVISION lines • FD, SD, RD and CD definitions • 01, 88 and 77 level items • OTHER-DATA level items, (i.e., 02 - 49 level) • PROCEDURE DIVISION statements • Program elements from multiple DIVISIONs

Program Summary - Sections/Paragraphs Pop-up

The Program Summary - Sections/Paragraphs pop-up, shown in Figure 48, shows the number of lines in the various program sections and paragraphs.

To display the Program Summary - Sections/Paragraphs pop-up, follow this step:

- Select 4. Sections on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Figure 48 • Program Summary - Sections/Paragraphs Pop-up

```
_Program Summary - Sections/Paragraphs

 Location                      Lines
 -------------------------    ------
 Data Division . . . . . . :    193
   File Section  . . . . . :     56
   Working-Storage Section :    127
   Linkage Section . . . . :      8
 Procedure Division  . . . :    210
   Number of SECTIONs  . . :      0
   Number of labels  . . . :     14
      "EXIT only"  . . . . :      2
      GOTO/ALTER targets . :      1
      In-line code . . . . :      0
      PERFORMed  . . . . . :     14
      Deadcode . . . . . . :      0
   Number of statements  . :    109
      Within PERFORMs  . . :    102
      Deadcode . . . . . . :      0
 Total . . . . . . . . . . :    434
```

Fields

Field	Description
Data Division	The total number of source lines, including comments, for each of these sections: • File Section • Working-Storage Section • Linkage Section

Field	Description
Procedure Division	The total number of PROCEDURE DIVISION source lines, including comments. Subtotals are included for these: • Number of SECTIONs - The number of SECTIONs in the PROCEDURE DIVISION. • Number of labels - The number of labels in the PROCEDURE DIVISION. Labels include these: — EXIT only - The number of labels containing only an EXIT statement. — GOTO/ALTER targets - The number of labels that are targets of GO TO or ALTER statements. — In-line code - The number of labels that are not reached using PERFORMs, (i.e., they are executed as a result of fall through code). — PERFORMed - The number of labels that are reached using PERFORMs. — Deadcode - The number of labels containing dead code, (i.e., code that is never reachable). • Number of statements - The number of statements (not source lines) in the PROCEDURE DIVISION. Statements include these: — Within PERFORMs - The number of statements executed by PERFORM statements. — Deadcode - The number of statements that are unreachable.
Total	The number of source lines in the program, including comment lines.

Program Summary - Control Flow Pop-up

The Program Summary - Control Flow pop-up, shown in Figure 49, shows the number of each type of control flow statement in the program.

To display the Program Summary - Control Flow pop-up, follow this step:

- Select 5. Control flow on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Figure 49 • Program Summary - Control Flow Pop-up

```
-          Program Summary - Control Flow

                                                  Count
                                                 ------
 External CALL statements  . . . . . . . . . :      0
    Programs CALLed  . . . . . . . . . . . . :      0
 PERFORM statements  . . . . . . . . . . . . :     16
    Number of PERFORM ranges . . . . . . . . :     12
       Ranges PERFORMed more than once . . . :      1
 GO TO statements  . . . . . . . . . . . . . :      2
    Number of labels reached . . . . . . . . :      1
       Labels reached more than once . . . . :      1
    Number of ALTERed GO TOs . . . . . . . . :      0
 Multi-way branches  . . . . . . . . . . . . :      0
    GO TO...DEPENDING ON...  . . . . . . . . :      0
    EVALUATE...  . . . . . . . . . . . . . . :      0
 ON CONDITION statements . . . . . . . . . . :      2
    End-of-File (AT END...)  . . . . . . . . :      1
    Other Input/Output conditions  . . . . . :      1
    Arithmetic conditions (ON OVERFLOW...) :      0
```

Fields

Field	Description
External CALL statements	The number of CALL statements in the PROCEDURE DIVISION. A subtotal of the number of unique programs called is included.
PERFORM statements	The number of PERFORM statements in the program. Subtotals are included for the number of unique perform range targets, and the number of ranges performed multiple times.
GO TO statements	The number of GO TO statements in the program. Subtotals are included for the number of unique labels that are targets of GO TOs, the number of labels that can be reached from multiple GO TOs, and the number of GO TOs that are the targets of ALTER GOTO statements.

Field	Description
Multi-way branches	The number of GO TO DEPENDING ON and EVALUATE statements in the program. Subtotals are included for the number of GO TO DEPENDING ON statements, and the number of EVALUATE statements.
ON CONDITION statements	The number of statements containing exception condition clauses, including SIZE, OVERFLOW, INVALID KEY, NO DATA, END OF PAGE, and END OF FILE. Subtotals are included for the number of AT END condition statements, the number of other I/O exceptions (e.g., INVALID KEY), and the number of arithmetic exceptions (e.g., ON OVERFLOW, ON SIZE ERROR, etc.).

Program Summary - Preprocessor Statements Pop-up

The Program Summary - Preprocessor Statements pop-up, shown in Figure 50, displays the number of data base statements in the program, including SQL, IDMS, CICS, and DL/I.

To display the Program Summary - Preprocessor Statements pop-up, follow this step:

- Select 6. CICS, DB2, etc. on the View - Program Summary pop-up (see Figure 44 on page 92) and press Enter.

Note:

The Program Summary - Preprocessor Statements pop-up only displays if the current program contains data base statements.

Figure 50 • Program Summary - Preprocessor Statements Pop-up

```
_   Program Summary - Preprocessor Statements

                          SQL   IDMS   CICS   DL/I
                         -----  -----  -----  -----
 Usage:
   Input only . . . :      0      0    N/A    N/A
   Output only  . . :      0      0    N/A    N/A
   Input/Output . . :      0      0    N/A    N/A
 Total Count  . . . :      0      0    N/A    N/A

 Input:
   Statements . . . :      0      0      4      0
   Paragraphs . . . :      0      0      4      0
 Output:
   Statements . . . :      0      0      6      0
   Paragraphs . . . :      0      0      6      0
```

Fields

Field	Description
SQL	Tables and Views referenced in EXEC SQL statements.
IDMS	IDMS Record Names referenced in IDMS COBOL statements.
CICS	EXEC CICS statements.
DL/I	EXEC DL/I statements, including CALL 'CBLTDLI' statements.
Usage	• Input only -The number of data base definitions referenced only by statements that query the data base. • Output only - The number of data base definitions referenced only by statements that update the data base. • Input/Output - The number of data base definitions referenced by statements that query or update the data base.
Total Count	The number of data base definitions.
Input	• Statements - The number of query-type statements that reference a data base definition. • Paragraphs - The number of paragraphs containing query-type statements that reference a data base definition.
Output	• Statements - The number of update-type statements that reference a data base definition. • Paragraphs - The number of paragraphs containing update-type statements that reference a data base definition.

View - Paragraph Cross-Reference Request Pop-up

The View - Paragraph Cross-Reference Request pop-up, shown in Figure 51, is used to select a target to be viewed on the View - Paragraph Cross Reference pop-up. To display the View - Paragraph Cross-Reference Request pop-up, follow this step:

- Select View ▸ Paragraph X-ref (see Figure 33 on page 71) and press Enter.

Figure 51 • View - Paragraph Cross-Reference Request Pop-up

```
                    View - Paragraph Cross-Reference Request

To list transfers of control to or from paragraphs in the program,
type a target name and select a target type and direction.  Then
press Enter.  For a name selection list for a target type (other
than types 1 and 5), type a pattern (e.g. ABC*) in the target name
input field.

Target name  ------------------------------------------------------

Target type                     Direction
__  1.  None - use cursor       1   1.  Previous
    2.  Perfrange name              2.  Next
    3.  Program name
    4.  Subset name
    5.  Line range
    6.  Label name
```

Fields

Field	Description
Target name	Optional. Enter the perform range name, program name, subset name, line range, or label name that is used as a basis for the Paragraph Cross Reference. If you want to use cursor positioning to select the target, make no entry in this field. The entry in the Target name field must correspond to the entry in the Target type field. Read the Target type field description for valid entries in the Target name field. If this field is left blank, and a Target type other than None or Line range is chosen, a selection list pop-up displays to allow selection of the desired target name. Wildcard characters (a question mark (?) for 1 character; an asterisk (*) for zero or more characters) can be entered in this field to reduce the size of the selection list.

Field	Description
Target type	Optional. Select the type of target represented by the entry in the Target name field. 1. None - use cursor - The target is specified by positioning the cursor on the desired target. When this Target type is selected, the previous View screen redisplays to allow cursor placement. 2. Perfrange name - Specifies that the entry in the Target name field is a perform range name. All paragraphs containing the perform range display on the View - Paragraph Cross Reference pop-up. 3. Program name - Specifies that the entry in the Target name field is a program name. All paragraph names containing the program display on the View - Paragraph Cross Reference pop-up. 4. Subset name - Specifies that the entry in the Target name field is a subset. For descriptions of the subsets, select List ▸ Subsets. 5. Line range - Specifies that the entry in the Target name field is a single line number or range of lines. Line numbers display in columns 1 - 6 of Source View. All paragraph names containing the specified line(s) display on the View - Paragraph Cross Reference pop-up. 6. Label name - Specifies that the entry in the Target name field is a label, (i.e., any PROCEDURE DIVISION paragraph or section name, or the literals PROCEDURE and PROC). All paragraph names containing the specified label display on the View - Paragraph Cross Reference pop-up.
Direction	Optional. Specify the type of transfer. The default is 1, Previous. 1. Previous - Displays paragraphs from which control is transferred to the target paragraph. 2. Next - Displays paragraphs to which control is transferred from the target paragraph.

View - Paragraph Cross Reference Pop-up

The View - Paragraph Cross Reference pop-up, shown in Figure 52, shows how control is transferred to or from the target paragraphs. It answers these questions:

- How do I get to this paragraph?
- What are all the paragraphs that transfer control to this paragraph?
- What are all the paragraphs to which this paragraph transfers control and how?

This pop-up displays when Enter is pressed on the View - Paragraph Cross-Reference Request pop-up, or by entering the PREF command on any screen. The LPRINT * command can be entered on this pop-up to copy the data to the List file.

Figure 52 • View - Paragraph Cross Reference Pop-up

```
                        View - Paragraph Cross Reference
Command ===> _____________________________________________ Scroll ===> CSR

Target(s): PROCEDURE DIVISION
Direction: PREVIOUS

   A : Add to target       P : Execute PREF       S : Select for viewing

S   Comes from                   How            Target paragraph(s)
- ---------------------------- -------------- ----------------------------
_                              ENTRY          PROCEDURE DIVISION
******************************** BOTTOM OF DATA ******************************
```

Fields

Field	Description
Target(s)	The paragraph name(s) of the requested target.
Direction	The type of transfer. PREVIOUS highlights paragraphs from which control is transferred. NEXT highlights paragraphs to which control is transferred. The default is PREVIOUS.
Line command area	The line command area, to the left of the Comes from or Target paragraph(s) field, accepts these entries: • A - Add the selected paragraph to the target paragraph(s). If the direction is PREVIOUS, the target paragraph is added to the top of the list. If the target is NEXT, the target paragraph is added to the end of the list. • P - Execute another paragraph cross-reference. • S - Return to the source code with the first line of the paragraph displayed at the top of the screen.

Field	Description
Comes from	This field displays when the direction is PREVIOUS. It lists the paragraph names that transfer control to the target paragraph(s). All paragraphs that relate to the target paragraph are grouped together.
Goes to	This field displays when the direction is NEXT. It lists the paragraph names that receive control from the target paragraph(s).
How	Indicates the way in which control is transferred from or to the target paragraph. If the target is reached through more than one occurrence of the same transfer method, the number of occurrences, in parentheses, follows the method of transfer.This example means that the target was reached through 2 PERFORMs and 3 GOTOs: `PERF(2), GO(3)` **Note:** For a complete list of paragraph control transfer methods and abbreviations, see Appendix C "Paragraph Control Transfer Methods" on page 525 for more information.
Target Paragraph(s)	The paragraphs requested as the target in the PREF command. Ditto marks (") are shown when multiple paragraphs transfer control to the same paragraph.

View - Levels Request Pop-up

The View - Levels Request pop-up is used to redisplay the current Tree View (see Figure 38 on page 80) or Structure View screen (see Figure 41 on page 87) to the specified level.

To display the View - Levels Request pop-up, follow this step:

- Select View ▸ Levels (see Figure 33 on page 71) and press Enter.

Figure 53 • View - Levels Request Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-----------                                                  ------------
                 ┌──────────────────────────────────────────┐  LINE 1 OF 6
Command ===      │              View - Levels Request        │  ll ===> CSR
                 │ To re-build a Tree View or Structure View with a
****** ****      │ different level of program hierarchy, type a
000000  PRO      │ number or MAX.  Then press Enter.         │  ************
000002 +         │                                           │  E,
000002 +         │        Levels 2    of    4     of depth   │
000002           └──────────────────────────────────────────┘
000002
000002 +         CLOSE-PGM.
****** ************************* BOTTOM OF DATA ********************************
```

Field

Field	Description
Levels	Optional. Specify the number of hierarchical levels to display. Change the level depth value to increase or decrease the amount of detail displayed. The amount of detail increases as the level depth increases. The total number of levels in the program displays to the right of this field. Enter a number equal to or less than this number, or enter the value MAX. The default is 2.

View - Reset Request Pop-up

The View - Reset Request pop-up, shown in Figure 54, is used to remove highlighting and tags, redisplay all excluded lines, cancel pending line commands, and terminate message lines.

To display the View - Reset Request pop-up, follow this step:

- Select View ▸ Reset and press Enter.

Figure 54 • View - Reset Request Pop-up

```
                View - Reset Request

To reset some or all program display features,
select an option.  Then press Enter.

                Options
                ✓  All
                _  Highlights
                _  Tags
                _  Excluded lines
                _  Line commands
                _  Messages
```

Fields

Field	Description
All	Optional. If specified, all conditions indicated by each of the other fields on this pop-up are reset. If not specified, only those options selected are reset. The default is to reset all conditions.
Highlights	Optional. If specified, any line that has been highlighted is returned to nonhighlighted. If not specified, highlighted lines remain highlighted unless the All option is specified. The default is lines remain highlighted.
Tags	Optional. If specified, tags in columns 73-80 are erased. If not specified, tags remain unless the All option is specified. The default is lines remain tagged.
Excluded lines	Optional. If specified, excluded lines redisplay. If not specified, excluded lines remain excluded unless the All option is specified. The default is lines remain excluded.

Field	Description
Line commands	Optional. If specified, pending line commands are canceled. If not specified, pending line commands remain on the screen unless the All option is specified. The default is pending line commands remain displayed.
Messages	Optional. If specified, short or long messages are removed from the display. If not specified, messages remain displayed on the screen unless the All option is specified. The default is messages remain displayed.

View - Exclude Request Pop-up

The View - Exclude Request pop-up, shown in Figure 55, is used to specify the lines to be excluded from the screen display.

To display the View - Exclude Request pop-up, follow this step:

- Select View ▸ Exclude and press Enter.

Excluded lines are represented by a line of dashes and text stating:

```
n LINE(S) NOT DISPLAYED
```

The Exclude action is valid on Source View (see Figure 34 on page 73) and Tree View (see Figure 38 on page 80).

Figure 55 • View - Exclude Request Pop-up

```
                    View - Exclude Request

To exclude specific lines from the program being
viewed, select the desired option.  Then press
Enter.  For the "string" option, type the string
to be excluded.

Options
1   1.  All lines
    2.  All highlighted lines
    3.  All non-highlighted lines
    4.  All lines containing a string

String ----------------------------------------
```

Fields

Field	Description
Options	Optional. Specify which lines to exclude from the screen display. The default is 1, All lines. 1. All lines - Excludes all lines from the screen display. 2. All highlighted lines - Excludes all lines that have been highlighted as the result of one or more commands from the screen display. 3. All non-highlighted lines - Excludes all non-highlighted lines from the screen display. 4. All lines containing a string - Excludes all lines containing the string specified in the String field. If this Option is selected, the String field is required.
String	Required if Option 4 is selected. Enter a string of alphanumeric or DBCS characters. This field is ignored if any other Option is selected.

5 Extract Menu

This chapter describes Encore's Extract pull-down menu functions and contains these sections:

Section	Page
Extract Overview	114
Perform Range Extract	116
Report Extract	116
Computation Variable Extract	117
Transaction Extract	118
Statement Extract	118
Complement Extract	119
Common Code Extract	119
Test Driven Extract	120
CICS Server Extract	121
Extract Objective	122
Extract Pull-down	124
Extract - Name Logic Segment Pop-up	126
Perform Range Extract	127
Report Extract	137
Computation Variable Extract	147
Transaction Extract	157

Section	Page
Statement Extract	171
Complement Extract	178
Common Code Extract	182
Test Driven Extract	184
CICS Server Extract	188

Extract Overview

Extraction is the process of identifying and selecting logically connected pieces of code within a program.

Encore's primary input is COBOL source programs. These programs must exist on the Application Knowledge Repository (AKR), and must have been Analyzed using a standard ESW Analyze method. The Extract facility is initiated through a task-oriented user interface that leads you by prompting for information and allowing you to select from various extract objective options.

The Extract facility identifies a piece of a COBOL program that is referred to in Encore as a Logic Segment. The process of identifying a Logic Segment is called extract. A Logic Segment typically defines a specific function performed by the program.

Note:

The Extract facility does not support extracting from COBOL II Release 3 modules containing subprograms.

To extract a Logic Segment from a program

1. Select a program using the File action.
2. Select Extract on the action bar to specify the type of extract to be performed. The extract process creates the Logic Segment.

The extract objective is to select a type of extract and perform the steps necessary to accomplish the extract. The type of extract that should be selected depends on the particular goal to be reached. The extract types, and the reasons for selecting each type, are explained in the sections in this chapter.

The result of each task-oriented objective is a set of Logic Segments. Once a Logic Segment is identified, these options are available to determine the Logic Segment disposition:

Objective	Description
Module Generation	You can request that a semantically and syntactically correct callable module be created for any Logic Segment isolated by an extract objective. Encore determines the required supporting DATA DIVISION, generates the necessary COBOL boilerplate sections, creates a USING parameter list, and includes the identified procedural code of the Logic Segment. Encore interfaces with the ISPF Editor to enable you to save the created module. After saving the module under a particular source manager, you can manage the location and contents of the module.
CASE Tool Interface	You can create a member that contains the set of statements of the Logic Segment. One use for this output might be to export or feed source code to a CASE tool.
Repository Knowledge	You can assign a name to the Logic Segment. Along with the name, free-form text can be entered that describes the purpose and scope of the Logic Segment.

When a Logic Segment is saved, it is associated with a program and becomes permanent information within the AKR. In addition to the name and description, the AKR contains detail data identifying the individual source code statements that make up the Logic Segment. The criteria necessary to regenerate the Logic Segment is retained as part of the Logic. These are the available Logic Segment regeneration/view options:

Item	Description
Segment definition	When an Analyze is performed on a program with defined Logic Segments, the detailed information regarding specific statements is deleted. However, the criteria information is retained, and in most cases is sufficient to regenerate the Logic Segment source statements.
Edit Interface	You may request that an Edit session be initiated. For this request, the Logic Segment definition is made available to the Editor. You can also issue the DISPLAY command, which excludes all of the statements not in the Logic Segment.

Perform Range Extract

Isolates all executable statements of a Perform Range. This includes not only the statements within the range, but also all code that is reachable using a GOTO or PERFORM from within the range. These are the Perform Range extract items and associated actions:

Item	Action
Extract Purpose	Identifies a specific named PEROFRMed range of code within the COBOL program.
Logic Segment Contents	All the code that could be executed by the perform range, (i.e., the Logic Segment contains not only the paragraph range of the PERFORM, but also all subordinate code that is PERFORMed or reachable by a GO TO or other statement).
User Action	Select the desired perform range name.
Applications	Use the perform range extract when a perform range should be removed from the program and then used as a CALLed routine instead of a PERFORMed routine. Also, the perform range extract can be used when the perform range is to be made into a reusable or sharable module.

Report Extract

Isolates all the code that produces a Report. A Report is defined in general terms as a logical collection of output statements. Individual output statements may be selected. These are the Report extract items and associated actions:

Item	Action
Extract Purpose	Identifies all code necessary to produce a set of output (WRITE or CALL) statements.
Logic Segment Contents	The minimum set of statements, from the beginning of the program, necessary to produce a set of user-identified output statements, (i.e., all statements required to produce a specified set of record or print line types are collected).

Item	Action
User Action	Select the report name. Identify the statements referencing the selected file that comprise the report.
Applications	Use the Report extract to separate a program that produces multiple reports into several programs that each produce one report. Also use the Report extract to create a separate program that produces only the report and does not perform any other functions of the original program.

Computation Variable Extract

Isolates executable statements that determine the value of a particular data variable, known as the Computation Variable, at a particular statement in the program. This provides a method to extract the logic that is used to determine the value of a particular Computation Variable, (i.e., the business rule). These are the Computation Variable extract items and associated actions:

Item	Action
Extract Purpose	Identifies all code that calculates a specific data variable, from the beginning of the program to a specific statement in the program.
Logic Segment Contents	The minimum set of statements that contribute to, or influence the value of, the identified data variable, (i.e., the Logic Segment contains only the code from the beginning of the program necessary to calculate the user-identified data variable at a user-identified statement in the program).
User Action	Select the name of the data variable. Identify the point in the program (i.e., the statement) where the calculation of the data variable is complete.
Applications	Use the Computation Variable extract to create a callable module that calculates, edits, derives, or manipulates a specific output data variable.

Transaction Extract

Isolates all the statements required to execute a Transaction. This is accomplished by selecting data variables and conditional statement paths that are executed for the desired transaction code value(s). These are the Transaction extract items and associated actions:

Item	Action
Extract Purpose	Identifies all code required for a specific transaction, (i.e., all code between a start point and multiple end points that is (or could be) executed based on the value(s) of a data variable).
Logic Segment Contents	All statements between the start and end points necessary to execute the path(s) selected as valid for the desired transaction values.
User Action	• Specify the start and end points. The default start point is the PROCEDURE DIVISION, and the default end points are all program EXITs. • Identify the data name(s). • Select the conditional path(s) which would be executed based on the value of the transaction, (i.e., determine whether a conditional statement is TRUE or FALSE based on the desired value).
Applications	Use the Transaction extract to isolate the code that processes several transaction code values, (e.g., ADD, CHANGE, and DELETE).

Statement Extract

Isolates specific executable Statements that you have identified. These are the Statement extract items and associated actions:

Item	Action
Extract Purpose	Identifies a user-selected set of COBOL source statements.
Logic Segment Contents	All statements you have identified.

Item	Action
User Action	Select the desired PROCEDURE DIVISION statement(s).
Applications	Use the Statement extract when the code to be isolated is not logically connected.

Complement Extract

Isolates the statements of the original program, excluding statements of a previous extract. These are the Complement extract items and associated actions:

Item	Action
Extract Purpose	Identifies a set of COBOL source statements that do not include a previous extract.
Logic Segment Contents	All statements from the original program, excluding statements of the previously-extracted Logic Segment, but retaining any statements needed by other functional paths.
User Action	Select the previously-extracted Logic Segment.
Applications	Use the Complement extract when a Logic Segment should be removed from the program.

Common Code Extract

Identifies code that is common to two or more previous extracts. These are the Common Code extract items and associated actions:

Item	Action
Extract Purpose	Identifies the code that is common to two or more previously extracted Logic Segments.
Logic Segment Contents	All statements that appear in a selected Logic Segment, as well as appearing in all other selected Logic Segments.

Item	Action
User Action	Select the previously extracted Logic Segments.
Applications	Use the Common code extract to identify code in a Logic Segment that also displays in one or more other Logic Segments.

Test Driven Extract

Creates a Logic Segment from an input file containing instructions executed during a SmartTest test session. These are the Test Driven extract items and associated actions:

Item	Action
Extract Purpose	Creates a Logic Segment from an input file that was created during a SmartTest test session.
Logic Segment Contents	All statements that were executed during the SmartTest test session.
User Action	Type the name of the input file on the extract criteria pop-up.
Applications	Use the Test Driven extract to understand how an unfamiliar program executes under different input variable values. This is helpful in gaining program understanding. The extract can also be used to identify possible dead code.

CICS Server Extract

Locates all of the necessary statements, files, and data elements required to create a CICS Server extract program. This program is then used to create a COMMAREA-based server program from a CICS pseudo-conversational program. These are the CICS Server extract items and associated actions:

Item	Action
Extract Purpose	Identifies all of the necessary statements, files, and data elements required to create a CICS Server Extract program, which are then used to create a COMMAREA-based server program from a CICS pseudo-conversational program.
Logic Segment Contents	All statements between the SEND and RECEIVE map that modify the data variables used by the SEND and RECEIVE map.
User Action	• Specify the end points of the server, (i.e., the CICS SEND map). • Specify the start points of the server, (i.e., the CICS RECEIVE maps of the SEND). • Specify the server path.
Applications	Use the Server extract to create a COMMAREA-based server program.

Extract Objective

Overview

The pop-ups used for the Perform Range, Computation Variable, Transaction, Report, and Statement objectives are fundamentally the same. Each of these objectives has a list pop-up and a filter pop-up. The list pop-ups identify the names of all selectable extract criteria in the program. The filter pop-up is used to collect the necessary information for Encore to apply as search criteria against the source code. The results of the search are then used to populate the list of selectable extract criteria on the list pop-ups.

Some name fields on the filter pop-ups can be entered as wildcard sequences. For example, when trying to locate the name of a data variable, partial names can be suffixed with the wildcard character(s). Entering a name with an embedded wildcard character displays an alphabetical list of names. The initial display is positioned at the beginning of the list.

In addition to viewing and selecting extract criteria on the list pop-ups, you can also view and select in one of these three view formats:

- Source - Textual display of the COBOL source program code.
- Tree - Textual representation of the execution structure of a COBOL program.
- Structure - Graphical representation of the PERFORM structure of a COBOL program.

These formats can be selected by pressing PF4 on the list pop-ups. In each view, you can select the desired items or gather additional information necessary to identify the desired item. See Chapter 4, "View Menu," on page 69 for more information.

A Complement extract or Common Code extract is based on a previous extract of a Perform Range, Computation Variable, Transaction, Report, or Statement. The item to be extracted is selected from a list of names.

A Test Driven extract is based on a previous SmartTest test session. The Logic Segment is created by processing a file created during the SmartTest test session.

A CICS Server extract is based on all statements between the SEND and RECEIVE map that modify the data variables used by the SEND and RECEIVE map.

To complete an objective

1. Display the available extract criteria in list, source, structure, or tree format. If you are in list format, filter the list of criteria as needed.

2. From the list pop-up or view screen, select an item appropriate to the objective.

3. Repeat steps 1 and 2 until all items have been selected.

4. For Computation Variable and Report extraction, an additional step is required to identify the desired in-context location of the referenced variables within the program. Transaction extract involves selecting a control variable data name and identifying the TRUE/FALSE conditional paths, and the START and END point statements.

5. For Complement and Common Code extraction, select the desired item name(s) from the list on the information pop-up. For a Complement extraction, if a Transaction item type was selected, the TRUE/FALSE conditional paths are then identified.

6. For Test Driven extraction, type the dataset name of the file created from the SmartTest test session on the Test Driven pop-up.

7. For a CICS Server extraction, select the CICS Server Endpoints from the available statements on the Endpoints pop-up and then select the corresponding CICS Server Startpoints from the available statements on the Startpoints pop-up that you want to use for the server extract.

Prior to saving or creating modules from the Logic Segment, the results can be previewed using the View facility. Based on the contents of these displays, you can cancel the objective and start over or request one of the available output options.

Extract Pull-down

The Extract pull-down, as shown in Figure 56, presents the various task-oriented extract options available.

To display the Extract pull-down, follow this step:

- Select Extract on the action bar displays the Extract pull-down.

Figure 56 • Extract Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
 -----------                                  -------------------------------
                 _ 1. Perform range...                   Program: VIARDEMO
 Command ===       2. Report...
                   3. Variable...                 ---------------------------------
                   4. Transaction...              ****************************
                   5. Statement...                ****************************
                   6. Complement...                       *****
                   7. Common Code...                     *****
           *       8. Test Driven...              ***   *****     *******
          **       9. Edit segment criteria...    ***  *****     ********
         ***      10. Server...                   **  *****          ***
        ****                                      *  *****           ***
       *****            *************************  *******************
      *****            *************************   ******************

         Copyright Allen Systems Group, Inc., an unpublished work.
  A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
          Visit the ASG Support Web Site at www.asg.com
```

Actions

Note:

Actions on the Extract pull-down cannot be selected unless there is an active program, [i.e., a program has been opened on the File pull-down (see Figure 9 on page 26)].

Action	Description
1. Perform range	Extract based on a perform range. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the Logic Segment name, then the Extract - Perform Name List pop-up (see Figure 58 on page 128) displays.
2. Report	Extract based on a report. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the Logic Segment name, then the Extract - Report Name List pop-up (see Figure 62 on page 138) displays.

Action	Description
3. Variable	Extract based on computation variable logic. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the Logic Segment name, then the Extract - Data Name List pop-up (see Figure 68 on page 148).
4. Transaction	Extract based on a transaction. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the Logic Segment name, then the Extract -Transaction Paths List pop-up (see Figure 72 on page 159).
5. Statement	Extract based on a user-selected statement. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the Logic Segment name, then the Extract - Source Statement List pop-up (see Figure 77 on page 171).
6. Complement	Extract based on a complement of a previously extracted Logic Segment. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the name for the new Logic Segment, then the Extract - Complement pop-up (see Figure 80 on page 179).
7. Common Code	Extract based on code that is common to two or more Logic Segments. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the name for the new Logic Segment, then the Extract - Common Code pop-up (see Figure 82 on page 183) displays.
8. Test Driven	Extract based on input of List Counts file created by a SmartTest test session. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the name for the new Logic Segment, then the Extract - Test Driven pop-up (see Figure 83 on page 184).
9. Edit segment criteria	Returns you to continue extracting an existing Logic Segment. Selecting this action displays the Extract - Edit Segment pop-up (see Figure 84 on page 187) used to select the Edit segment to extract, then returns you to continue extracting an existing Logic Segment.
10. CICS Server	Extract based on CICS SEND/RECEIVE maps. Selecting this action displays the Extract - Name Logic Segment pop-up (see Figure 57 on page 126) used to specify the name of the new Logic Segment, then the Extract - Server pop-up (see Figure 85 on page 188).

Extract - Name Logic Segment Pop-up

The Extract - Name Logic Segment pop-up, shown in Figure 57, is used to assign a name to the Logic Segment that is extracted.

To display the Extract - Name Logic Segment pop-up, follow this step:

- Select an extract type on the Extract pull-down (see Figure 56 on page 124).

Figure 57 • Extract - Name Logic Segment Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-----------                                              --------------
                 Extract - Name Logic Segment            ram: VIARDEMO
Command ===                                              ---------------
                Type name of logic segment. Then press Enter.
                                                         **********
                Name  . . . PERFORM-RANGE-0001           *********

          **********************  *****************   *****     *******
         **********************   *****************  *****     ********
        *****            *****               ******  *****          ***
       *****            *****               ******  *****          ***
      *****            *************************  ******************
     *****            ************************    *****************

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

Field

Field	Description
Name	Required. A default name displays, based on the type of extract selected, and the number of extracts already performed. The name may be changed if desired. The Logic Segment name may be a maximum of 32 characters, must be unique within the program, and may not contain embedded blanks.

Perform Range Extract

The purpose of the perform range extract objective is to identify the name of a perform range that contains the code that accomplishes a specific function. These are the steps necessary to accomplish a Perform Range extract:

1. Analyze the program, if this has not already been done. The desired program is analyzed by selecting File ▸ Analyze from the File pull-down (see Figure 9 on page 26 or one of the methods described in Chapter 12, "Analyze Functions," on page 431 for more information).

2. Select the perform ranges for extract on the Extract - Perform Name List pop-up (see Figure 58 on page 128). To filter the list of perform ranges in the list, press PF5.

To initiate a perform range extract, follow this step:

- Select Extract ▸ Perform range (see Figure 56 on page 124) and press Enter. This extract mode allows selection of one perform range or multiple perform ranges.

The pop-ups used to complete the perform range extract objective are described in this section in the order that they are used.

Extract - Perform Name List Pop-up

The Extract - Perform Name List pop-up (see Figure 58 on page 128) is used to select the name of the perform range to be extracted. This pop-up displays by selecting Extract ▸ Perform range from the Extract pull-down (see Figure 56 on page 124), then pressing Enter on the Extract - Name Logic Segment pop-up.

When this pop-up initially displays, all perform ranges in the program are listed. To narrow the selection list, press PF5 to display the Extract - Perform Range Filter pop-up (see Figure 59 on page 132). When this pop-up first displays, a short message displays indicating the number of perform ranges available for selection.

To select the desired perform range name, use the SELECT primary command or the S line command. Selected perform range names are highlighted.

Use the SORT command to sort the perform range list by any of the columns on the pop-up, as shown in Figure 58. The Extract - Sort PERFORM Range pop-up (see Figure 61 on page 136) displays so you can specify the sort column.

Note:

The screen may be scrolled left and right to view all the fields. The column heading underscore line contains a less than (<) indicator if the screen can be scrolled to the left and a greater than (>) indicator if the screen can be scrolled to the right.

Figure 58 • Extract - Perform Name List Pop-up

```
                          Extract - Perform Name Li   12 PERFORM RANGE NAMES
Command ===> ____________________________________________ Scroll ===> CSR

Select PERFORM Range name(s) for extract.  Then press Enter.  Press
Enter when complete.  Use SORT command to order column by value.
  S - Select          V - View Source          U - Unselect

                                     Size       Times PERF Direct Subord
  PERFORM Range Name               Lines   %     PERF   by  PERFs  PERFs
- ------------------------------ <----- -----   ----  ---- ----   ----   >
_ ACCT-MAINTENANCE THRU ACCT-MAI     27 12.86      1     1    3      4
_ CLOSE-FILES                         7  3.33      1     1    0      0
_ CLOSE-PGM                           5  2.38      1     1    1      1
_ EXCEPTION-REPORT-1                 16  7.62      1     1    0      0
_ GET-NUM-OF-DAYS                    10  4.76      1     1    1      1
_ INIT-ACTION-RECORD                  9  4.29      1     1    0      0
_ INITIALIZE-PGM                     21 10.00      1     1    2      3
_ INITIATE-ACTION                    13  6.19      5     2    0      0
_ OPEN-FILES                          5  2.38      1     1    0      0
_ REPORT-FINAL-CNTRS                  5  2.38      1     1    0      0

    PF4=View  PF5=Filter  PF17=Sort
```

Fields.

Field	Description
Line command area	The line command area, to the left of the PERFORM Range Name field, accepts these line commands: • S - Selects the perform range and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements in the specified perform range are highlighted, and tagged to the right with PERF RNG. Any perform range can be selected on the View - Source pop-up with the SELECT command. • U - Unselects a previously-selected perform range name. The name is no longer highlighted.
PERFORM Range Name	Lists the perform ranges in the program. To filter the list, press PF5 to display the Extract - Perform Range Filter pop-up to enter the filter criteria (see Figure 59 on page 132).

Field	Description
Size Lines	The number of source lines, including comments, in the perform range.
Size%	The number of source lines in the perform range as a percentage of the total number of lines in the program.
Times PERF	The number of PERFORM statements that reference the perform range.
PERF by	The number of unique statements PERFORMing the perform range.
Direct PERFs	The number of perform ranges that are PERFORMed by the listed perform range.
Subord PERFs	The number of subordinate PERFORM units that could be invoked when the listed perform range is PERFORMed.
Extern CALLs	The number of external CALLs from the perform range to unique program names.
Data Mod	The number of data items that are modified, but not used, within the perform range.
Data Use	The number of data items that are used, but not modified, within the perform range.
Data Ref	The number of data items that are modified or used within the perform range.
Parms (01/77)	The number of data items that would be present in the USING clause if this perform range was selected for extract.
Dead Code	An X indicates that the perform range contains dead code.
Pgm EXIT	An X indicates that the perform range contains a program exit, (i.e., the PERFORM may not return).
Live Exit	An X indicates that the perform range contains a live exit.
Overlap Paragraph	An X indicates that the perform range contains code that overlaps with another perform range, (i.e., the code is contained by two or more perform ranges).
Recursion	An X indicates that the perform range contains code that could be recursive, (i.e., it could perform itself).

PF Keys

Key	Description
PF4 = View	Displays the Select View Type pop-up (see Figure 60 on page 134) to select a view format. After selecting the view format, the current program displays in the specified view screen. Use the SELECT primary command to select a perform range in the view pop-up.
PF5 = Filter	Displays the Extract - Perform Range Filter pop-up (see Figure 59 on page 132) used to filter the list of Perform Range that displays on the Extract - Perform Range List pop-up.
PF17 = Sort	Displays the Extract - Sort PERFORM Range pop-up (see Figure 61 on page 136) to sort the list of perform range names by any of the columns listed on the Extract - Perform Name List pop-up.

Commands

Command	Description
EXPORT	Allows you to write relevant data from the Extract - Perform Name List pop-up (in comma delimited format) to a specified, or default, dataset name. These are the operands for this command: • ALL - selects all available perform range data for export. • An asterisk (*) or a question mark (?) - allows for wildcard selection criteria. • PERFRANGE name - selects the specified perform range name(s). • TO - directs the export to the desired output dsn. • dsn(member) - specifies export to a default dsn member. • ’dsn(member)’ - specifies export to a specific dsn member. • dsn - specifies export to a default dsn. • ’dsn’ - specifies export to a specific dsn.

See "EXPORT Command" on page 351 for a detailed description of this command and all associated operands.

Usage Notes

A COBOL program usually contains a number of perform range names. If the perform range to be extracted is not known, the Extract - Perform Range Filter pop-up (see Figure 59 on page 132) can be used to reduce the number of perform range names listed on the Extract - Perform Name List pop-up, so that selecting the desired perform range is easier. The reduction of the list of perform ranges is accomplished by processing the qualifying criteria from the fields on the Extract - Perform Name Filter pop-up in this order:

Number	Description
1.	The list is reduced to those names that match the entered perform range name pattern. If a perform range name is not entered, and all other fields on the pop-up are left blank, all perform ranges are included in the selectable list.
2.	The list can be altered by specifying data names that must be referenced within the perform range. The Data Name field restricts the list to those perform ranges that reference data names matching the Data Name field. Use Context restricts the list to only those data names that are referenced, used or modified. If Aliasing is specified, the list of data names is expanded to include all data names that are aliases of the data name in the Data Name field.
3.	The list can be further restricted by specifying a line range. The list is reduced to contain those perform ranges whose lines intersect the range of lines specified.
4.	The list can be reduced by specifying a COBOL subset. The list is reduced to contain the perform ranges that have at least one of the requested COBOL subset statements.
5.	The list can be further reduced by specifying a text pattern that must be contained in the perform range set of source lines.

The resultant list of perform range names displays according to the entry in the Selection Screen field.

Extract - Perform Range Filter Pop-up

The Extract - Perform Range Filter pop-up, shown in Figure 59, is used to specify criteria to filter the list of selectable perform ranges on the Extract - Perform Range List pop-up. This pop-up displays by pressing PF5 on the Extract - Perform Name List pop-up (see Figure 58 on page 128).

When this pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

A question mark (?) can be entered in text fields to display a help pop-up about that particular field. The help pop-ups provide information about the field, list the valid values, and allow input of the desired value.

Figure 59 • Extract - Perform Range Filter Pop-up

```
                    Extract - Perform Range Filter
Type information for filtering PERFORM Range Name list.

PERFORM Name    ----------------------------------------------------

Data Name  . . ----------------------------------------------------
Use Context                 Options
1   1.  References          /  Aliasing
    2.  Uses
    3.  Modifications

Line Range . . ----------------------------------------------------
COBOL Subset   ----------------------------------------------------
Text Pattern   ----------------------------------------------------
```

Fields

Field	Description
PERFORM Name	Optional. Type the name of the perform range you want to appear on the selection list. Enter a name pattern with embedded wild card characters (a question mark (?) for one character; an asterisk (*) for zero or more characters) to list all perform ranges with that name pattern. For example, enter `A*` to list all perform ranges that begin with an A. If the PERFORM Name field is left blank, the list of selectable perform ranges includes all perform ranges meeting the criteria entered on this pop-up. The desired perform range is then selected from the list.
Data Name	Optional. Enter a qualified data variable name. The list of selectable perform range names on the Extract - Perform Name List pop-up includes only those using, modifying, or referencing the variable, according to the entry in the Use Context field. If left blank, the list is not restricted by data name.

Field	Description
Use Context	Optional. Enter a qualifier to identify a specific use context of the data name. The default is 1, References. 1. References - The list of selectable perform ranges only includes occurrences that test, use, set or modify the specified data name. 2. Uses - The list of selectable perform ranges only includes occurrences that test or use the specified data name. 3. Modifications - The list of selectable perform ranges only includes occurrences that modify the specified data name.
Aliasing	Optional. Specify Aliasing to include data names that are aliases of the specified data name. If left blank, aliases of the data name are not included. The default is no aliasing. Aliases consist of these: • Parent - higher level group item • Child - lower level item • Rename/redefinition - renamed, redefined or 88 level items **Note:** The Use Context and Aliasing fields are ignored if the Data Name field is left blank.
Line Range	Optional. Enter a single line number or a range of lines. If Line Range is left blank, the list is not restricted by line range.
COBOL Subset	Optional. Enter a predefined COBOL language subset. If COBOL Subset is left blank, the list is not restricted by subset type. Subsets can be concatenated by placing a plus sign (+) between them. See Chapter 2, "Product Overview," on page 7 for a list of all the COBOL subsets. Field help for the COBOL Subset field may be obtained by entering a question mark (?) as the only character in this field.
Text Pattern	Optional. Enter an alphanumeric string. If Text Pattern is left blank, the list is not restricted by perform range.

Select View Type Pop-up

The Select View Type pop-up, shown in Figure 60, is used to specify the format in which you want to view the source for the current program. After the source displays, use the SELECT primary command to select the extract criteria. This pop-up displays when you select PF4 on one of the Extract List pop-ups.

Figure 60 • Select View Type Pop-up

```
                          Extract - Perform Name List
C                                                        Scroll ===> CSR
                 Select View Type
S                                                ract.  Then press Enter.  Press
E          Select the view format.               nd to order column by value.
                                                          U - Unselect
           Selection Screen
           _ 1.  Source View                      Size       Times PERF Direct Subord
           _ 2.  Tree View                       nes   %     PERF   by  PERFs  PERFs
-          _ 3.  Structure View                  --- -----   ----  ---- ----   ----   >
_                                                 27 12.86     1     1    3      4
_                                                  7  3.33     1     1    0      0
_ CLOSE-PGM                                        5  2.38     1     1    1      1
_ EXCEPTION-REPORT-1                              16  7.62     1     1    0      0
_ GET-NUM-OF-DAYS                                 10  4.76     1     1    1      1
_ INIT-ACTION-RECORD                               9  4.29     1     1    0      0
_ INITIALIZE-PGM                                  21 10.00     1     1    2      3
_ INITIATE-ACTION                                 13  6.19     5     2    0      0
_ OPEN-FILES                                       5  2.38     1     1    0      0
_ REPORT-FINAL-CNTRS                               5  2.38     1     1    0      0

     PF4=View  PF5=Filter  PF17=Sort
```

Fields

Field	Description
Option	Type the number corresponding to the view pop-up or screen to be used to select extract criteria. 1. Source View - Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the program in COBOL source statement order. If extract criteria have been selected from the extract list, any code selected is highlighted and tagged and the cursor is positioned on the first target found. To select extract criteria, use the SELECT primary command followed by the item name. 2. Tree View - Displays the View - Tree View Request pop-up (see Figure 36 on page 77) to format the display of the program. After specifying format options and pressing Enter, the View - Tree pop-up with a shortened action bar displays showing logical program units and the relationships between related units. If extract criteria have been selected on the extract list, all program labels containing found targets are highlighted and tagged, and the cursor is positioned at the first label containing the found target(s). Select the desired extract criteria with the SELECT primary command. 3. Structure View - Displays the View - Structure View Request pop-up (see Figure 40 on page 85) to format the display of the program. After specifying format options and pressing Enter, the View - Structure pop-up displays showing a graphical representation of the PERFORM structure of the program. If extract criteria have been selected on the extract list, all boxes representing the criteria are highlighted. Select the desired extract criteria with the SELECT primary command.

Extract - Sort PERFORM Range Pop-up

The Extract - Sort PERFORM Range pop-up, shown in Figure 61, is used to specify the sort column for the Extract - Perform Name List pop-up. This pop-up displays by pressing PF17 (or when the SORT command is entered on the command line) on the Extract - Perform Name List pop-up (see Figure 58 on page 128).

Figure 61 • Extract - Sort PERFORM Range Pop-up

```
                Extract - Sort PERFORM Range

Sort Column                          Sort Order
_  1.  PERFORM Range Name            /  Ascending
   2.  Size (lines)
   3.  Times PERFORMed
   4.  PERFORMers
   5.  PERFORMs
   6.  Subordinate PERFORMs
   7.  External CALLs
   8.  Data Modifications
   9.  Data Uses
   10. Data References
   11. 01/77 Parameters
   12. DEADCODE
   13. Program EXITs
   14. "Live" EXITs
   15. Recursion
   16. Overlapping Code
```

Fields

Field	Description
Sort Column	Optional. Type the number of the column to be sorted. The default is 1. PERFORM Range Name.
Ascending	Optional. Specify Ascending if the entries on the Extract - Perform Name List pop-up (see Figure 58 on page 128) are to be sorted in ascending order. This is the default. To sort the entries in descending order, enter a blank space to remove the slash (/).

Report Extract

The purpose of the Report extract is to isolate all the code that produces a report. A report is defined in general terms as a logical collection of output statements.

Usually, a program that produces multiple reports would output each report as a separate file description (FD) or report description (RD), or possibly using a general purpose CALLed subroutine, although Encore does not require this as a prerequisite to report extraction. Also, reports are usually output to a hardcopy device, although Encore does not require that reports be destined for hardcopy output. The report extract could be used to separate multiple record types from a file.

A report name can be on of these:

- An FD
- An RD
- An SD
- A CALLed program name (static)
- DISPLAY statements (SYSOUT file)

To accomplish the report extract objective

1. Analyze the program, if this has not already been done. The desired program by selecting File ▸ Analyze from the File pull-down (see Figure 9 on page 26), or one of the methods described in Chapter 12, "Analyze Functions," on page 431.

2. Select the desired report name on the Extract - Report Name List pop-up (see Figure 62 on page 138). To filter the list of selectable report names, press PF5 on the Extract - Report Name List pop-up.

3. After the report name is selected, Encore presents a list of statements that reference the selected report name. From this list, you select the COBOL statement(s) which define the report desired.

To initiate a report extract, follow this step:

▸ Select Extract ▸ Report and press Enter (see Figure 56 on page 124). This extract mode allows selection of one report name or multiple report names.

The pop-ups used to complete a report extract objective are described in this section in the order that they are used.

Extract - Report Name List Pop-up

The Extract - Report Name List pop-up, shown in Figure 62, is used to select the name of the report definition to be extracted. This pop-up displays by selecting Extract ▸ Report from the Extract pull-down (see Figure 56 on page 124), and then pressing Enter on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).

When this pop-up initially displays, all report names in the program are listed. To narrow the selection list, press PF5 to display the Extract - Report Filter pop-up (see Figure 63 on page 141). When this pop-up initially displays information is listed in ascending alphabetical order by report name. To change the sort order, press PF17 to display the Extract - Sort Report Name pop-up (see Figure 66 on page 145).

To select the desired report name, use the SELECT primary command or the S line command. More than one report name can be selected on this screen, but they must be selected one at a time. Selected report names are highlighted.

After a report name is selected, press Enter to display the Extract - Source Ordered View pop-up (see Figure 67 on page 146) to select the appropriate report statements.

Figure 62 • Extract - Report Name List Pop-up

```
                         Extract - Report Name List         4 REPORT NAMES
Command ===> _____________________________________________ Scroll ===> CSR

Select Name of Report Definition for extract.  Then press Enter.  Use SORT
command to order column by value.  Press Enter when complete.

  S - Select     V - View Source     U - Unselect

                                                          OPEN     Num
  Name                                Type     Organization Options RECS
  ------------------------------      -------  ------------ -------  ---->
_ ACCT-FILE                           FD       INDEXED      I/O        1
_ ACTION-FILE                         FD       SEQUENTIAL   OUTPUT     1
_ EXCEPTION-FILE                      FD       SEQUENTIAL   OUTPUT     1
_ SYSOUT                              DDNAME   SEQUENTIAL   OUTPUT     0
******************************** BOTTOM OF DATA ****************************


     PF4=View  PF5=Filter  PF17=Sort
```

Fields

Note:

The screen may be scrolled left and right to view all the columns. The column heading underscore line contains a less than (<) indicator if the columns can be scrolled to the left and a greater than (>) indicator if the columns can be scrolled to the right.

Field	Description
Line command area	The line command area, to the left of the Name field, accepts these line commands: • S - Selects the identified report name, and displays the Extract - Source Ordered View pop-up (see Figure 67 on page 146) for selection of the corresponding statement(s). • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements in the report name specified to be viewed are highlighted and tagged to the right with DATA DEF. Any report name can be selected on the View - Source pop-up with the SELECT command. • U - Unselects a previously-selected report name. The name is no longer highlighted. **Note:** Unselecting a report name implies the unselection of the related statements.
Name	Lists the report names in the program. To filter the list, press PF5 to display the Extract - Report Filter pop-up (see Figure 63 on page 141). Report names that have already been selected are highlighted.
Type	Identifies the name as a FD (File Description), SD (Sort Description), RD (Report Description), DDNAME, CALL, TABLE, or VIEW.
Organization	Identifies the file organization as INDEXED or SEQUENTIAL.
OPEN Options	Identifies the type of input/output operation that accesses the report name as I/O, INPUT, or OUTPUT.
Num RECS	The number of 01 records associated with the report.
Min LRECL	The minimum logical record length defined for the report.

Field	Description
Max LRECL	The maximum logical record length defined for the report.
READ INTOs	The number of READ INTO statements for the report.
WRITE FROMs	The number of WRITE FROM statements for the report.
DDNAME	The name assigned to the DDNAME of the file by the SELECT statement.
Num Keys	The number of variables assigned as KEYs to the file.
Num Reads	The number of READ statements that reference the file.
Num Writes	The number of WRITE statements that reference the file.
Num DELETEs	The number of DELETE statements that reference the file.
Num OPENs	The number of OPEN statements that reference the file.
Num CLOSEs	The number of CLOSE statements that reference the file.
Num STARTs	The number of START statements that reference the file.
FILE-STATUS variable	The data variable name assigned to the FILE-STATUS for the file.

PF Keys

Key	Description
PF4 = View	Displays the Select View Type pop-up (see Figure 60 on page 134) to select the view format. After selecting the view format, the current program displays. Use the Select primary command to select a report name in the view pop-up.
PF5 = Filter	Displays the Extract - Report Filter pop-up (see Figure 63 on page 141) used to filter the report name selection list.
PF17 = Sort	Displays the Extract - Sort Report Names pop-up (see Figure 66 on page 145) to sort the list of report names by any of the columns listed on the Extract - Report Name List pop-up.

Usage Notes

A COBOL program contains a number of report definition names. If the specific report definition name to be extracted is not known, the Extract - Report Filter pop-up (see Figure 63 on page 141) can be used to reduce the number of report names listed on the Extract - Report Name List pop-up, so that selecting the desired report name is easier. The reduction of the list of report names is accomplished by processing the qualifying criteria from the fields on the Extract - Report Filter pop-up in this order:

- The list is reduced to those names that match the entered report name pattern. If a report name is not entered, and all other fields on the pop-up are left blank, all report names are included in the selectable list.
- The list can be altered by specifying data names that must be contained in a record description for the report name. The Data Name field restricts the list to those report names that have record descriptions containing data names matching the pattern entered in the Data Name field. Level Number, if entered, further restricts data names to the specified level number.
- The list can be further restricted by specifying a copy book name. The list is reduced to contain report names that are defined or referenced within the copy book name specified.
- The list can be further restricted by specifying a line range. The list is reduced to contain those report names that are referenced within the specified range of lines.

Extract - Report Filter Pop-up

The purpose of the Extract - Report Filter pop-up, shown in Figure 63, is to specify the name of the desired report to extract. The Extract - Report Filter pop-up displays by pressing PF5 on the Extract - Report Name List pop-up (see Figure 62 on page 138).

When this pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

A question mark (?) can be entered in text fields to display a help pop-up about that particular field. The help pop-ups provide information about the field, list the valid values, and allow input of the desired value.

Figure 63 • Extract - Report Filter Pop-up

```
                    Extract - Report Filter

Type information for filtering Report Name list.

Report Name    _
Data Name  . . ____________________________________________________
COPYBOOK Name  ________
Line Range . . ____________________________________________________

Options
_  Limit to within an existing Segment
```

Fields

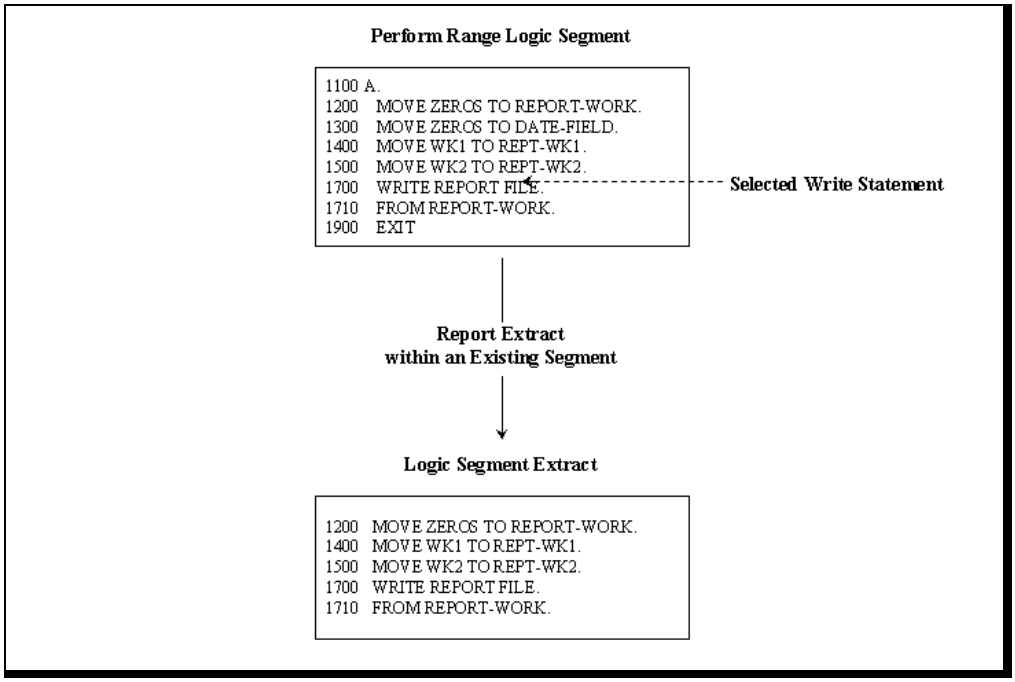
Field	Description
Report Name	Required. Type the name of the report you want to appear on the selection list. Enter a name pattern with embedded wild card characters (a question mark (?) for one character, an asterisk (*) for zero or more characters) list all reports with that name pattern. For example, enter `A*` to list all reports that begin with A. If the Report Name field is left blank, the list of selectable report names includes all report names meeting the criteria entered on this pop-up.
Data Name	Optional. Enter a qualified data variable name.
COPYBOOK Name	Optional. Type the name of a copy book member included in the program. The list of selectable report names on the Extract - Report Name List pop-up (see Figure 62 on page 138) includes only those that reference data names defined in the copy book member. If COPYBOOK Name is left blank, the list is not restricted by COPY member.
Line Range	Optional. Enter a single line number or a range of lines. The list of selectable report names on the Extract - Report Name List pop-up (see Figure 62 on page 138) includes only those that contain statements in the specified line range. If Line Range is left blank, the list is not restricted by line range.
Limit to within an existing Segment	Optional. This extract option (specified by entering a slash (/)) limits the report extract to the Logic Segment. Perform Range and Transaction Logic Segments display for selection on the Select Extract-Within Logic Segment pop-up (see Figure 65 on page 144). The result is a limited Report Extract.

Usage Notes

When the Limit to within an existing Segment option is not selected (i.e., the extract encompasses the entire program), the result is a standard Report extract. The Logic Segment includes all of the code necessary to produce the report.

When the Limit to within an existing Segment option is selected, the result is a Report extract limited to the contents of the Perform Range or Transaction Logic Segment, as shown in Figure 64. The resulting Logic Segment may not have all the code necessary to produce the report.

Figure 64 • Limit to Within an Existing Segment Option



The Write Statement at lines 1700 - 1710 is selected. The statements at lines 1200, 1400, and 1500 are included (with the Write Statement) in the Report extract. The statement at line 1300 (MOVE ZEROS TO DATE-FIELD) is not included in the Report extract, because it does not contain a value or contribute to the value of the report record.

Select Extract-Within Logic Segment Pop-up

The Select Extract-Within Logic Segment pop-up, shown in Figure 65, is used to select a Perform Range or Transaction Logic Segment for an extract. This selection limits the extract to the contents of the Logic Segment. This pop-up displays by selecting the Limit to within an existing Segment field on the Extract - Report Filter pop-up (see Figure 63 on page 141) or Extract - Computation Variable pop-up (see Figure 69 on page 152).

Figure 65 • Select Extract-Within Logic Segment Pop-up

```
                    Select Extract-within Logic Segment   1 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

Select the Logic segment which limits or bounds the current Extract
definition.  Then press Enter.
   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
_  VIARDEMO   PERFORM-RANGE-0001             Perform Range
******************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select the desired Logic Segment.
Program	The name of the program from which the Logic Segment was extracted.
Logic Segment Name	List of Logic Segment names defined for the current program.
Description	The type of extract (Perform Range or Transaction) that was used to create the Logic Segment.

Extract - Sort Report Names Pop-up

The Extract - Sort Report Names pop-up, shown in Figure 66, is used to specify the sort column for the Extract - Report Name List pop-up (see Figure 62 on page 138). This pop-up displays by pressing PF17 on the Extract - Report Name List pop-up, or when the SORT command is entered.

Figure 66 • Extract - Sort Report Name Pop-up

```
                    Extract - Sort Report Names

Sort Column                          Sort Order
_  1.  Name                          /  Ascending
   2.  Type
   3.  Organization
   4.  OPEN options
   5.  Number of records
   6.  Minimum record size
   7.  Maximum record size
   8.  READ INTOs
   9.  WRITE FROMs
   10. DDNAME
   11. Number of keys
   12. Number of READs
   13. Number of WRITEs
   14. Number of DELETEs
   15. Number of OPENs
   16. Number of CLOSEs
   17. Number of STARTs
   18. FILE STATUS variable
```

Fields

Field	Description
Sort Column	Type the number of the column to be sorted. The default is column 1. Name.
Ascending	Optional. Specify Ascending if the entries on the Extract - Report Name List pop-up are to be sorted in ascending order. This is the default. To sort the entries in descending order, enter a blank space to remove the slash (/).

Extract - Source Ordered View Pop-up

The Extract - Source Ordered View pop-up, shown in Figure 67, is used to select the individual statements that make up the report being extracted. This pop-up displays after a report name is selected on the appropriate supporting pop-up.

All WRITE statements (for FDs), GENERATE statements (for RDs), DISPLAY statements, or CALL statements that reference the selected report name are listed on this pop-up in source code order.

Figure 67 • Extract - Source Ordered View Pop-up - Report Extract

```
                        Extract - Source Ordered View
Command ===> _____________________________________________ Scroll ===> CSR

Select statements for Report Extract, press Enter.  When complete press
Enter.

Report Name: ACCT-FILE

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
_ 000322      REWRITE ACCT-RECORD.
******************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
Report Name	The name of the currently-selected report definition.
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the identified source line and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the COBOL source statement selected for viewing positioned in the middle of the screen to see the statement in context. The statement is highlighted and tagged to the right with LINE RNG. Use the END command to return to the Extract - Source Ordered View pop-up. • U - Unselects a previously-selected source line. The report name is no longer highlighted on the unselected line.

Field	Description
line number	The source code line number of the report statement.
statement	The COBOL source code statement(s) contained in the currently-selected report definition. Statements that have been selected so far are highlighted.

Usage Notes

Which of the displayed statements to select is determined by known characteristics of the report. For example, some reports can be identified by a set of output variables. Others can be identified based on where, in the program source sequence, the output records are created.

Computation Variable Extract

The purpose of the Computation Variable extract objective is to isolate the logic that determines the value of a specific data variable at a specific statement. This logic consists of the code that contributes to, or influences the value of, a data variable from the beginning of the program to a user-specified statement. This logic contains the statements that assign values to the variable and any additional variables in those statements.

To accomplish the Computation Variable extract objective

1. Analyze the program, if this has not already been done. The desired program is analyzed by selecting File ▸ Analyze from the File pull-down (see Figure 9 on page 26), or one of the methods described in Chapter 12, "Analyze Functions," on page 431.

2. Select the COBOL data variable name on the Extract - Data Name List pop-up (see Figure 68 on page 148). To filter the list of selectable data names, press PF5 on the Extract - Data Name List pop-up (see Figure 68 on page 148).

3. After the data name is selected, Encore presents a list of source statements that reference the data name selected. From this list, you select the specific statement(s) at which the selected data variable has the desired value.

A Computation Variable extract is initiated by selecting Extract ▸ Variable from the Extract pull-down (see Figure 56 on page 124). This extract mode allows selection of one computation variable data name or multiple computation variable data names.

The pop-ups used to complete the Computation Variable extract objective are described in this section in the order that they are used.

Extract - Data Name List Pop-up

The Extract - Data Name List pop-up, shown in Figure 68, is used to select the data name to be extracted. This pop-up displays by selecting Extract ▸ Variable from the Extract pull-down (see Figure 56 on page 124), and then pressing Enter on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).

When this pop-up initially displays, all data names in the program are listed. To narrow the selection list, press PF5 to display the Extract - Computation Variable Filter pop-up (see Figure 69 on page 152). The data names display in alphabetical order. To change the sort order, press PF17 to display the Extract - Sort Data Name pop-up (see Figure 70 on page 154).

More than one data name can be selected on this screen, but they must be selected one at a time. Selected data names are highlighted.

When a data name is selected and Enter is pressed, the Extract - Source Ordered View pop-up (see Figure 71 on page 155) displays. The appropriate statements are then selected on this pop-up.

Figure 68 • Extract - Data Name List Pop-up

```
                          Extract - Data Name List            59 DATANAMES
Command ===> _____________________________________________ Scroll ===> CSR

Select Data Name.  Then press Enter to display referencing statements.  Use
SORT command to order column by value.  Press Enter when complete.
                        Data Name Information
                        1   1.  Qualification
                            2.  Profile
  S - Select     V - View Source     U - Unselect

  LV Data Name                          Qualification
- -- --------------------------------  ----------------------------------
_ 05 ACCT-AVG-BALANCE                  01 ACCT-RECORD
_ 05 ACCT-BALANCE                      01 ACCT-RECORD
_ 05 ACCT-CONTROL-CODE                 01 ACCT-RECORD
_ 05 ACCT-INT-YTD                      01 ACCT-RECORD
_ 05 ACCT-INTEREST                     01 ACCT-RECORD
_ 77 ACCT-IO-CNT
_ 05 ACCT-KEY                          01 ACCT-RECORD
_ 05 ACCT-LAST-UPDATE                  01 ACCT-RECORD

     PF4=View  PF5=Filter  PF17=Sort
```

Fields

Field	Description
Data Name Information	Optional. Specifies the type of information displayed on the Extract - Data Name List pop-up. The default is 1. Qualification. 1. Qualification - Displays the parent(s) of the data variable name. 2. Profile - Displays profile information.
Line command area	The line command area, to the left of the LV field, accepts these line commands: • S - Selects the identified data name, and displays the Extract - Source Ordered View pop-up (see Figure 71 on page 155) for selection of the corresponding statement(s). • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the COBOL source code of the data definition. The line is highlighted and tagged to the right with DATA DEF. A long message displays indicating the number of data definitions found. Use the END command to return to the Extract - Data Name List pop-up. • U - Unselects a previously-selected data name. The name is no longer highlighted. **Note:** Unselecting a data variable name implies the unselection of the related statements.
LV	The COBOL level number of the data name.
Data Name	The name of the data variable. Data names that have already been selected are highlighted.

Field	Description
Qualification	These field displays when the Data Name Information field is set to 1. Qualification. The parent(s) of the data variable name. The qualifying parent name includes the level number and data name. The qualification displays only for qualified data variable names, (i.e., data names that are not unique). A greater than (>) indicator denotes that the name is truncated on the display and the rest of the name may be viewed by scrolling to the right. A less than (<) indicator denotes that the display may be scrolled to the left.
Profile	**Note:** These fields display when the Data Name Information field is set to 2. Profile. • Line - The source line number containing the definition of the data variable. • Loca - The SECTION containing the source line definition. • Size - The data variable length, in bytes. • RED - An X indicates that the data name REDEFINEs another data structure. • REN - An X indicates that the data name RENAMEs another data structure. • OCC - An X indicates that the data name is defined with an OCCURS clause. • DED - An X indicates that the data item is DEADDATA, (i.e., it is never referenced in the program). • CAL - An X indicates that the data name is used in a CALL statement, (i.e., the data item is used as a parameter to another program). • SQL - An X indicates that the data name is used as an SQL host variable. • CIC - An X indicates that the data name is used in an EXEC CICS statement. • ID - An X indicates that the data name is used in an IDMS statement.

PF Keys

Key	Description
PF4 = View	Displays the Select View Type pop-up (see Figure 60 on page 134) to select the view format. After selecting the view format, the current program displays. Use the SELECT primary command to select a data name in the view pop-up.
PF5 = Filter	Displays the Extract - Computation Variable Filter pop-up (see Figure 69 on page 152) used to filter the list of variable names that displays on the Extract - Data Name List pop-up.
PF17 = Sort	Displays the Extract - Sort Data Name pop-up (see Figure 70 on page 154) to change the sort order for the data name list.

Usage Notes

A COBOL program contains a number of data variable names. If the data variable name to be extracted is not known, the Extract - Computation Variable Filter pop-up (see Figure 69 on page 152) can be used to reduce the number of data names listed on the Extract - Data Name List pop-up (see Figure 68 on page 148), so that selecting the desired data variable name is easier. The reduction of the list is accomplished by processing the qualifying criteria from the fields on this pop-up in this order:

- The list is reduced to those names that match the entered data name pattern. If a data name is not entered, and all other fields on the pop-up are left blank, all data names are included in the selectable list. Level Number, if entered, further restricts data names to the specified level number. If Aliasing is specified, the list of data names is expanded to include all data names that are aliases of the data names qualified by the entered Data Name.
- The list can be further restricted by specifying a copy book member name. The list is reduced to contain those data names that are defined or referenced within the COPYBOOK Name specified.
- The list can be restricted by specifying a line range. The list is reduced to contain those data names that are referenced within the specified range of lines.
- The list can be restricted by specifying a paragraph label. The list is reduced to contain those data names referenced within the paragraph specified.
- The list can be reduced by specifying a COBOL subset. The list is reduced to contain data name references in statements corresponding to the specified COBOL subset type(s).
- You can also use the Use Context or Limit to within an existing segment option to further reduce the list. See "Extract - Computation Variable Filter Pop-up" on page 152 for more information.

Extract - Computation Variable Filter Pop-up

The Extract - Computation Variable Filter pop-up, shown in Figure 69, is used to specify a data variable name to extract. This pop-up displays by pressing PF5 on the Extract - Data Name List pop-up (see Figure 68 on page 148).

When this pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

A question mark (?) can be entered in text fields to display a help pop-up about that particular field. The help pop-ups provide information about the field, list the valid values, and allow input of the desired value.

Figure 69 • Extract - Computation Variable Filter Pop-up

```
                    Extract - Computation Variable Filter

Type information for filtering Data Variable Name list.

Data Name  . . _____________________________________________________
Use Context                    Options
__  1.  References             _  Aliasing          Level Number . . __
    2.  Uses
    3.  Modifications

COPYBOOK Name  ________
Line Range . . ________________________________________________
Label Name . . ________________________________________________
COBOL Subset   ________________________________________________

Options
_  Limit to within an existing Segment
```

Fields

Field	Description
Data Name	Optional. Type the data variable name to be selected and press Enter. If a fully-qualified data name is entered, the Extract - Source Ordered View pop-up (see Figure 71 on page 155) displays to allow selection of the corresponding statement(s). Field help for the Data Name field may be obtained by entering a question mark (?) as the only character in this field.
Use Context	Optional. Enter a qualifier to identify a specific use context of the data name. The default is 1. References. 1. References - The list of selectable data names only includes occurrences that test, use, set or modify the specified data name. 2. Uses - The list of selectable data names only includes occurrences that test or use the specified data name. 3. Modifications - The list of selectable data names only includes occurrences that modify the specified data name.

Field	Description
Aliasing	Optional. Specify Aliasing to include data names that are aliases of the specified data name. If left blank, aliases of the specified data name are not included in the list of selectable data names. The default is no aliasing. Aliases consist of these: • Parent - higher level group item • Child - lower level item • Rename/redefinition - renamed, redefined or 88 level items
Level Number	Optional. Enter a valid data name level number. The list of selectable data names on the Extract - Data Name List pop-up (see Figure 68 on page 148) includes only those at the specified level. If Level Number is left blank, the list is not restricted by level.
COPYBOOK Name	Optional. Enter the name of a copy book member included in the program. The list of selectable data names on the Extract - Data Name List pop-up (see Figure 68 on page 148) includes only those that reference data names defined in the copy book member. If COPYBOOK Name is left blank, the list is not restricted by COPY member.
Line Range	Optional. Enter a single line number or a range of lines. If Line Range is left blank, the list is not restricted by line range.
Label Name	Optional. Enter a PROCEDURE DIVISION paragraph or section name. The list of selectable data names on the Extract - Data Name List pop-up (see Figure 68 on page 148) includes only those referenced in the specified paragraph or section. If left blank, the list is not restricted by label name.
COBOL Subset	Optional. Enter a predefined COBOL language subset. If left blank, the list is not restricted by subset type. Subsets can be concatenated by placing a plus sign (+) between them. See "COBOL Subsets" on page 17 for a list of all the COBOL subsets.
Limit to within an existing Segment	Optional. This extract option (specified by entering a slash (/)) limits the Computation Variable extract to the Logic Segment. Perform Range and Transaction Logic Segments display for selection on the Select Extract-Within Logic Segment pop-up (see Figure 65 on page 144). The result is a limited Computation Variable extract.

Usage Notes

Using the Limit to Within an Existing Segment Option

Similar to the processing described for limited Report extracts (see Figure 64 on page 143), when the Limit to within an existing Segment option is selected, the result is a Computation Variable extract limited to the contents of the Perform Range or Transaction Logic Segment. The resulting Logic Segment may not have all the code necessary to calculate the data variable.

When the Limit to within an existing Segment option is not selected (i.e., the extract encompasses the entire program), the result is a standard Computation Variable extract. The Logic Segment includes all of the code necessary to calculate the data variable.

Note:

If this option is not specified, the extract is not limited to a Logic Segment, but encompasses the entire program. The result is a standard Computation Variable extract.

Extract - Sort Data Name Pop-up

The Extract - Sort Data Name pop-up, shown in Figure 70, is used to specify the sort column for the Extract - Data Name List pop-up (see Figure 68 on page 148). This pop-up displays when the SORT command is entered on the Extract - Data Name List pop-up.

Figure 70 • Extract - Sort Data Name Pop-up

```
                    Extract - Sort Data Name

Sort Column                          Sort Order
__  1.  Level Number                 /  Ascending
    2.  Data Name
    3.  Defined Source Line
    4.  Defined SECTION
    5.  Size
    6.  REDEFINEs
    7.  RENAMEs
    8.  OCCURs
    9.  DEADDATA
    10. Use in CALLs
    11. Use in SQL
    12. Use in CICS
    13. Use in IDMS
```

Fields

Field	Description
Sort Column	Type the number of the column to be sorted. The default is column 2. Data Name.
Ascending	Optional. Specify Ascending if the entries on the Extract - Data Name List pop-up (see Figure 68 on page 148) are to be sorted in ascending order. This is the default. To sort the entries in descending order, enter a blank space to remove the slash (/).

Extract - Source Ordered View Pop-up

The Extract - Source Ordered View pop-up, shown in Figure 71, is used to select the statement(s) in the program that determines the Computation Variable Logic Segment termination point, (i.e., the point(s) at which the data variable contains the desired value). This pop-up displays after a data name is selected and Enter is pressed on the Extract - Data Name List pop-up (see Figure 68 on page 148).

All COBOL source statements that have USE or MOD references to the selected data variable name (and aliases if requested) are listed on this pop-up in COBOL source program order.

Figure 71 shows the Extract - Source Ordered View pop-up for a Computation Variable extract.

Figure 71 • Extract - Source Ordered View Pop-up - Computation Variable

```
                         Extract - Source Ordered View
Command ===> _____________________________________________ Scroll ===> CSR

Select statements for Computation Variable Extract, press Enter.  When
complete press Enter.
Referenced Data Name: ACCT-AVG-BALANCE
            Aliasing:

  S - Select       V - View Source       U - Unselect
- ------ ------------------------------------------------------------------
_ 000303      COMPUTE WK-INTEREST =
  000304            (ACCT-AVG-BALANCE / DAYS-IN-YEAR) *
  000305            DAYS-IN-PERIOD * (INTEREST-RATE / 100).
******************************** BOTTOM OF DATA ******************************
```

Fields

Field	Description
Referenced Data Name	The data name that was selected on the Extract - Data Name List (see Figure 68 on page 148) pop-up or Extract - Computation Variable pop-up (see Figure 69 on page 152).
Aliasing	The current status of whether aliases are included in the list.
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the identified source line and highlights the data name. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the COBOL source statement selected for viewing positioned in the middle of the screen to see the statement in context. The statement is highlighted and tagged to the right with LINE RNG. Use the END command to return to the Extract - Source Ordered View pop-up. • U - Unselects a previously-selected source line. The name is no longer highlighted.
line number	The source code line number of the computation variable statement.
statement	The COBOL source code statement(s) contained in the currently-selected data variable name. Statements that have been selected so far are highlighted.

Usage Notes

When Aliasing is specified on the Extract - Computation Variable Filter pop-up (see Figure 69 on page 152), the Extract - Source Ordered View pop-up (see Figure 71 on page 155) displays all statements referencing the selected data name and all statements referencing the aliases of the selected data name. If a statement is then selected that references an aliased variable, instead of the selected data variable, the extracted logic is the derivation/computation of the aliased variable, not the selected data name. For example:

```
01  GROUP-ITEM.
05  ITEM-01 . . .
05  ITEM-02 . . .
05  ITEM-03 . . .
```

If GROUP-ITEM is selected as the computation variable and Aliasing is specified, the Extract -Source Ordered View pop-up displays all references to GROUP-ITEM, ITEM-01, ITEM-02 and ITEM-03. If a statement referencing GROUP-ITEM (the specified data name) is selected, the Logic Segment consists of all statements necessary to compute GROUP-ITEM (including ITEM-01, ITEM-02 and ITEM-03) at the selected statement(s).

However, if a statement referencing ITEM-`nn` is selected, the Logic Segment reflects the derivation of ITEM-`nn` only. Therefore, for the selected statement, the aliased data item effectively replaces the selected computation variable.

Transaction Extract

The purpose of the Transaction extract objective is to isolate the logic that executes when a specific data variable reaches a particular value. This logic consists of the code, between a start point and one or more end points, that executes when a particular transaction variable is assigned a specified set of values.

Usually, a program processing several types of transactions has a control variable (the transaction variable) that determines the logic flow. The Transaction objective isolates the sections of code based on the different values of the control variable. A transaction is defined by one or more nodes of conditional statements that define the code for a specific transaction, a START statement, and one or more END statements.

Note:

A Transaction Logic Segment consists of those statements that can be reached from the START point and that reach an END point without passing beyond an END point or program exit, nor without passing through a blocked conditional path.

To accomplish the transaction extract objective

1. Analyze the program, if this has not already been done. The desired program is analyzed by selecting File ▸ Analyze from the File pull-down (see Figure 9 on page 26), or one of the methods described in Chapter 12, "Analyze Functions," on page 431.

2. If necessary, the data variable name must then be selected on the Extract - Transaction Paths List pop-up (see Figure 75 on page 167). The list of names available for selection can be filtered by pressing PF5 on the Extract - Transaction Paths List pop-up.

3. Define the boundaries, or scope, of the transaction by specifying the START statement and one or more END statements. This step is only necessary if the defaults need to be changed, (i.e., to limit the scope to include only relevant processing sections of the program). This is useful in large programs to focus on only that section of the program of current interest. If the START or END statements are changed, a subprogram (callable module) is created when the Logic Segment is generated.
   The default for the start point is the beginning of the PROCEDURE DIVISION. The default(s) for the end point(s) consist of all program exits.

4. Select the paths of the program that make up the logic for the transaction. Encore displays a list of conditional statements, within the START and END boundaries, which reference the previously-selected data variable(s). You determine which transaction code values are desired, then select those condition paths that would be executed based on the desired values.

A Transaction extract can be initiated by selecting Extract ▸ Transaction from the Extract pull-down (see Figure 56 on page 124).

The pop-ups used to complete the Transaction variable extract objective are described in this section in the order that they are used.

Extract - Transaction Paths List Pop-up

The Extract - Transaction Paths List pop-up (see Figure 72) is used to select the logic path(s) that define the code for the transaction.

Figure 72 • Extract - Transaction Paths List Pop-up

```
                        Extract - Transaction Paths List
Command ===> ____________________________________________ Scroll ===> CSR

Identify conditionals which are definitely TRUE for the transaction, those
which are definitely FALSE, and leave blank the conditionals which may be
either TRUE or FALSE. Then press Enter.

  T - TRUE      F - FALSE      blank - TRUE/FALSE     V - View Source
- ------ ------------------------------------------------------------------
_ 000266      IF ACCT-MIN-BALANCE IS LESS THAN ACCT-BALANCE
_ 000277      IF ACCT-TYPE OF ACCT-RECORD = 'INT'
_ 000279      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
_ 000281      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
_ 000283      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
_ 000286      IF ACCT-MULTI-ACCT = 'YES'
_ 000291      IF ACCT-MIN-BALANCE < FREE-SERVICE-MIN
_ 000299      IF ACCT-MIN-BALANCE < 0
_ 000332      IF EX-RPT-LINE-CNT IS GREATER THAN 60
_ 000382      IF  FREE-SERVICE-MIN NOT NUMERIC
_ 000384      IF  CD-REVIEW-MIN    NOT NUMERIC

     PF5=Filter  PF6=Start/End Point
```

To display the Extract - Transaction Paths List pop-up, follow this step:

- Select Extract ▸ Transaction and press Enter on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).

When this pop-up initially displays, all conditional statements in the program are listed. To filter the list, press PF5 to display the Extract - Data Name List Filter pop-up (see Figure 68 on page 148). The list can also be reduced by changing the START and END points using the PF6 function.

This pop-up displays in source order the conditional statements that reference one or more of the data names previously identified on the Extract - Data Name List Filter pop-up. The conditional statements displayed include IF, EVALUATE, and GO TO DEPENDING ON.

During the Transaction extract, the default START statement is the PROCEDURE DIVISION statement. The default END statements are all program exits. If the default START and END statements are not valid for the Transaction extract to be performed, press PF6 to change them.

The initial display of this screen shows all paths selected and highlighted (i.e., TRUE/FALSE). The T or F line command must then be entered on relevant conditional statements.

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • T - Identifies the indicated path as TRUE, and includes the code as part of the transaction. • F - Identifies the indicated path as FALSE, and excludes the code from the transaction. • blank - Identifies the indicated path as either TRUE or FALSE, and includes the code as part of the transaction. • V - Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the conditional statement positioned in the middle of the screen to see the statement in context. The statement is highlighted and tagged to the right with LINE RNG. Use the END command to return to the Extract - Transaction Paths List pop-up (see Figure 72 on page 159). • An X displays in the line command area for conditionals not designated as T or F, and that are blocked by another conditional and therefore not reachable. In this case, the line command field is protected and cannot be changed. **Note:** If a conditional is designated as T or F, but is blocked by another conditional, an error message displays.
line number	The source code line number of the statement. Encore-generated path lines are indicated by ..... instead of a number.
statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

PF Keys

Field	Description
PF5=Filter	Displays the Extract - Data Name List Filter pop-up (see Figure 74 on page 164) used to filter the list of transaction paths.
PF6=Start/End Point	Displays the Extract - START/END statements pop-up (see Figure 73 on page 162) to change the starting and ending points for the Transaction extract.

Usage Notes

To define the transaction paths, first determine the anticipated values of the selected data variables. Then, based on the desired values, specify for each conditional whether that statement is definitely TRUE, definitely FALSE, or indeterminate (i.e., could be either TRUE or FALSE). Selection can be the TRUE or FALSE branches of an IF statement, individual branch paths of a GO TO DEPENDING ON statement, and individual WHEN conditions of an EVALUATE statement.

- For IF statements, indicate whether the statement is TRUE, FALSE, or either. Indicating that the statement is TRUE causes the code of the THEN clause to be included in the transaction, and the code of the ELSE clause to be excluded. If no ELSE clause exists, no code is excluded. Indicating that the statement is FALSE causes the THEN clause code to be excluded, and the ELSE clause code (if it exists) to be included.
- For EVALUATE statements, indicate which of the WHEN conditions are TRUE and which are FALSE. The EVALUATE statement displays to show the context of the WHEN statements, but no TRUE/FALSE indication is allowed for the actual EVALUATE statement. Specifying TRUE for a WHEN statement condition includes the code invoked by the WHEN. Specifying FALSE excludes the code.
- GO TO DEPENDING ON statements control various numbers of paths. Encore provides one line for each possible value of the DEPENDING variable. The value is indicated by `=n`, where `n` is the number corresponding to the position in the GO TO statement. Following the number is the label name that represents the flow for the indicated number value. To specify that code should be included, specify the number is TRUE. To exclude code, specify the number is FALSE.

Extract - START/END Statements Pop-up

Note:

This screen should only be used when the default START and END statements are not valid for the Transaction extract to be performed. If the START or END statements are changed, a subprogram (callable module) is created when the Logic Segment is generated.

The Extract - START/END Statements pop-up, shown in Figure 73, is used to change the default START and END statements of the transaction bounds.

Figure 73 • Extract - START/END Statements Pop-up

```
                        Extract - START/END Statements
Command ===> _____________________________________________ Scroll ===> CSR

Select start and end statement(s) for transaction.  When complete press
Enter.

      1  - List START/END(s)            2 - List All statements

  S - Select START E - Select END(s) U - Unselect V - View Source
- ---------------------------------------------------------------------------
_ 000225  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE START
_ 000226                           BANK-BALANCE.                      START
_ 000240      GOBACK.                                                   END
******************************** BOTTOM OF DATA ******************************
```

To display the Extract - START/END Statements pop-up, follow this step:

- Press PF6 on the Extract - Transaction Paths List pop-up (see Figure 72 on page 159).

The default START statement is the PROCEDURE DIVISION statement, and the default END statements are all program exits. The initial display of this pop-up contains only the START and END statements currently defined, which are highlighted.

To reset the display to contain the entire COBOL source program, follow this step:

- Use option 2 - List All statements.

To identify a new START statement

1 Unselect the current START statement with the U line command.

2 Select the new START statement with the S line command.

To reset Individual END statements follow this step:

- Use the U line command.

To select new END statements, follow this step:

- Use the E line command.

Options

Option	Description
List START/END(s)	Displays only the currently-selected START and END statements. This is the default.
List ALL statements	Displays all statements in the program, with the currently-selected START and END statements highlighted and tagged.

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Start point. Selects the identified statement as the START point, and highlights it. • E - End point. Selects the identified statement as an END point, and highlights it. • U - Unselects a previously selected START or END statement. The statement is no longer highlighted.
line number	The source code line number of the statement.
statement	The COBOL source code statements. Statements that have been selected so far are highlighted and tagged.
status	Indicates whether the statement status is START, END, or not selected.

Extract - Data Name List Filter Pop-up

The Extract - Data Name List Filter pop-up (see Figure 74 on page 164) is used to select the data names that control the transaction flow listed on the Extract - Transaction Paths List pop-up (see Figure 72 on page 159).

Figure 74 • Extract - Data Name List Filter Pop-up

```
                        Extract - Data Name List Filter         17 DATANAMES
Command ===> ____________________________________________ Scroll ===> CSR

Select Data Name(s) for filtering Transaction Path List.  The SORT
command may be used to order Data Names.
                         Data Name Information
                         1   1.  Qualification
                             2.  Profile
  S - Select      V - View Source     U - Unselect

  LV Data Name                          Qualification
- -- ---------------------------------  ----------------------------------
_ 05 ACCT-BALANCE                       01 ACCT-RECORD
_ 05 ACCT-KEY                           01 ACCT-RECORD
_ 05 ACCT-MIN-BALANCE                   01 ACCT-RECORD
_ 05 ACCT-MULTI-ACCT                    01 ACCT-RECORD
_ 01 ACCT-RECORD
_ 10 ACCT-TYPE                          05 ACCT-KEY, 01 ACCT-RECORD
_ 01 BANK-BALANCE
_ 01 BRANCH-BALANCE

     PF5=Filter  PF17=Sort
```

To dispay the Extract - Data Name List Filter pop-up, follow this step:

- Press PF5 on the Extract - Transaction Paths List pop-up.

When the Extract - Data Name List Filter pop-up initially displays, all data names in the program are listed. To filter the list, press PF5 to display the Extract - Transaction Variable pop-up (see Figure 75 on page 167). To change the sort order for the list, press PF17.

The effect of selecting a data name is that when the Extract - Transaction Paths List pop-up (see Figure 72 on page 159) is presented, only conditional statements referencing the data name display.

Use the SELECT primary command or the S line command to select the desired data name. More than one name can be selected. When all desired statements have been selected, use the END command to return to the Extract - Transaction Variable pop-up.

Fields

Field	Description
Data Name Information	Optional. Specify the type of information to display on the Extract - Data Name List pop-up. The default is 1. Qualification. 1. Qualification - Displays the parent(s) of the data variable name. 2. Profile - Displays profile information.
Line command area	The line command area, to the left of the LV field, accepts these line commands: • S - Selects the identified data name, and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the COBOL source code of the data definition. The line is highlighted and tagged to the right with DATA DEF. A long message displays indicating the number of data definitions found. Use the END command to return to the Extract - Data Name List pop-up. • U - Unselects a previously-selected data name. The name is no longer highlighted. **Note:** Unselecting a data variable name has no effect on any previously selected statements.
LV	The COBOL level number of the data name.
Data Name	The names of the data variables. Data names that have already been selected are highlighted.

Field	Description
Qualification	**Note:** This field displays when the Data Name Information field is set to 1. Qualification. The parent(s) of the data variable name. The qualifying parent name includes the level number and data name. The qualification displays only for qualified data variable names, (i.e., data names that are not unique). A greater than (>) indicator denotes that the name is truncated, and the rest of the name may be viewed by scrolling to the right. A less than (<) indicator denotes that the display may be scrolled to the left.
Profile	These fields display when the Data Name Information field is 2. Profile. • Line - The source line number containing the definition of the data variable. • Loca - The SECTION containing the source line definition. • Size - The data variable length, in bytes. • RED - An X indicates that the data name REDEFINEs another data structure. • REN - An X indicates that the data name RENAMEs another data structure. • OCC - An X indicates that the data name is defined with an OCCURS clause. • DED - An X indicates that the data item is DEADDATA, (i.e., it is never referenced in the program). • CAL - An X indicates that the data name is used in a CALL statement, (i.e., the data item is used as a parameter to another program). • SQL - An X indicates that the data name is used as an SQL host variable. • CIC - An X indicates that the data name is used in an EXEC CICS statement. • ID - An X indicates that the data name is used in an IDMS statement.

PF Keys

Field	Description
PF5=Filter	Displays the Extract - Transaction Variable pop-up (see Figure 75) to filter the list of selectable data names.
PF17=Sort	Displays the Extract - Sort Data Name pop-up (see Figure 76 on page 170) to change the sort order of the data names listed.

Extract - Transaction Variable Pop-up

The Extract - Transaction Variable pop-up, shown in Figure 75, is used to filter the list of data names that appear on the Extract - Data Name List Filter pop-up (see Figure 74 on page 164).

Figure 75 • Extract - Transaction Variable Pop-up

```
                    Extract - Transaction Variable

Type information for filtering Data Variable Name list.

Data Name  . . ____________________________________________________
/  Aliasing    Level Number  . . . . __

COPYBOOK Name  ________
Line Range . . ____________________________________________________
Label Name . . ____________________________________________________
COBOL Subset   COND
```

To display the Extract - Transaction Variable pop-up, follow this step:

- Press PF5 on the Extract - Data Name List Filter pop-up.

When this pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

A question mark (?) can be entered in text fields to display a help pop-up about that particular field. The help pop-ups provide information about the field, list the valid values, and allow input of the desired value.

Fields

Field	Description
Data Name	Optional. Type the data variable name to be selected. If a fully-qualified data name is entered, a message displays that confirms selection of the specified data name. You can also enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). Field help for the Data Name field may be obtained by entering a question mark (?) as the first or only character in this field.
Aliasing	Optional. Specify Aliasing to include data names that are aliases of the specified data name. If left blank, aliases of the specified data name are not included in the list of selectable data names. The default is no aliasing. Aliases consist of these: • Parent - higher level group item • Child - lower level item • Rename/redefinition - renamed, redefined or 88 level items
Level Number	Optional. Enter a valid data name level number. The list of selectable data names on the Extract - Data Name List pop-up (see Figure 68 on page 148) includes only those at the specified level. If left blank, the list is not restricted by level.
COPYBOOK Name	Optional. Enter the name of a copy book member included in the program. The list of selectable data names on the Extract - Data Name List pop-up includes only those that reference data names defined in the copy book member. If left blank, the list is not restricted by COPY member. Field help for the COPYBOOK Name field may be obtained by entering a question mark (?) as the only character in this field.
Line Range	Optional. Enter a single line number or a range of lines. If left blank, the list is not restricted by line range. Field help for the Line Range field may be obtained by entering a question mark (?) as the only character in this field.

Field	Description
Label Name	Optional. Enter a PROCEDURE DIVISION paragraph or section name. The list of selectable data names on the Extract - Data Name List pop-up includes only those contained in the specified paragraph or section. If left blank, the list is not restricted by label. Field help for the Label Name field may be obtained by entering a question mark (?) as the only character in this field.
COBOL Subset	Optional. Enter a predefined COBOL language subset. If left blank, the list is not restricted by subset type. For example, to list data names containing input or output statements, enter the COBOL subset type IO. See "COBOL Subsets" on page 17 for a list of all the COBOL subsets. The default is COND. Subsets can be concatenated by placing a plus sign (+) between them. Field help for the COBOL subsets field may be obtained by entering a question mark (?) as the only character in this field.

Usage Notes

A COBOL program contains a number of data variable names. If the data variable name to be extracted is not known, the Extract - Transaction Variable pop-up (see Figure 75 on page 167) can be used to reduce the number of data names listed on the Extract - Data Name List pop-up (see Figure 68 on page 148), so that selecting the desired data variable name is easier. The reduction of the list is accomplished by processing the qualifying criteria from the fields on this screen in this order:

- The list is reduced to those names that match the entered data name pattern. If a data name is not entered, and all other fields on the screen are left blank, all data names are included in the selectable list. Level Number, if entered, further restricts data names to the specified level number. If Aliasing is specified, the list of data names is expanded to include all data names that are aliases of the data names qualified by the entered Data Name.
- The list can be restricted by specifying a copy book name. The list is reduced to contain those data names that are defined or referenced within the copy book name specified.
- The list can be restricted by specifying a line range. The list is reduced to contain those data names that are referenced within the specified range of lines.
- The list can be restricted by specifying a paragraph label. The list is reduced to contain only those data names referenced within the paragraph specified.
- The list can be reduced by specifying one or more COBOL subsets. The list is reduced to contain only the data name references in statements corresponding to the specified COBOL subset type(s).

The resultant list of data names display on the Extract - Data Name List Filter pop-up (see Figure 74 on page 164).

Extract - Sort Data Name Pop-up

The Extract - Sort Data Name pop-up, shown in Figure 76, is used to specify the sort column for the Extract - Data Name List Filter pop-up (see Figure 74 on page 164).

To display the Extract - Sort Data Name pop-up, follow this step:

- Enter the SORT command on the Extract - Data Name List Filter pop-up.

Figure 76 • Extract - Sort Data Name Pop-up - List Filter

```
                    Extract - Sort Data Name

Sort Column                          Sort Order
__  1.  Level Number                 /  Ascending
    2.  Data Name
    3.  Defined Source Line
    4.  Defined SECTION
    5.  Size
    6.  REDEFINEs
    7.  RENAMEs
    8.  OCCURs
    9.  DEADDATA
    10. Use in CALLs
    11. Use in SQL
    12. Use in CICS
    13. Use in IDMS
```

Fields

Field	Description
Sort Column	Type the number of the column to be sorted. The default is column 2, Data Name.
Ascending	Optional. Specify Ascending if the entries on the Extract - Data Name List pop-up are to be sorted in ascending order. This is the default. To sort the entries in descending order, enter a blank space to remove the slash (/).

Statement Extract

The purpose of the Statement extract objective is to isolate a specific COBOL source statement.

To accomplish the Statement extract objective

1. Analyze the program, if this has not already been done. The desired program is analyzed by selecting File ▸ Analyze from the File pull-down (see Figure 9 on page 26), or one of the methods described in Chapter 12, "Analyze Functions," on page 431.

2. The statement(s) must then be selected on the Extract - Source Statement List pop-up (see Figure 77 on page 171). To filter the list of selectable statements, press PF5 on the Extract - Source Statement List pop-up.

A Statement extract is initiated by selecting Extract ▸ Statement from the Extract pull-down (see Figure 56 on page 124).

The pop-ups and screens used to complete the Statement extract objective are described in this section in the order that they are used.

Extract - Source Statement List Pop-up

The Extract - Source Statement List pop-up, shown in Figure 77, is used to provide a list of source lines of the COBOL program from which to select individual statements.

Figure 77 • Extract - Source Statement List Pop-up

```
                        Extract - Source Statement List
Command ===> ____________________________________________ Scroll ===> CSR

Select desired Source statements. Press Enter when complete.

  S,SS - Select      V - View Source      U,UU - Unselect

   Line   Source Statement
-- ------ ----------------------------------------------------------------
__ 000225  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,
__ 000226                           BANK-BALANCE.
__ 000227
__ 000228      OPEN OUTPUT EXCEPTION-FILE.
__ 000229
__ 000230      PERFORM INITIALIZE-PGM.
__ 000231
__ 000232      PERFORM ACCT-MAINTENANCE
__ 000233         THRU ACCT-MAINTENANCE-EXIT
__ 000234        UNTIL TRAN-FILE-ENDS.
__ 000235

     PF5=Filter  PF16=And/Or
```

To display the Extract - Source Statement List pop-up

1. Select Extract ▸ Statement from the Extract pull-down and press Enter. The Enter on the Extract - Name Logic Segment pop-up displays (see Figure 57 on page 126).

2. Press Enter again.

When this pop-up initially displays, all statements in the program are listed. To filter the list, press PF5 to display the Extract - Source Statement Filter pop-up (see Figure 78 on page 174).

To select the desired statement(s), use the S or SS line command. More than one statement can be selected. Selected statements are highlighted.

Fields

Field	Description
Line command area	The line command area, to the left of the Line field, accepts these line commands: • S, SS - Selects the identified source statement and highlights it. To select a block of statements, type the `SS` command in the line command areas for the statements at the beginning and end of the block. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the COBOL source code of the program. The statement specified to be viewed is highlighted and tagged to the right with LINE RNG. Use the END command to return to the Extract - Source Statement List pop-up (see Figure 77 on page 171). • U, UU - Unselects a previously selected statement and removes the highlighting. To unselect a block of statements, type the `UU` command in the line command areas for the statements at the beginning and end of the block.
Line	The source code line number of the statement.
Source Statement	The COBOL source code statement. Statements that have been selected so far are highlighted.

PF Keys

Field	Description
PF5=Filter	Displays the Extract - Statement Filter pop-up (see Figure 78 on page 174) used to filter the list of statements that appear on the Extract - Source Statement List pop-up.
PF16=AND/OR	Displays the Statement - AND/OR Logic Segment pop-up (see Figure 79 on page 177) to select or unselect statements using previously defined Logic Segments.

Usage Notes

A COBOL program contains a number of source statements. If the statements to be extracted is not known, the Extract - Statement Filter pop-up (see Figure 78 on page 174) can be used to reduce the number of source statements listed on the Extract - Source Statement List pop-up (see Figure 77 on page 171), so that selecting the desired statements is easier. The reduction of the list is accomplished by processing the qualifying criteria from the fields on this screen in this order:

- The list is reduced to those statements that reference the entered data name. If a data name is not entered, and all other fields on the pop-up are left blank, all statements are included in the selectable list. The Use Context field selections restricts the list to only those data names that are referenced, used, or modified. If Aliasing is specified, the list of statements is expanded to include all statements that reference aliases of the data names qualified by the entered data name.
- The list can be restricted by specifying a copy book name. The list is reduced to include only those that reference data names defined in the copy book member. If left blank, the list is not restricted by COPY member.
- The list can be restricted by specifying a line range. The list is reduced to contain only those statements within the specified range of lines.
- The list can be restricted by specifying a paragraph label. The list is reduced to contain only those statements that are contained within the paragraph label. The list can be further reduced by specifying a COBOL subset. The list is restricted to source statements that correspond to the specified COBOL subset type(s).

Extract - Statement Filter Pop-up

The purpose of the Extract - Statement Filter pop-up, shown in Figure 78, is to qualify PROCEDURE DIVISION statements for selection.

Figure 78 • Extract - Statement Filter Pop-up

```
                        Extract - Statement Filter

Type information for filtering PROCEDURE Division statements.

Data Name  . . ------------------------------------------------------
Use Context                  Options
1   1.  References           /  Aliasing
    2.  Uses
    3.  Modifications

Copy Book Name    --------
Line Range  . . . ------------------------------------------------------
Label Name  . . . ------------------------------------------------------
COBOL Subset  . . ------------------------------------------------------
```

To display the the Extract - Statement Filter pop-up, follow this step:

- Press PF5 on the Extract - Source Statement List pop-up (see Figure 77 on page 171).

When this pop-up initially displays, each field contains its default values. If these defaults are changed, the new values are retained for subsequent sessions.

A question mark (?) can be entered in text fields to display a help pop-up about that particular field. The help pop-ups provide information about the field, list the valid values, and allow input of the desired value.

Fields

Field	Description
Data Name	Optional. Enter a data variable name. If Data Name is left blank, all data names are included on the Extract - Source Statement List pop-up (see Figure 77 on page 171). You can also enter a pattern with wildcard characters (a question mark (?) for one character, an asterisk (*) for zero or more characters). Field help for the Data Name field may be obtained by entering a question mark (?) as the only character in this field.
Use Context	Optional. Enter a qualifier to restrict the list of selectable source statements. The default is 1. References. 1. References - The list of selectable statements only includes occurrences of the data name where its value is tested, used, set or modified. 2. Uses - The list of selectable statements only includes occurrences of the data name where its value is being tested or used. 3. Modifications - The list of selectable statements only includes occurrences of the data name where its value is being set or modified.
Aliasing	Optional. If specified, the list of selectable source statements on the Extract - Source Statement List pop-up includes those with data names that are aliases of the specified data name. If left blank, statements containing aliases of the data name are not included in the selectable list. Aliases include these: • Parent - higher level group item • Child - lower level item • Rename/redefinition - renamed, redefined or 88 level items The default is no aliasing. **Note:** The Use Context and Aliasing fields are ignored if the Data Name field is left blank.

Field	Description
Copy Book Name	Optional. Type the name of a copy book member included in the program. The list of selectable statements on the Extract - Source Statement List pop-up (see Figure 77 on page 171) includes only those that reference data names defined in the copy book member. If left blank, the list is not restricted by COPY member. Field help for the Copy Book Name field may be obtained by entering a question mark (?) as the only character in this field.
Line Range	Optional. Enter a single line number or a range of lines. The list of selectable statements on the Extract - Source Statement List pop-up includes only those in the specified line range. If left blank, the list is not restricted by line range. Field help for the Line Range field may be obtained by entering a question mark (?) as the only character in this field.
Label Name	Optional. Enter a PROCEDURE DIVISION paragraph or section name. The list of selectable statements on the Extract - Source Statement List pop-up includes only those contained in the specified paragraph or section. If left blank, the list is not restricted by label. Field help for the Label Name field may be obtained by entering a question mark (?) as the only character in this field.
COBOL Subset	Optional. Enter a predefined COBOL language subset. The list of selectable statements on the Extract - Source Statement List pop-up includes only those corresponding to the specified subset type. If left blank, the list is not restricted by subset type. Subsets can be concatenated by placing a plus sign (+) between them. For example, to list input and output statements, enter the COBOL subset type IO. See "COBOL Subsets" on page 17 for a list of the COBOL subsets. Field help for the COBOL Subset field may be obtained by entering a question mark (?) as the only character in this field.

Statement - And/Or Logic Segment Pop-up

The Statement - And/Or Logic Segment pop-up, shown in Figure 79, is used to populate the Statement extract criteria using an existing Logic Segment.

Figure 79 • Statement - And/Or Logic Segment Pop-up

```
                        Statement - And/Or Logic Segment    4 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

The Statement Selection will be modified based on the operator.
   * - AND      + - OR      - - DIFF      X - XOR

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
_  VIARDEMO   PERFORM-RANGE-0001             Perform Range
_  VIARDEMO   TRANSACTION-0007               Transaction
_  VIARDEMO   TRANSACTION-0009               Transaction
_  VIARDEMO   TRANSACTION-0010               Transaction
******************************** BOTTOM OF DATA ******************************
```

This pop-up can also be used to generate Statement extracts from Logic Segments created using the Test Driven extract feature (see "Test Driven Extract" on page 120).

To display the Statement - And/Or Logic Segment pop-up, follow this step:

- Press PF16 on the Extract - Source Statement List pop-up (see Figure 77 on page 171).

Fields

Field	Description
Line Commands	• * - AND - Type * next to the Program name to select statements that are in both the extract source statement list and the Logic Segment, or to remove the highlighting from selected statements in the source list that are in the specified Logic Segment. • + - OR - Type + next to the Program name to select source statements that are in the specified logic statement. • - - DIFF - Type − next to the Program name to select statements that are in both the extract source statement list and the Logic Segment, or to remove the highlighting from selected statements in the source list that are in the specified Logic Segment. • X - XOR - Type X next to the Program name to unselect statements that are selected in the source statement list or that are in the Logic Segment.

Field	Description
Program	The name of the program for which the Logic Segment was generated.
Logic Segment Name	The name of the Logic Segment.
Description	A description of the Logic Segment.

Complement Extract

The purpose of the Complement extract objective is to identify a set of COBOL source statements that do not include a previous extract. The Complement Logic Segment contains all statements from the original program, excluding statements of the previously-extracted Logic Segment, but retaining any statements needed by other functional paths.

To accomplish the Complement extract objective

1. Perform the desired extract type to contain the desired statements to be removed from the program and save the resulting Logic Segment in the AKR. See the previous sections in this chapter for specific information.

2. The Logic Segment is then selected on the Extract - Complement pop-up (see Figure 80 on page 179). For a Transaction Complement, logic paths are then selected on the Extract - Transaction Complement Paths pop-up (see Figure 81 on page 180).

A Complement extract is initiated by selecting Extract ▸ Complement from the Extract pull-down (see Figure 56 on page 124).

The pop-ups used to complete the Complement extract objective are described in this section in the order that they are used.

Note:

Logic segments that were defined using the Limit to within and existing segment option do not appear on this list. See "Extract - Computation Variable Filter Pop-up" on page 152 for more information.

Extract - Complement Pop-up

The Extract - Complement pop-up, shown in Figure 80, is used to select the Logic Segment to create a Complement Logic Segment.

Figure 80 • Extract - Complement Pop-up

```
                              Extract - Complement          4 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

Select segments for complement extract and press ENTER.  When selections
are complete, press Enter.
   S - Select           U - Unselect

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
_  VIARDEMO   PERFORM-RANGE-0001             Perform Range
_  VIARDEMO   TRANSACTION-0007               Transaction
_  VIARDEMO   TRANSACTION-0009               Transaction
_  VIARDEMO   TRANSACTION-0010               Transaction
******************************** BOTTOM OF DATA ******************************
```

To display the Extract - Complement pop-up

1. Select Extract ▸ Complement and press Enter. The Extract - Name Logic Segment pop-up displays (see Figure 57 on page 126).

2. Press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts these line commands: • S - Selects the Logic Segment to use for a Complement Logic Segment, consisting of all statements of the original program, excluding statements of the selected Logic Segment, but retaining any statements needed by other functional paths. • U - Unselects a previously-selected Logic Segment.
Program	The name of the program from which the Logic Segment was extracted.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).
Description	The type of extract that was used to create the Logic Segment.

Extract - Transaction Complement Paths Pop-up

The Extract - Transaction Complement Paths pop-up, shown in Figure 81, is used to select the logic path(s) that define the code for the Transaction Complement.

Figure 81 • Extract - Transaction Complement Paths Pop-up

```
                    Extract - Transaction Complement Paths
Command ===> ____________________________________________ Scroll ===> CSR

For the transaction complement, identify those conditionals which are
definitely TRUE, those which are definitely FALSE, and leave blank
those which may be either TRUE or FALSE.  Then press Enter.

    T - TRUE      F - FALSE      blank - TRUE/FALSE     V - View Source
- - ------ ------------------------------------------------------------------
_   000266      IF ACCT-MIN-BALANCE IS LESS THAN ACCT-BALANCE
_   000277      IF ACCT-TYPE OF ACCT-RECORD = 'INT'
_   000279      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
_   000281      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
_   000283      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
_   000286      IF ACCT-MULTI-ACCT = 'YES'
_   000291      IF ACCT-MIN-BALANCE < FREE-SERVICE-MIN
_   000299      IF ACCT-MIN-BALANCE < 0
_   000332      IF EX-RPT-LINE-CNT IS GREATER THAN 60
_   000382      IF  FREE-SERVICE-MIN NOT NUMERIC
_   000384      IF  CD-REVIEW-MIN    NOT NUMERIC
 ASG4345I SEGMENT(S) "TRANSACTION-0007" SELECTED.
```

To display the Extract - Transaction Complement Paths pop-up, follow this step:

- Select a Logic Segment on the Extract Compliment pop-up (see Figure 80 on page 179) that uses a Transaction extract Logic Segment.

This screen displays in source order the conditional statements that reference one or more of the data names previously identified for the Transaction extract objective. The conditional statements displayed include IF, EVALUATE, and GO TO DEPENDING ON. This pop-up is similar to the Extract - Transaction Paths List pop-up (see Figure 72 on page 159) used to define the Transaction extract objective.

The initial display of this pop-up shows all paths selected and highlighted (i.e., TRUE/FALSE). The T or F line command must then be entered on relevant conditional statements. See "Usage Notes" on page 181 for more information.

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • T - Identifies the indicated path as TRUE, and includes the code as part of the transaction. • F - Identifies the indicated path as FALSE, and excludes the code from the transaction. • blank - Identifies the indicated path as either TRUE or FALSE, and includes the code as part of the transaction. • V - Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the conditional statement in the context of the COBOL source code.
Line number	The source code line number of the statement. Encore-generated path lines are indicated by..... instead of a number.
Statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

Usage Notes

To define the transaction complement, specify for each conditional identified for the Transaction objective Logic Segment whether that statement is definitely TRUE, definitely FALSE, or indeterminate (i.e., could be either TRUE or FALSE). Selection can be the TRUE or FALSE branches of an IF statement, individual branch paths of a GO TO DEPENDING ON statement, and individual WHEN conditions of an EVALUATE statement.

- For IF statements, indicate whether the statement is TRUE, FALSE, or either. Indicating that the statement is TRUE causes the code of the THEN clause to be included in the transaction, and the code of the ELSE clause to be excluded. If no ELSE clauses exists, no code is excluded. Indicating that the statement is FALSE causes the THEN clause code to be excluded, and the ELSE clause code (if it exists) to be included.
- For EVALUATE statements, indicate which of the WHEN conditions are TRUE and which are FALSE. The EVALUATE statement displays to show the context of the WHEN statements, but no TRUE/FALSE indication is allowed for the actual EVALUATE statement. Specifying TRUE for a WHEN statement condition includes the code invoked by the WHEN. Specifying FALSE excludes the code.

- GO TO DEPENDING ON statements control various numbers of paths. Encore provides one line for each possible value of the DEPENDING variable. The value is indicated by `=n`, where `n` is the number corresponding to the position in the GO TO statement. Following the number is the label name that represents the flow for the indicated number value. To specify that code should be included, specify the number is TRUE. To exclude code, specify the number is FALSE.

Also displayed on this screen is the identification that was specified for the transaction on the Extract - Transaction Paths List pop-up (see Figure 72 on page 159). This information is provided to assist in relating the transaction results to the Transaction Complement results, determining which conditionals of the complement are TRUE and which are FALSE, based on the specification for the transaction.

Common Code Extract

The purpose of the Common Code extract objective is to identify a set of COBOL source statements that are common to two or more previously-extracted Logic Segments. The Common Code Logic Segment contains all statements that appear in a selected Logic Segment, as well as appearing in all other selected Logic Segments.

To accomplish the Common Code extract objective

1 Perform the desired extracts and save the resulting Logic Segments in the AKR. See the previous sections in this chapter for specific information.

2 The Logic Segments are then selected on the Extract - Common Code pop-up (see Figure 82 on page 183).

A Common Code extract is initiated by selecting Extract ▸ Common Code from the Extract pull-down (see Figure 56 on page 124).

Figure 82 on page 183 is used to complete the Common Code extract objective and is described in this section.

Extract - Common Code Pop-up

The Extract - Common Code pop-up, shown in Figure 82, is used to select the Logic Segments to create a Common Code Logic Segment.

Figure 82 • Extract - Common Code Pop-up

```
                          Extract - Common Code          8 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

Select segments for common code extract.  When selections are complete,
press Enter.
   S - Select           U - Unselect

   Program    Logic Segment Name              Description
-  ---------- ------------------------------- ------------------------------
_  VIARDEMO   COMPLEMENT-0012                 Complement Perform Range
_  VIARDEMO   COMPLEMENT-0014                 Complement Perform Range
_  VIARDEMO   COMPLEMENT-0018                 Complement Perform Range
_  VIARDEMO   COMPLEMENT-0019                 Complement Transaction
_  VIARDEMO   PERFORM-RANGE-0001              Perform Range
_  VIARDEMO   TRANSACTION-0007                Transaction
_  VIARDEMO   TRANSACTION-0009                Transaction
_  VIARDEMO   TRANSACTION-0010                Transaction
******************************** BOTTOM OF DATA ******************************
```

To display the Extract - Common Code pop-up

1. Select Extract ▸ Common Code from the Extract pull-down (see Figure 56 on page 124) and press Enter. The Extract - Name Logic Segment pop-up displays (see Figure 57 on page 126).

2. Press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts these line commands: • S - Selects the Logic Segment to use for a Common Code Logic Segment. At least two Logic Segments must be selected. • U - Unselects a previously-selected Logic Segment
Program	The name of the program from which the Logic Segment was extracted.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).
Description	A description of the type of extract that was used to create the Logic Segment.

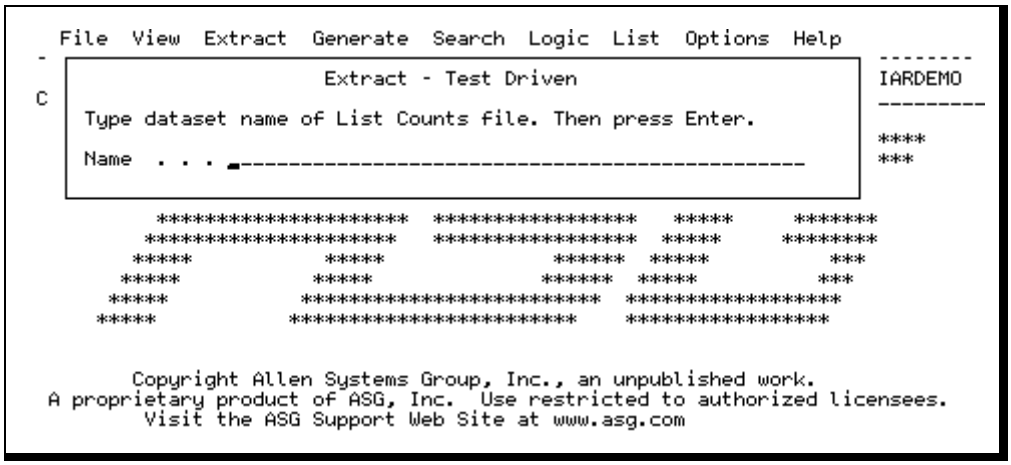
Test Driven Extract

The purpose of the Test Driven extract is to create a Logic Segment from an input file created during a SmartTest test session. The Test Driven Logic Segment contains all statements that were executed during a SmartTest test session.

Extract - Test Driven Pop-up

The Extract - Test Driven pop-up, shown in Figure 83, is used to import a file created by a previous SmartTest test session by entering an LPRINT on a LIST COUNTS screen.

Figure 83 • Extract - Test Driven Pop-up



To display the Extract - Test Driven pop-up

1. Select Extract ▸ Test Driven and press Enter. Tthe default Extract - Name Logic Segment selection pop-up displays.

2. Press Enter.

The Test Driven extract is based upon the run-time behavior of a program. In the context of Encore, the Logic Segment is comprised of statements that were executed during a particular execution run of a program.

Fields

Field	Description
Name	Required. Enter the name of the file that you want to import. The content of the file is used to create a Logic Segment.

To accomplish the Test Driven extract objective from SmartTest

1. Perform the desired SmartTest test scenario, making sure that the List Counts option is active.

2. At the completion of the test scenario, from the List Counts panel, type `LPRINT *`. This creates a file containing the information of the statement execution information, as represented by the List Counts screen. The name of the file must be retained for entry into Encore.

3. The name of this file is acquired by doing a PLOG command and processing the list file.

To accomplish the Test Driven extract objective from Encore

1. Analyze the program that was used in the SmartTest test session, if this has not already been done as an Encore analyze. The desired program is analyzed using the File ▸ Analyze selection from the File pull-down (see Figure 9 on page 26), or one of the methods described in Chapter 12, "Analyze Functions," on page 431.

2. Open the program to Encore.

3. Type the dataset name of the file created by the SmartTest test session on the Extract - Test Driven pop-up (see Figure 83 on page 184) to generate the Logic Segment.

Figure 84 on page 187 is used to complete the Test Driven extract objective and is described in this section.

Usage Notes

To generating a COBOL module from a Test Driven extract

1. Use the AND/OR feature on the Extract - Source Statement List pop-up (see Figure 77 on page 171) to select all statements in the Test Driven extract.

2. Using the selected source statements as the extract criteria, generate the source Statement extract. The resulting Logic Segment contains the code generated from the Test Driven extract results.

Test Driven extract advantages

The results of a Test Driven extract are completely dependent on the data that was used to run the program. If you intended to capture specific functionality, and the data did not trigger those statements, then they are not part of the resultant extract.

In the case of a Computation Variable or Report extract objectives, the resultant extract is based on a static analysis or "slice" of the program; only those statements that value or influence the value of the selected variable are extracted. For a Test Driven extract the resultant extract is based on a dynamic slice of the program; only those statements that are executed during a particular execution-run of the program are extracted.

As a way of comparing the results of a static slice as opposed to a dynamic slice, assume this program fragment:

```
100         ACCEPT X.
200         ACCEPT Y.
300         IF X > 100 THEN
400             DISPLAY 'X = ', X.
500         IF X > Y THEN
600             COMPUTE Z = X + Y
700         ELSE
800             COMPUTE Z = X - Y.
900         DISPLAY Z.
```

For the static slice (on line 900, DISPLAY Z.), these statements are extracted:

```
100         ACCEPT X.
200         ACCEPT Y.
500         IF X > Y THEN
600             COMPUTE Z = X + Y
700         ELSE
800             COMPUTE Z = X - Y.
900         DISPLAY Z.
```

Notice that lines in 500 and 600, the code checking for X being greater than 100 is missing, since by strict definition of slicing they did not contribute to the value of Z at line 900. Nevertheless, it could be argued that the code checking for the value of X is crucial.

For the dynamic slice, the set of statements that are extracted is dependent on this input data:

- If X is always greater than Y, then Z is always the sum of X and Y, i.e. the statement at line 800, `COMPUTE Z = X - Y` is never part of the dynamic slice.
- If X is greater than 100, the IF statement at lines 300 and 400 is included.

The Test Driven extract objective can play a fundamental role in the understanding phase of the re-engineering process. It allows you to focus on the core code executed when a specific transaction or set of input data is selected.

This speeds the understanding of what program re-engineering does for a complete extraction. After you understand the executed core code, you can perform a complete extract, transaction driven for example, so that all the necessary supporting code (i.e., error routines and exception handling) can also be included. In addition, the extracted Logic Segment can be used in viewing common code, thereby making it a simple task, for example, to see what code is shared between two transactions.

Extract - Edit Segment Pop-up

The Extract - Edit Segment pop-up, shown in Figure 84, is used to select the Logic Segment to edit. This pop-up displays by selecting Extract ▸ Edit segment criteria from the Extract pull-down (see Figure 56 on page 124).

Figure 84 • Extract - Edit Segment Pop-up

```
                              Extract - Edit Segment         1 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
S  VIARDEMO   PERFORM-RANGE-0009             Perform Range
******************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select the Logic Segment.
Program	The name of the program from which the Logic Segment was extracted.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).
Description	The description of the type of extract that was used to create the Logic Segment.

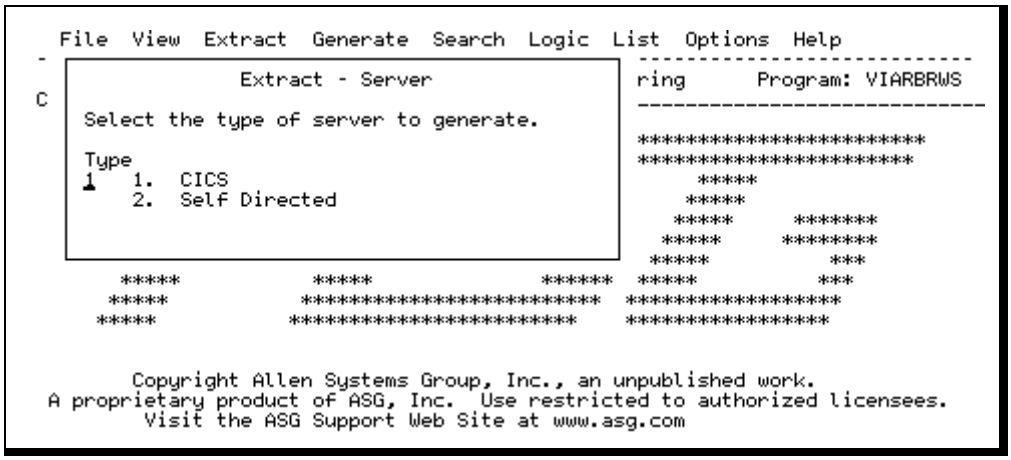
CICS Server Extract

Encore provides the capability to separate presentation logic and business logic in both conversational and pseudo-conversational CICS programs. The resulting program is the CICS Server program, which contains business logic. Once isolated from the presentation logic, the server program becomes a business logic component that can be invoked from a Web server, as well as other types of clients (e.g., Java servlets, MQSeries, and CICS clients on other platforms).

Extract - Server Pop-up - CICS Selection

The Extract - Server pop-up, shown in Figure 85, is used to select the type of server to be generated. Select 1. CICS to create a CICS server, or 2. Self Directed to designate any CICS SEND or CICS RECEIVE statements for the Endpoints and Startpoints.

Figure 85 • Extract - Server Pop-up - CICS Selection



To display the Extract - Server pop-up, follow this step:

- Select Extract ▸ Server and press Enter (see Figure 56 on page 124).

Fields

Field	Description
Type	The type of extract that is used to create the server: 1. CICS - indicates that a CICS server is to be created from the extract. 2. Self Directed - indicates that a self-directed server is to be created from the extract.

Extract - CICS Server Endpoints Pop-up

The Extract - CICS Server Endpoints pop-up, shown in Figure 86, is used to select the statement for each CICS Server Endpoint.

Figure 86 • Extract - CICS Server Endpoints Pop-up

```
                        Extract - CICS Server Endpoints
Command ===> ____________________________________________ Scroll ===> CSR

Select the statement for each CICS Server Endpoint and press Enter to view
the corresponding Startpoints.  This screen will be redisplayed to allow
additional Endpoints to be selected.  Press Enter when all selections are
complete.

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
_ 000481      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW')
  000482           ERASE END-EXEC.
_ 000615      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW')
  000616           ERASE END-EXEC.
_ 000652      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW') END-EXEC.
_ 000672      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW')
  000673           ERASE END-EXEC.
_ 000694      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW')
  000695           ERASE END-EXEC.
_ 000735      EXEC CICS SEND MAP('MENU') MAPSET('VIARMNU') ERASE END-EXEC.
******************************** BOTTOM OF DATA ******************************
```

To display the Extract - CICS Server Endpoints pop-up, follow this step:

- Select CICS on the Extract Server pop-up (see Figure 85 on page 188).

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the statement for each CICS server endpoint and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements of the specified extracted CICS server endpoints are highlighted, and tagged to the right with LINE RNG. • U - Unselects a previously selected CICS server endpoint statement. The name is no longer highlighted.
line number	The source code line number of the statement. Encore-generated path lines are indicated by..... instead of a number.
statement	The EXEC CICS SEND MAP statements.

Extract - CICS Server Startpoints Pop-up

The Extract - CICS Server Startpoints pop-up, shown in Figure 87, is used to select the statements for CICS Server Startpoints.

Figure 87 • Extract - CICS Server Startpoints Pop-up

```
                        Extract - CICS Server Startpoints
Command ===> _____________________________________________ Scroll ===> CSR

Select statements for CICS Server Startpoints and press Enter.  Press Enter
when selections are complete to return to the Endpoints screen.

Endpoint

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
_ 000629      EXEC CICS RECEIVE MAP('BROWSE') MAPSET('VIARBRW')
  000630           RESP(RESPONSE) END-EXEC.
******************************** BOTTOM OF DATA ****************************
```

To display the Extract - CICS Server Startpoints pop-up, follow this step:

- Select a CICS Server Endpoint on the Extract - CICS Server Endpoints pop-up (see Figure 86 on page 189) and press Enter. After making your selection, you are returned to the Extract - CICS Server Endpoints pop-up.

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the statement for each CICS server startpoint and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements of the specified extracted CICS server endpoints are highlighted, and tagged to the right with LINE RNG. • U - Unselects a previously selected CICS server endpoint statement. The name is no longer highlighted.
Line number	The source code line number of the statement. Encore-generated path lines are indicated by..... instead of a number.
Statement	The EXEC CICS RECEIVE MAP statements.

Extract - CICS Server Paths Pop-up

The Extract - CICS Server Paths pop-up, shown in Figure 88, is used to identify conditionals that are definitely TRUE (T) or FALSE (F) for the server, or to leave blank those conditionals which may be either.

Figure 88 • Extract - CICS Server Paths Pop-up

```
                         Extract - CICS Server Paths
Command ===> ____________________________________________ Scroll ===> CSR

Identify conditionals which are definitely TRUE for the Server, those which
are definitely FALSE, and leave blank the conditionals which may be either
TRUE or FALSE.  Press Enter when all selections are complete.

  T - TRUE     F - FALSE     blank - TRUE/FALSE     V - View Source
- ------ ------------------------------------------------------------------
T 000298      IF EIBCALEN NOT = 0 THEN
X 000377      IF EIBCALEN = 0 THEN GO TO NEXT-LINE.
_ 000506      IF EIBCALEN = 0 THEN GO TO TEST-STATS.
F 000634      IF EIBAID = DFHCLEAR THEN
F 000640      IF EIBAID = DFHPF1 OR DIRI = 'F' THEN GO TO PAGE-FORWARD.
T 000641      IF EIBAID = DFHPF2 OR DIRI = 'B' THEN GO TO PAGE-BACKWARD.
********************************* BOTTOM OF DATA ******************************
```

To display the Extract - CICS Server Paths pop-up, follow this step:

- Press Enter after returning to the Extract - CICS Server Endpoints pop-up (see Figure 86 on page 189).

Fields

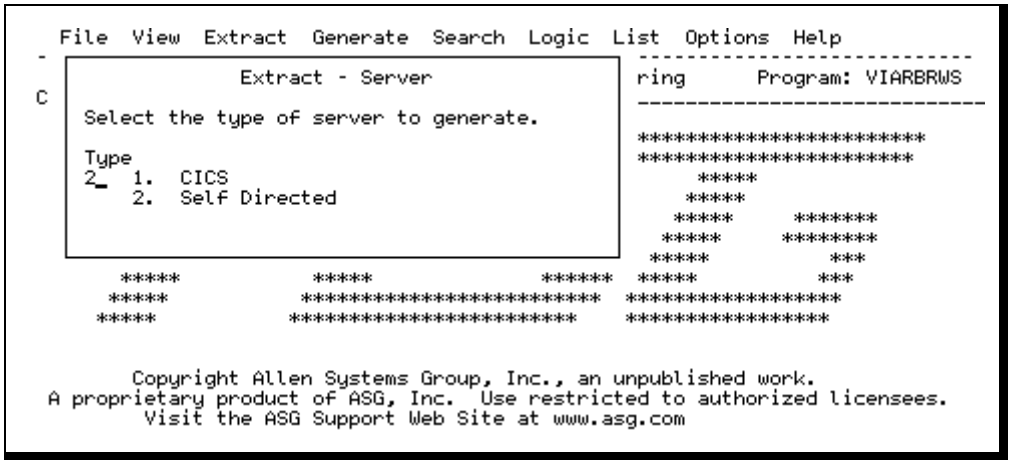
Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • T - Identifies the indicated path as TRUE and includes the code as part of the transaction. • F - Identifies the indicated path as FALSE and excludes the code from the transaction. • blank - Identifies the indicated path as either TRUE or FALSE and includes the code as part of the transaction.

Field	Description
	• V - Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the conditional statement positioned in the middle of the screen to see the statement in context. The statement is highlighted and tagged to the right with LINE RNG. Use the END command to return to the Extract - CICS Server Paths pop-up. • An X displays in the line command area for conditionals not designated as T or F, and which are blocked by another conditional and therefore not reachable. In this case, the line command field is protected and cannot be changed. **Note:** If a conditional is designated as T or F, but is blocked by another conditional, an error message displays.
Line number	The source code line number of the statement. Encore-generated path lines are indicated by..... instead of a number.
Statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

Extract - Server Pop-up - Self-Directed Selection

The Extract - Server pop-up, shown in Figure 89, is used to select the type of server to be generated.

Figure 89 • Extract - Server Pop-up - Self Directed Selection



To display the Extract - Server pop-up, follow this stepL

- Select Server on the Extract pull-down (see Figure 56 on page 124).

Select 1. CICS to generate a CICS server, or 2. Self Directed to designate any CICS SEND or CICS RECEIVE statements for the Endpoints and Startpoints.

Fields

Field	Description
Type	The type of extract that is used to create the server: 1. CICS - displays only CICS server extraction screens. 2. Self Directed - displays only self-directed server extraction screen.

Extract - Self Directed Server Endpoints Pop-up

The Extract - Self Directed Server Endpoints pop-up, shown in Figure 90, is used to select the statement for each Self Directed Server Endpoint.

Figure 90 • Extract - Self Directed Server Endpoints Pop-up

```
                    Extract - Self Directed Server Endpoints
Command ===> ____________________________________________ Scroll ===> CSR

Select the statement for each Server Endpoint and press Enter to select the
appropriate Data Item(s).  Press Enter when all selections are complete.

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
  000296 *    WORKING STORAGE IN THE PROGRAM.
  000297 *
_ 000298      IF EIBCALEN NOT = 0 THEN
_ 000299         MOVE DFHCOMMAREA TO COMMAREA
  000300         GO TO PROMPT.
  000301 *
  000302 *    THIS COMMAND MAPS IN THE ACCOUNT NUMBER FROM THE OPERATOR
  000303 *    INSTRUCTION MENU. NOTE THE EXPLICIT TESTING OF THE RESPONSE
  000304 *    TO THE COMMAND.
  000305 *
_ 000306      EXEC CICS RECEIVE MAP('MENU') MAPSET('VIARMNU')
  000307           RESP(RESPONSE) END-EXEC.

  PF5=Filter
```

To display the Extract - Self Directed Server Endpoints pop-up, follow this step:

- Select Self Directed on the Extract - Server pop-up (see Figure 89 on page 192).

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the data name for each self directed server endpoint and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements of the specified extracted self directed server endpoints are highlighted, and tagged to the right with LINE RNG. • U - Unselects a previously selected self directed server endpoint statement. The name is no longer highlighted.
line number	The source code line number of the statement. Encore-generated path lines are indicated by..... instead of a number.
statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.
PF5=Filter	Displays the Extract - Self Directed Server Filter pop-up (see Figure 91 on page 194) used to filter the list of statements that appear on the Extract - Self Directed Server Endpoints pop-up.

Extract - Self Directed Server Filter Pop-up

The Extract - Self Directed Server Filter pop-up, shown in Figure 91, is used to filter the variable selection list that displays on the Extract - Self Directed Server Endpoints (see Figure 90 on page 193) or Extract - Self Directed Server Startpoints pop-ups (see Figure 92 on page 196).

Figure 91 • Extract - Self Directed Server Filter Pop-up

```
                    Extract - Self Directed Server Filter

      Enter the desired Filtering parameters and press Enter.

      Data Name  . . _
      Use Context                    Options
      2   1.  References             /  Aliasing
          2.  Uses
          3.  Modifications

      Copy Book Name    ________
      Line Range  . . . ____________________________________________________
      Label Name  . . . ____________________________________________________
      COBOL Subset  . . ____________________________________________________
```

To display the Extract - Self Directed Server Filter pop-up, follow this step:

- Press PF5 on the Extract - Self Directed Server Endpoints pop-up.

Fields

Field	Description
Data Name	Optional. Type the name of the data item you want to appear on the selection list. Enter a name pattern with embedded wild card characters (a question mark (?) for one character, an asterisk (*) for zero or more characters) to list all data items with the name pattern. For example, enter `A*` to list all data names that begin with A. If the Data Name field is left blank, the list of selectable items includes all data names meeting the criteria entered on this pop-up. The desired data name can then be selected from the list.
Use Context	Optional. Enter a qualifier to identify a specific use context of the data name. Depending on the option selected, the list of selectable data names on the Extract - Self Directed Server Endpoints pop-up (see Figure 90 on page 193) includes only those data names that are referenced, used, or modified. The default is 1. References. 1. References - The list of selectable data names only includes those that test, use, set or modify the specified data name. 2. Uses - The list of selectable data names only includes occurrences that test or use the specified data name. 3. Modifications - The list of selectable data names only includes occurrences that modify the specified data name.
Aliasing	Optional. If specified, the list of selectable data names includes those that are aliases of the specified data name. If left blank, data names containing aliases of the specified data name are not included in the list. The default is no aliasing. Aliases consist of these: • Parent - higher level group item • Child - lower level item • Rename/redefinition - renamed, redefined, or 88 level items **Note:** The Use Context and Aliasing fields are ignored if the Data Name field is left blank.
COPYBOOK Name	Optional. Type the name of a copy book member included in the program. The list of selectable data names includes only those that reference data names defined in the copy book member.

Field	Description
Line Range	Optional. Enter a single line number or a range of lines. The list of selectable data names includes only those that are referenced within the specified line range.
Label Name	Optional. Enter a PROCEDURE DIVISION paragraph or section name. The list of selectable data names includes only those referenced in the specified paragraph or section.
COBOL Subset	Optional. Enter a predefined COBOL language subset. The list of selectable data names includes only those containing statements corresponding to the specified subset type.

Extract - Self Directed Server Data Names Pop-up

The Extract - Self Directed Server Data Names pop-up, shown in Figure 92, is used to select the data names for the selected server endpoint.

Figure 92 • Extract - Self Directed Server Data Names Pop-up

```
                    Extract - Self Directed Server Data Names
Command ===> ____________________________________________ Scroll ===> CSR

Select the Data Names for this Server Endpoint and press Enter.  Press
Enter when all selections are complete.

  S - Select      V - View Source      U - Unselect

  LV Data Name                        Qualification
- -- ------------------------------   ------------------------------------
_ 00 EIBCALEN                         00 DFHEIBLK
_ 00 BLL      1
******************************** BOTTOM OF DATA ****************************
```

To display the Extract - Self Directed Server Data Names pop-up, follow this step:

- Select an endpoint on the Extract - Self Directed Server Endpoints pop-up (see Figure 90 on page 193) and press Enter.

Fields

Field	Description
Line command area	The line command area, to the left of the LV field, accepts these line commands: • S - Selects the data name for each self directed server endpoint and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL data names in the specified self directed server endpoints are highlighted and tagged to the right with DATA DEF. • U - Unselects a previously-selected self directed server endpoint data name. The name is no longer highlighted.
LV	The COBOL level number of the data name.
Data Name	The name of the data variable. Data names that have already been selected are highlighted.
Qualification	The parent(s) of the data variable name. The qualifying parent name includes the level number and data name. The qualification displays only for qualified data variable names, (i.e., data names that are not unique). • A greater than (>) indicator denotes that the name is truncated on the display, and the rest of the name may be viewed by scrolling to the right. • A less than (<) indicator denotes that the display may be scrolled to the left.

Extract - Self Directed Server Startpoints Pop-up

The Extract - Self Directed Server Startpoints pop-up, shown in Figure 93, is used to display and select the EXEC RECEIVE MAP statements that correspond to the EXEC SEND MAP statements selected on the Extract - Self Directed Server Endpoints pop-up (see Figure 90 on page 193). Only those RECEIVE MAP statements whose control flows may reach the endpoints are shown.

Figure 93 • Extract - Self Directed Server Startpoints Pop-up

```
                    Extract - Self Directed Server Startpoints
Command ===> ____________________________________________ Scroll ===> CSR

Select the statement for each Server Startpoint and press Enter.  Press
Enter when all selections are complete.

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
  000296 *    WORKING STORAGE IN THE PROGRAM.
  000297 *
_ 000298      IF EIBCALEN NOT = 0 THEN
  000299         MOVE DFHCOMMAREA TO COMMAREA
  000300         GO TO PROMPT.
  000301 *
  000302 *    THIS COMMAND MAPS IN THE ACCOUNT NUMBER FROM THE OPERATOR
  000303 *    INSTRUCTION MENU. NOTE THE EXPLICIT TESTING OF THE RESPONSE
  000304 *    TO THE COMMAND.
  000305 *
  000306      EXEC CICS RECEIVE MAP('MENU') MAPSET('VIARMNU')
  000307           RESP(RESPONSE) END-EXEC.

  PF5=Filter
```

To display the Extract - Self Directed Server Startpoints pop-up, follow this step:

- Select a data name on the Extract - Self Directed Server Data Names pop-up (see Figure 92 on page 196).

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the statement for each self directed server startpoint and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements of the specified extracted self directed server startpoints are highlighted, and tagged to the right with LINE RNG. • U - Unselects a previously selected self directed server startpoint statement. The name is no longer highlighted.

Field	Description
Line number	The source code line number of the statement. Encore-generated path lines are indicated by ..... instead of a number.
Statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

PF Keys

Field	Description
PF5 = Filter	Displays the Extract - Self Directed Server Filter pop-up (see Figure 91 on page 194) used to filter the variable selection list that displays on the Extract - Self Directed Server Endpoints (see Figure 90 on page 193) pop-up or the Extract - Self Directed Server Startpoints pop-up (see Figure 93 on page 198).

Extract - Self Directed Server Paths Pop-up

The Extract - Self Directed Server Paths pop-up, shown in Figure 94, lists the conditional statements that contain references to system defined names (e.g., EIBCALEN) and are contained within the execution paths between the selected startpoints and endpoints.

Figure 94 • Extract - Self Directed Server Paths Pop-up

```
                    Extract - Self Directed Server Paths
Command ===> ____________________________________________ Scroll ===> CSR

Identify conditionals which are definitely TRUE for the Server, those which
are definitely FALSE, and leave blank the conditionals which may be either
TRUE or FALSE.  Press Enter when all selections are complete.

  T - TRUE     F - FALSE     blank - TRUE/FALSE     V - View Source
- ------ ------------------------------------------------------------------
F 000298      IF EIBCALEN NOT = 0 THEN
F 000311      IF RESPONSE = DFHRESP(MAPFAIL) THEN
F 000314      IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
_ 000320      IF KEYL NOT = ZERO THEN
T 000321         IF KEYI IS NUMERIC THEN
T 000350      IF RESPONSE = DFHRESP(NOTFND) THEN
_ 000353      IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
_ 000358      IF RIDF IN COMMAREA NOT EQUAL '999999' THEN
_ 000377      IF EIBCALEN = 0 THEN GO TO NEXT-LINE.
_ 000383      IF RESPONSE = DFHRESP(NOTFND) THEN
_ 000386      IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
_ 000391      IF RIDF IN COMMAREA = '000000' THEN GO TO NEXT-LINE.
_ 000408      IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
```

To display the Extract - Self Directed Server Paths pop-up, follow this step:

- Select a startpoint on the Extract - Self Directed Server Startpoints Names pop-up (see Figure 93 on page 198)

You should indicate which statements are TRUE and which statements are FALSE in relation to the server code. In particular, statements in the context, is this the first time?, should be resolved since the value contained within the system generated name is always non-zero. (Within the original program, on the first invocation of the program, the system generated name usually contains a zero value.)

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, identifies the indicated path as either TRUE or FALSE, and includes the code as part of the Logic Segment. The line command area accepts these line commands: • T - Indicates that the conditional is TRUE for the server. • F - Indicates that the conditional is FALSE for the server. • blank - Allows you to leave the field blank or enter either TRUE or FALSE for a given conditional. • V - Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar, showing the conditional statement positioned in the middle of the screen to see the statement in context. The statement is highlighted and tagged to the right with LINE RNG. • An X displays in the line command area for conditionals not designated as T or F, and which are blocked by another conditional and therefore not reachable. In this case, the line command field is protected and cannot be changed.
line number	The source code line number of the statement. Encore-generated path lines are indicated by ..... instead of a number.
statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

Extract - Self Directed Server Outputs Pop-up

The Extract - Self Directed Server Outputs pop-up, shown in Figure 95, is used to display the Output statements that are contained within the execution paths between the startpoints and endpoints. When initially displayed, all output statements are automatically selected. You can unselect any statements that are not to be included in the server module.

Figure 95 • Extract - Self Directed Server Outputs Pop-up

```
                    Extract - Self Directed Server Outputs
Command ===> ____________________________________________ Scroll ===> CSR

Select the Output statements to be included in the Server and press Enter.
Press Enter when all selections are complete.

  S - Select      V - View Source      U - Unselect
- ------ ------------------------------------------------------------------
  000296 *    WORKING STORAGE IN THE PROGRAM.
  000297 *
  000298      IF EIBCALEN NOT = 0 THEN
  000299         MOVE DFHCOMMAREA TO COMMAREA
  000300         GO TO PROMPT.
  000301 *
  000302 *    THIS COMMAND MAPS IN THE ACCOUNT NUMBER FROM THE OPERATOR
  000303 *    INSTRUCTION MENU. NOTE THE EXPLICIT TESTING OF THE RESPONSE
  000304 *    TO THE COMMAND.
  000305 *
  000306      EXEC CICS RECEIVE MAP('MENU') MAPSET('VIARMNU')
  000307           RESP(RESPONSE) END-EXEC.

  PF5=Filter
```

To display the Extract - Self Directed Server Outputs pop-up, follow this step:

- Press Enter on the Extract - Self Directed Server Paths pop-up (see Figure 94 on page 199)

Fields

Field	Description
Line command area	The line command area, to the left of the line number field, accepts these line commands: • S - Selects the statement for each self directed output statement and highlights it. • V - View source. Displays the View - Source pop-up (see Figure 35 on page 76) with a shortened action bar. The COBOL source statements of the specified extracted self directed output statement are highlighted, and tagged to the right with LINE RNG. • U - Unselects a previously selected self directed server output statement. The name is no longer highlighted.

Field	Description
Line number	The source code line number of the statement. Encore-generated path lines are indicated by ..... instead of a number.
Statement	The conditional statements and their corresponding paths. Each path of EVALUATE and GO TO DEPENDING ON statements is identified on a separate line.

PF Keys

Field	Description
PF5 = Filter	Displays the Extract - Self Directed Server Filter pop-up (see Figure 91 on page 194) used to filter the variable selection list that displays on the Extract - Self Directed Server Endpoints pop-up (see Figure 90 on page 193), the Extract - Self Directed Server Startpoints pop-up (see Figure 92 on page 196), or the Extract - Self Directed Server Outputs pop-up (see Figure 95 on page 201).

6 Generate Menu

This chapter describes Encore's Generate pull-down menu functions and contains these sections:

Section	Page
Introduction	203
Generate Pull-down	204
COBOL Modules	206
File IO Modules	230
Output Files	233
Batch Computation	236
CICS and Self-Directed Server Generation	241

Introduction

Each extract objective produces a Logic Segment, which is the collection of DATA DIVISION and PROCEDURE DIVISION statements required to define the criteria of the extract objective. After a Logic Segment has been defined through the Encore Extract facility, the next step is to specify the disposition of the Logic Segment.

A Logic Segment contains only the data items identified by the extract objective. The Logic Segment is used as input to the Generate facility, which generates the necessary statements to create a program or callable module from the Logic Segment. A generated Logic Segment may be viewed, edited, printed, saved, or used to create replacement or callable modules.

A complement module contains all statements from the original program, excluding statements of the extracted Logic Segment, but retaining any statements needed by other functional paths.

The Generate COBOL Module facility is not valid for a Common Code extract.

Generate Pull-down

Selecting Generate on the action bar displays the Generate pull-down, shown in Figure 96, which is used to generate a program or callable module, and to print or punch the Logic Segment.

Note:

Actions on the Generate pull-down cannot be selected unless there is an open program with an active Logic Segment.

Figure 96 • Generate Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
--------------------  1. COBOL module...        --------------------------
                      2. File IO Module...            Program: VIARDEMO
Command ===> _______  3. Output file...   ----------------------------
                      4. Batch Computation...
                      5. Server...
```

Copyright Allen Systems Group, Inc., an unpublished work.
A proprietary product of ASG, Inc. Use restricted to authorized licensees.
Visit the ASG Support Web Site at www.asg.com

Actions

Action	Description
1. COBOL module	Displays the Generate - Select Logic Segment pop-up (see Figure 97 on page 207) used to select the desired Logic Segment, then displays the Generate - COBOL Module pop-up (see Figure 98 on page 208) used to specify the disposition of the Logic Segment. The Generate COBOL Module facility is not valid for a Common Code extract.
2. File IO Module	Displays the Generate - FD Name List pop-up (see Figure 106 on page 230) used to select the name of the FD for the IO module generation.
3. Output file	Displays the Output File - Select Logic Segment pop-up (see Figure 108 on page 234) used to select the desired Logic Segment, then displays the Generate - Output File pop-up (see Figure 109 on page 235) used to print or punch the Logic Segment.
4. Batch Computation	Displays the Batch Computation - Select Logic Segments pop-up (see Figure 110 on page 236) used to select the Logic Segments for generation through a batch job.
5. Server	Displays the Generate - Select Logic Segment pop-up (see Figure on page 241) used to select the Server Logic Segments for generation.

COBOL Modules

The COBOL Module facility is used to create a syntactically correct COBOL source module from an extracted Logic Segment. The generated COBOL module contains a descriptive header containing information about the origin of the module, and comments throughout to assist in comprehending the contents of the module and in making any necessary adjustments.

The source code contents of each module varies, depending on the type of extract that was used to create the Logic Segment, and on certain options specified when generating the module.

After the module is generated, it is presented in an Edit session, which allows modification of the code.

See "COBOL Module Contents" on page 216 for more information.

Note:

The generated COBOL module is automatically saved or deleted, according to the value of the Save EDIT File field on the Options - Product Parameters pop-up (see Figure 141 on page 296). The generated COBOL module can also be saved by entering the ISPF CREATE command.

Generate - Select Logic Segment Pop-up

The Generate - Select Logic Segment pop-up, shown in Figure 97, is used to select the Logic Segment for generating a COBOL module. This pop-up displays by selecting Generate ▸ COBOL module from the Generate pull-down (see Figure 96 on page 204).

Figure 97 • Generate - Select Logic Segment Pop-up

```
                        Generate - Select Logic Segment     1 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
S  VIARDEMO   PERFORM-RANGE-0001             Perform Range
********************************* BOTTOM OF DATA ******************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select the Logic Segment. The Generate - COBOL Module pop-up (see Figure 98 on page 208) displays.
Program	The name of the program containing the Logic Segment definition.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up. See Figure 57 on page 126 for more information.
Description	The description of the extract that was used to create the Logic Segment.

Generate - COBOL Module Pop-up

The Generate - COBOL Module pop-up, shown in Figure 98, is used to create a new COBOL source module from the extracted Logic Segment. This pop-up displays after selecting a Logic Segment on the Generate - Select Logic Segment pop-up (see Figure 97 on page 207) and pressing Enter.

When this pop-up first displays, an informational message displays as a reminder that a default program ID has been assigned to the Logic Segment. This program ID should be verified for appropriateness and changed if necessary.

Figure 98 • Generate - Cobol Module Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-                                                  -----------------
C                 Generate - COBOL Module          Program: VIARDEMO
                                                   -------------------
  Type Program ID and select generation options.
                                                   **************
  Objective . . : Perform Range                    *************
  Program ID  . . VIARDEMV

  Options                                            *******
  _  CALLs to IO Module(s)                          ********
                                                       ***
                                                      ***

  ASG4312I DEFAULT PROGRAM ID VALUE ASSIGNED. VERIFY, AND CHANGE IF
  NECESSARY.

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

Note:

To set format and output options for the COBOL Modules generated by Encore, see the Options - Generate pop-up (Figure 150 on page 312) for more information.

Fields

Field	Description
Objective	This field automatically displays the type of COBOL module being generated.
Program ID	Required. Enter the name for the COBOL program or subprogram. This name becomes the PROGRAM-NAME of the PROGRAM-ID paragraph. A default program ID is assigned by Encore, which should be changed if appropriate. The default program ID is determined by these conditions: • If the original source program ID is 8 characters, the default program ID is the same, with the last character changed to a V. • If the original source program ID is 7 characters, the default program ID is the same, with a V appended to the name. • If the original source program ID is less than 7 characters, the default program ID is the same, with 0V appended to the name.
CALLs to IO Module(s)	Optional. If specified, the Select FD Names pop-up (see Figure 99 on page 210) displays to select the FD name(s) for which CALLs to an I/O module should be generated. If left blank, the Select FD Names pop-up does not display. This is the default.

Usage Notes

After specifying all necessary information, the module is generated and the Edit screen displays. The Edit screen shows the generated module and allows modification of the code. Encore generates several comments of informative text to assist you in comprehending the contents of the Logic Segment and in making any necessary adjustments. These comments include:

- A header that contains a description of the Logic Segment.
- A sample CALL statement identifying inputs and outputs.
- Comments throughout the generated COBOL module identifying possible discrepancies that should be reviewed and verified manually.

See "COBOL Module Contents" on page 216 for more information.

Select FD Names Pop-up

The Select FD Names pop-up, shown in Figure 99, is used to select the FD name(s) for which CALLs to a File I/O module are created. This pop-up displays by selecting the Generate ▸ Calls to IO Modules field from the Generate - COBOL Module pop-up (see Figure 98 on page 208).

The IO statements are replaced by CALLs to a generated File I/O module. When Enter is pressed, after selecting the desired FDs and specifying the other information about this pop-up, the Generate - COBOL Module pop-up redisplays to continue with the generation of the COBOL module.

Figure 99 • Select FD Names Pop-up

```
                          Generate - FD Name List                4 FD NAMES
Command ===> ____________________________________________ Scroll ===> CSR

Select the name of the FD for the IO module generation.  Then press Enter.
'*' indicates names have been previously generated.

    FD Name                          Access  Organization  READs  WRITEs
-   -------------------------------- ------  ------------  -----  ------
_ * ACCT-FILE                        I/O     INDEXED           1      1
_ * ACTION-FILE                      OUTPUT  SEQUENTIAL        0      1
_ * EXCEPTION-FILE                   OUTPUT  SEQUENTIAL        0      4
_ * TRAN-FILE                        INPUT   SEQUENTIAL        1      0
****************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
IO Parm Block Prefix	Required. Enter a prefix for the IO parm block to provide an identification for items pertaining to the file IO. The maximum is 8 alphanumeric characters. The default is ASG.
Line command area	The line command area, to the left of the FD Name field, accepts the S line command to select the desired FD name(s).
FD Name	The names of all File Descriptions (FDs) in the active program.
ENTRY Name	Required for selected FD Names. Enter an ENTRY name for each name selected. The default is the first eight characters of the FD name, with any hyphen replaced with a zero.

Generate - Specify Perform Range ENTRY Names Pop-up

The Generate - Specify Perform Range ENTRY Names pop-up, shown in Figure 100, is used to specify the ENTRY names for each Perform Range in the extracted Logic Segment. This pop-up displays after pressing Enter on the Generate - COBOL Module pop-up (see Figure 98 on page 208) for a Perform Range extract type.

Figure 100 • Generate - Specify Perform Range ENTRY Names Pop-up

```
                    Generate - Specify Perform Range ENTRY Names
Command ===> ____________________________________________ Scroll ===> CSR

Specify the ProgramID and ENTRY names of the modules to be generated.
S - Split   M - Move   I - Into   + - Expand   - - Collapse

  Program  ENTRY     PERFORM Range Name
- -------- --------  ------------------------------------------
******************************** TOP OF DATA ******************************
_ VIARDEMV           *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
_                      PERFORM INITIATE-ACTION
_                      PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT
_                       PERFORM INIT-ACTION-RECORD
_                      PERFORM EXCEPTION-REPORT-1
_          CLOSE0PG  *CLOSE-PGM
_                    * PERFORM CLOSE-FILES
_          CLOSE0FI  *CLOSE-FILES
******************************** BOTTOM OF DATA ***************************
```

Options

Field	Description
S - Split	Changes a subroutine performed by an extracted Perform Range into a callable module.
M - Move	Reverses the effects of selecting multiple entry points during the extract. This option must be used in conjunction with the I - Into option. This option causes the selected program to be moved into the target program as a Perform Range with an entry point.
I - Into	Specifies the target of a Move option. This option must be used in conjunction with the M - Move option. **Note:** The Move and Into actions can only be used to reverse a split that creates separate modules from entries for multiple Perform Range extracts. You cannot reverse a nested Perform Range split.
+ - Expand	Displays any subordinate PERFORM statements of the selected Perform Range. Subordinate PERFORMs display by default when this pop-up initially displays. If a Perform Range is collapsed, a plus sign (+) displays next to the PERFORM range name.
- - Collapse	Prevents subordinate PERFORM statements of a selected Perform Range from displaying.

Fields

Field	Description
Line command area	The line command area, to the left of the Program field, accepts options S, M, I, +, or -.
Program	Required for a Logic Segment with a single Perform Range extracted. Type the module name.
ENTRY Name	Required for each Perform Range in a Logic Segment with multiple Perform Ranges extracted. Enter an ENTRY name for each name required, as indicated by an underscore in the ENTRY Name field.
PERFORM Range Name	The name of the Perform Range you have entered or modified. If a Perform Range is split to create a callable module and it executes a Perform Range that was not split, the Perform Range code is duplicated in the module. The legend Dup (`nn`), where `n` is the number of times the Perform Range code is duplicated, displays to the right of the duplicated Perform Range Name field.

Generate - Multi-module PERFORM Ranges Pop-up

The Generate - Multi-module PERFORM Ranges pop-up, shown in Figure 101, is used to select the program to be generated. This pop-up displays after pressing Enter on the Generate - Specify Perform Range ENTRY Names pop-up (see Figure 100 on page 211).

Select a program to be generated by typing `S` next to the program ID. Only one program can be selected at a time. Once the selected program has been generated, the Generate - Multi-module PERFORM Ranges pop-up redisplays so that another program can be selected for generation.

Figure 101 • Generate - Multi-module PERFORM Ranges Pop-up

```
                    Generate - Multi-module PERFORM Ranges
Command ===> _____________________________________________ Scroll ===> CSR

Select the module to generate, then press Enter.

Program ID  ENTRY Name   PERFORM Range Name
-  -------- --------  ---------------------------------------------------
********************************* TOP OF DATA ******************************
_ VIARDEMV            ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
           CLOSE0PG   CLOSE-PGM
           CLOSE0FI   CLOSE-FILES
******************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program ID field, accepts the S line command to select the desired Program ID.
Program ID	Name of the program to be generated.
ENTRY Name	Name of the ENTRY point name assigned to the Perform Range.
PERFORM Range Name	Name of the selected PERFORM range.

Generate - Specify Perform Range CALL Names Pop-up

The Generate - Specify Perform Range CALL Names pop-up, shown in Figure 102, is used to specify the program ID or ENTRY names for each Perform Range in the extracted Logic Segment. This pop-up displays after pressing Enter on the Generate - COBOL Module pop-up (see Figure 98 on page 208) for a Perform Range Complement extract type.

For a Logic Segment with a single Perform Range extracted, enter the Program ID. For a Logic Segment with multiple Perform Ranges extracted, enter an ENTRY Name for each Perform Range required, as indicated by an underscore in the ENTRY Name field.

Figure 102 • Generate - Specify Perform Range CALL Names Pop-up

```
                    Generate - Specify Perform Range CALL Names
Command ===> _____________________________________________ Scroll ===> CSR

The complement module includes CALLs to extracted PERFORM Range(s).  Type
the PROGRAM-ID or ENTRY Names of the programs to be CALLed.  When complete
press Enter.

Program ID  ENTRY Name   PERFORM Range Name
--------    --------     -------------------------------------------
********************************** TOP OF DATA ******************************
            VIARDEMV     ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
            CLOSE0PG     CLOSE-PGM
            CLOSE0FI     CLOSE-FILES
********************************* BOTTOM OF DATA ****************************
```

Fields

Field	Description
Program ID	Required for a Logic Segment with a single Perform Range extracted. Enter the module name.
ENTRY Name	Required for each Perform Range in a Logic Segment with multiple Perform Ranges extracted. Enter an ENTRY name for each name required, as indicated by an underscore in the ENTRY Name field.
PERFORM Range Name	The name of the Perform Range you have entered or modified.

COBOL Module Contents

When a request is made to generate a COBOL module, the COBOL code produced by Encore is dependent on the extract objective used to create the Logic Segment, and also on the type of module requested on the Generate - COBOL Module pop-up (see Figure 98 on page 208) or the Generate - IO Module pop-up (see Figure 107 on page 231).

Module Header Box

When a module is generated, a header box is placed at the beginning of the source code, shown in Figure 103. The template used for the header box contents is contained in the CNTL member VIARCBSK. This template may be modified to conform to site standards.

Figure 103 • Encore-Generated Header Box

```
ASG-Encore EDIT - VIASHG.ENC00057.COBOL                    COLUMNS 00001 00072
Command ===> ____________________________________________ Scroll ===> CSR
****** ****************************** Top of Data ******************************
000001        ***********************************************************************
000002        *                          ASG, INC.                                  *
000003        *                                                                     *
000004        * GENERATED BY ASG-ENCORE        RELEASE 6.0(000)                    *
000005        *                                                                     *
000006        * PROGRAM:         VIARDEMV      COMPLEMENT                          *
000007        *                                                                     *
000008        * DATE:            MM-DDD-YYYY HH:MM:SS                              *
000009        * REQUESTOR:       USERID                                            *
000010        *                                                                     *
000011        * AKR INFORMATION                                                    *
000012        *     DS NAME:     USERID.TEST.AKR                                   *
000013        *     MEMBER:      VIARDEMO                ALIAS: (NONE)             *
000014        *                                                                     *
000015        * SEGMENT NAME:    COMPLEMENT-0002                                   *
000016        *                                                                     *
000017        * OBJECTIVE TYPE: COMPLEMENT (PERFORM RANGE)                         *
00001                                                                               *
00001   ASG1608I ENTER THE CREATE COMMAND TO SAVE THE COBOL PROGRAM.                *
00002                                                                               *
000021                                                                              *
```

Default Fields

Field	Description
PROGRAM	The program ID of the generated program, entered in the PROGRAM ID field of the Generate - COBOL Module pop-up (see Figure 98 on page 208).
DATE	Date and time the COBOL module was created.
REQUESTOR	TSO USERID of the person who created the COBOL module.
AKR INFORMATION	The dataset name, member name, and alias name of the original source program from which the Logic Segment was generated.
SEGMENT NAME	The name of the Logic Segment entered on the File - Save Segment pop-up (see Figure 15 on page 37). If a name has not been entered on the File - Save Segment pop-up, UNASSIGNED displays in this field. See Chapter 3, "File Menu," on page 25 for more information.
OBJECTIVE TYPE	The extract objective type that was used to generate the Logic Segment.

Callable Module USING Clause

When a callable COBOL subprogram is created, Encore generates the necessary USING clause in the subprogram, as shown in Figure 104.

Figure 104 • COBOL Module USING Clause

```
ASG-Encore EDIT - VIASHG.ENC00059.COBOL                    COLUMNS 00001 00072
Command ===> ____________________________________________  Scroll ===> CSR
000351        PROCEDURE DIVISION USING
000352                           ACCT-RECORD
000353                           ACTION-RECORD
000354                           TRAN-FILE-FLAG
000355                           INTEREST-RATE
000356                           DAYS-IN-PERIOD
000357                           DAYS-IN-YEAR
000358                           WORK-ACTION-REASON
000359                           ACCT-IO-CNT
000360                           TRAN-CNT
000361                           EX-RPT-LINE-CNT
000362                           EX-RPT-PAGE-CNT
000363                           ACTION-CNT
000364                           WORK-RETURN-CODE
000365                           TODAY-DATE
000366                           EXCEPT-HEAD-LINE1
000367                           CONTROL-PARAMETERS
000368                           BRANCH-BALANCE
000369                           BANK-BALANCE.
000370
000371
000372            PERFORM ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT.
```

Callable Module CALL Statement

Encore also generates comments that specify the corresponding CALL statement to be inserted into each calling program. You should review and manually insert these comments in each calling program.

Figure 105 shows the CALL statement that Encore has generated as a comment that you should review and insert into each calling program.

Figure 105 • Sample CALL Statement

```
ASG-Encore EDIT - VIASHG.ENC00059.COBOL                    COLUMNS 00001 00072
Command ===> ____________________________________________ Scroll ===> CSR
000033      * ASG-ENCORE NOTE:
000034      *     THE FOLLOWING ARE SAMPLE CALLS SHOWING INPUTS AND OUTPUTS:
000035      *
000036      *     CALL 'VIARDEMV' USING
000037      *                ACCT-RECORD  <<OUTPUT>>
000038      *              ACTION-RECORD  <<OUTPUT>>
000039      *             TRAN-FILE-FLAG  <<OUTPUT>>
000040      *              INTEREST-RATE  <<INPUT/OUTPUT>>
000041      *             DAYS-IN-PERIOD  <<INPUT>>
000042      *               DAYS-IN-YEAR  <<INPUT>>
000043      *         WORK-ACTION-REASON  <<INPUT/OUTPUT>>
000044      *                ACCT-IO-CNT  <<INPUT/OUTPUT>>
000045      *                   TRAN-CNT  <<INPUT/OUTPUT>>
000046      *            EX-RPT-LINE-CNT  <<INPUT/OUTPUT>>
000047      *            EX-RPT-PAGE-CNT  <<INPUT/OUTPUT>>
000048      *                 ACTION-CNT  <<INPUT/OUTPUT>>
000049      *           WORK-RETURN-CODE  <<OUTPUT>>
000050      *                 TODAY-DATE  <<INPUT>>
000051      *          EXCEPT-HEAD-LINE1  <<INPUT/OUTPUT>>
000052      *         CONTROL-PARAMETERS  <<INPUT>>
000053      *             BRANCH-BALANCE  <<INPUT/OUTPUT>>
000054      *               BANK-BALANCE  <<INPUT/OUTPUT>> .
```

DATA DIVISION COPY Member Processing

When the entire contents of a COPY member remain in the original section (WORKING-STORAGE or LINKAGE), the generated module contains the original COPY statement.

When the entire contents of a COPY member is relocated from WORKING-STORAGE to LINKAGE, the generated module contains the original COPY statement unless the COPY member contains VALUE clauses. In this case, the contents of the COPY member are included, (i.e., copied into the generated program in order to comment out the VALUE clauses).

When a portion of a COPY member must be relocated, the contents of the COPY member are included, (i.e., copied into the generated program). This can occur when a COPY member contains two or more 01 or 77 level data variables.

Subprogram and Program Contents

This section describes the contents of the generated modules for each extract objective and module type.

Note:

Modules generated by Encore from Logic Segments, except for Statement Selection and CICS Server extracts, provides execution equivalent code from the original program. However, certain COBOL constructs can occur whereby the module may not compile or execute without manual intervention. See the *ASG-Encore User's Guide* for additional information about COBOL constructs and anomalies for each extract objective.

Perform Range Extract

These are the contents of a COBOL module that has been generated for a subprogram using a Perform Range extract. If multiple Perform Ranges have been selected, an ENTRY is generated for each Perform Range.

DATA DIVISION	
Section	**Contents**
LINKAGE	All 01 and 77 level data variables recognized as inputs and outputs to the Perform Range.
WORKING-STORAGE	All data items used and modified in the Perform Range not already included in the LINKAGE SECTION.
FILE	Included if any files are used by the Perform Range unit, and contains an FD for each file-named reference.
REPORT	Included if any reports are used by the Perform Range unit, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the Perform Range unit, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements within the identified Perform Range, plus all code which can be reached by a PERFORM or GO TO statement within the range.	
A USING clause is generated if the performed range has external input or output variable references.	
The first executable statements of the PROCEDURE DIVISION are a generated PERFORM of the range selected, followed by a GOBACK statement.	

Report Extract

Subprogram Report extract

These are the contents of a COBOL module that has been generated for a subprogram using a Report extract. A subprogram is generated for a Report extract when the Limit to within an existing Segment option is selected on the Extract - Report Filter pop-up (see Figure 63 on page 141). See Chapter 5, "Extract Menu," on page 113 for more information.

DATA DIVISION	
Section	**Contents**
LINKAGE	All 01 and 77 level data variables recognized as input and output to the report subprogram module.
WORKING-STORAGE	All data variables required to generate the report not already defined in the LINKAGE SECTION.
FILE	Included if any files are used by the report, and contains a FD for each file-name referenced.
REPORT	Included if any reports are used by the report, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the report, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements from the beginning of the program identified as contributing to, or influencing the value of, the selected report output statements.	
All flow-of-control statements (GO TOs, PERFORMs) required to reach all statements of the report.	
A Encore-generated GOBACK following every report termination statement.	
A USING clause is generated if the report subprogram has input or output variables.	
All statements and referenced variables identified as influencing the flow of control to all report statements.	

Program Report extract

These are the contents of a COBOL module that has been generated for a program using a Report extract. A program is generated for a Report extract unless the Limit to within an existing Segment Option is selected on the Extract - Report Pop-up (in which case a subprogram is generated). See Chapter 5, "Extract Menu," on page 113 for more information.

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed.
WORKING-STORAGE	All data items required to generate the report output.
FILE	Included if any files are used by the report, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the report, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the report, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements identified as contributing to, or influencing, the selected report generation statements.	
All flow-of-control statements (GO TOs, PERFORMs) required to reach all statements of the report.	
A Encore-generated STOP RUN statement following every report termination statement.	
All statements and referenced variables identified as influencing the flow of control to the report.	

Computation Variable Extract

Subprogram Computation Variable extract

These are the contents of a COBOL module that has been generated for a subprogram using a Computation Variable extract. A subprogram is generated for a Computation Variable extract when the Limit to within an existing Segment option is selected on the Extract - Computation Variable Filter pop-up (see Figure 69 on page 152). See Chapter 5, "Extract Menu," on page 113 for more information.

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed, plus the Computation Variable itself if not already included in the LINKAGE SECTION.
WORKING-STORAGE	All data items required to compute the Computation Variable not already defined in the LINKAGE SECTION.
FILE	Included if any files are used by the Computation Variable, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the Computation Variable, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the Computation Variable, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements from the beginning of the program identified as contributing to, or influencing the value of, the selected Computation Variable at the selected statement(s).	
All flow-of-control statements (GO TOs, PERFORMs) required to reach all statements of the Logic Segment.	
All statements and referenced variables identified as influencing the flow of control to the Computation Variable derivation.	
A GOBACK statement following the Logic Segment statements.	
A USING clause is generated if the Computation Variable has input or output variables.	

Program Computation Variable extract

These are the contents of a COBOL module that has been generated for a program using a Computation Variable extract. A program is generated for a Computation Variable extract unless the Limit to within an existing Segment option is selected on the Extract - Computation Variable Filter pop-up (in which case a subprogram is generated, see Figure 69 on page 152). See Chapter 5, "Extract Menu," on page 113 for more information.

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed.
WORKING-STORAGE	All data items required to compute the Computation Variable, including the definition of the Computation Variable itself, if it was there (otherwise it was already in the LINKAGE SECTION).
FILE	Included if any files are used by the Computation Variable, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the Computation Variable, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the Computation Variable, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION	Contents
All statements of the Logic Segment, which is defined as all statements from the beginning of the program identified as contributing to, or influencing the value of, the selected Computation Variable at the selected statement(s).	
All flow-of-control statements (GO TOs, PERFORMs) required to reach all statements of the Computation Variable derivation.	
All statements and referenced variables identified as influencing the flow of control to the statements of the Computation Variable.	
A STOP RUN statement following all Logic Statement termination statements.	

Transaction Extract

Subprogram Transaction extract

These are the contents of a COBOL module that has been generated for a subprogram using a Transaction extract. A subprogram is generated for a Transaction extract when the default START or END statements are changed.

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed, plus variables identified as inputs to the transaction.
WORKING-STORAGE	All data items required to compute the transaction code, unless already defined in the LINKAGE SECTION.
FILE	Included if any files are used by the transaction, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the transaction, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the transaction, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements between the identified start and end point(s), as identified in the transaction objective.	
STOP RUN statements are inserted in the blocked conditional paths.	
Encore-generated GOBACKs are inserted following all END points.	
A Encore-generated label and corresponding GO TO are inserted when the transaction start is not the PROCEDURE DIVISION.	

Program Transaction extract

These are the contents of a COBOL module that has been generated for a program using a Transaction extract. A program is generated for a Transaction extract unless the default START or END statements are changed (in which case a subprogram is generated).

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed.
WORKING-STORAGE	All data items required to compute the transaction code.
FILE	Included if any files are used by the transaction, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the transaction, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used by the transaction, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements of the Logic Segment, which is defined as all statements between the identified START and END point(s), as identified in the Transaction objective.	
STOP RUN statements are inserted in the blocked conditional paths.	
A Encore-generated label and corresponding GO TO are inserted when the transaction start is not the PROCEDURE DIVISION.	

Statement Extract

Subprogram Statement extract

These are the contents of a COBOL module that has been generated for a subprogram using a Statement extract.

DATA DIVISION	
Section	**Contents**
LINKAGE	All 01 and 77 level data variables recognized as input and output to the subprogram module.
WORKING-STORAGE	All required data variables not already defined in the LINKAGE SECTION.
FILE	Included if any files are used by the statements, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the statements, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements explicitly selected.	

Program Statement extract

These are the contents of a COBOL module that has been generated for a program using a Statement extract:

DATA DIVISION	
Section	**Contents**
LINKAGE	All data variables that existed in the LINKAGE SECTION of the program, if one existed.
WORKING-STORAGE	All data items required.
FILE	Included if any files are used by the statements, and contains an FD for each file-name referenced.
REPORT	Included if any reports are used by the statements, and contains an RD for each report-name referenced.
COMMUNICATION	Included if any data communication names are used, and contains a CD entry for each communication-name referenced.
PROCEDURE DIVISION Contents	
All statements explicitly selected.	

File IO Modules

The File IO Module facility is used to generate an IO module for an FD (File Description). The IO module contains the necessary program functions for the file, and other modules may call the IO module instead of performing the input/output functions internally. To call the new IO module, select the CALLS to IO Module(s) field on the Generate - COBOL Module pop-up (see Figure 98 on page 208) when generating a Logic Segment. The Select FD Names pop-up (see Figure 99 on page 210) displays to specify which FDs should have IO functions replaced with CALLs.

When the File IO Module facility is selected on the Generate pull-down (see Figure 96 on page 204), the Generate - FD Name List pop-up (see Figure 106 on page 230) displays so you can select the desired FD name. Only one FD may be selected at a time. After a name is selected and Enter is pressed, the Generate - IO Module pop-up (see Figure 107 on page 231) displays to specify the disposition of the module. After entering any options on the Generate - IO Module pop-up, the module is generated and an Edit session is established.

Generate - FD Name List Pop-up

The Generate - FD Name List pop-up, shown in Figure 106, is used to select the name of the FD for the IO module generation. This pop-up displays by selecting Generate ▸ File IO Module from the Generate pull-down (see Figure 96 on page 204).

All FDs in the active program are listed on this pop-up. Any FDs that have already had IO modules generated are marked with an asterisk (*). Only one FD may be selected at a time. After a name is selected and Enter is pressed, the Generate - IO Module pop-up (see Figure 107 on page 231) displays to specify the disposition of the module.

Figure 106 • Generate - FD Name List Pop-up

```
                          Generate - FD Name List                4 FD NAMES
Command ===> ____________________________________________ Scroll ===> CSR

Select the name of the FD for the IO module generation.  Then press Enter.
'*' indicates names have been previously generated.

    FD Name                          Access  Organization  READs  WRITEs
-   -------------------------------- ------  ------------  -----  ------
_ * ACCT-FILE                        I/O     INDEXED           1       1
_ * ACTION-FILE                      OUTPUT  SEQUENTIAL        0       1
_ * EXCEPTION-FILE                   OUTPUT  SEQUENTIAL        0       4
_ * TRAN-FILE                        INPUT   SEQUENTIAL        1       0
******************************** BOTTOM OF DATA ****************************
```

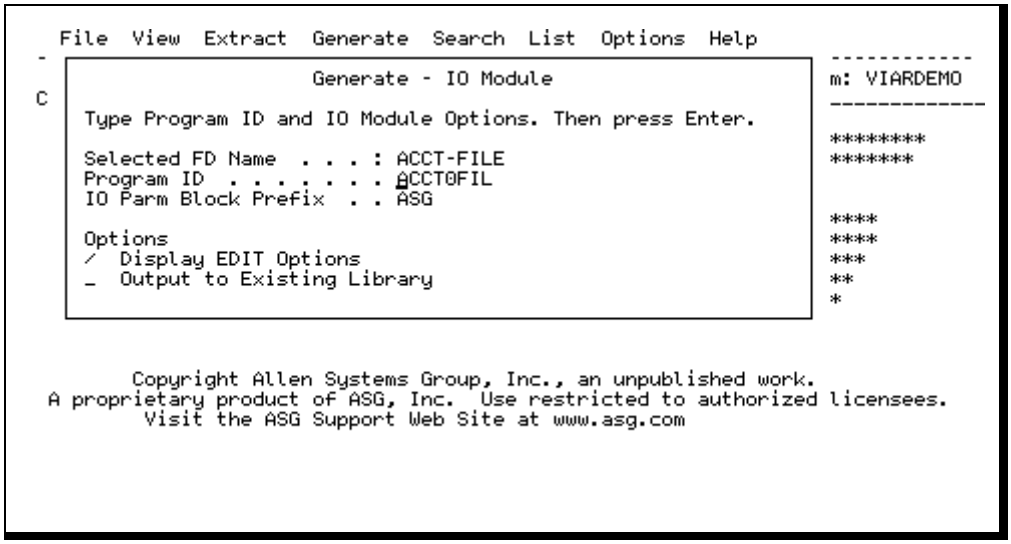
Fields

Field	Description
Line command area	The line command area, to the left of the FD Name field, accepts the S line command to select the desired FD name.
FD Name	The names of all File Descriptions (FDs) in the active program.
Access	The type of Input/Output processing performed.
Organization	The file organization type.
READs	The number of READ statements for the file in the program.
WRITEs	The number of WRITE statements for the file in the program.

Generate - IO Module Pop-up

The Generate - IO Module pop-up, shown in Figure 107, is used to create a new COBOL source module containing all the Input/Output code for a file. This pop-up displays after selecting a FD name and pressing Enter on the Generate - FD Name List pop-up (see Figure 106 on page 230).

Figure 107 • Generate - IO Module Pop-up



Fields

Field	Description
Selected FD Name	The name that was selected on the Generate - FD Name List pop-up (see Figure 106 on page 230).
Program ID	Required. Enter the name for the COBOL module. This name becomes the PROGRAM-NAME of the PROGRAM-ID paragraph. A default program ID is assigned by Encore, which should be changed if appropriate. The default program ID consists of the first eight characters of the FD name, with any hyphen replaced by a zero.
IO Parm Block Prefix	Required. Enter a prefix for the IO parm block to provide an identification for items pertaining to the file I/O. The maximum is 8 alphanumeric characters. The default is ASG.
Display EDIT Options	Optional. If specified, the Edit Options pop-up displays so you can specify the source manager. If left blank, the Edit Options pop-up is bypassed, and the Editor screen displays.
Output to Existing Library	Optional. If specified, the Edit Entry Panel displays to initiate an Edit session. If Display EDIT Options is also specified, the Edit Options pop-up displays first to specify the source manager. On the Edit Entry Panel, a member name can be entered for a new source member library. If left blank, an Edit session is established against a temporary file containing the generated module. Use the ISPF CREATE or REPLACE command to save the file. This is the default.

Output Files

The Output File facility is used to print or punch a Logic Segment. The Logic Segment is printed or punched as the set of COBOL statements identified in the extract objective phase. The resulting output can then be used to interface with an existing CASE tool.

Note:

Printing or punching a Logic Segment does not ensure compilable modules.

The Logic Segment statements are copied to the List or Punch file, based on the option selected. The List and Punch files can be printed on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300). See Chapter 9, "Options Menu," on page 293 for more information.

When the Output file option is selected on the Generate pull-down (see Figure 96 on page 204), the Output File - Select Logic Segment pop-up (see Figure 108 on page 234) displays so you can select the desired Logic Segment. After the Logic Segment is selected and Enter is pressed, the Generate - Output File pop-up (see Figure 109 on page 235) displays so you can specify the disposition of the Logic Segment.

Output File - Select Logic Segment Pop-up

The Output File - Select Logic Segment pop-up, shown in Figure 108, is used to select the Logic Segment to print or punch. This pop-up displays by selecting Generate ▸ Output file from the Generate pull-down (see Figure 96 on page 204).

Figure 108 • Output File - Select Logic Segment Pop-up

```
                    Output File - Select Logic Segment   3 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
_  VIARDEMO   COMPLEMENT-0002                Complement Perform Range
_  VIARDEMO   PERFORM-RANGE-0001             Perform Range
S  VIARDEMO   PERFORM-RANGE-0003             Perform Range
******************************** BOTTOM OF DATA ******************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select the Logic Segment. The Generate - Output File pop-up displays (see Figure 109 on page 235).
Program	The name of the program containing the Logic segment definition.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up. See Chapter 5, "Extract Menu," on page 113 for more information.
Description	The description of the extract that was used to create the Logic Segment.

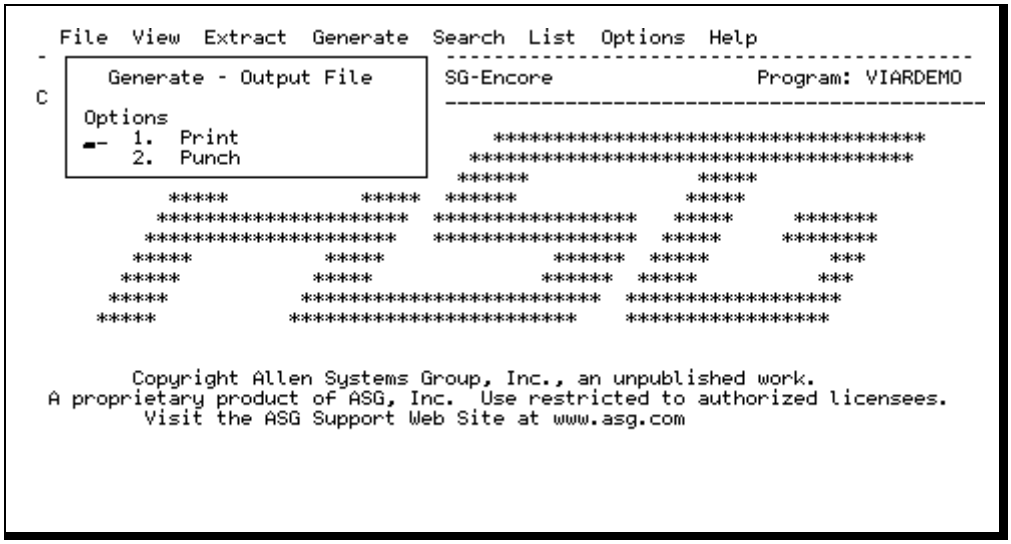
Generate - Output File Pop-up

The Generate - Output File pop-up, shown in Figure 109, is used to output source COBOL statements. This pop-up displays after selecting a Logic Segment on the Output File - Select Logic Segment pop-up (see Figure 108 on page 234) and pressing Enter.

The Logic Segment is printed/punched as the set of COBOL statements identified in the extract objective phase. The resulting output can then be used to interface with an existing CASE tool. Note that printing or punching a Logic Segment does not ensure compilable modules.

The Logic Segment statements are copied to the List or Punch file, based on the option selected. The List and Punch files can be printed on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300). See Chapter 9, "Options Menu," on page 293 for more information.

Figure 109 • Generate - Output File Pop-up



Field

Field	Description
Options	Required. Specify the disposition of the Logic Segment. 1. Print - Transfers a copy of the Logic Segment statements to the List file. 2. Punch - Transfers a copy of the Logic Segment statements to the Punch file.

Batch Computation

The Batch Computation facility is used to select and save Logic Segments that are generated through a batch job. A COBOL module is generated from the selected Logic Segment.

When the Batch Computation facility is selected on the Generate pull-down (see Figure 96 on page 204), the Batch Computation - Select Logic Segments pop-up (see Figure 110) displays so you can select the desired Logic Segments. Only one Logic Segment may be selected at a time. After a Logic Segment is selected and Enter is pressed, the Batch Computation - Save Segment pop-up (see Figure 111 on page 237) displays to specify segment definition information. After the segment information is typed and Enter is pressed, the Logic Segment - Batch Submit pop-up (see Figure 112 on page 239) displays. Information for the dataset is specified and the batch job may be submitted for generation.

Batch Computation - Select Logic Segments Pop-up

The Batch Computation - Select Logic Segments pop-up, shown in Figure 110, is used to select the Logic Segment that is generated during a batch job. This pop-up displays by selecting Generate ▸ Batch Computation from the Generate pull-down (see Figure 96 on page 204).

After a Logic Segment is selected and Enter is pressed, the Batch Computation - Save Segment pop-up (see Figure 111 on page 237) displays to save segment information.

Figure 110 • Batch Computation - Select Logic Segments Pop-up

```
                    Batch Computation - Select Logic Segme   3 LOGIC SEGMENTS
Command ===> ____________________________________________ Scroll ===> CSR

Select segment for Batch Computation and press Enter.  When selections are
complete, press Enter.

   S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
_  VIARDEMO   COMPLEMENT-0002                Complement Perform Range
_  VIARDEMO   PERFORM-RANGE-0001             Perform Range
_  VIARDEMO   PERFORM-RANGE-0003             Perform Range
******************************** BOTTOM OF DATA ****************************
```

Fields

Field	Description
Line command area	The line command area, to the left of the Program field, accepts the S line command to select the Logic Segment.
Program	The name of the program containing the Logic Segment displays in this field.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).
Description	The description of the extract that was used to create the Logic Segment.

Batch Computation - Save Segment Pop-up

The Batch Computation - Save Segment pop-up, shown in Figure 111, is used to save a Logic Segment that is generated using a batch job. This pop-up displays after selecting a Logic Segment and pressing Enter on the Batch Computation - Select Logic Segments pop-up (see Figure 110 on page 236). After entering segment information and pressing Enter, the Logic Segment - Batch Submit pop-up (see Figure 112 on page 239) displays to submit the batch job for generation.

Figure 111 • Batch Computation - Save Segment Pop-up

```
                        Batch Computation - Save Segment
Command ===> ____________________________________________ Scroll ===> CSR

Type segment definition information. Then press Enter.

Objective  . . . . : Perform Range
Author . . . . . . : USERID
Created  . . . . . : DD-MMM-YYYY HH:MM:SS

Name  . . . DAYS-IN-PERIOD
Description DAYS-IN-INTEREST-PERIOD_____________________________

Long Description:
THIS SEGMENT CONTAINS CODE TO DETERMINE THE NUMBER OF DAYS IN THE INTEREST
PERIOD. IT WILL BE USED TO GENERATE A CALLABLE SUB-MODULE._______________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
_______________________________________________________________________
```

Fields

Field	Description
Objective	Displays the extract type used to create the Logic Segment.
Author	Displays the TSO USERID used to create the Logic Segment.
Created	Displays the creation date and time for the Logic Segment. If the Logic Segment was copied from another definition, then the date is the copy date.
Name	If the Logic Segment has not been saved, enter a unique name up to 32 characters to identify the extract criteria. The name must be unique within the program and may not contain embedded blanks.
Description	The short description of the Logic Segment being copied. This description may be changed before pressing Enter to complete the copy.
Long Description	The long description of the Logic Segment being copied.

Logic Segment - Batch Submit Pop-up

The Logic Segment - Batch Submit pop-up, shown in Figure 112, is used to specify information for the dataset, specify where the COBOL module generated from the selected Logic Segment is stored, and to submit the batch job for generation. This pop-up displays by pressing Enter on the Batch Computation - Save Segment pop-up (see Figure 111 on page 237).

Figure 112 • Logic Segment - Batch Submit Pop-up

```
                        Logic Segment - Batch Submit         SAVE SUCCESSFUL

Logic Segment Name  . . : DAYS-IN-PERIOD
AKR Name  . . . . . . . : USER.TEST.AKR
Program Name  . . . . . : VIARDEMO

Generate COBOL Module:

Enter the following fields to generate a COBOL module.

Generate Program ID . . . ________
Generate Program DSN  . . ____________________________________________

Job statement information:
  //NAME     JOB (ACCOUNT),'NAME'
  //           MSGCLASS=A
  //*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
  //*

     PF3=End  PF5=Submit  PF6=Edit
```

Fields

Field	Description
Logic Segment Name	The name of the Logic Segment for generation.
AKR Name	The name of the current AKR.
Program	The name of the current program from which the segment is generated.
Generate Program ID	Optional. The program ID for the generated COBOL module.
Generated Program DSN	Optional. The name of the dataset where the generated module resides.
Job Statement Information	Job statement information for the batch generate job.

PF Keys

Field	Description
PF3=End	Cancels the batch submit job and redisplays the Batch Computation - Select Logic Segments pop-up (see Figure 110 on page 236).
PF5=Submit	Submits the batch generation job for the specified Logic Segment.
PF6=Edit	Displays the Edit screen, which allows you to edit the JCL for the batch generation job.

CICS and Self-Directed Server Generation

The Server Generation facility is used to select all of the necessary statements, files, and data elements from an extract used to produce the CICS or Self-Directed Server.

Note:

A module generated from a CICS or Self-Directed Server extract may contain compile errors or code that must be reviewed for correctness or optimal performance. When these conditions occur, Encore inserts comment messages in the generated modules. See the *ASG-Encore User's Guide* for more information.

Generate - Select Logic Segment Pop-up

The Generate - Select Logic Segment pop-up, shown in Figure 113, is used to select the appropriate Server Logic Segment that is used to create a stand-alone CICS server module. The module contains all of the statements required to produce the CICS Server. This pop-up displays by selecting Generate ▸ Server from the Generate pull-down (see Figure 96 on page 204).

After a Logic Segment name is selected and Enter is pressed, the Generate - HOST Copy Member pop-up (see Figure 114 on page 243) displays, which allows you to enter new COPY member information.Generate - Select Logic Segment Pop-up

Figure 113 • Generate - Select Logic Segment Pop-up

```
                        Generate - Select Logic Segment     1 LOGIC SEGMENTS
Command ===> _____________________________________________ Scroll ===> CSR

 S - Select

   Program    Logic Segment Name             Description
-  ---------- ------------------------------ ------------------------------
S  VIARBRWS   CLIENT-SERVER-0001             Client Server
******************************** BOTTOM OF DATA ******************************
```

Fields

Action	Description
Line command area	The line command area, to the left of the Program field, accepts the S line command to select the Logic Segment.
Program	The name of the program containing the Logic Segment displays in this field.
Logic Segment Name	The name of the Logic Segment entered on the Extract - Name Logic Segment pop-up (see Figure 57 on page 126).
Description	The description of the Logic Segment being copied that was used to locate the Logic Segment.

Generate - HOST Copy Member Pop-up

The Generate - HOST Copy Member pop-ups, shown in Figure 114 and Figure 115 on page 244, show each existing copybook used within the program that requires a new corresponding server copybook. These new copybooks are required for the screen map records and for the actual server communications area. The new copybooks are created containing the renumbered record definitions. These pop-ups display once for each screen map record and once for the communications area (the DFHCOMMAREA for CICS).

On the first Generate - HOST Copy Member pop-up, shown in Figure 114, enter the new name you want to assign to the DFHCOMMAREA copy member in the Copy Member field.

Assign the Starting Level Number and Level Number Increment you want, or accept the defaults. Press Enter to see the second Generate - HOST Copy Member pop-up (see Figure 115 on page 244).

Figure 114 • Generate - HOST Copy Member Pop-up - 1 of 2

```
                Generate - HOST Copy Member         1 OF 2

Enter new COPY member information.  Press Enter when
complete.

 Existing Copybook Information
 Copy Member . . . . . . . :
 Data Name . . . . . . . . : DFHCOMMAREA

 New Copybook Information
    Copy Member  . . . . . . _-------
    Starting Level Number    5
    Level Number Increment   5
```

On the second Generate - HOST Copy Member pop-up, shown in Figure 115, accept the new copy member defaults or change the name and level numbers and press Enter to display the Generate - HOST Server Module pop-up (see Figure 116 on page 245).

Figure 115 • Generate - HOST Copy Member Pop-up - 2 of 2

```
                Generate - HOST Copy Member         2 OF 2

Enter new COPY member information.  Press Enter when
complete.

 Existing Copybook Information
 Copy Member . . . . . . . : VIARBRWC
 Data Name . . . . . . . . : BROWSEO

 New Copybook Information
    Copy Member  . . . . . . VIARBRCS
    Starting Level Number    5
    Level Number Increment   5
```

Fields

Existing Copybook Information	
Field	**Description**
Copy Member	The name of the existing copy member being updated (if applicable).
Data name	The data name of existing data name being updated.

New Copybook Information	
Field	**Description**
Copy Member	The name for the updated copy member.
Starting Level Number	The starting level number for the new copy member. The default record level is 5.
Level Number Increment	The starting level for the level number increments. The default level is 5.

Generate - HOST Server Module Pop-up

The Generate - HOST Server Module pop-up, shown in Figure 116, is used to specify the Program ID for the HOST server module. This pop-up displays after pressing Enter on the second Generate - HOST Copy Member pop-up (see Figure 115 on page 244).

Figure 116 • Generate - HOST Server Module Pop-up

```
                Generate - HOST Server Module

Type Program ID and select generation options.

Objective  . . : Client Server
Program ID . . . SERVER

Functions
-- 1.  Generate options...
   2.  Generate COPY members...
   3.  Generate server module
   4.  Exit
```

Fields

Action	Description
Objective	The objective of this generation process.
Program ID	The name to be given to the generated code. The default is SERVER, which can be changed.
Functions	1. Generate options - Displays the COBOL Generation Options pop-up, which allows you to review or modify these generation options: • Display Edit Options - displays the Edit Options panel. • Output to Existing Library - allows Encore to insert the generated output directly into an existing library. • Full DATA DIVISION - specifies whether the full DATA DIVISION should be carried over into the generated module. • Retain COPY statements - specifies whether to retain or expand the copy statements. Expand is the default. This option is only used if the items in the COPY member are being modified by the product. If no modification takes place, COPY statements are always retained, regardless of the setting on this option. • In-line Communication Area - specifies that you want to include the expanded COPY Statements in-stream. • Retain Source Line Numbers - specifies that you want to preserve the original code columns in the generated module. • Retain PERFORMed Paragraphs - specifies that existing PERFORM Statements of extracted Perform Ranges are replaced with a PERFORM to a ASG-GENERATED CALL Section. If this option is not selected, existing PERFORM statements are replaced with CALL statements (unless the statement is a PERFORM VARYING). 2. Generate COPY members - Displays the Generate - Copy Members pop-up, which allows you to generate, edit, and save the HOST copy members. 3. Generate server module - Generates the actual HOST Server Module and invokes the editor to allow you to save the server module. 4. Exit - Exits this pop-up.

7 Search Menu

This chapter describes Encore's Search pull-down menu functions and contains these sections:

Section	Page
Introduction	250
Search Pull-down	251
Search - Data Name Pop-up	253
Search - Label Name Pop-up	256
Search - Paragraph Name Pop-up	258
Search - String Pop-up	260
Search - COBOL Subset Name Pop-up	262
Search - Perform Range Name Pop-up	265
Search - Program Name Pop-up	267
Search - Line Range Pop-up	269
Search - Any/Unknown Type Pop-up	270
Search - Branch Request Pop-up	272
Branch Previous Options Pop-up	275
Selection List Pop-ups	276
IN-Clause Option Pop-up	277

Introduction

The Search facility is used to find or exclude one of these specified targets:

- Data name
- Label name
- Paragraph name
- Pattern
- COBOL subset
- Perform range
- Program
- Line number

These are the different view methods available to assist you in isolating, understanding, and identifying important sections of the source program:

- Source View
- Tree View
- Structure View

Each view is created from the same original COBOL source program, but presents a different perspective, and is concurrently available and synchronized with the other views. You can switch back and forth between views and retain the results of certain actions.

Current search command options

Find - Searches the source code for one or all occurrences of the specified target. The results are presented in this manner:

- Source View - Positions the cursor at the first target in the specified direction, highlights all occurrences found, and places tags indicating the type of target found to the right of each statement.
- Tree View - Positions the cursor at the first perform range in a top/down manner which contains the target, highlights all occurrences found, and places tags indicating the type of target found to the right of each statement.
- Structure View - Positions the cursor at the first perform range in the specified direction and highlights all results found.

Exclude - Searches the source code for one or all occurrences of the specified target and omits the resulting lines from the display. This action is applicable to Source View only.

Search Pull-down

Selecting Search on the action bar displays the Search pull-down, shown in Figure 117. The Search pull-down is used to conduct a COBOL intelligent search of the source code for one or all occurrences of a specified target.

Figure 117 • Search Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
-----------------------------  ------------------------
                                     Program: VIARBRWS
Command ===> _________________  -------------------------
                                 1. Data...
                                 2. Label...
                                 3. Paragraph...
                                 4. String...
                                 5. Subset...
                                 6. Perform...
                                 7. Program...
                                 8. Line...
                                 9. Any...
                                10. Branch...

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

Note:

If other ESW products are installed at your site, other selections may appear on the Search pull-down, and other fields may appear on Search pop-ups. For information about these features of other products, see the corresponding product User's Guide or online help.

Actions on the Search pull-down cannot be selected unless there is an active program, (i.e., a program has been opened on the File pull-down).

Actions

Action	Description
1. Data	Displays the Data - Displays the Search - Data Name pop-up (see Figure 118 on page 253) used to search for data items.
2. Label	Displays the Search - Label Name pop-up (see Figure 119 on page 256) used to search for a specified label name.
3. Paragraph	Displays the Search - Paragraph Name pop-up (see Figure 120 on page 258) used to search for statements in a paragraph.
4. String	Displays the Search - String pop-up (see Figure 121 on page 260) used to search for a character string.
5. Subset	Displays the Search - COBOL Subset Name pop-up (see Figure 122 on page 262) used to search for COBOL subsets.
6. Perform	Displays the Search - Perform Range Name pop-up (see Figure 123 on page 265) used to search for statements within a perform range.
7. Program	Displays the Search - Program Name pop-up (see Figure 124 on page 267) used to search for statements in subprograms.
8. Line	Displays the Search - Line Range pop-up (see Figure 125 on page 269) used to search for statements within line ranges.
9. Any	Displays the Search - Any/Unknown Type pop-up (see Figure 126 on page 270) used to search for targets of unknown types.
10. Branch	Displays the Search - Branch Request pop-up (see Figure 127 on page 272) used to branch to a specific location.

Search - Data Name Pop-up

The Search - Data Name pop-up, shown in Figure 118, is used to find or exclude data names, depending on the entry in the Action field, and also depending on the current View screen. This pop-up displays by selecting Search ▸ Data from the Search pull-down (see Figure 117 on page 251).

Figure 118 • Search - Data Name Pop-up

```
                          Search - Data Name

Type a data name and select search options.  Then press Enter.  For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data Name  ------------------------------------------------------

Direction                 Options                   Action
1   1.  All               _  No Data Aliasing       1   1.  Find
    2.  Next              _  IN-Clause...               2.  Exclude
    3.  Previous
    4.  First
    5.  Last

References
1   1.  All
    2.  Defs
    3.  Uses
    4.  Mods
```

Fields

Field	Description
Data Name	Required. Enter a COBOL data name or qualified COBOL data name. Data name refers to any valid COBOL reference for a data element. To display the Selection List - Data Names pop-up, leave the Data name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character, an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information. Data names can be concatenated by placing a plus sign (+) between the data names.

Field	Description
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the data name, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the data name. 3. Previous - Searches backward from the cursor position to the previous occurrence of the data name. 4. First - Searches from the top of the source file to the first occurrence of the data name. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the data name.
Options	Optional. Specifies the options you can use for the Data Name search. • No Data Aliasing - Optional. If specified, aliases of the data name are ignored. Aliases include: — Parent - higher level group item — Child - lower level item — Rename/Redefinition - renamed, redefined, or 88 level items — If left blank, aliases of the data name are included. This is the default. • IN-Clause - Optional. If specified, displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.

Field	Description
Action	Optional. Specifies the action to be performed on the data name. The default is 1. Find. 1. Find - Searches for one or all occurrences of the data name. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default Action. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the data name from the display. This Action is applicable to Source View only.
References	Optional. The search can be restricted to only those data names that are defined, used, or modified, by specifying the type of data name reference. The default is 1. All. 1. All - Includes all occurrences of the data name where its value is defined, tested, used, set, or modified. This is the default. 2. Defs - Includes definitions of the data name in the DATA DIVISION. 3. Uses - Includes occurrences of the data name where its value is being tested or used. 4. Mods - Includes occurrences of the data name where its value is being set or modified.

Search - Label Name Pop-up

The Search - Label Name pop-up, shown in Figure 119, is used to find or exclude a specified label, paragraph, or section name, depending on the entry in the Action field, and also depending on the current View screen.

Figure 119 • Search - Label Name Pop-up

```
                          Search - Label Name

To find label name references, type a name and select search
options.  Then press Enter.  For selection list, type a pattern
(e.g. ABC*) in the name area.                                 -

Label name  ------------------------------------------------------

Direction                 Options                 Action
1   1.  All               _  IN-Clause...         1   1.  Find
    2.  Next                                          2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Label Name pop-up, follow this step:

- Select Search ▸ Label and press Enter.

Fields

Field	Description
Label name	Required. Enter any PROCEDURE DIVISION paragraph name or section name. PROCEDURE and PROC literals can also be specified. To display the Selection List pop-up, leave the Label name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information. Label names may be concatenated by placing a plus sign (+) between the label names.

Field	Description
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the label name, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the label name. 3. Previous - Searches backward from the cursor position to the previous occurrence of the label name. 4. First - Searches from the top of the source file to the first occurrence of the label name. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the label name.
Options	Optional. Specifies the options you can use for the Label Name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the label name. The default is 1. Find. 1. Find - Searches for one or all occurrences of the label name. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the label name from the display. This action is applicable to Source View only.

Search - Paragraph Name Pop-up

The Search - Paragraph Name pop-up, shown in Figure 120, is used to find or exclude all statements in a specified paragraph, depending on the entry in the Action field, and also depending on the current View screen.

Figure 120 • Search - Paragraph Name Pop-up

```
                    Search - Paragraph Name

To find all statements in a paragraph, type a name and select the
search options.  Then press Enter.  For a name list, type a pattern
(e.g. ABC*) in the name area.

Paragraph name  _------------------------------------------------

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Paragraph Name pop-up, follow this step:

- Select search ▸ Paragraph and press Enter (see Figure 117 on page 251).

Fields

Field	Description
Paragraph name	Required. Enter the name of a paragraph on which to perform the search. To display the Selection List pop-up, leave the Paragraph name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information. Paragraph names may be concatenated by placing a plus sign (+) between the paragraph names.

Field	Description
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the paragraph name, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the paragraph name. 3. Previous - Searches backward from the cursor position to the previous occurrence of the paragraph name. 4. First - Searches from the top of the source file to the first occurrence of the paragraph name. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the paragraph name.
Options	Optional. Specifies the options you can use for the Paragraph Name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the paragraph name. The default is 1. Find. 1. Find - Searches for one or all occurrences of the paragraph name. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the paragraph name from the display. This action is applicable to Source View only.

Search - String Pop-up

The Search - String pop-up, shown in Figure 121, is used to find or exclude a specified character string, depending on the entry in the Action field, and also depending on the current View screen.

Figure 121 • Search - String Pop-up

```
                              Search - String

Type a string and select search options.  Then press Enter.

String  ------------------------------------------------------------

String Type              Location
1   1.  Simple           1   1.  Any
    2.  Hexadecimal          2.  Word
    3.  Text                 3.  Prefix
    4.  Picture              4.  Suffix

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
Begin column ___         End column ___
```

To display the Search - String pop-up, follow this step:

- Select Search ▸ String and press Enter.

Fields

Field	Description
String	Required. Enter a string of alphanumeric or DBCS characters.
String Type	Optional. Specifies the type of pattern string. The default is 1. Simple. 1. Simple - Indicates a string that is not a hexadecimal, text, or a picture string. 2. Hexadecimal - Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value.

Field	Description
	3. Text - A character string may be entered regardless of upper or lowercase by using the text option. 4. Picture - A string profile may be entered instead of exact characters. These nine special characters are defined by Encore for use in picture strings and can be combined with other characters: = Any character ¬ Any nonblank character . Any nondisplay character # Any numeric character - Any non-numeric character @ Any alphabetic character (upper or lowercase) < Any lowercase alphabetic character > Any uppercase alphabetic character $ Any special character (not alphabetic or numeric)
Location	Optional. Specifies the location to search for the pattern string. The default is 1. Any. 1. Any - A string anywhere in the source code. 2. Word - A string preceded and followed by any non-alphanumeric character (except hyphen). 3. Prefix - A word that begins with the specified string. 4. Suffix - A word that ends with the specified string.
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the string, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the string. 3. Previous - Searches backward from the cursor position to the previous occurrence of the string. 4. First - Searches from the top of the source file to the first occurrence of the string. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the string.

Field	Description
Options	Optional. Specifies the options you can use for the String name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the string. The default is 1. Find. 1. Find - Searches for one or all occurrences of the string. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the string from the display. This action is applicable to Source View only.
Begin column	Optional. Enter the column number where the search is to begin. The default is column 7.
End column	Optional. Enter the column number where the search is to end. The default is column 66.

Search - COBOL Subset Name Pop-up

The Search - COBOL Subset Name pop-up, shown in Figure 122, is used to find or exclude COBOL subset references, depending on the entry in the Action field, and also depending on the current View screen.

Figure 122 • Search - COBOL Subset Name Pop-up

```
                    Search - COBOL Subset Name

Type a subset name and select search options.  Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

Subset name  _____________________________________________________

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - COBOL Subset Name pop-up, follow this step:

- Select Search ▸ Subset and press Enter.

Fields

Field	Description
Subset name	Required. Enter the name of a predefined COBOL subset. To display the Selection List pop-up, leave the Subset Name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information. These are the predefined COBOL subsets: ASsignment, DEbug, IO CAll, DEFinition, LABel CIcs, DIRective, MAINline COBOLII, DIVision, MATH COBOL/370, DL/I | DL/1, Output COMment, DML, PARagraph CONditional, ENtry, PERform COPy, EXIt | PGMExit, RETurn DB2/SQL, FALLthrough, SECtion DDL, GOto, SORTMerge DEAD, IDMS, STructure DEADCode, INClude, TAG DEADData, Input, TESted | UNTested **Note:** The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.

Field	Description

	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or  NX`
	
	A tagged lines subset of tags appearing in columns 73 - 80.
	
	Subsets may be concatenated by placing a plus sign (+) between the subset names. See Chapter 2, "Product Overview," on page 7 for a description of each subset.

Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the subset, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the subset. 3. Previous - Searches backward from the cursor position to the previous occurrence of the subset. 4. First - Searches from the top of the source file to the first occurrence of the subset. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the subset.
Options	Optional. Specifies the options you can use for the COBOL Subset Name search.IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search.

Field	Description
	If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the subset. The default is 1. Find. 1. Find - Searches for one or all occurrences of the subset. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the subset from the display. This action is applicable to Source View only.

Search - Perform Range Name Pop-up

The Search - Perform Range Name pop-up, shown in Figure 123, is used to find or exclude perform ranges and all code executed by the perform range, depending on the entry in the Action field, and also depending on the current View screen.

Figure 123 • Search - Perform Range Name Pop-up

```
                    Search - Perform Range Name

Type a perfrange name and select search options.  Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

Perfrange name  _______________________________________________

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Perform Range Name pop-up, follow this step:

- Select Search ▸ Perform and press Enter.

Fields

Field	Description
Perfrange name	Required. Enter the name specified in a PERFORM statement, or the name of any section contained in the DECLARATIVES. To display the Selection List pop-up, leave the Perfrange name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information.
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the perform range, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the perform range. 3. Previous - Searches backward from the cursor position to the previous occurrence of the perform range. 4. First - Searches from the top of the source file to the first occurrence of the perform range. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the perform range.
Options	Optional. Specifies the options you can use for the Perform Range Name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the perform range. The default is 1. Find. 1. Find - Searches for one or all occurrences of the perform range. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the perform range from the display. This action is applicable to Source View only.

Search - Program Name Pop-up

The Search - Program Name pop-up, shown in Figure 124, is used to find or exclude statements in programs or subprograms, depending on the entry in the Action field, and also depending on the current View screen.

Figure 124 • Search - Program Name Pop-up

```
                         Search - Program Name

Type a (sub)program name and select search options.  Then press
Enter.  For a selection list, type a pattern (e.g. ABC*) in the name
area.

Program name  _______________________________________________________

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Program Name pop-up, follow this step:

- Select Search ▸ Program and press Enter.

Fields

Field	Description
Program Name	Required. Enter the name of the main program or any nested (sub)program. To display the Selection List pop-up, leave the Program name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information. Program names may be concatenated by placing a plus sign (+) between the program names.

Field	Description
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the program name, and displays the number found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the program name. 3. Previous - Searches backward from the cursor position to the previous occurrence of the program name. 4. First - Searches from the top of the source file to the first occurrence of the program name. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the program name.
Options	Optional. Specifies the options you can use for the Program Name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the program name. The default is 1. Find. 1. Find - Searches for one or all occurrences of the program name. Double Byte Character Set (DBCS) identifiers or patterns can be specified to locate DBCS strings. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the program name from the display. This action is applicable to Source View only.

Search - Line Range Pop-up

The Search - Line Range pop-up, shown in Figure 125, is used to find or exclude a single line or range of lines, depending on the entry in the Action field, and also depending on the current View screen.

Figure 125 • Search - Line Range Pop-up

```
                         Search - Line Range

Type a line range (or line) and select search options.  Then press
Enter.

Line range  _____________________________________________________

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Line Range pop-up, follow this step:

- Select Search ▸ Line and press Enter.

Fields

Field	Description
Line range	Required. Enter a single line number or range of lines. Line numbers display in columns 1 - 6 in Source View.
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the line range, and displays the number of lines found in the short or long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the line range. 3. Previous - Searches backward from the cursor position to the previous occurrence of the line range. 4. First - Searches from the top of the source file to the first occurrence of the line range. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the line range.

Field	Description
Options	Optional. Specifies the options you can use for the Line Range name search. IN-Clause - Optional. Displays the IN-Clause Option pop-up used to restrict the source lines to be considered for a search. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the line range. The default is 1. Find. 1. Find - Searches for one or all occurrences of the line range. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the line range from the display. This action is applicable to Source View only.

Search - Any/Unknown Type Pop-up

The Search - Any/Unknown Type pop-up, shown in Figure 126, is used to find or exclude targets when the type is unknown, depending on the entry in the Action field, and also depending on the current View screen.

Figure 126 • Search - Any/Unknown Type Pop-up

```
                    Search - Any/Unknown Type

To find references to a target of "unknown" type (data, label,
etc.), type the target and select search options.  Then press Enter.

Target  _------------------------------------------------------

Direction                Options                Action
1   1.  All              _  IN-Clause...        1   1.  Find
    2.  Next                                        2.  Exclude
    3.  Previous
    4.  First
    5.  Last
```

To display the Search - Any/Unknown Type pop-up, follow this step:

- Select Search ▸ Any and press Enter.

Fields

Field	Description
Target	Required. Enter the name of the target. Target names may be concatenated by placing a plus sign (+) between the target names. To re-use the target of the previous search, enter an asterisk (*) in the Target field.
Direction	Optional. Indicates the direction for the search. The default is 1. All. 1. All - Searches for all occurrences of the target and displays the number found in the short/long message field. This is the default. 2. Next - Searches forward from the cursor position to the next occurrence of the target. 3. Previous - Searches backward from the cursor position to the previous occurrence of the target. 4. First - Searches from the top of the source file to the first occurrence of the target. 5. Last - Searches backward from the bottom of the source file to the last occurrence of the target.
Options	Optional. Specifies the options you can use for the Type name search. IN-Clause - Optional. If specified, the IN-Clause Option pop-up displays so you can restrict the search to the specified target type. If left blank, the entire source code program is searched. The default is the entire program. See Figure 130 on page 277 for more information.
Action	Optional. Specifies the action to be performed on the target. The default is 1. Find. 1. Find - Searches for one or all occurrences of the target. All lines containing the target are highlighted and tagged on the current view screen. Find is the default. Results depend on the current view screen. See Chapter 2, "Product Overview," on page 7 for more information. 2. Exclude - Omits lines containing occurrences of the target from the current view screen. This action is applicable to Source View only.

Search - Branch Request Pop-up

The Search - Branch Request pop-up, shown in Figure 127, is used to position the cursor to the specified target on the Source View screen.

Figure 127 • Search - Branch Request Pop-up

```
                                                          .

                         Search - Branch Request

To branch to another area of the program, select the Option desired.
For Option 1, type the branch location (Target) information.  Then
press Enter.  For a name selection list (for Target type 2, 3 or 4),
type a pattern (e.g. ABC*) in the name field.

Option
1   1.  Branch to target
    2.  Return to previous "Branch to target" location
    3.  Branch to transfer(s) of control to cursor position

Target name  ------------------------------------------------------

Target type
1   1.  None - use cursor
    2.  Label name
    3.  Perfrange name
    4.  Program name
```

To display the Search - Branch Request pop-up, follow this step:

- Select Search ▸ Branch and press Enter.

Fields

Field	Description
Option	Required. Enter the type of branch to perform. The default is 1. Branch to target. 1. Branch to target - Positions the cursor to the name specified in the Target name field. If this Option is selected, the Target name field is required. 2. Return to previous Branch to target location - Positions the cursor to the statement from which the previous branch occurred.

Field	Description
	3. Branch to transfer(s) of control to cursor position - Redisplays the Source View screen with a long message stating to position the cursor to the desired branch location and press Enter. The cursor is positioned to the statement that branches to the specified branch location. If more than one statement transfers control to the specified branch location, the Branch Previous Options pop-up (see Figure 128 on page 275) displays to select the desired statement. If this Option is selected, the Target name and Target type fields are invalid.
Target name	Required if Option 1 is selected. Enter a Perform Range name, label name, paragraph name, or section name. For COBOL II Release 3 (minimum supported release level) programs, a subprogram name can also be entered. • For Perform Ranges and label names, if the specified name is not fully qualified and there are multiple Perform Ranges or labels with the specified name, the cursor is positioned on the first statement found. • For COBOL II Release 3 (minimum supported release level) programs, a data item, label or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. • To display the Selection List pop-up, leave the Target name blank, or enter a pattern with wildcard characters ((a question mark (?) for one character; an asterisk (*) for zero or more characters), and select Target type 2, 3, or 4. On the Selection List pop-up, select the desired name with the S line command. See "Selection List Pop-ups" on page 276 for more information.

Field	Description
Target type	Optional. Enter the number of the target type corresponding to the entry in the Target name field. This field is valid for Option 1 only. The default is 1. None - use cursor. 1. None - use cursor - Redisplays the Source View screen with a long message stating to position the cursor to the selected statement and press Enter. If the statement is a GOTO, PERFORM, or internal CALL, the cursor is positioned to the target of that statement. If any other statement type is selected, the cursor is positioned to the next sequential statement to be executed. The PROCEDURE DIVISION label is located if the cursor is positioned outside the PROCEDURE DIVISION. 2. Label name - Positions the cursor to the specified paragraph or section name. If the specified name is not fully qualified and there are multiple paragraphs or sections with the specified name, the cursor is positioned to the first paragraph or section found. 3. Perfrange name - Positions the cursor to the specified Perform Range. If the specified name is not fully qualified and there are multiple Perform Ranges with the specified name, the cursor is positioned to the first Perform Range found. 4. Program name - Positions the cursor to the specified nested program. For COBOL II Release 3 and later programs, a data item, label, or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name.

Branch Previous Options Pop-up

The Branch Previous Options pop-up, shown in Figure 128, is used to select a statement for branching.

Figure 128 • Branch Previous Options Pop-up

```
                          Branch Previous Options        5 STATEMENTS FOUND
Command ===> _____________________________________________ Scroll ===> CSR

Select the statement to be viewed.  Then press Enter.

S  Line    Source statement.
- ------ ------------------------------------------------------------------
_ 000258              PERFORM INITIATE-ACTION
  000289          PERFORM INITIATE-ACTION.
  000295          PERFORM INITIATE-ACTION
_ 000301          PERFORM INITIATE-ACTION.
_ 000318       PERFORM INITIATE-ACTION.
****************************** BOTTOM OF DATA ****************************
```

This pop-up displays when more than one statement transfers control to the specified branch location on the Search - Branch Request pop-up (see Figure 127 on page 272), or when you use the BRANCH PREVIOUS command.

When this pop-up first displays, the short message field indicates the number of statements found.

Fields

Field	Description
Line command area	The line command area, to the left of the Line field, accepts the S line command.
Line	The source statement line number.
Source statement	The source code statement.

Selection List Pop-ups

Selection list pop-ups are available for the Data, Label, Paragraph, Subset, Perform Range and Program searches.

To display the selection list, follow this step:

- Leave the appropriate field blank

  Or

  Enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters).

Selection List pop-ups are also available in the Extract facility. See Chapter 5, "Extract Menu," on page 113 for more information.

Figure 129 shows an example of a selection list pop-up. Selection list pop-ups are the same except for the data displayed on the pop-up.

Figure 129 • Selection List - Data Names Pop-up

```
                        Selection List - Subset Names          LINE 1 OF 45
Command ===> ____________________________________________ Scroll ===> CSR

S    Subset          Description
-  ---------------------------------------------------------------------
_  ASsignment    - ADD, COMPUTE, MOVE, SQL FETCH, IDMS GET, IDMS LOAD...
_  CAll          - CALL, ENTRY, IDMS LINK, CICS LINK...
_  CIcs          - EXEC CICS and EXEC DLI commands.
_  COBOLII       - CALL/PERFORM (COBOLII), EVALUATE, end verbs...
_  COBOL/370     - Intrinsic function calls, procedure pointers...
_  COMment       - Comment, Identification Division, SQL COMMENT...
_  CONditional   - IF, ELSE, AT END, WHEN, SEARCH, IDMS ON, IDMS IF...
_  COPy          - COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, -INC
_  DB2           - EXEC SQL statements.
_  DDL           - SQL ALTER, SQL CREATE, SQL DECLARE, SQL DROP.
_  DEAD          - DEADCode and DEADData subsets.
_  DEADCode      - Procedure Division code which cannot be executed.
_  DEADData      - Unreferenced data names.
_  DEBug         - DEBUG, EXHIBIT, "D" in col 7, CICS DUMP...
_  DEFinition    - SPECIAL-NAMES, Data Division, SQL CREATE, SQL UPDATE.
_  DIRective     - COPY, EJECT, ENTER, SKIP, IDMS BIND, IDMS CHANGE...
_  DIVision      - Division names.
```

Use the S line command to select the desired name from the list. The Search pop-up redisplays with the selected name(s) in the appropriate field.

In the Search facility, more than one name may be selected on these pop-ups:

- Selection List - Data Names
- Selection List - Label/Paragraph Names
- Selection List - Subset Names
- Selection List - Program Names

IN-Clause Option Pop-up

The IN-Clause Option pop-up, shown in Figure 130, is used to restrict the source lines to be considered for a search.

Figure 130 • IN-Clause Option Pop-up

```
                          IN-Clause Option

To restrict the source lines to be considered, type one or more
IN-Clause names.  Then press Enter.  For a selection list, type a
pattern in the appropriate name field.

Line range  . . . ------------------------------------------------
Subset name . . . ------------------------------------------------
Paragraph name    ------------------------------------------------
Perfrange name    ------------------------------------------------
Program name  . . ------------------------------------------------
```

To display the IN-Clause Option pop-up, follow this step:

- Specify the IN-Clause option on a Search pop-up.

Fields

Field	Description
Line range	Optional. Enter a single line number or range of lines.
Subset name	Optional. Enter a predefined COBOL language subset name. To display the Selection List pop-up, leave the Subset Name field blank, or enter a pattern with wildcard characters (a question mark (?) for one character; an asterisk (*) for zero or more characters). See "Selection List Pop-ups" on page 276 for more information.

Field	Description

	These are the predefined COBOL subsets:
	ASsignment DEbug IO
	CAll DEFinition LABel
	CIcs DIRective MAINline
	COBOLII DIVision MATH
	COBOL/370 DL/I | DL/1 Output
	COMment DML PARagraph
	CONditional ENtry PERform
	COPy EXIt | PGMExit RETurn
	DB2/SQL FALLthrough SECtion
	DDL GOto SORTMerge
	DEAD IDMS STructure
	DEADCode INClude TAG
	DEADData Input TESted | UNTested
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or NX`
	A tagged lines subset of tags appearing in columns 73 - 80.
	Subsets may be concatenated by placing a plus sign (+) between the subset names. See Chapter 2, "Product Overview," on page 7 for a description of each subset.

Paragraph name	Optional. Enter a PROCEDURE DIVISION paragraph or section name. The PROCEDURE and PROC literals can also be specified. Paragraphs may be concatenated by placing a plus sign (+) between the paragraph names.

Field	Description
Perfrange name	Optional. Enter the name specified in a PERFORM statement, or the name of any section contained in the DECLARATIVES. The search includes all statements that are executed as a result of the PERFORM statement.
Program name	Optional. Enter the name of the main program or any nested program. The search includes all code contained in the program, including all programs physically nested inside the specified program. Programs may be concatenated by placing a plus sign (+) between the program names.

Usage Notes

All fields on this pop-up are optional. If all fields are left blank, the search is not restricted, (i.e., the entire program is searched).

8 List Menu

This chapter describes Encore's List pull-down menu functions and contains these sections:

Section	Page
List Pull-down	282
List - CALL Statements Pop-up	284
List - Equates Pop-up	285
List - Perform Range Names Pop-up	286
List - Program/Subprogram Names Pop-up	287
List - COBOL Subset Names Pop-up	288
List - Segments Pop-up	289
Copy Logic Segment Pop-up	291
Rename Logic Segment Pop-up	292

List Pull-down

The List pull-down is used to access prompt list pop-ups that list information about calls, equates, performs, programs, subsets, and segments (see Figure 131). This chapter describes the actions available on the List pull-down.

Figure 131 • List Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
-------------------------------------                  -----------------
                                ASG-E   _  1. Calls...     rogram: VIARBRWS
Command ===> ________________________      2. Equates...  ------------------
                                           3. Performs...
                                           4. Programs...  *************
                                           5. Subsets...   ************
                                           6. Segments...
```

To display the List Pull-down, follow this step:

- Select List on the action bar and press Enter.

Note:

If other ESW products are installed at your site, other actions may appear on the List pull-down. For information about these features of other products, see the corresponding product User's Guide or online help.

Actions on the List pull-down cannot be selected unless there is an active program, (i.e., a program has been opened on the File pull-down (see Chapter 3, "File Menu," on page 25 for more information)).

Actions

Action	Description
1. Calls	Displays the List - CALL Statements pop-up (see Figure 132 on page 284) used to list programs that are CALLed by the active program.
2. Equates	Displays the List - Equates pop-up (see Figure 133 on page 285) used to list all Equates for the active program.
3. Performs	Displays the List - Perform Range Names pop-up (see Figure 134 on page 286) used to list all COBOL perform ranges in the active program.
4. Programs	Displays the List - Program/Subprogram Names pop-up (see Figure 135 on page 287) used to list the internal subprograms defined within a COBOL II Release 3 (minimum supported release level) program.
5. Subsets	Displays the List - COBOL Subset Names pop-up (see Figure 136 on page 288) used to list the subsets along with a brief description of each.
6. Segments	Displays the List - Segments pop-up (see Figure 137 on page 289) used to list the Logic Segments for the active program. On the List - Segments pop-up, Logic Segments can be copied, renamed, deleted, generated from criteria, selected for viewing, and printed.

List - CALL Statements Pop-up

The List - CALL Statements pop-up, shown in Figure 132, is used to display the programs that are CALLed by the active program.

Figure 132 • List - CALL Statements Pop-up

```
                          List - CALL Statements                  2 CALLS
Command ===> ____________________________________________ Scroll ===> CSR

Select the program CALL(s) to be viewed.  Then press Enter.

S    Called program                                              Mode
-   ------------------------------------------------------------ --------
_   'DSNTIAR'                                                    STATIC
_   'DSN8MCG'                                                    STATIC
****************************** BOTTOM OF DATA ******************************
```

To display the List - CALL Statements pop-up, follow this step:

▶ Select List ▶ Calls from the List pull-down.

Or

Enter the LIST CALLS command on any screen.

When this pop-up first displays, the short message field displays the number of CALLs listed. The LPRINT * command can be entered on this pop-up to copy the list of called subprograms to the List file.

Fields

Field	Description
Line command area	The line command area, to the left of the Called program field, accepts the S line command to select a called program for viewing. The View - Source pop-up displays with the selected program highlighted and tagged.
Called program	The name of the called module.
Mode	The type of CALL: • STATIC - The object of the CALL is a literal and is link-edited into the same load module as the calling program. • DYNAMIC - The object of the CALL is an identifier and is loaded at run time. • INTERNAL - A call to a nested source subprogram.

List - Equates Pop-up

The List - Equates pop-up, shown in Figure 133, is used to display all equates for the active program.

Figure 133 • List - Equates Pop-up

```
                              List - Equates
Command ===> ____________________________________________ Scroll ===> CSR

   Name      Program            Substitution String     Columns 001-053
---------- ---------- ------------------------------------------------------
EQUATE1    VIARDEMO   ZIPCODE-COUNTRY
****************************** BOTTOM OF DATA ******************************
```

To display the List - Equates pop-up, follow this step:

- Select List ▸ Equates from the List pull-down (see Figure 131 on page 282).

  Or

  Enter the LIST EQUATES command on any screen. The LPRINT * command can be entered on this pop-up to copy the list of equates to the List file.

Fields

Field	Description
Name	The name assigned to the equate.
Program	The name of the program containing the equate.
Substitution String	The actual data the Name represents. Prior to execution, this string is substituted into any command which uses the Equate name. This field is left and right scrollable to show the entire string. The substitution string can be changed by typing over it. The right-most column contains a greater than (>) indicator when the substitution string extends past the screen column capability. Typing blanks over the substitution string deletes the equate.
Columns `xxx-xxx`	Indicates the total column width of the substitution string characters per scrollable page.

List - Perform Range Names Pop-up

The List - Perform Range Names pop-up, shown in Figure 134, is used to display all COBOL perform ranges in the program being viewed.

Figure 134 • List - Perform Range Names Pop-up

```
                          List - Perform Range Names      12 PERFORM RANGES
Command ===> ____________________________________________ Scroll ===> CSR

Select the perform range to be viewed.  Then press Enter.

    Perform range name
-   ----------------------------------------------------------------------
_   ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
_   CLOSE-FILES
_   CLOSE-PGM
_   EXCEPTION-REPORT-1
_   GET-NUM-OF-DAYS
_   INIT-ACTION-RECORD
_   INITIALIZE-PGM
_   INITIATE-ACTION
_   OPEN-FILES
_   REPORT-FINAL-CNTRS
_   SUM-TOTAL-DAYS
_   UPDATE-ACCT THRU UPDATE-ACCT-EXIT
******************************** BOTTOM OF DATA ******************************
```

To display the List - Perform Range Names pop-up, follow this step:

- Select List ▸ Performs from the List pull-down (see Figure 131 on page 282).

  Or

  Enter the LIST PERFORMS command on any screen.

When this pop-up first displays, a short message indicates the number of perform ranges found. The LPRINT * command can be entered on this pop-up to copy the list of perform range names to the List file.

Fields

Field	Description
Line command area	The line command area, to the left of the Perform range name field, accepts the S line command to select a perform range for viewing. The View - Source pop-up displays with the selected perform range highlighted and tagged.
Perform range name	The perform label name.

List - Program/Subprogram Names Pop-up

The List - Program/Subprogram Names pop-up, shown in Figure 135, displays the internal subprograms defined within a COBOL II Release 3 (minimum supported release level) program. Internal subprograms are indented in relation to their hierarchy.

Figure 135 • List - Program/Subprogram Names Pop-up

```
                        List - Program/Subprogram Names        1 PROGRAM(S)
Command ===> ____________________________________________ Scroll ===> CSR

Select the program name to be viewed.  Then press Enter.

    Program name
-   ------------------------------------------------------------------
_   VIARDEMO
******************************** BOTTOM OF DATA ******************************
```

To display the List - Program/Subprogram Names pop-up, follow this step:

- Select List ▸ Programs from the List pull-down (see Figure 131 on page 282).

  Or

  Enter the LIST PROGRAMS command on any screen.

When this pop-up first displays, a short message indicates the number of programs found. The LPRINT * command can be entered on this pop-up to copy the list of program names to the List file.

Fields

Field	Description
Line command area	The line command area, to the left of the Program name field, accepts the S line command to select a program for viewing. The View - Source pop-up displays with the selected program highlighted and tagged.
Program name	The program label name.

List - COBOL Subset Names Pop-up

The List - COBOL Subset Names pop-up, shown in Figure 136, displays the subsets along with a brief description of each. Various Encore commands accept subsets as operands, which limit their scope to a particular type of data.

Figure 136 • List - COBOL Subset Names Pop-up

```
                              List - Subset Names
Command ===> ____________________________________________ Scroll ===> CSR

Select the subset to be viewed.  Then press Enter.

S     Subset                            Description
- ---------------------------------------------------------------------------
_ ASsignment      - COBOL: ACCEPT, ADD, SUBTRACT, MULTIPLY, DIVIDE,
                           BY, COMPUTE, MOVE, FROM, INSPECT, EXAMINE,
                           STRING, UNSTRING, SET, SEARCH, TRANSFORM;
                    IDMS: ACCEPT, CHECK, ERASE, GET, INQUIRE, LOAD,
                          PUT, RETURN;
                    SQL: FETCH, SELECT INTO, SET.
_ CAll            - COBOL: CALL, CANCEL, ENTRY,
                           intrinsic function calls;
                    CICS: LINK, XCTL;
                    IDMS: ATTACH, LINK, XCTL.
_ CIcs            - EXEC CICS, EXEC DLI commands.
_ COBOLII         - CALL (COBOLII), PERFORM (COBOLII), CONTINUE, SET,
                    SERVICE LABEL, EVALUATE, INITIALIZE, end verbs.
_ COBOL/370       - Statements and clauses unique to COBOL/370, such as
                    intrinsic function calls, procedure pointers and
```

To display the List - COBOL Subset Names pop-up, follow this step:

- Select List ▸ Subsets from List pull-down (see Figure 131 on page 282).

  Or

  Enter the LIST SUBSETS command on any screen.

The LPRINT * command can be entered on this pop-up to copy the list of subsets to the List file.

Fields

Field	Description
S	Line command area. Type `S` to select a subset for viewing. The View - Source pop-up with a shortened action bar displays with statements of the specified subset type highlighted and tagged. See Chapter 4, "View Menu," on page 69 for more information about the View - Source pop-up.
Subset	The subset name.
Description	The description of the subset.

List - Segments Pop-up

The List - Segments pop-up, shown in Figure 137, is used to provide a set of operations that manage saved extract criteria for a program.

Figure 137 • List - Segments Pop-up

```
                                  List - Segments
Command ===> ____________________________________________ Scroll ===> CSR

   C - Copy Criteria           G - Calculate Logic Segment from Criteria
   R - Rename Criteria         P - Print Logic Segment Criteria
   D - Delete Criteria         V - View Description

  * - indicates Calculation required

   Logic Segment Name                Created     Description
- ---------------------------------- ----------- ---------------------------
_  PERFORM-RANGE-0001                DD-MMM-YYYY Perform Range
****************************** BOTTOM OF DATA ****************************
```

To display the List - Segments pop-up, follow this step:

▶ Select List ▶ Segments from List pull-down (see Figure 131 on page 282).

The LPRINT* command can be entered on this pop-up to copy the list of segments to the List file.

Fields

Field	Description
Line command area	The line command area, to the left of the Logic Segment Name field, accepts these line commands: • C - Copy criteria. Displays the Copy Logic Segment pop-up (see Figure 138 on page 291) used to create a new copy of the Logic Segment. • R - Rename criteria. Displays the Rename Logic Segment pop-up (see Figure 139 on page 292) used to change the name of the Logic Segment. • D - Delete criteria. Removes the Logic Segment from the current program. The criteria and statement definitions are also deleted. • G - Calculate Logic Segment from Criteria. Generates the source statements from the criteria selected. This option is valid only when an asterisk (*) precedes the Logic Segment name, indicating that the program has been reanalyzed.

Field	Description
	• P - Print Logic Segment criteria. Causes a formatted output of the criteria to be transferred to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up (see Chapter 9, "Options Menu," on page 293 for more information). It is not necessary to exit Encore to print the List file. • V - View Description. Displays the Logic Segment Description pop-up (see Chapter 3, "File Menu," on page 25) used to view the long description.
Logic Segment Name	The names of all the current extract criteria in the program. The name is the complete extract criteria name entered when the Logic Segment was created, saved, copied, or renamed. An asterisk (*) preceding the name indicates criteria without statement definition. This occurs under one of these conditions: • The program containing the Logic Segment has been reanalyzed since the Logic Segment was created. • The $$SEGMENTS member in the AKR has been erroneously modified and no longer contains the Logic Segment criteria.
Created	The date that the Logic Segment was created. If the Logic Segment was copied from another definition, the date is the COPY date.
Description	The short description entered on the File - Save Segment pop-up (see Figure 15 on page 37) when the extract criteria was saved or the default description created when the Logic Segment was created. The description may be truncated on the screen display due to column limitations.

Copy Logic Segment Pop-up

The Copy Logic Segment pop-up, shown in Figure 138, is used to create a new version of an existing Logic Segment definition.

Figure 138 • Copy Logic Segment Pop-up

```
                                Copy Logic Segment
Command ===> ____________________________________________ Scroll ===> CSR

Type Logic Segment Name for COPY "TO". Then press Enter.

"From" name  . . . : PERFORM-RANGE-0001
Author . . . . . . : USERID
Created  . . . . . : DD-MMM-YYYY HH:MM:SS

"To" name   PERFORM-RANGE-0001
Description ____________________________________________________

Long Description:
****************************** BOTTOM OF DATA ****************************
```

To display the Copy Logic Segment pop-up, follow this step:

- Enter the C line command on the List - Segments pop-up (see Figure 137 on page 289). After successfully copying the Logic Segment, the two Logic Segment definitions are identical.

The fields on this pop-up contain information entered on the File - Save Segment pop-up (see Figure 15 on page 37 for more information) when the Logic Segment was created or modified.

Fields

Field	Description
"From" name	The name of the Logic Segment being copied, from the List - Segments pop-up (see Figure 137 on page 289).
Author	The user ID of the person who created the Logic Segment being copied.
Created	The date and time the Logic Segment being copied was created.
"To" name	Required. Enter the name of the new Logic Segment.
Description	The short description of the Logic Segment being copied. This description can be changed before pressing Enter to complete the copy.
Long Description	The long description of the Logic Segment being copied.

Rename Logic Segment Pop-up

The Rename Logic Segment pop-up, shown in Figure 139, is used to change the name of an existing Logic Segment definition.

Figure 139 • Rename Logic Segment Pop-up

```
                                Rename Logic Segment
Command ===> ____________________________________________ Scroll ===> CSR

Type new Logic Segment name. Then press Enter.

"Old" Name . . . . : PERFORM-RANGE-0001
Author . . . . . . : USERID
Created  . . . . . : DD-MMM-YYYY HH:MM:SS

"New" Name  PERFORM-RANGE-COBOL-EXTRACT
Description ____________________________________________________

Long description:
****************************** BOTTOM OF DATA ****************************
```

To display the Rename Logic Segment pop-up, follow this step:

- Enter the R line command on the List - Segments pop-up (see Figure 137 on page 289). The rename function changes only the name, and does not affect the criteria or statement definitions of the Logic Segment.

The fields on this pop-up contain information entered on the File - Save Segment pop-up (see Figure 15 on page 37 for more information) when the Logic Segment was created.

Fields

Field	Description
"Old" Name	The name of the Logic Segment being renamed, from the List - Segments pop-up (see Figure 137 on page 289).
Author	The user ID of the person who created the Logic Segment.
Created	The date and time that the Logic Segment was created.
"New" Name	Required. Enter the new name of the Logic Segment.
Description	The short description of the Logic Segment being renamed. This description may be changed before pressing Enter to complete the rename.
Long description	The long description of the Logic Segment.

9 Options Menu

This chapter describes Encore's Options pull-down menu functions and contains these sections:

Section	Page
Options Pull-down	294
Options - Product Parameters Pop-up	296
Options - Product Allocations Pop-up	298
Options - Log/List/Punch Definition Pop-up	300
Options - Script File Allocations Pop-up	304
Options - PF Key Definition Pop-up	305
Options - Processing Modes Pop-up	308
Options - Scratchpad Equate Pop-up	310
Options - Generate Pop-up	312

Options Pull-down

Selecting Options on the action bar displays the Options pull-down, shown in Figure 140. The Options pull-down is used to access the pop-ups that customize the Encore environment.

Figure 140 • Options Pull-down

```
  File  View  Extract  Generate  Search  List  Options  Help
-----------------------------------------
                                ASG-Encor
Command ===> ____________________________

                 ********************    ****
               **********************  ******
              *****            *****  ******
             *****            *****  ******
            **********************  *********
           **********************   *********
          *****            *****               ******  *****          ***
         *****            *****               ******  *****          ***
        *****            *************************  ******************
       *****            *************************    *****************

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

```
_  1. Product Parameters...
   2. Product Allocations...
   3. Log/list/punch...
   4. Script file allocations...
   5. PF keys...
   6. Modes...
   7. Equates...
   8. Generate...
```

Note:

If other ESW products are installed at your site, other options may appear on the Options pull-down, and additional pop-ups may be available. For information about these features of other products, see the corresponding product User's Guide or online help.

Customization of the Encore environment includes these functions:

- Determining which screen displays when you open a program in Encore.
- Defining the Log, List, Punch, and Work files used by Encore.
- Specifying the default script data names.
- Determining the values of the PF keys used with Encore.
- Defining the cursor substitution character that is used with Encore commands.
- Determining whether Equates are to be saved automatically.
- Specifying the output and format settings for COBOL modules generated by Encore.

Actions

Action	Description
1. Product Parameters	Displays the Options - Product Parameters pop-up (see Figure 141 on page 296) used to control the online operation of Encore. This includes whether an alarm is sounded when an error message displays and other processing options.
2. Product Allocations	Displays the Options - Product Allocations pop-up (see Figure 142 on page 298) used to specify the DASD volumes for the Log, List, Punch, and Work files, and to specify the space for the Work file.
3. Log/list/punch	Displays the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300) used to define the Log, List and Punch files and to customize the dataset numbers for these files. These files are used to log system messages and to output the results of some Encore commands.
4. Script file allocations	Displays the Options - Script File Allocations pop-up (see Figure 145 on page 304) used to specify or modify the default script dataset names.
5. PF Keys	Displays the Options - PF Key Definition pop-up (see Figure 146 on page 305) used to display or change the PF keys used with Encore.
6. Modes	Displays the Options - Processing Modes pop-up (see Figure 148 on page 308) used to display or change the setting for each mode.
7. Equates	Displays the Options - Scratchpad Equate pop-up (see Figure 149 on page 310) used to define a name for a character string.
8. Generate	Displays the Options - Generate pop-up (see Figure 150 on page 312) used to specify output and format settings for COBOL modules generated by Encore.

Options - Product Parameters Pop-up

The Options - Product Parameters pop-up, shown in Figure 141, is used to set parameters that affect the online operation of Encore.

Figure 141 • Options - Product Parameters Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
------------------------
                                 Options - Product Parameters
Command ===> ___________
                                 Type desired product parameters. Then press
                 *********       Enter.
               ***********
             *****             Cursor Substitution Character    %
            *****
           ***************       COBOL Module Exit  . . . . . . . --------
          ***************
         *****                   Options                    Default View
        *****           *        _  Alarm                   3   1.  Structure View
       *****           **        _  Confirm on Exit             2.  Tree View
      *****           ***        /  Save EQUATES                3.  Source View
                                 _  Save EDIT File              4.  None

        Copyright Allen
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

To display the Options - Product Parameters pop-up, follow this step:

- Select Options ▸ Product Parameters on the Options pull-down (see Figure 140 on page 294).

  Or

  Enter the PARMDEF command on any screen.

Fields

Field	Description
Cursor Substitution Character	Optional. Identifies the cursor token substitution character. The cursor substitution character can be used in any primary command on the Source View, Tree View, and Structure View screens, and on the View - Source, View - Tree, and View - Structure pop-ups. See Chapter 4, "View Menu," on page 69 for more information. Cursor substitution saves time by allowing use of this character in a command, then placement of the cursor on the token in the source code. Any character can be specified as the token substitution character, however, it should be a character that is rarely used in commands. The default substitution character is the percent (%) sign.

Field	Description
COBOL Module Exit	Optional. Enter the name of the module to be CALLed when abnormal exit conditions are detected in the generated program. In order to avoid creating a new execution sequence path, the program should be aborted. The name specified on the pop-up is used to generate a CALL statement which replaces the default statement sequence. The default is a STOP RUN statement. The COBOL Module Exit is valid only for Transaction and Transaction Complement extract types.
Options	• Alarm - Optional. If specified, the audible alarm on the terminal sounds when an error message displays. If left blank, the alarm does not sound. The default is the alarm sounds. • Confirm on Exit - Optional. If specified, confirmation panels appear when you exit Encore and some screens within the product. • Save EQUATES - Optional. If specified, Equates are saved when closing a program. If left blank, Equates are not saved. See the EQUATE command for more information about Equates (see Chapter 11, "Commands," on page 321). The default is that Equates are saved. • Save EDIT File - Optional. If specified, the generated module source program file is retained upon return from the Edit session. If left blank, the file is deleted at the end of the Edit session. The default is the file is not saved.
Default View	Optional. Selects the View type that displays when a program is first opened. If 1, 2, or 3 is selected, the corresponding View screen displays with the opened program. The default is 4. None, which indicates that when a program is first opened, the primary Encore screen remains displayed. A View may then be selected on the View pull-down. See Chapter 4, "View Menu," on page 69 for more information.

Options - Product Allocations Pop-up

The Options - Product Allocations pop-up, shown in Figure 142, is used to specify the DASD volumes for the Log, List, Punch, and Work files, and to specify the space for the Work file.

Figure 142 • Options - Product Allocations Pop-up

```
                    Options - Product Allocations
Command ===> ____________________________________________________________

  Log file:
     Generic unit . . . SYSDA     (generic group name or unit address)
     Volume serial  . .           (blank for authorized default volume)
  List file:
     Generic unit . . . --------  (generic group name or unit address)
     Volume serial  . .           (blank for authorized default volume)
  Punch file:
     Generic unit . . . SYSDA     (generic group name or unit address)
     Volume serial  . .           (blank for authorized default volume)
  Work file:
     Generic unit . . . SYSDA_    (generic group name or unit address)
     Volume serial  . . ______    (blank for authorized default volume)
     Space units  . . . CYLS      (BLKS, TRKS or CYLS)
     Primary space  . . 5         (space units)
     Secondary space    5         (space units)
```

To display the Options - Product Allocations pop-up, follow this step:

- Select Options ▸ Product Allocations from the Options pull-down (see Figure 140 on page 294).

  Or

  Enter the ALLOCDEF command on any screen.

Fields

Field	Description
Log file	Specifies the Generic unit and volume serial number for the Log file which is allocated upon entry into Encore. The Log file is used for error messages and log commands. File characteristics are specified on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300).
List file	Specifies the Generic unit and volume serial number for the List file which is allocated the first time a request is made to print output, (e.g., printing a Logic Segment description). The List file is used for all printed output. File characteristics are specified on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300).

Field	Description
	For more information, see the LPRINT command (see Chapter 11, "Commands," on page 321), the File - Open Segment pop-up (see Chapter 3, "File Menu," on page 25), the Generate - Output File pop-up (see Chapter 6, "Generate Menu," on page 203), and the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300).
Punch file	Specifies the Generic unit and volume serial number for the Punch file which is allocated the first time a request is made to punch output, (e.g., punch a Logic Segment source). The Punch file is used for all punched output. File characteristics are specified on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300). For more information, see the Generate - Output File (see Chapter 6, "Generate Menu," on page 203) and the Options - Log/List/Punch Definition pop-ups (see Figure 143 on page 300).
Work file	Specifies Generic unit, volume serial number, and space requirements for the Work file which is allocated upon entry into Encore. The Work file is a temporary file. The sizes of the programs being worked with should be considered when determining space requirements for the Work file. Approximately one cylinder is required for a 5,000 line program on a 3380 device.

Options - Log/List/Punch Definition Pop-up

The Options - Log/List/Punch Definition pop-up, shown in Figure 143, is used to set values for allocating, formatting, and processing the Encore Log, List, and Punch files and to customize the dataset names for these files. Once a file is processed, a new file is allocated for subsequent print and punch requests.

Figure 143 • Options - Log/List/Punch Definition Pop-up

```
                    Options - Log/List/Punch Definition
Command ===> _____________________________________________________________

1 - Process log file  2 - Process list file  3 - Process punch file
                      4 - Customized data set name

Options                     Log            List           Punch
----------------            ---            ----           -----
Process option  . . . . . . PD             PD             PK
Primary tracks      . . . . 1              1              1
Secondary tracks      . . . 2              5              5
Lines per page  . . . . . . 56             56             56
Sysout class  . . . . . . . *              *              *

Process options:  PK (print/keep), PD (print/delete), K, or D.


Job statement information:
  //USER     JOB (ACCOUNT),'NAME',
  //         MSGCLASS=A
  //*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
  //*
```

To display the Options - Log/List/Punch Definition pop-up, follow this step:

- Select Options ▸ Log/list/punch from the Options pull-down (see Figure 140 on page 294).

  Or

  Enter the PRINTLOG or PRINTLST command on any screen.

The Log file is allocated upon entry into Encore. The List file is allocated the first time a print request is made. The Punch file is allocated the first time a punch request is made.

Volumes for the Log, List, and Punch files are specified on the Options - Product Allocations pop-up (see Figure 142 on page 298).

Options

Field	Description
1 - Process log file	Verify or change the Options for the Log file, then select option 1 to process the Log file. A new file is allocated to collect additional data, if required. The file name is in this format: `userid.ENCxxxxx.VIALOG` where *xxxxx* is a unique number assigned by Encore. If the PK or PD Process option is specified, Job statement information must be entered prior to selecting option 1. The Log file is processed when you press Enter. It is not necessary to exit Encore to process the Log file.
2 - Process List file	Verify or change the Options for the List file, then select option 2 to process the List file. A new file is allocated to collect additional data, if required. The file name is in this format: `userid.ENCxxxxx.VIALIST` where *xxxxx* is a unique number assigned by Encore. If the PK or PD Process option is specified, Job statement information must be entered prior to selecting option 2. The List file is processed when you press Enter. It is not necessary to exit Encore to process the List file.
3 - Process punch file	Verify or change the Options for the Punch file, then select option 3 to process the Punch file. This file is defined as fixed block with 80 byte records. The file name is in this format: `userid.ENCxxxxx.VIAPUNCH` where *xxxxx* is a unique number assigned by Encore. If the PK or PD Process option is specified, Job statement information must be entered prior to selecting option 3. The Punch file is processed when you press Enter. It is not necessary to exit Encore to process the Punch file.
4 - Customized dataset name	Displays the Options - Log/List/Punch Name Customization pop-up (Figure 144 on page 302), which enables you to specify a datatset where these files are saved.

Figure 144 • Options - Log/List/Punch Name Customization Pop-up

```
                Options - Log/List/Punch Name Customization
Command ===> ____________________________________________________________

You can define a customized name when you choose option PK (print/keep) or
K (keep) by specifying U(ser).  Specifying Y(es) on Prompt later lets you
define a custom name as you process each log/list/punch file.  Otherwise,
specify data set name and file mode.  Then press Enter.

Options                   Log     List    Punch
------------------------- ---     ----    -----
File Naming . . . . . . . S       S       S      U(ser) or S(ystem)
Prompt later for DSN  . . _       _       _      Y(es)  or N(o)

The following are needed if U(ser) and N(o) are specified above

Log Data set name   ____________________________________________    (Seq)
      File Mode . . _      O(verwrite) or A(ppend)
List Data set name  ____________________________________________    (Seq)
      File Mode . . _      O(verwrite) or A(ppend)
Punch Data set name ____________________________________________    (Seq)
      File Mode . . _      O(verwrite) or A(ppend)
```

If the K or PK process option is specified, you can customize the dataset where the log, list, or punch file is allocated. By default, these files are allocated as:

`userid.ENC`*`xxxxx`*`.VIA`*`nnnnn`*

where:

`xxxxx` is a sequential number from 00001 to 99999.

`nnnnn` is *`LOG`* for Log.

`LIST` for List.

`PUNCH` for Punch files.

If you specified a TSO prefix, the prefix is appended to the beginning of the file name allocated.

Type `U` in the File Naming field for Log, List, or Punch to indicate a user-defined dataset name. If you specify `N` in the Prompt later for DSN field, you must enter a dataset name in the corresponding dataset name field, and specify Overwrite or Append in the File Mode field.

Fields

Options	
Field	**Description**
Process option	Required. Enter one of the listed processing options. The default is PD for the Log and List files, and PK for the Punch file.
Primary tracks	Required. Enter the number of primary tracks to allocate. A size change does not take effect until the next allocation occurs. The default is 1 for the Log, List, and Punch files.
Secondary tracks	Required. Enter the number of secondary tracks to allocate. A size change does not take effect until the next allocation occurs. • 2 is the default for the Log file. • 5 is the default for the List file. • 5 is the default for the Punch file.
Lines per page	Required. Enter the number of print lines per page. Typical maximum values are 60 for six lines per inch and 80 for eight lines per inch. The default is 56.
SYSOUT class	Required. Enter the SYSOUT class value. The default is an asterisk (*), which sends the SYSOUT to the destination specified in the MSGCLASS parameter on the JOB statement.
Process options	Lists these available options for the Log, List, and Punch files: • PK - Print and keep • PD - Print and delete • K - Keep without printing • D - Delete without printing
XXXXX FILE IS ALLOCATED	This message displays when the Log, List, or Punch file has been properly allocated. If the message does not appear, check the assignments on the Options - Product Allocations pop-up (see Figure 142 on page 298). *XXXXX* is `LOG`, `LIST`, or `PUNCH` depending on the file that was allocated.
Job statement information	Enter the appropriate JOB statement information for your installation. These JCL statements are required if the PK or PD processing option is specified.

Options - Script File Allocations Pop-up

The Options - Script File Allocations pop-up, shown in Figure 145, is used to display or modify the default script file concatenation sequence for your session.

Figure 145 • Options - Script File Allocations Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-                                          ---------------------
          Options - Script File Allocations   Program: VIARDEMO
C  Command ===> ____________________________________  ----------------------

       R - Restore default Script allocations        *****************
                                                     ****************
  Enter desired Script file concatenation.           ***
                                                     **
  Script file data set names:                        *     *******
                                                          ********
    ----------------------------------------------            ***
    ----------------------------------------------           ***
    ----------------------------------------------        **********
    -                                                     *********

       Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

To display the Options - Script File Allocations pop-up, follow this step:

- Select Options ▸ Script file allocations from the Options pull-down (see Figure 140 on page 294).

Fields

Field	Description
Restore default Script allocations	Enter R to restore the defaults set during the install process.
Script file dataset names	Enter the dataset names for the desired default script files. The files are concatenated in the order entered. You can override an existing file by typing over its dataset name. You can remove an existing dataset name from the concatenation by blanking out the name. If you blank out all dataset names, the default concatenation is restored. **Note:** When initializing for defaults, the lines are filled from the bottom to allow concatenation.

Usage Notes

When Encore is installed, the default script files are specified for the site. You can use the Options - Script File Allocations pop-up (see Figure 145 on page 304) to modify the default dataset names or the concatenation sequence for the script files. The changes are maintained in your profile.

Options - PF Key Definition Pop-up

The Options - PF Key Definition pop-up, shown in Figure 146, is used to display or redefine PF key values.

Figure 146 • Options - PF Key Definition (01-12) Pop-up

```
                    Options - PF Key (01-12) Definition
Command ===> _____________________________________________________

Press Enter to process changes and/or to display alternate keys.
Press PF3/15 (END) to exit.

       Number of PF keys: 24        Terminal type: 3278

PF01 HELP
PF02 SPLIT
PF03 END
PF04 ZOOMIN
PF05 RFIND
PF06 FX %
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 LEFT
PF11 RIGHT
PF12 RECALL
```

To display the Options - PF Key Definition pop-up

1. Select Options ▸ PF keys from the Options pull-down (see Figure 140 on page 294).

   Or

   Enter the KEYS command on any Encore screen. PF keys 1-12 display initially.

2. Press Enter to display PF keys 13-24. Any command or data value can be assigned to a PF key.

A similar screen exists for PF keys 13-24, shown in Figure 147, and displays when you press Enter. Any command or data value can be assigned to a PF key.

Figure 147 • Options - PF Key Definition (13-24) Pop-up

```
                    Options - PF Key (13-24) Definition
Command ===> _____________________________________________________

Press Enter to process changes and/or to display alternate keys.
Press PF3/15 (END) to exit.

       Number of PF keys: 24        Terminal type: 3278

PF13 HELP
PF14 SPLIT
PF15 END
PF16 ZOOMIN
PF17 RFIND
PF18 _______________________________________________________
PF19 UP
PF20 DOWN
PF21 SWAP
PF22 BRANCH
PF23 BRANCH BACKUP
PF24 RECALL
```

Fields

Field	Description
Number of PF keys	The number of supported PF keys.
Terminal type	This number indicates the type of terminal being used. Encore supports these 3270 type terminals: • Model 2 (24 lines x 80 columns) • Model 3 (27 lines x 80 columns) • Model 4 (43 lines x 80 columns) • Model 5 (27 lines x 133 columns)
PF01 - PF12 or PF13 - PF24	Required. The value assigned to the PF keys, which can be changed by typing over it. To reset a PF key to its default value, delete the entry and press Enter. To set a PF key to have no value, (i.e., to turn off a PF key). Enter the value `NOP`.

Usage Notes

When Encore is installed, all PF keys are set to these default values:

Primary Key	Default	Alternate Key	Default
PF01	HELP	PF13	HELP
PF02	SPLIT	PF14	SPLIT
PF03	END	PF15	END
PF04	ZOOMIN	PF16	ZOOMIN
PF05	RFIND	PF17	RFIND
PF06	FX%	PF18	
PF07	UP	PF19	UP
PF08	DOWN	PF20	DOWN
PF09	SWAP	PF21	SWAP
PF10	LEFT	PF22	BRANCH
PF11	RIGHT	PF23	BRANCH BACKUP
PF12	RECALL	PF24	RECALL

Note:

The alternate PF keys may be PF01-12 or PF13-24, depending on your site's specifications.

Options - Processing Modes Pop-up

The Options - Processing Modes pop-up, shown in Figure 148, is used to enable or disable the mode indicated by the specified option.

Figure 148 • Options - Processing Modes Pop-up

```
                          Options - Processing Modes
Command ===> ____________________________________________ Scroll ===> CSR

Type option settings.  Then press PF3/15 (END) to save options and exit.

 Option     Set    Description
----------  ---  -------------------------------------------------------
LEARN       OFF  Display internally generated Primary Commands
SCRIPT      OFF  Script facility is disabled
XMODE       OFF  Exclude all lines before executing Primary Commands
```

To display the Options - Processing Modes pop-up, follow this step:

- Select Modes on the Options pull-down (see Figure 140 on page 294).

  Or

  Enter the SET command without operands on any screen.

A setting can be changed by typing over it, or by using the SET command. See Chapter 11, "Commands," on page 321 for more information about the SET command.

Fields

Field	Description
Option	The mode that is to be enabled or disabled. Each mode is described in this table. All options can be reset to their default values by entering SET DEFAULT.
Set	Indicates the status of the mode or operand. Type over the status in the Set column to change the status of the corresponding operand. Values that are not set to the default status display in bold.
Description	Describes the mode (option) that is being set. The Script facility also indicates whether it is enabled or disabled.

Field	Description
LEARN	When ON, all primary commands that are generated internally when actions are executed display on the LEARN Mode - Generated Command pop-up (see Figure 158 on page 418). See the SET command (Chapter 11, "Commands," on page 321) for more information about the LEARN Mode - Generated Command pop-up. The default for this Option is OFF.
SCRIPT	The SCRIPT option is used to automatically generate a script file. When this mode is ON, all Encore primary commands entered during a session are saved. Using this mode allows inclusion of a predefined command sequence in any session. This option, or the SET SCRIPT ON command, allocates and opens a dataset and saves all Encore primary commands in this dataset. The dataset name is in this format: `userid.ENCnnnnn.VIASCRIP` where *nnnnn* is a number assigned by Encore, beginning with 00001, and incremented by 1 for each new script file. The name of the dataset displays when it is opened or closed. Scripts support all Encore primary commands. The script file is closed (ignored) when this mode is OFF or the SET SCRIPT OFF command is entered. Once a script file is closed, it can be used as the dataset name in the EXECUTE command. The default is OFF. See Chapter 11, "Commands," on page 321 for more information.
XMODE	The XMODE option determines if lines are excluded from the screen before a primary command is executed. When this mode is ON, all lines are excluded before a primary command is executed. This mode affects all primary commands except BRANCH, LOCATE, RFIND, and SCROLL. The default is OFF.

Options - Scratchpad Equate Pop-up

The Options - Scratchpad Equate pop-up, shown in Figure 149, is used to define a name for a character string.

Figure 149 • Options - Scratchpad Equate Pop-up

```
 File  View  Extract  Generate  Search  List  Options  Help
-----
Comma
        Options - Scratchpad Equate

        To create an equate, type a name and string.  Then press Enter.

        Equate name    _________

        Equated string ___________________________________________________

         *****          *****            ******  *****         ***
        *****          *****            ******  *****        ***
       *****          *************************  *******************
      *****          ************************    *****************

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

To display the Options - Scratchpad Equate pop-up, follow this step:

▶ Select Options ▶ Equates from the Options pull-down (see Figure 140 on page 294).

Fields

Field	Description
Equate name	Required. Names must conform to these standards: • 1-10 alphanumeric characters • First character must be alphabetic • A hyphen is the only special character allowed The name may also be 1-4 DBCS characters
Equated string	Optional. A character string to be substituted. A character string can be a long command, pattern, data name, concatenated data name, etc. Literals and blanks can be specified in the character string if desired. Character strings that include blanks must be enclosed with single or double quotes. The string may be a DBCS string. If the Equated string is not entered, the Equate name is deleted.

Usage Notes

Equated names can be used during a View or Extract session to reduce the number of keystrokes. Multiple equate statements can be used if desired.

If an equated name is used in another EQUATE command, it must first be defined. For example:

```
EQ CC TOT-COST
EQ BB CC + PROD-COST
```

where:

BB now contains `TOT-COST + PROD-COST`

Equates created in View or Extract can be saved in the AKR, by using the SAVE command or the Options - Product Parameters pop-up (see Figure 141 on page 296).

The substitution string can also be changed on the List - Equates pop-up (see Figure 133 on page 285) by typing over it with the new value. See Chapter 8, "List Menu," on page 281 for more information.

For more information, see the LIST command and the SAVE command (see Chapter 11, "Commands," on page 321), the Options - Product Parameters pop-up (see Figure 141 on page 296), and the List - Equates pop-up (see Chapter 8, "List Menu," on page 281).

Options - Generate Pop-up

The Options - Generate pop-up, shown in Figure 150, is used to specify output and format settings for COBOL modules generated by Encore.

Figure 150 • Options - Generate Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-                                                          -------
C                      Options - Generate                  ARDEMO
                                                           --------
  Select the desired COBOL Module Generate Options.
                                                           ***
  ✓ Display EDIT Options          _ Output to Existing Library  **

  COBOL Source
  _ Full DATA DIVISION            _ Retain Source Line Numbers
  _ Retain COPY statements        _ Retain PERFORMed Paragraphs
  _ In-line Communication Area

    *****          ************************    *****************

       Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
        Visit the ASG Support Web Site at www.asg.com
```

To display the Options - Generate pop-up, shown in Figure 150, follow this step:

- Select Options ▸ Generate from the Options pull-down (see Figure 140 on page 294).

Fields

Field	Description
Display EDIT Options	Optional. If specified, the Edit Options pop-up displays so you can specify the source manager. If left blank, the Edit Options pop-up is bypassed, and the Editor screen displays. The Edit Options pop-up displays only if you select Output to Existing Library.
Output to Existing Library	Optional. If specified, the Edit Entry Panel displays so you can initiate an Edit session. If Display EDIT Options is also specified, the Edit Options pop-up displays first. On the Edit Entry Panel, a member name can be entered for a new source member library. After the member is identified and the Edit session established, use the RNSCOPY command to copy the generated module into the current Edit member. If left blank, an Edit session is established against a temporary file containing the generated module. Use the ISPF CREATE or REPLACE command to save the file. This is the default.

Field	Description
COBOL Source	
Full DATA DIVISION	Optional. If specified, the DATA DIVISION is an exact copy of the DATA DIVISION from the program being re-engineered (i.e., including COPY statements). If left blank, a DATA DIVISION is generated that supports only the derived PROCEDURE DIVISION. All data items which have parents are resolved to the 01 level, so that major program data structures are not destroyed. Also, if data items are defined within a COPY statement, then the COPY statement itself is generated in the new module. This is the default.
Retain COPY Statements	Optional. If specified, the generated module does not expand DATA DIVISION statements if the only reason for expansion was due to the VALUE clauses. The default is to expand COPY statements.
In-line Communication Area	Optional. If specified, COPY Statements are included instream.
Retain Source Line Numbers	Optional. If specified, existing PROCEDURE DIVISION line numbers are retained in the generated module. If left blank, statements in the generated module contain new line numbers. The default is not to use existing line numbers.
Retain PERFORMed Paragraphs	Optional. If specified, the statements for all PERFORM ranges executed are retained in the generated module. If left blank, only PERFORM ranges specifically selected in the extract criteria are included in the generated module. The default is not to retain PERFORMed paragraphs.

10 Help Menu

This chapter describes Encore's Help pull-down menu functions and contains these sections:

Section	Page
Introduction	315
Help Pull-down	316
Help - Specific ASG Command Pop-up	318
Help - Specific ASG Message Number Pop-up	319
Help - About Pop-up	320

Introduction

Help commands display information about the current screen or pop-up, commands, messages, or abend codes. Help information is also obtained by selecting an action on the Help pull-down, which is explained in this chapter.

Help Pull-down

The Help pull-down, shown in Figure 151, is used to access the online help facility.

Figure 151 • Help Pull-Down

```
  File  View  Extract  Generate  Search  List  Options  Help
-----------------------------------------------
                                  ASG-Encore
Command ===> ____________________________________
                                                  _ 1. Current screen
                                                    2. Current message
                                                    3. All commands
                                                    4. Specific command...
                                                    5. Specific message...
                                                    6. Common abends
                                                    7. Table of contents...
                                                    8. Index...
                                                    9. Action bar...
                                                   10. About...

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

To display the Help pull-down, follow this step:

- Select Help on the action bar and press Enter.

Actions

Action	Description
1. Current screen	Displays help for the screen or pop-up that is currently displayed.
2. Current message	Displays help for the message that is currently displayed.
3. All commands	Displays a complete list of all Encore primary commands, from which information about a specific command displays when you select the appropriate number
4. Specific command	Displays the Help - Specific ASG Command pop-up (see Figure 152 on page 318) used to obtain help about a specific Encore primary command.
5. Specific message	Displays the Help - Specific ASG Message Number pop-up (see Figure 153 on page 319) used to obtain help about a specific message number.

Action	Description
6. Common abends	Displays the Abends screen, from which information about a specific abend displays when you select the appropriate number. Select topic 2 - ASG Abends on this screen to display the ASG Abend Codes screen, which lists all the ASG user abends and their explanations.
7. Table of contents	Displays the online help table of contents, which lists general information topics about Encore.
8. Index	Displays the first online help index screen.
9. Action bar	Displays the Help Tutorial for the Encore action bar.
10. About	Displays the Help - About pop-up (see Figure 154 on page 320) that lists information about the currently installed levels of Encore and Center.

See Appendix A, "Help Facility," on page 503, or consult online help, for more information.

Help - Specific ASG Command Pop-up

The Help - Specific ASG Command pop-up, shown in Figure 152, is used to obtain help about a specific Encore primary command.

Figure 152 • Help - Specific ASG Command Pop-up

```
                    Help - Specific ASG Command

To display a message that describes an ASG
command, type the command below.  Then press
Enter.

       Command . . . . _---------------

To display additional command information, enter
the HELP command (PF1/13) from the screen that
displays the command message.
```

To display the Help - Specific ASG Command pop-up, follow this step:

- Select Help ▸ Specific command from the Help pull-down (see Figure 151 on page 316).

Field

Field	Description
Command	Required. Enter an Encore primary command. The Help Tutorial for that command displays. See online help or Appendix A, "Help Facility," on page 503 for more information.

Help - Specific ASG Message Number Pop-up

The Help - Specific ASG Message Number pop-up, shown in Figure 153, is used to display help for a specific ASG message.

Figure 153 • Help - Specific ASG Message Number Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-                                                        ----------------------
C          Help - Specific ASG Message Number                Program: VIARDEMO
                                                         -----------------------
 To display the description of a message, type
 the message number below.  Then press Enter.            ******************
                                                         *****************
                Message . . . . _-------                 ****
                                                         ***
                                                         **      *******
        **********************   ******************  *****     ********
       *****            *****               ******  *****          ***
      *****            *****               ******  *****           ***
     *****            *************************  ******************
    *****            *************************    ******************

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

To display the Help - Specific ASG Message Number pop-up, follow this step:

- Select Help ▸ Specific message from the Help pull-down (see Figure 151 on page 316).

Field

Field	Description
Message	Required. Enter an ASG message number. The HELP Explanation and Action Panel for that message displays. See Appendix A, "Help Facility," on page 503, for more information.

Help - About Pop-up

The Help - About pop-up, shown in Figure 154, is used to display information about the currently installed releases of Encore and Center.

Figure 154 • Help - About Pop-up

```
                    Help - About

The following is release information for this ASG
product.

 Product name . . . . . . . : ASG-Encore
  Release number  . . . . . : x.x
  Maintenance level . . . . : xxx

 ASG-CENTER release number  : x.x
  Maintenance level . . . . : xxx

 Operating system . . . . . : OS(390)
```

To display the Help - About pop-up, follow this step:

- Select Help ▸ About on the Help pull-down (see Figure 151 on page 316).

Fields

Field	Description
Product name	ASG-Encore, or the name of the ESW product you are currently running.
Release number	The currently installed release number of Encore.
Maintenance level	The currently installed Encore PRS number.
Center release number	The currently installed release number of Center.
Maintenance level	The currently installed Center PRS number.
Operating system	The operating system under which you are running.

Note:

This information is requested when the ASG Customer Support group is contacted for assistance. This information also displays when you enter the PRODLVL command on any screen.

11 Commands

This chapter contains command syntax diagrams and descriptions for all of Encore's commands and contains these sections:

Section	Page
Introduction	324
Command Processing	324
Command Syntax Diagrams	325
Cursor Position	327
Cursor Substitution Character	327
& (Retain) Command	328
ALLOCDEF Command	329
ANALYZE Command	330
BRANCH Command	331
CALL Command	334
CANCEL Command	335
CURRENT Command	336
DISPLAY Command	337
END Command	338
EQUATE Command	339
EXCLUDE Command	340
EXECUTE Command	346

Section	Page
EXIT Command	350
EXPORT Command	351
FIND Command	353
FINDXTND Command	358
GOBACK Command	368
HELP Command	369
JUMP Command	371
KEYS Command	372
LEVELS Command	374
LIST Command	375
LOCATE Command	377
LPRINT Command	379
LPUNCH Command	381
PARMDEF Command	389
PREF Command	390
PRINTLOG Command	393
PRINTLST Command	394
PROCESS Command	395
PRODLVL Command	399
PROGRAM Command	400
QUALIFY Command	401
RECALL Command	402
REDO Command	404

Section	Page
REPEAT Command	405
RESET Command	406
RETURN Command	408
RFIND Command	409
RNSCOPY Command	410
RSTRUCTURE-VIEW Command	411
RTREEVW Command	412
SAVE Command	413
SCROLL Command	414
SELECT Command	415
SET Command	417
SORT Command	419
SOURCE-VIEW Command	420
STRUCTURE-VIEW Command	421
TREEVIEW Command	423
UNSELECT Command	425
VIEW Command	427
ZOOMIN/ZOOMOUT Commands	428

Introduction

Encore is an interactive product driven by menus and screens. Easily-understood commands are also available to make your task easier. Encore accepts primary and line commands that are entered in the same manner as ISPF commands. Primary commands are entered in the designated command input area. On most screens, line commands are entered in a selection column field; in Source View, line commands are entered over the line numbers (columns 1-6). See the specific screen for more information about entering line commands.

Encore supports all ISPF system commands on the appropriate screens. All Encore commands are described in this chapter.

Some of the most frequently used commands have been assigned to PF keys. The KEYS command can be entered from any screen to change the PF key assignments. The Options - PF Key Definition pop-up displays.

See online help and Chapter 9, "Options Menu," on page 293 for more information about the Options - PF Key Definition pop-up, and a list of the defaults assigned to the PF keys.

Note:

If other ESW products are installed at your site, other commands may be available. For information about the features of other products, see the corresponding product User's Guide or online help.

Command Processing

Encore line commands are processed before primary commands. Commands entered by pressing a PF key are handled as if entered in the command input area. If a command is entered by pressing any PF key and there is a command in the command input area, the contents of the command input area are appended to the PF key command. The combined commands are then executed as a whole. Multiple commands entered in the command input area are separated by a semicolon (;) and are processed in a left-to-right sequence. Command results display when one of these conditions occurs:

- The last command is processed and all commands in the sequence completed successfully.
- A command is not successfully processed. If an error occurs in the sequence, processing stops and an error message displays. Results display for the commands that successfully complete. The command causing the error and all remaining commands in the sequence display in the command input area.

Each recallable command in a command sequence is recognized separately for use with the RECALL command.

Command Syntax Diagrams

These notational conventions and symbols are used to describe command syntax:

Item	Description
ABBREViations	Command abbreviations are shown in uppercase letters; lowercase letters in the command are optional.
lowercase	Lowercase values indicate user-supplied variable information.
UPPERCASE	Uppercase words indicate commands or keywords.
Bold	Bold operands are available only if Insight is installed and a Insight analysis has been run on the COBOL program being tested.
Underline	The default value of an operand is underlined.
|	A vertical bar separates synonymous commands or operands.
——▶	A right ending arrow indicates that the command syntax is continued on the next line.
▶——	A right beginning arrow indicates the command syntax is continued from the previous line.
——▶◀—	Right and left ending arrows indicate the end of the command syntax.
—— required ——	An operand or keyword appearing on the main command line is required.
choice1 choice2 choice3	Stacked operands on the main line indicate a choice of one required item.
optional	An operand or keyword appearing below the main command line is optional.
choice1 choice2	Stacked operands below the main line show a choice of one optional item.

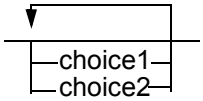
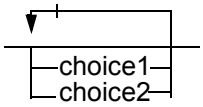
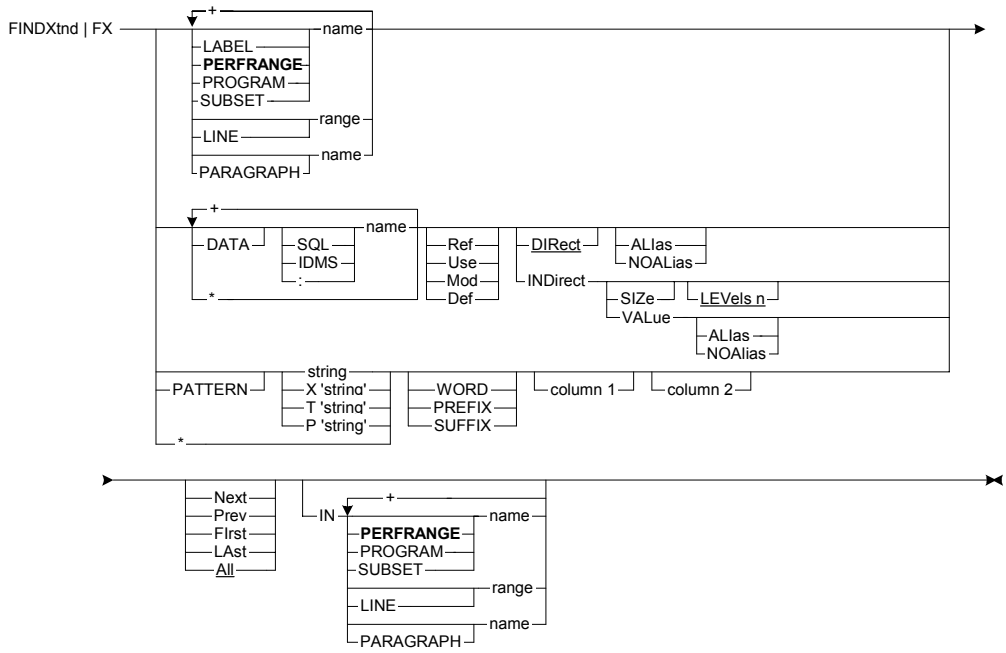
Item	Description
	A returning arrow indicates that more than one operand can be chosen.
	A returning arrow with a plus sign (+) indicates that operands can be concatenated by placing a + between them.

Figure 157 shows a syntax diagram for the FINDXTND command. FINDXTND or a minimum of FINDX or FX can be entered. FINDXTND must be entered with a target operand as indicated by the name variable on the main path of the line. If the path is followed down the first or second vertical line, any of the target operands can be specified. If more than one operand is specified, separate each with a space. Operands that can be concatenated are indicated by a returning arrow that includes a plus sign (+) in the line. Data name operands (Ref, Use, etc.) pertain to all data names in a concatenated series. The direction and intarget operands are optional since they are below the main path of the line.

Figure 155 • FINDXTND Syntax Diagram Example



Bold operands are only available with Insight

Cursor Position

The result of the command depends on the starting point of the search. A starting point can be specified by entering a line number or label name, or by using the cursor as the starting point. If the cursor is used as the starting point, these rules apply:

If the Cursor is in	The search is started
Command input area	In the first column of the first source line on the screen.
Line command area	From the first column on that line.
Display area	From the cursor position.

Cursor Substitution Character

The cursor character is a token substitution character which may be used in some primary commands. A token is a contiguous set of characters preceded by and followed by a blank, period, comma, or parenthesis. The cursor substitution character is defined on the Options - Product Parameters pop-up (see Figure 141 on page 296). Any character may be chosen, but it should be one which is rarely used in commands. The default is percent (%).

The cursor substitution character saves time and typing in commands. The cursor substitution character is typed anywhere on the command line, then the screen cursor is placed on the token that is to take its place in the command. For example, to find all occurrences of HLD-ZIP-PREFIX, enter this FINDXTND command in the command input area:

```
FX %
```

Next, place the screen cursor anywhere on HLD-ZIP-PREFIX in the source code. The command FX HLD-ZIP-PREFIX is executed when Enter is pressed. Note that the cursor substitution character must have a space before it.

Multiple cursor tokens can be used to specify consecutive tokens in the source code. For example, enter this in the command input area:

```
FX % % %
```

If the screen cursor is then placed on HLD-ZIP-PREFIX, the next two tokens are also picked up in the command, resulting in the command FX HLD-ZIP-PREFIX OF HLD-ZIP. The first token is HLD-ZIP-PREFIX, the second token is OF, and the third token is HLD-ZIP. The tokens must be consecutive.

& (Retain) Command

&any primary command

Function

Executes the specified primary command and keeps it displayed in the command input area for repeated use or modification.

Operands

Operand	Description
&	Must be followed by a primary command.

Usage Notes

The Retain command is useful if the same primary command is to be executed repeatedly or if minor changes to a command are desired.

ALLOCDEF Command

ALLOCDEF | ADEF

Function

Displays the Options - Product Allocations pop-up (see Figure 142 on page 298) used to specify the DASD volumes for the Log, List, Punch and Work files, and to specify space for the Work file.

Operands

None.

Usage Notes

Note:

You can also display the Options - Product Allocations pop-up by selecting Options ▸ Product Allocations.

The ALLOCDEF command can be entered on any Encore screen. See online help for more information about the Options - Product Allocations pop-up.

ANALYZE Command

ANalyze

Function

Displays the File - Analyze Submit pop-up (see Figure 16 on page 39) that can be used to submit a compile/analyze job without ending the current Encore function. A program must be analyzed before it can be used in Encore.

Operands

None.

Usage Notes

Note:

The File - Analyze Submit pop-up can also displays when you select File ▸ Analyze.

The ANALYZE command can be entered on any Encore screen.

See Chapter 12, "Analyze Functions," on page 431 for more information about submitting an analyze job. See online help for more information about the File - Analyze Submit pop-up.

BRANCH Command

Bold operands are available only with Insight

Function

The BRANCH command is used to position the cursor at the specified target on the Source View screen (see Figure 34 on page 73) or the View - Source pop-up (see Figure 35 on page 76). This command can be used to scroll from a statement, such as a PERFORM, to the paragraph being performed. The BACKUP operand can be used to return to the statement from which the BRANCH occurred.

Operands

Operand	Description
Blank	No operand is required if the cursor is positioned on a GO TO or PERFORM statement. The BRANCH command automatically locates the target of the GO TO or PERFORM statement. If the cursor is positioned on any other statement, the BRANCH command locates the next sequential statement to be executed. The PROCEDURE DIVISION label is located if the cursor is positioned outside the PROCEDURE DIVISION.
PERFRANGE name	The BRANCH command locates the specified perform range. If the specified name is not fully qualified and there are multiple perform ranges with the specified name, the cursor is positioned on the first perform range found. A message displays indicating the number of perform ranges found.
PROGRAM name	The BRANCH command locates the specified nested program. For COBOL II Release 3 (minimum supported release level) programs, a data item, label, or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. For example, if the label P120-READ exists in another program and also in the program VIARDEM1, then it can be qualified as P120-READ OF VIARDEM1.

Operand	Description
LABEL name	The BRANCH command locates the specified paragraph or section name. If the specified name is not fully qualified and there are multiple paragraphs or sections with the specified name, the cursor is positioned on the first paragraph or section found. A message displays indicating the number of paragraphs or sections found.
PARAGRAPH name	A PROCEDURE DIVISION paragraph or section name, or the literals PROCEDURE and PROC. The entire paragraph is included.
BACKup	BACKUP is used to position the cursor at the location from which the branch occurred.
PREVious	PREVIOUS is used in conjunction with the current cursor location, to position the cursor at the GO TO, PERFORM, or CALL statement from which a target was reached. If more than one previous location is found, the Branch Previous Options pop-up displays (see Figure 128 on page 275). See online help for more information about the Branch Previous Options pop-up.

Usage Notes

The Branch function can also be initiated by selecting Search ▸ Branch, which displays the Search - Branch Request pop-up (see Figure 127 on page 272). See online help for more information.

The BRANCH command positions the cursor to the COBOL source statement that contains the requested target (perform range name, PROGRAM name, label, or BACKUP). This command can only be used on the Source View screen (see Figure 34 on page 73) or View - Source pop-up (see Figure 35 on page 76).

The BRANCH command is very helpful when tracking branching logic. This command can be used to track branching logic several levels deep into PERFORMed code, then return to each PERFORM statement. The screen position is saved for use with the BACKUP operand based on these conditions:

- A perform range or label name is entered on the command line.
- The cursor is placed on a GO TO or PERFORM statement.

The PROCEDURE DIVISION label displays when the BRANCH command is entered with no operands and the cursor is positioned in a part of the program other than the PROCEDURE DIVISION.

If a PROGRAM EXIT or STOP RUN is encountered, a pause occurs and a message displays indicating that a logical end has been reached.

Note:

Selecting Branch on the Search pull-down also allows you to branch to a specified target.

See online help for more information about the Search pull-down.

Examples

Example 1

Type this command, place the cursor on PERFORM READ-ALL, and press Enter.

```
BRANCH
```

The screen is scrolled and the cursor is positioned on the READ-ALL paragraph.

Example 2

Type this command and press Enter:

```
B P120-READ
```

The screen is scrolled and the cursor is positioned on the P120-READ paragraph.

Example 3

Type this command and press Enter:

```
B PROGRAM-INIT
```

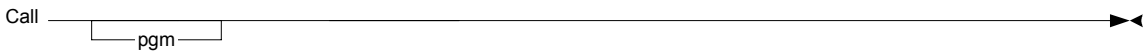
The screen is scrolled and the cursor is positioned on the PROGRAM-INIT paragraph.

Type this command and press Enter:

```
B BACKUP
```

The screen is scrolled and the cursor is positioned at the location from where the B PROGRAM-INIT command was entered.

CALL Command



Note:

This command is valid only if Insight is installed.

Function

The CALL command is used to start a Source View session for another program without ending the current session. The GOBACK or END command can be used to return to the original program.

Operands

Operand	Description
Blank	Displays the AKR Program Selection pop-up for selection of a program. See the online help for more information.
pgm	Specifies a COBOL program or entry point name. If the specified program or entry point is not available in the current AKR, an error message displays.

Usage Notes

The CALL command provides you a means of following logic paths across CALLed programs. CALL remembers the location within the CALLing program and displays the CALLed program. When finished with the CALLed program, use the GOBACK command to return to the location in the CALLing program. See the "GOBACK Command" on page 368 and the "PROGRAM Command" on page 400 for additional information.

Examples

Example 1. In this example the current Source View program is VIARDEMO, this command calls the VIARBRWS program:

```
CALL VIARBRWS
```

Example 2. Returns to the VIARDEMO Source View session without saving marks or equates that you created in VIARBRWS:

```
GOBACK VIARDEMO NOSAVE
```

CANCEL Command

CANcel

Note:

This command is valid only if Insight is installed.

Function

The CANCEL command terminates the current function without saving changes to marks and equates made during the session.

Operands

None.

Usage Notes

CANCEL is the equivalent of an END NOSAVE command. It terminates the function without saving new marks or equates, or changes to existing marks or equates.

CURRENT Command

CURRent

Function

The CURRENT command is used to save the current location in the program source. The LOCATE &CURRENT command can then be used to reposition the screen to the saved location.

Only one CURRENT command can be in effect. A new CURRENT command replaces the prior one.

The CURRENT command is valid only on the Source View screen (see Figure 34 on page 73).

Operands

None.

Usage Notes

The CURRENT and LOCATE &CURRENT commands increase the ability to control the position in the source during script processing.

The CURRENT command can also be used to save your current location in the program source while you browse related data items.

DISPLAY Command

DISplay

Function

The DISPLAY command excludes from the screen all statements in the program that are not contained in the active Logic Segment. If the DISPLAY command is entered on any View other than Source View (see Figure 34 on page 73), the Source View screen displays with the Logic Segment displayed. A Logic Segment must be currently specified before the DISPLAY command is entered. On the Edit screen, if there is no active Logic Segment when DISPLAY is entered, the File - Display Logic Segment Entry pop-up (see Figure 25 on page 57) displays to specify a Logic Segment.

Operands

None.

Usage Notes

Note:

Logic Segments also displays when you select View ▸ Logic segment.

The DISPLAY command refreshes the current member display to contain only the statements defined for the current Logic Segment; all other lines of the program are excluded.

See online help for more information about displaying Logic Segments.

END Command

Function

The END command is used to terminate the current function and redisplay the previous screen or function.

Operands

Operand	Description
blank	If the END command is entered with no operand, the value specified on the Options - Product Parameters pop-up (see Figure 141 on page 296) are used to determine if Equates are to be saved.
Save	Saves Equates created for this program in the AKR. This operand overrides the value set on the Options - Product Parameters pop-up.
Nosave	Equates created for this program are not saved. This operand overrides the value set on the Options - Product Parameters pop-up.

Usage Notes

The END command terminates any Encore function. The previous function or screen displays.

The END command is the default for PF03/15. The command and the desired operand can be entered in the command input area, or the operand can be entered in the command input area and the PF key pressed.

See online help for more information about PF keys and the Options - Product Parameters pop-up.

EQUATE Command

```
EQuate ──name──┬──────────┬──────────────────────────────────────►◄
               └──string──┘
```

Function

The EQUATE command is used to define a name for a character string.

Operands

Operand	Description
name	Names must conform to these standards: • 1-10 alphanumeric characters. • First character must be alphabetic. • A hyphen is the only special character allowed. The name may also be 1-4 DBCS characters.
string	A character string to be substituted by the EQUATE command. A character string can be a long command, pattern, data name, concatenated data name, etc. Literals and blanks can be specified in the character string if desired. Character strings that include blanks must be enclosed with single or double quotes. The string may be a DBCS string. If the string operand is not entered, the equated name is deleted.

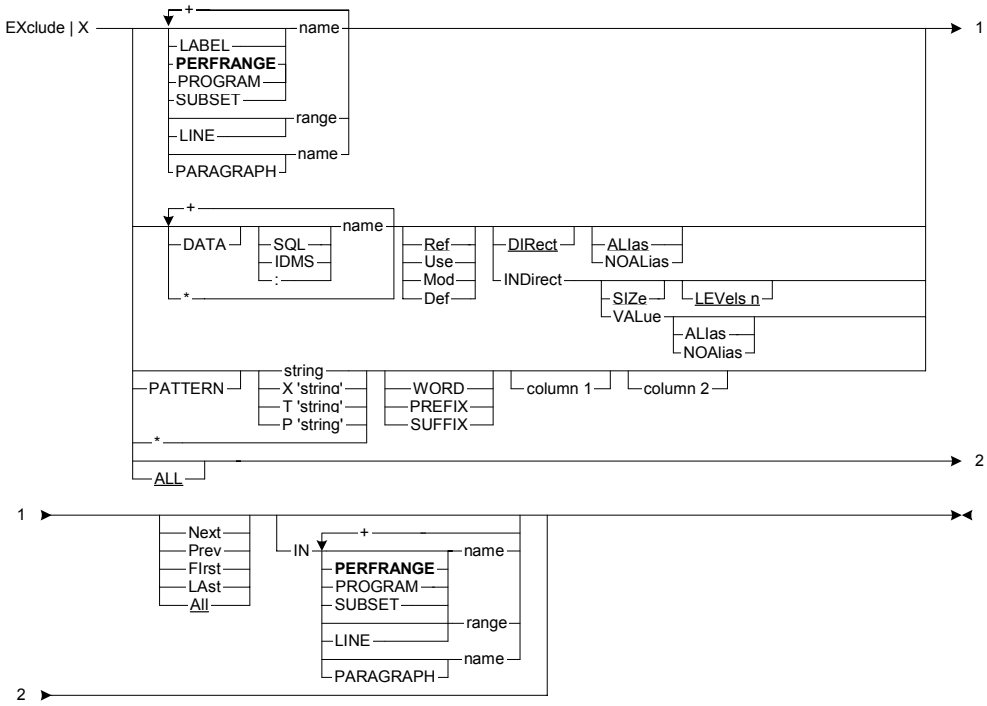
Usage Notes

Note:

Equates can also be defined and deleted on the Options - Scratchpad Equate pop-up (see Figure 149 on page 310), which displays by selecting Options ▸ Equates. In View or Extract only, the substitution string can be changed on the List - Equates pop-up (see Figure 133 on page 285) by typing over it with the new value. Equates can also be defined with the `.label` line command on the Source View screen (see Figure 34 on page 73).

See online help for more information about these screens and pop-ups.

EXCLUDE Command



Bold operands are available only with Insight

Function

The EXCLUDE command performs a FINDXTND command on the specified target, excluding the resulting lines.

Operands

Operand	Description
LABEL name	A PROCEDURE DIVISION paragraph name or section name or the literals PROCEDURE and PROC. All transfers of control to the label name are excluded.
PERFRANGE name	The name specified in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the DECLARATIVES.

Operand	Description
PROGRAM name	The name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program.

SUBSET name	These are predefined COBOL language subsets:
	ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, COPy, DB2/SQL, DDL, DEAD, DEADCode, DEADData, DEbug, DEFinition, DIRective, DIVision, DL/I | DL/1, DML, ENtry, EXIt | PGMExit, FALLthrough, GOto, IDMS, INClude, Input, IO, LABel, MAINline, MATH, Output, PARagraph, PERform, RETurn, SECtion, SORTMerge, STructure, TAG, TESted | UNTested
	Note:
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or NX`
	A tagged lines subset of tags appearing in columns 73 - 80.
	Subsets may be concatenated by placing a plus sign (+) between the subset names. See Chapter 2, "Product Overview," on page 7 for a description of each subset.

LINE range	A single line number or range of lines. Line numbers display in columns 1-6 in Source View.
PARAGRAPH name	A PROCEDURE DIVISION paragraph or section name or the literals PROCEDURE and PROC. The entire paragraph is excluded.

Operand	Description
*	Reuses the target of the previous EXCLUDE command. See "Usage Notes" on page 345 for information about concatenating this operand.
DATA name	A COBOL data name or qualified COBOL data name. DATA name refers to any valid COBOL reference for a data element. These subordinate operands apply to the data name and can be used to further qualify the EXCLUDE command: REF, USE, MOD, DEF, ALIAS, NOALIAS, and DIRECT. The defaults are REF, ALIAS, and DIRECT.
SQL name	Excludes data names that are DB2/SQL variables only.
IDMS name	Excludes data names that are IDMS variables only.
: name	Excludes data names that are COBOL variables only.
Ref	Excludes occurrences of the data name where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand.
Use	Excludes occurrences of the data name where its value is being tested or used.
Mod	Excludes occurrences of the data name where its value is being set or modified.
Def	Excludes definitions of the data name in the DATA DIVISION.
DIRect	Includes occurrences of the data name (and aliases if specified) where it is directly tested, used, set, modified, or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT, however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed.
INDirect	Includes occurrences of any data name indirectly affected by the specified data name (and aliases if specified). This can be very useful when working with data names. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. See "Usage Notes" on page 345 for more information.

Operand	Description
ALIas	Includes these aliases for the data name: • Parent - higher level group item • Child - lower level item • Rename/Redefinition - renamed, redefined, or 88 level items This is the default value for the DATA name operand.
NOAlias	Ignores aliases for the specified data name.
SIZe	Includes occurrences of data names that could be directly or indirectly affected if the size of the specified data name were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. See "Usage Notes" on page 345 for more information.
VALue	Includes occurrences of data names that could be directly or indirectly affected if the specified data name were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand.
LEVels `n`	Includes all data items that are affected within the specified levels. This subordinate operand is not used with the VALUE subordinate operand. All levels is the default for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. See "Usage Notes" on page 345 for more information.
PATTERN	An optional keyword indicating the characters that follow are part of a string.
string	A string of alphanumeric or DBCS characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used.
X'string'	The hexadecimal string option. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes.
T'string'	The text string option. A character string may be entered regardless of upper or lowercase by using the text option. The string must be enclosed in single or double quotes.

Operand	Description
P'string'	The picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. These nine special characters are defined by Encore for use in picture strings and can be combined with other characters: P’=’ Any character P’¬’ Any nonblank character P’.’ Any nondisplay character P’#’ Any numeric character P’-’ Any non-numeric character P’@’ Any alphabetic character (upper or lowercase) P’<’ Any lowercase alphabetic character P’>’ Any uppercase alphabetic character P’$’ Any special character (not alphabetic or numeric)
WORD	The specified pattern string preceded and followed by any non-alphanumeric character (except a hyphen).
PREFIX	A word that begins with the specified pattern string.
SUFFIX	A word that ends with the specified pattern string.
column1	The column number where the search is to begin.
column2	The column number where the search is to end.
blank	Entering EXCLUDE with no operands excludes all lines from the display.
ALL	Excludes all lines from the display. The ALL operand cannot be entered with any other operands. ALL is the default if EXCLUDE is entered with no operands.
Next	Searches the source code forward from the cursor position to the next occurrence of the requested target.
Prev	Searches the source code backward from the cursor position to the previous occurrence of the requested target.
FIrst	Searches from the top of the source code to the first occurrence of the requested target.
LAst	Searches backward from the bottom of the source code to the last occurrence of the requested target.

Operand	Description
All	Searches for all occurrences of the requested target and displays the number found in the short/long message field. This is the default value for the FINDXTND command.
IN	The IN target operand is optional and restricts the EXCLUDE command to the specified target type.

Usage Notes

Note:

Lines can also be excluded with the X (Exclude) and XX (Exclude Block) line commands; or by selecting View ▸ Exclude, which displays the View - Exclude Request pop-up (see Figure 55 on page 111); or by using one of the Search pop-ups available on the Search pull-down (see Figure 117 on page 251).

The EXCLUDE command can be used to remove specific lines from the screen resulting from a previous command. Excluded lines are represented by a line of dashes and text stating `n` LINE(S) NOT DISPLAYED. On the Structure View screen (see Figure 41 on page 87), a short message lists the number of lines excluded.

To concatenate targets, place a plus sign (+) between the target names. For example, IO+MYSET. These rules apply to concatenation:

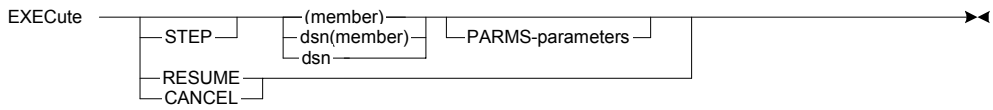
- A concatenated data name can be used wherever a data name is valid. Data name subordinate operands (REF, ALIAS, etc.) pertain to all data names in a concatenated series.
- A concatenated set name can be used wherever a set name is valid. These sets can be concatenated:

  ```
  LABEL    PERFRANGE    PROGRAM
  LINE     SUBSET       PARAGRAPH
  ```

- The * operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

If a data name or set name is ambiguous or used multiple times, it can be qualified (e.g., ZIP-CODE OF MASTER-IN). See online help for more information about the View - Exclude Request pop-up (see Figure 55 on page 111), the Search pop-ups, and Search pull-down (see Figure 117 on page 251).

EXECUTE Command



Function

The EXECUTE command is used to read and execute the commands in a Encore script file. Script files can contain EXECUTE commands that execute lower level (nested) script files. The STEP operand allows stepping through each script file command.

Operands

Operand	Description
STEP	The STEP operand allows stepping through each command in the script file. Each command in the script file displays in the command input area. The displayed command can be changed or erased from the command input area. Press Enter to execute the displayed command. Processing of the script file continues in this manner until all commands have displayed or executed.
RESUME	EXECUTE RESUME is entered after a script file has been executed with the STEP operand specified. The RESUME operand is used to execute the remaining script file commands without stepping through each.
CANCEL	EXECUTE CANCEL can be entered after a script file has been executed with the STEP operand specified. The CANCEL operand is used to cancel the remaining script file command.
(*member*)	A script member in a default script file defined during product installation. The default script files, or their concatenation sequence, can be modified for your profile using the Options - Script File Allocations pop-up (see Figure 145 on page 304). The parentheses are required.
dsn	The dataset name of a sequential file that contains the script commands to be executed. This dataset must be in card image format (LRECL=80). The specified dataset name must be entered following TSO usage conventions in this format: `'sequential.dataset.name'` If the quotes are omitted, your TSO prefix or userID is the high level qualifier.

Operand	Description
dsn(member)	The dataset and member name of a partitioned file that contains the script commands to be executed. This dataset must be in card image format (LRECL=80). The specified dataset member must be entered following TSO usage conventions in this format: `'pds.dsn(member)'` If the quotes are omitted, your TSO prefix or userID is the high level qualifier.
PARMS	Parameters to be passed by the script being executed.
parameters	Parameter values may contain any non-quoted literal without embedded blanks.

Commands within a script file may contain substitution variables numbered &1 through &9. Parameter values are passed at execution time to the substitution variables using the PARMS operand of the EXECUTE command. Enter the parameters values in the numerical order 1 through 9. If more than 9 parameter values are entered, the first nine are used and the remainder are ignored. If substitution variables are included in the script and parameters are not included in the EXECUTE command, the substitution variable values are passed as null.

Substitution variables can be used anywhere in the executable script commands, but are not substituted if they occur in comment lines.

Usage Notes

The EXECUTE command reads the specified dataset and executes the script commands contained in the file. Each script file can contain EXECUTE commands that execute lower level (nested) script files. Script files that create a loop are recognized and an error message displays.

The STEP operand provides a means of stepping through each script command.

Scripts must be sequential datasets or members of partitioned datasets. In both cases, scripts must be in card image format (LRECL=80).

Scripts can be used to perform these repetitive tasks:

- Initiate a test session
- Set default values
- Set up a test session
- Execute a test
- Re-execute a test
- Execute a predefined command sequence

Comments can be included in script files by entering an asterisk (*) in column one followed by the comment text.

Note:

Script files can also be executed by selecting File ▸ Execute, which displays the File - Execute Script pop-up (see Figure 27 on page 59). See online help for more information about the File - Execute Script pop-up.

Examples

Example 1

In this example, the EXECUTE command causes the first command in the script file to display in the command input area.

```
EXECUTE STEP 'USERID.ENCxxxx.VIASCRIP'
```

You can modify the displayed script command if desired and then press Enter.

The command is executed and the next command in the script file displays in the command input area. Any command can be changed or bypassed (erased) when it displays in the command input area.

Example 2

In this EXECUTE command, all script commands in the member SCRIPT01 in the default script datasets are sequentially executed. These commands can consist of BREAK, BRANCH, LPRINT, or any valid Encore primary command.

```
EXECUTE (SCRIPT01)
```

Example 3

In this example, the program name MYPROG and the data name HIRE-DATE are passed as parameters to the script in member SCRIPT02 in the default script files.

```
EXECUTE (SCRIPT02) PARMS MYPROG HIRE-DATE
```

If the script contains these commands, the parameters are substituted in the commands in place of the substitution variables to allow flexibility in using the script.

```
QUALIFY &1

FX &2
```

These are the actual commands resulting from the parameter substitution:

```
QUALIFY MYPROG

FX HIRE-DATE
```

See the "PROCESS Command" on page 395 for additional uses of EXECUTE parameters.

EXIT Command

EXIT

Function

The EXIT command terminates the current function and quits the Encore session. This command is only available on the first Encore Primary screen.

Operands

None.

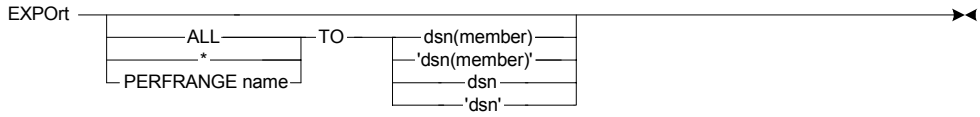
Usage Notes

Note:

Encore can also be exited by selecting File ▸ Exit. See online help for more information.

The EXIT command immediately ends the Encore session.

EXPORT Command



Function

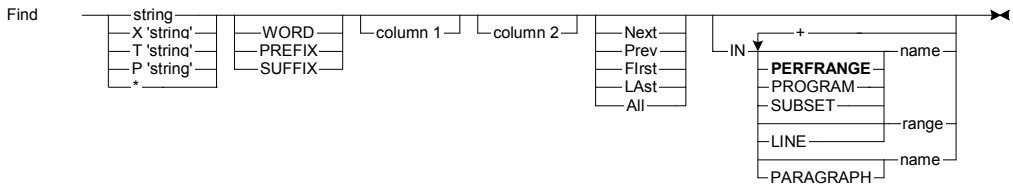
The EXPORT command allows you to write the relevant data from the Extract - Perform Name List pop-up (see Figure 58 on page 128), in comma delimited format, to a specified, or default, dataset name. A file can be created using the EXPORT command to write information to other mainframe files for downloading to spreadsheets or PC database managers.

Operands

Operand	Description
ALL	Allows all displayed perform range names and associated information to be selected. The specified command must be entered in this format: `EXPORT ALL`
* or ?	Allows you to enter a perform range name pattern with embedded wild card characters (a question mark (?) for one character or an asterisk (*) for zero or more characters) to list all perform ranges with that name pattern. For example, enter R* to list all perform ranges that begin with R. The specified command must be entered in this format: `EXPORT * (or) EXPORT A?B (or) EXPORT R*`
PERFRANGE *name*	A name that identifies the perform range. The specified command must be entered in this format: `EXPORT perfrange` Where `PERFRANGE` must conform to these standards: • Complete perform range name • First character must be alphabetic • A hyphen is the only special character allowed The name may also be 1 through 4 DBCS characters.

Operand	Description
TO	Designates that you want to export the selected information to a specified output file. The specified dataset member must be entered following TSO usage conventions in this format: `EXPORT operand TO specified dataset`
dsn (member)	The default dataset and member name of a partitioned file that contains the exported perform range information. Your TSO userID is added as the prefix to DSN. The specified dataset member must be entered following TSO usage conventions in this format: *userid.dsn(member)*
'*dsn (member)*'	A specific dataset and member name of a partitioned file that contains the exported perform range information. The specified dataset member must be entered following TSO usage conventions in this format: '*dsn(member)*'
dsn	The default dataset name of a sequential file that contains the exported perform range information. If the quotes are omitted, your TSO prefix or userID is the high level qualifier. The specified dataset name must be entered following TSO usage conventions in this format: *userid.sequential.dataset.name*
'*dsn*'	A specific dataset name of a sequential file that contains the exported perform range information. The specified dataset name must be entered following TSO usage conventions in this format: '*sequential.dataset.name*'

FIND Command



Bold operands are available only with Insight

Function

The FIND command searches for and highlights one or all occurrences of the specified character string. The syntax is similar to the ISPF/PDF FIND command. Double Byte Character Set (DBCS) identifiers or patterns can be specified with the FIND command to locate DBCS strings.

The FIND command results depend on the view method currently in use.

Operands

Operand	Description
string	A string of alphanumeric or DBCS characters. If the string contains blanks, it must be enclosed in single or double quotes. The string cannot contain mixed Single Byte Character Set (SBCS) and DBCS characters. The string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the string is to be used.
X'string'	The hexadecimal string option. Unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes.
T'string'	The text string option. A character string may be entered regardless of upper or lowercase by using the text option. The string must be enclosed in single or double quotes.

Operand	Description
P'*string*'	The picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. These nine special characters are defined by Encore for use in picture strings and can be combined with other characters: P’=’ Any character P’¬’ Any nonblank character P’.’ Any nondisplay character P’#’ Any numeric character P’-’ Any non-numeric character P’@’ Any alphabetic character (upper or lowercase) P’<’ Any lowercase alphabetic character P’>’ Any uppercase alphabetic character P’$’ Any special character (not alphabetic or numeric)
*	Reuses the target of the previous FIND command.
WORD	The specified string preceded and followed by any non-alphanumeric character (except a hyphen).
PREFIX	A word that begins with the specified string.
SUFFIX	A word that ends with the specified string.
column1	The column number where the search is to begin.
column2	The column number where the search is to end.
Next	Searches the source code forward from the current cursor position to the next occurrence of the requested target. This is the default value for the FIND command.
Prev	Searches the source code backward from the current cursor position to the previous occurrence of the requested target.
FIrst	Searches from the top of the source code to the first occurrence of the requested target.
LAst	Searches backward from the bottom of the source code to the last occurrence of the requested target.
All	Searches for all occurrences of the requested target in the source code and displays the number found in the short and long message fields.

Operand	Description
IN	The IN target operand is optional and restricts the FIND command to the specified target type.
PERFRANGE *name*	The name specified in a PERFORM statement including all statements that are executed as a result of a PERFORM statement or the name of any section contained in the DECLARATIVES.
PROGRAM *name*	The name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program.

SUBSET *name*	These are the predefined COBOL language subsets:
	ASsignment, DEbug, IO
	CAll, DEFinition, LABel
	CIcs, DIRective, MAINline
	COBOLII, DIVision, MATH
	COBOL/370, DL/I | DL/1, Output
	COMment, DML, PARagraph
	CONditional, ENtry, PERform
	COPy, EXIt | PGMExit, RETurn
	DB2/SQL, FALLthrough, SECtion
	DDL, GOto, SORTMerge
	DEAD, IDMS, STructure
	DEADCode, INClude, TAG
	DEADData, Input, TESted | UNTested
	Note:
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or NX`
	A tagged lines subset of tags appearing in columns 73 - 80.
	See Chapter 2, "Product Overview," on page 7 for a description of each subset.

Operand	Description
LINE *range*	A single line number or range of lines. Line numbers display in columns 1-6 of Source View.
PARAGRAPH *name*	A PROCEDURE DIVISION paragraph or section name, or the literals PROCEDURE and PROC. The entire paragraph or section is included.

Usage Notes

Note:

A search can also be conducted by selecting a Search pop-up on the Search pull-down (see Figure 117 on page 251). See online help for more information.

The FIND command results depend on the view method currently in use. This table shows the results for each view method. See online help for more information about view methods.

View Method	FIND Command Results
Source View	Highlights and tags all lines found, and positions the cursor at the first target in the specified direction. The short message indicates the number of occurrences found.
Tree View	Highlights and tags all lines found, and positions the cursor at the first perform range in a top/down manner which contains the target. The short message indicates the number of occurrences found.
Structure View	Positions the cursor at the first box containing an occurrence of the target, and highlights all boxes containing occurrences of the target. The short message indicates the number of occurrences found.

In Source View only, the search begins from the current line when the NEXT or PREV operands are specified. All lines are searched regardless of the current line or direction when the ALL operand is specified. Any excluded lines containing FIND targets are included and displayed. The IN operand restricts the FIND command to only the specified targets.

Examples

Example 1

This command highlights all occurrences of XYZ.

```
FIND 'XYZ' ALL
```

Example 2

This FIND command searches for the DATA NAME string without regard to case. The DATA NAME string can be uppercase, lowercase, or mixed case.

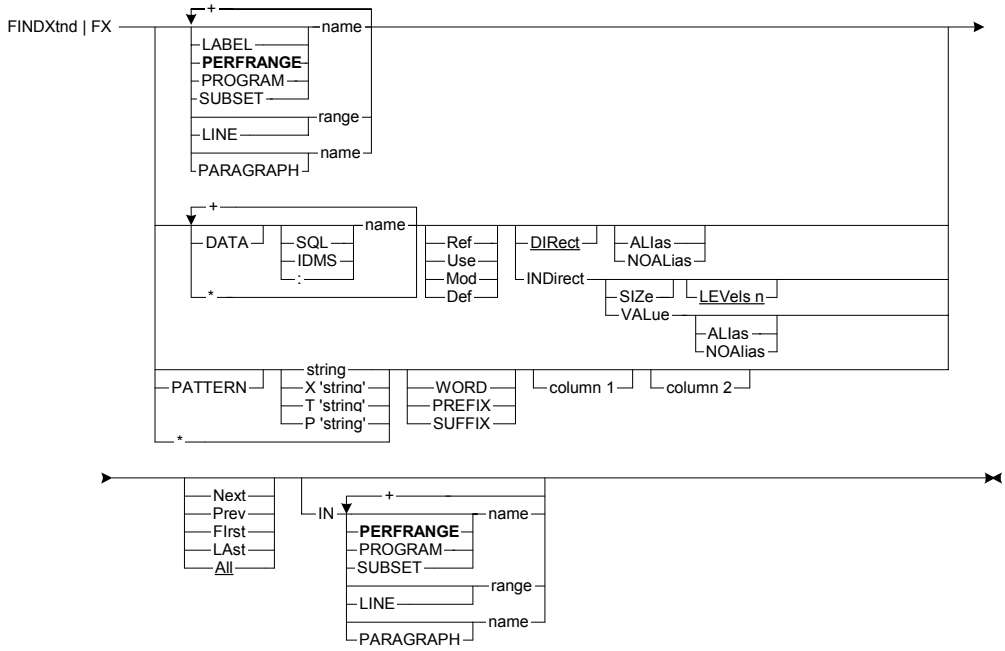
```
FIND T'DATA NAME'
```

Example 3

This FIND command searches for two non-blank characters separated by a blank.

```
FIND P'¬ ¬'
```

FINDXTND Command



Bold operands are available only with Insight

Function

The FINDXTND command performs a COBOL intelligent search of the source code for one or all occurrences of the specified target and highlights the results.

The FINDXTND command results depend on the view method currently in use.

Operands

Operand	Description
LABEL name	A PROCEDURE DIVISION paragraph or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included.
PERFRANGE name	The name specified in a PERFORM statement including all statements that are executed as a result of a PERFORM statement, or the name of any section contained in the DECLARATIVES.

Operand	Description
PROGRAM name	The name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program.

SUBSET name	These are predefined COBOL language subsets:
	ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, COPy, DB2/SQL, DDL, DEAD, DEADCode, DEADData, DEbug, DEFinition, DIRective, DIVision, DL/I | DL/1, DML, ENtry, EXIt | PGMExit, FALLthrough, GOto, IDMS, INClude, Input, IO, LABel, MAINline, MATH, Output, PARagraph, PERform, RETurn, SECtion, SORTMerge, STructure, TAG, TESted | UNTested
	Note:
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or NX`
	A tagged lines subset of tags appearing in columns 73 - 80.
	See Chapter 2, "Product Overview," on page 7 for a description of each subset.

LINE range	A single line number or range of lines. Line numbers display in columns 1 through 6 of Source View.
PARAGRAPH name	A PROCEDURE DIVISION paragraph or section name, or the literals PROCEDURE and PROC. The entire paragraph is included.
*	Reuses the target of the previous find or exclude action or command.

Operand	Description
DATA name	A COBOL data name or qualified COBOL data name. Data name refers to any valid COBOL reference for a data element. These subordinate operands apply to the data name and can be used to further qualify the FINDXTND command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT, and INDIRECT. The defaults are REF, ALIAS and DIRECT.
SQL name	Includes data names that are DB2/SQL variables only.
IDMS name	Includes data names that are IDMS variables only.
: name	Includes data names that are COBOL variables only.
Ref	Includes occurrences of the data name where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand.
Use	Includes occurrences of the data name where its value is being tested or used.
Mod	Includes occurrences of the data name where its value is being set or modified.
Def	Includes definitions of the data name in the DATA DIVISION.
DIRect	Includes occurrences of the data name (and aliases if specified) where it is directly tested, used, set, modified or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT, however, if the SIZE, VALUE, or LEVELS operand is specified, INDIRECT is assumed.
INDirect	Includes occurrences of any data name indirectly affected by the specified data name (and aliases if specified). This can be very useful when working with data names. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. See "Usage Notes" on page 363 for more information.

Operand	Description
ALIas	Includes aliases for the data name. These aliases include: • Parent - higher level group item • Child - lower level item • Rename/Redefinition - renamed, redefined, or 88 level items This is the default value for the DATA name operand.
NOAlias	Ignores aliases for the specified data name.
SIZe	Includes occurrences of data names that could be directly or indirectly affected if the size of the specified data name were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. See "Usage Notes" on page 363 for more information.
VALue	Includes occurrences of data names that could be directly or indirectly affected if the specified data name were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand.
LEVels `n`	Includes all data items that are affected within the specified levels. This subordinate operand is not used with the VALUE subordinate operand. All levels is the default for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. See Usage Notes for more information.
PATTERN	An optional keyword indicating the characters that follow are part of a string.
string	A string of alphanumeric or DBCS characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used.
X'string'	The hexadecimal string option. Unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes.
T'string'	The text string option. A character string may be entered regardless of upper or lowercase by using the text option. The string must be enclosed in single or double quotes.

Operand	Description
P'string'	The picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. These nine special characters are defined by Encore for use in picture strings and can be combined with other characters: P’=’ Any character P’¬’ Any nonblank character P’.’ Any nondisplay character P’#’ Any numeric character P’-’ Any non-numeric character P’@’ Any alphabetic character (upper or lowercase) P’<’ Any lowercase alphabetic character P’>’ Any uppercase alphabetic character P’$’ Any special character (not alphabetic or numeric)
WORD	The specified pattern string preceded and followed by any non-alphanumeric character (except a hyphen).
PREFIX	A word that begins with the specified pattern string.
SUFFIX	A word that ends with the specified pattern string.
column1	The column number where the search is to begin.
column2	The column number where the search is to end.
Next	Searches the source code forward from the cursor position to the next occurrence of the requested target.
Prev	Searches the source code backward from the cursor position to the previous occurrence of the requested target.
FIrst	Searches from the top of the source code to the first occurrence of the requested target.
LAst	Searches backward from the bottom of the source code to the last occurrence of the requested target.

Operand	Description
All	Searches for all occurrences of the requested target and displays the number found in the short/long message field. This is the default value for the FINDXTND command.
IN	The IN target operand is optional and restricts the FINDXTND command to the specified target type.

Usage Notes

Note:

A search can also be conducted by selecting a Search pop-up on the Search pull-down (see Figure 117 on page 251). See online help for more information.

The FINDXTND command results depend on the view method currently in use. This table shows the results for each view method. See online help for more information about view methods.

View Method	FINDXTND Command Results
Source View	Positions the cursor at the first target in the specified direction. All targets are highlighted and tagged, and the long or short message indicates the number of occurrences found.
Tree View	Positions the cursor at the first perform range in a top/down manner which contains the target. All Procedure Division targets are highlighted and tagged, and the long or short message indicates the number of occurrences found.
Structure View	Positions the cursor at the first box containing an occurrence of the target, and highlights all boxes containing occurrences of the target. The short message indicates the number of occurrences found.

To concatenate targets, place a plus sign (+) between the target names. For example, IO+MYSET. These rules apply to concatenation:

- A concatenated data name can be used wherever a data name is valid. Data name subordinate operands (REF, ALIAS, etc.) pertain to all data names in a concatenated series.
- A concatenated set name can be used wherever a set name is valid. These sets can be concatenated:

  ```
  LABEL    PERFRANGE    PROGRAM      PATTERN

  LINE     SUBSET       PARAGRAPH
  ```

- The * operand can be concatenated once with any number of other operands (e.g., * + IO + MYSET); however, the * operand cannot be concatenated to itself (* + * is invalid). The * operand may appear in any order in the concatenated list.

If a data name or set name is ambiguous or used multiple times, it can be qualified (e.g., ZIP-CODE OF MASTER-IN).

In Source View, highlighting is used to indicate the occurrences found. Lines that are already highlighted are reset, so that only the results of the current command are highlighted. If lines containing targets are excluded, they redisplay on the screen. Tags are placed on the source code lines indicating the type of target found. These tags are placed in columns 73-80. These tags include:

Tag	Corresponding Name(s)
LABEL name	LABELREF
PERFRANGE name	PERF RNG
PROGRAM name	• PROGRAM • PGM tag

Tag	Corresponding Name(s)		
SUBSET name - COBOL	ASsignment	DEbug	IO
	CAll	DEFinition	LABel
	CIcs	DIRective	MAINline
	COBOLII	DIVision	MATH
	COBOL/370	DL/I | DL/1	Output
	COMment	DML	PARagraph
	CONditional	ENtry	PERform
	COPy	EXIt | PGMExit	RETurn
	DB2/SQL	FALLthrough	SECtion
	DDL	GOto	SORTMerge
	DEAD	IDMS	STructure
	DEADCode	INClude	TAG
	DEADData	Input	TESted | UNTested
	Note: The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.		
SUBSET name - SCREEN	• HIGH • NONHIGH • EXCLUDE • NONX		
LINE range	LINE RNG		
PARAGRAPH name	PARAGRAP		
DATA name	• DATA REF • DATA USE • DATA MOD • DATA DEF		
PATTERN string	PATTERN		

The NEXT and PREV operands start the search from the current position and locate the closest occurrence. If ALL is specified, all lines are searched regardless of the current line or direction. A message displays indicating the number of targets found.

When an unqualified data name is entered and more than one occurrence of the specified data name exists, all occurrences are found. This provides a means of seeing all like data names but under different group level names, then selecting the appropriate data name. A message displays if an invalid qualification is entered.

When a data name is entered with the REF operand, the screen is positioned to show the first occurrence of the data name in the PROCEDURE DIVISION. DEF information in the DATA DIVISION is also highlighted.

When the REF operand is specified or used by default, all appropriate references are highlighted. If the reference is a DEF, USE, or MOD, the line is tagged accordingly. If the line contains a USE and a MOD, it is tagged as a REF.

DBCS Strings

The FINDXTND command supports DBCS strings. See the related discussion on the "FIND Command" on page 353 for detailed information.

FX INDIRECT

A data name can impact other data names in source statements. These data names can be subsequently propagated indirectly through other statements, and referenced at each level. For example, consider this code:

```
MOVE A TO B
MOVE B TO C
MOVE C TO D
```

When the data name of interest is A, the first statement contains a direct reference to A. The direct reference to a data name is level 1. The last two statements contain indirect references to A, based on the propagation of the data name A by moving its value to B, then to C, then to D. The second statement is a level 2 reference to data name A, and the last statement is a level 3 reference to data name A.

All levels of the indirect impact to a data name displays, or one level at a time can be viewed by using the LEVELS operand. For example, to locate all data names that could be indirectly affected by a change in the size of the specified data name, use the command FX A SIZE. Because the LEVELS operand is not specified, the default of all levels is used. This results in a complete list of all data names that must be reviewed (for a size change of the specified data name). If the LEVELS operand is specified (e.g., FX A SIZE LEVELS 2), only the results of that many levels are highlighted. The ripple effect can be more clearly identified by starting with LEVELS 1, then increasing the number of levels to be searched.

When using INDIRECT SIZE, any occurrence of an affected data name in the USING clause of a CALL statement is tagged as a MOD. The called subprogram must be examined for other data names that are affected as a result of the CALL. IN target can be used to restrict the search to the specified IN target.

When the REF operand is specified or used by default, all appropriate references are highlighted. If the reference is a DEFinition, USE, or MODification, the line is tagged accordingly. If the line contains a USE and a MODification, it is tagged as a REFerence.

Examples

Example 1

Enter this FINDXTND command to search backward for the previous occurrence of a use of DATE-CODE or its aliases:

```
FINDXTND DATE-CODE USE PREV
```

Example 2

Enter this FX command to highlight the definition of DATE-CODE in the DATA DIVISION:

```
FX DATE-CODE DEF NOALIAS ALL
```

Example 3

Enter this FX command to highlight all occurrences of the SWITCH field used in conditional statements:

```
FX SWITCH IN COND ALL
```

GOBACK Command

Note:

This command is valid only if Insight is installed.

Function

The GOBACK command is used to reverse a CALL or VIEW command and return to a Source View session.

Operands

Operand	Description
Blank	Specifies that the program where the last CALL or VIEW was executed is returned to the Source View. Marks and equates are saved or not saved, depending on the values specified on the Options - Product Parameters pop-up.
pgm	Specifies a program or entry point in the AKR. A Source View session must currently be active for this program.
Save	Saves marks and equates created for this program in the AKR. This operand overrides the values on the Options - Product Parameters pop-up.
Nosave	Specifies that marks and equates created for this program are not saved. This operand overrides the values on the Options - Product Parameters pop-up.

Usage Notes

The GOBACK command ends the current Source View session and returns to the Source View session where a CALL or VIEW command was entered. Specifying a program name returns to a particular program in the calling chain.

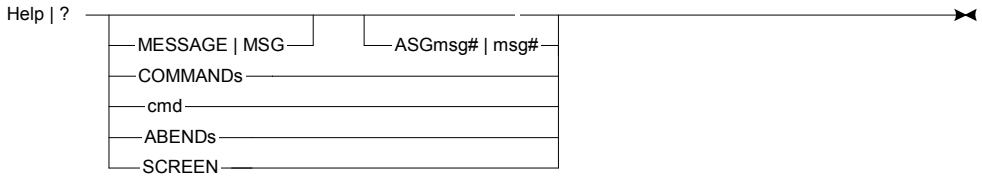
Example

The current program is VIARDEMO. To display the VIARBRWS program on the Source View screen, type:

```
CALL VIARBRWS
```

After the above command finishes executing, type `GOBACK` to return to the VIARDEMO Source View session.

HELP Command



Function

The HELP command is used to display information about Encore screens, pop-ups, commands, messages, or abend codes. See online help for more information about the Encore Help facility.

Operands

Operand	Description
blank	If no message displays on the screen, entering HELP with no operands displays the Help Tutorial for the current screen or pop-up, which describes all fields on the screen or pop-up and any special processing considerations. If a message displays, entering HELP with no operands displays the Help Explanation and Action Panel for the current message.
MESSAGE | MSG	Displays the Help Explanation and Action Panel for the message currently displayed, unless a ASG message number is also specified.
ASGmsg# | msg#	Displays the Help Explanation and Action Panel for the specified ASG message number. This number consists of 1-4 digits. It is not necessary to enter leading zeros.

Operand	Description
COMMANDs	Displays a list of all Encore primary commands. Then, information for a particular command displays when you select the appropriate number.
cmd	A Encore primary command. When help is requested for a command, a long message displays giving a brief description of that command. Requesting help again then displays more detailed information about the command.
ABENDs	Displays the Abends screen that lists the types of abend codes.
SCREEN	Displays the Help Tutorial for the current screen or pop-up, which describes all fields on the screen or pop-up and any special processing considerations.

Usage Notes

Help can also be viewed using these methods:

- To see help for a command, enter a command or keyword, then press PF1/13.
- To see screen information, press PF1/13 when the screen displays.
- To see an explanation of a message and actions (if any), enter the message number, then press PF1/13.
- Select Help from the action bar.

Example

This command displays the Help Explanation and Action Panel for message 1229. The explanation for the message describes why the message displays, and the action portion of the screen describes any required or suggested user actions.

```
HELP ASG1229
```

See online help for more information about the Help pull-down and its actions, the Help facility, and the Help Tutorial.

JUMP Command

JUMP

Function

The JUMP command is used on the Tree View screen to display the View - Source pop-up (see Figure 35 on page 76), with the cursor positioned at the Tree View cursor location.

Operands

None.

Usage Notes

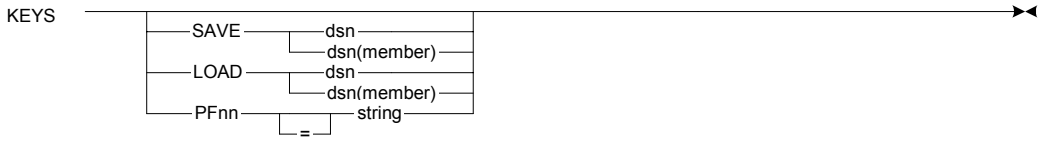
The JUMP command is used in conjunction with the cursor position. Enter the JUMP command in the command input area, and position the cursor to the line containing the target of the jump, (i.e., where the View - Source pop-up should be positioned). Then press Enter.

The JUMP command displays the View - Source pop-up at the Tree View screen's (see Figure 38 on page 80) cursor position.

Use the END command to return to the Tree View screen.

See online help for more information about the Tree View screen and View - Source pop-up.

KEYS Command



Function

This screen is used to display or modify the current Encore PF key assignments.

The KEYS command may also save, restore, or modify the PF key assignments without displaying the Options - PF Key Definition (01-12) pop-up (see Figure 146 on page 305).

Operands

Operand	Description
SAVE	This operand saves the current PF Key definitions in a sequential or partitioned dataset. If you do not specify a dataset name, this is the default name: `<TSO UserID>.ENCnnnnn.VIAPFKEY` Where *nnnnn* is a generated unique number. See the description of dsn and dsn(member) for additional dataset naming rules.
LOAD	Restores the PF Key definitions from the specified sequential or partitioned dataset. A dataset name must be specified for the LOAD operand.

dsn	A sequential dataset name. If quotes are included, the dataset name is used as is. If there are no quotes, your TSO prefix or userID is appended to the front of the dataset name and VIAPFKEY is appended to the end of the dataset name. For example, if you enter this command:
	`SAVE TEST1`
	this is the generated dataset name:
	`<TSO UserID>.TEST1.VIAPFKEY`

Operand	Description
dsn(member)	A partitioned dataset and member name. If quotes are included, the dataset name and member name are used as is. If there are no quotes, your TSO prefix or userID is appended to the front of the dataset name. If the PDS member already exists, it is overwritten.
PF`nn`	This operand provides a command line update to the PF Key definition. For example, this command: `KEYS PF21 RESET EXCLUDED` assigns the text RESET EXCLUDED to PF Key 21. • If the text to be assigned begins with an equal (=) symbol, you must enter two equal symbols separated by a space. • If a null string is entered for the PF`nn` operand, the PF Key assignment reverts to the default value for the current ESW product. The script facility verifies that the PF Key number is valid.

Usage Notes

The KEYS command can be entered without operands on any Encore screen to display or modify the current PF key assignments. Values assigned to the Encore PF keys have no effect on other ISPF applications.

The SAVE and LOAD operands can be used to save or restore PF Key settings. This allows the KEYS command to be used in scripts to set PF Key assignments as desired. This also provides a mechanism for multiple users to share a set of PF Key definitions.

The PF`nn` operand allows the KEYS command to be used in a script file to modify PF Key settings during script file execution.

Note:

The Options - PF Key Definition pop-up (see Figure 146 on page 305) also display when you select Options ▸ PF Keys.

See online help for more information about PF key assignments, the Options - PF Key Definition pop-up, and the Options pull-down.

LEVELS Command

Function

The LEVELS command redisplays the current screen to the specified level. This command is valid on the Structure View screen (see Figure 41 on page 87), View - Perform Structure pop-up (see Figure 42 on page 90), Tree View screen (see Figure 38 on page 80), and View - Tree pop-up (see Figure 39 on page 84).

Operands

Operand	Description
number	Redisplays the current screen to the level depth specified.
MAX	Redisplays the current screen to the maximum number of levels in the program.

Usage Notes

Note:

Levels can also be changed by selecting View ▸ Levels, which displays the View - Levels Request pop-up (see Figure 53 on page 108).

The LEVELS command redisplays the current screen to the level specified. If there are levels less than the number specified, they are expanded to the requested level. If there are levels greater than the number specified, they are removed. The LEVELS command works globally throughout the displayed program.

See online help for more information about the View screens and pop-ups.

LIST Command

```
LIst ──┬──────────────────────────────┬──────────────────────────────►◄
       ├──Calls──────────────────────┤
       ├──Equates────────────────────┤
       ├──Performs───────────────────┤
       ├──PRograms | PGms────────────┤
       └──Subsets────────────────────┘
```

Function

The LIST command is used to display a specified pop-up. Pop-ups are available for each operand shown in the command syntax.

Operands

Operand	Description
blank	Displays the List pop-up that is used to select any List pop-up for display. See "List Pop-up" on page 376 for more information.
Calls	Displays the List - CALL Statements pop-up (see Figure 132 on page 284) that shows all COBOL statements containing CALLs to other programs in the active program. Static and internal calls are noted.
Equates	Displays the List - Equates pop-up (see Figure 133 on page 285) that shows all equates for the active program.
Performs	Displays the List - Perform Range Names pop-up (see Figure 134 on page 286) that shows all COBOL Perform Ranges in the active program and lists the qualifications.
PRograms | PGms	Displays the List - Program/Subprogram Names pop-up (see Figure 135 on page 287) that shows the nested program hierarchy structure of the program.
Subsets	Displays the List - COBOL Subset Names pop-u (see Figure 136 on page 288)p that lists the subsets along with a brief description of each.

Usage Notes

These pop-ups are also available by selecting an action on the List pull-down (see Figure 131 on page 282). See online help for more information about these pop-ups and the List pull-down.

List Pop-up

The List pop-up, shown in Figure 156, is used to select the specified pop-up that displays pertinent information about the program being viewed.

Figure 156 • List Pop-up

```
  File  View  Extract  Generate  Search  List  Options  Help
-                                          ------------------------------------
                 List                      re                 Program: VIARBRWS
C
  Select the LIST command category.
  Then press Enter.                        ***********************************
                                           ***********************************
  Options                                                *****
  ==> 1.  CALLed subprograms                            *****
      2.  EQUATEs                          *********   *****     *******
      3.  PERFORM ranges                   *********  *****     ********
      4.  PROGRAMs                           ******  *****          ***
      5.  COBOL Subsets                     ******  *****          ***
      6.  Logic Segments                   ******  ******************
                                           ****    ******************

        Copyright Allen Systems Group, Inc., an unpublished work.
 A proprietary product of ASG, Inc.  Use restricted to authorized licensees.
         Visit the ASG Support Web Site at www.asg.com
```

To display the List pop-up, follow this step:

- Enter the LIST command with no operands.

Any item on this pop-up can be selected by entering these corresponding numbers in the Options entry field. The pop-up for the selected option then displays.

Field	Description
CALLed subprogram	Displays the List - CALL Statements pop-up.
EQUATEs	Displays the List - Equates pop-up.
PERFORM ranges	Displays the List - Perform Range Names pop-up.
PROGRAMs	Displays the List - Program/Subprogram Names pop-up.
COBOL Subsets	Displays the List - COBOL Subset Names pop-up.
Logic Segments	Displays the List - Segments pop-up.

LOCATE Command

```
LOCATE ──┬─ string ─┬──────────────────────────────────────────►◄
         ├─ .label ─┤
         └─ &CURRent ─┘
```

Function

The LOCATE command is used to scroll to the selectable entry that most closely matches the specified string. LOCATE is valid on most List type screens.

Operands

Operand	Description
string	An alphanumeric or DBCS character string to locate a particular item on a list or directory type screen.
.label	In Source View only, locates a line label entered in the line command area (columns 1-6).
&CURRent	Positions the Source View screen (see Figure 34 on page 73) to the last location marked using the CURRENT command. If no CURRENT command has preceded the LOCATE &CURRENT command, the message `&CURRENT NOT AVAIL` displays and the screen is not repositioned. The &CURRENT operand is only valid on the Source View screen.

Usage Notes

A character string can be specified when an exact program name or list item is not known. The LOCATE command displays the item that most closely matches the specified character string. Matching is done alphabetically.

See online help for more information about the List type screens associated with the LOCATE command.

Examples

Example 1

This LOCATE command displays the line containing an item that matches or follows S alphabetically:

```
LOCATE S
```

Example 2

This LOCATE command positions the display to the location marked by the most recent CURRENT command:

```
LOCATE &CURRENT
```

LPRINT Command

Function

The LPRINT command is used to copy lines containing the requested target to the List file. The List file is processed on the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300).

Operands

Operand	Description
*	Entering LPRINT * causes the entire virtual screen (all data that can be viewed by scrolling up and down) to be copied to the List file. All excluded lines are copied to the List file as excluded lines (as they appear on the screen at the time the LPRINT * command is entered).
NOTE comment	The NOTE comment operand is used to place descriptive comment lines in the List file. These comments are included in the printed output from the List file. Text of the comment may be up to 50 alphanumeric or 23 DBCS characters.

Usage Notes

The LPRINT command can be entered on the Source View screen (see Figure 34 on page 73), View - Source pop-up (see Figure 35 on page 76), Tree View screen (see Figure 38 on page 80), View - Tree pop-up (see Figure 39 on page 84), and on all List pop-ups.

See online help for more information about these pop-ups.

Examples

Example 1

Enter this LPRINT command to copy all lines containing the characters ZIP to the List file for subsequent printing:

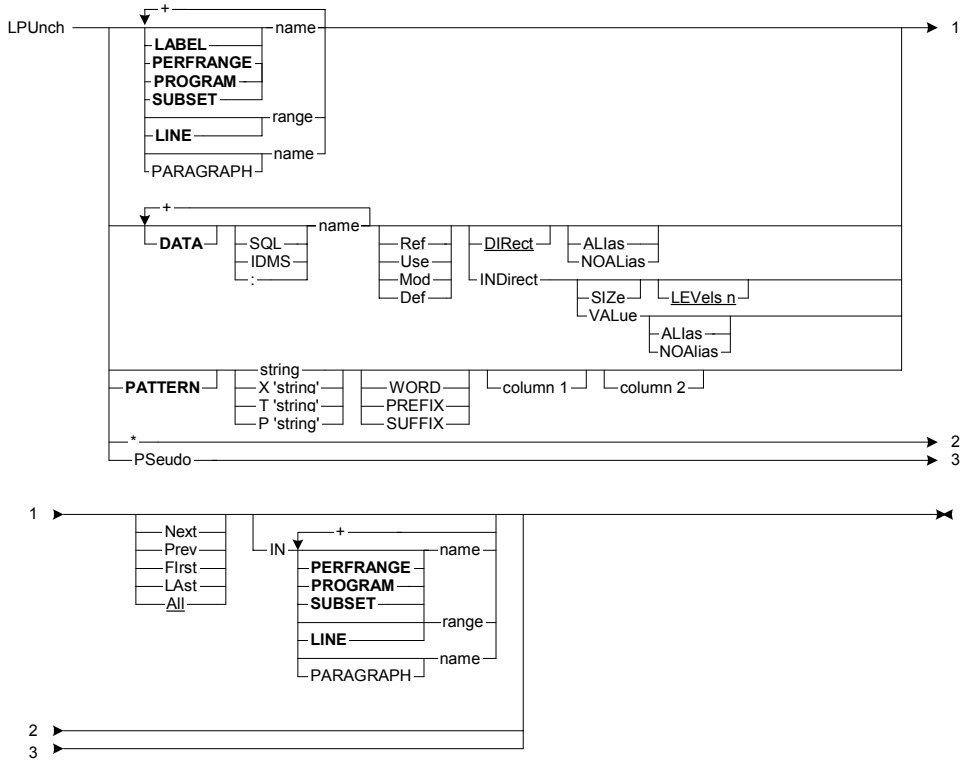
```
LPRINT ZIP
```

Example 2

Enter this LPRINT command to copy the entire virtual screen (including any Pseudo code, zoomed data, disassembled object code, etc.) to the List file, when Program View screen displays:

```
LPRINT *
```

LPUNCH Command



Bold operands are available only with Insight

Function

The LPUNCH command is used to copy lines containing the specified target to the Punch file for subsequent processing. The Punch file can be de-allocated from the Options - Log/List/Punch Definition pop-up (see Figure 144 on page 302).

Note:

If Insight is not installed, the operands PERFRANGE, PROGRAM, SUBSET, LINE, LABEL, DATA, and PATTERN are not available.

Operands

Operand	Description
LABEL name	A PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included.This operand is only available if Insight is installed.
PERFRANGE name	The name specified in a PERFORM statement including all statements that are executed as a result of a PERFORM statement or the name of any section contained in the DECLARATIVES. This operand is only available if Insight is installed.
PROGRAM name	The name of the main program or any nested program representing all the code contained in the program. This includes all the programs physically nested inside the specified program. This operand is only available if Insight is installed.

Operand	Description

SUBSET name	These are predefined COBOL language subsets:
	ASsignment, CAll, CIcs, COBOLII, COBOL/370, COMment, CONditional, COPy, DB2/SQL, DDL, DEAD, DEADCode, DEADData UNTested, DEbug, DEFinition, DIRective, DIVision, DL/I | DL/1, DML, ENtry, EXIt | PGMExit, FALLthrough, GOto, IDMS, INClude, Input, IO, LABel, MAINline, MATH, Output, PARagraph, PERform, RETurn, SECtion, SORTMerge, STructure, TAG, TESted
	Note:
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`Highlighted | HI`
	`NONHighlighted | NHI`
	`Excluded | X`
	`NONExcluded | NX`
	Tagged lines subset of tags appearing in columns 73 - 80.
	This operand is only available if Insight is installed.

LINE range	A single line number or range of lines. Line numbers display in columns 1-6. This operand is only available if Insight is installed.
PARAGRAPH name	A PROCEDURE DIVISION paragraph name or section name. The literals PROCEDURE and PROC can also be specified. The entire paragraph or section is included.

Operand	Description
DATA name	A COBOL data name or qualified COBOL data name. Data name refers to any valid COBOL reference for a data element. These subordinate operands apply to the data name and can be used to further qualify the LPUNCH command: REF, USE, MOD, DEF, ALIAS, NOALIAS, DIRECT and INDIRECT. The defaults are REF, ALIAS and DIRECT. This operand is only available if Insight is installed.
SQL name	Includes data names that are DB2/SQL variables only.
IDMS name	Includes data names that are IDMS variables only.
: name	Includes data names that are COBOL variables only.
Ref	Includes occurrences of the data name where its value is defined, tested, used, set, or modified. This is the default value for the DATA name operand.
Use	Includes occurrences of the data name where its value is being tested or used.
Mod	Includes occurrences of the data name where its value is being set or modified.
Def	Includes definitions of the data name in the DATA DIVISION.
DIRect	Includes occurrences of the data name (and aliases if specified) where it is directly tested, used, set, modified, or defined (based on the REF, USE, MOD, or DEF selection). The default for the DATA name operand is DIRECT. If the SIZE, VALUE or LEVELS operand is specified, INDIRECT is assumed.
INDirect	Includes occurrences of any data name call indirectly affected by the specified data name (and aliases if specified). This can be very useful when working with data names. The indirect data item can be further qualified using the SIZE, VALUE, and LEVELS subordinate operands. These subordinate operands indicate the type of indirect reference to be located. SIZE and All levels are the defaults for the INDIRECT operand. See "Usage Notes" on page 387 for more information.
SIZe	Includes occurrences of data names that could be directly or indirectly affected if the size of the specified data name were to be changed. SIZE is valid only with the INDIRECT operand. This is the default for the INDIRECT operand. See "Usage Notes" on page 387 for more information.

Operand	Description
VALue	Includes occurrences of data names that could be directly or indirectly affected if the specified data name were to be changed. The LEVELS subordinate operand is not used with VALUE. VALUE is valid only with the INDIRECT operand.
LEVels n	Includes all data items that are affected within the specified number of indirect levels. This subordinate operand is not used with the VALUE subordinate operand. All levels is the default for the LEVELS operand. LEVELS is valid only with the INDIRECT operand. See "Usage Notes" on page 387 for more information.
ALIas	Includes these aliases for the data name: • Parent - higher level group item • Child - lower level item • Rename/Redefinition - renamed, redefined, or 88 level items This is the default value for the DATA name operand.
NOAlias	Ignores aliases for the specified data name.
PATTERN	An optional keyword indicating the characters that follow are part of a string. This operand is only available if Insight is installed and a Insight analysis has been run on the COBOL program being tested.
string	A string of characters. If the pattern string contains blanks, it must be enclosed in single or double quotes. The pattern string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands. These subordinate operands describe how the pattern string is to be used.
X'string'	The hexadecimal string option. Specific unprintable characters may be specified by giving the EBCDIC hexadecimal value. The string must be enclosed in single or double quotes.
T'string'	The text string option. A character string is matched regardless of case by using the text option. The string must be enclosed in single or double quotes.

Operand	Description
P'string'	The picture string option. A string profile may be entered instead of exact characters. The string must be enclosed in single or double quotes. These nine special characters are defined by Encore for use in picture strings and can be combined with other characters: P'=' Any character P'¬' Any nonblank character P'.' Any nondisplay character P'#' Any numeric character P'-' Any non-numeric character P'@' Any alphabetic character (upper or lowercase) P'<' Any lowercase alphabetic character P'>' Any uppercase alphabetic character P'$' Any special character (not alphabetic or numeric)
WORD	The specified pattern string preceded and followed by any non-alphanumeric character (except hyphen).
PREFIX	A word that begins with the specified pattern string.
SUFFIX	A word that ends with the specified pattern string.
column1	The column number where the search is to begin.
column2	The column number where the search is to end.
* (asterisk)	Entering LPUNCH * causes the entire virtual screen (all data that can be viewed by scrolling forward and backward) to be copied to the Punch file. All excluded lines are copied to the Punch file as excluded lines (as they appear on the screen at the time the LPUNCH * command is entered).
PSeudo	Entering LPUNCH PSEUDO causes all Pseudo code within the program currently being tested to be copied to the Punch file.
Next	Searches forward from the current cursor position to the next occurrence of the requested target.
Prev	Searches backward from the current cursor position to the previous occurrence of the requested target.
FIrst	Searches from the top of the source file to the first occurrence of the requested target.

Operand	Description
LAst	Searches backward from the bottom of the source file to the first occurrence of the requested target.
All	Searches for all occurrences of the requested target and displays the number found in the short message field. This is the default value for the LPUNCH command.
IN	The IN target operand is optional and restricts the LPUNCH command to the specified target type.

Usage Notes

LPUNCH copies the specified targets to the Punch file for subsequent processing.

Any Pseudo code that is incompatible with COBOL syntax rules is automatically changed to a comment line when copied to the Punch file. These Pseudo code statements remain intact but contain an asterisk (*) in column seven.

To concatenate targets, place a plus sign (+) between the target names. These rules apply to concatenation:

- A concatenated data name can be used wherever a data name is valid. Data name subordinate operands (REF, ALIAS, etc.) pertain to all data names in a concatenated series.
- A concatenated target name can be used wherever a target name is valid. These targets can be concatenated:

  LABEL MARK SUBSET PARAGRAPH
  LINE PERFRANGE PROGRAM

For COBOL II Release 3 and later programs, a data name, label name, or program name that may be ambiguous or used multiple times can be qualified with OF. For example, if the label P120-READ exists in VIAPDEM2 and in the program VIAPDEM1, then it can be qualified with OF (e.g., P120-READ OF VIAPDEM1).

Note:

Selected lines may also be copied to the Punch file by selecting a Search pop-up on the Search pull-down (see Figure 117 on page 251).

See online help for more information about the Search pop-ups and the Search pull-down.

Examples

This LPUNCH command results in the entire source file being copied to the Punch file:

```
LPUNCH SUBSET PERFORM
```

Figure 157 illustrates the Punch file after the LPUNCH SUBSET PERFORM command is entered.

Figure 157 • Lpunch Subset Perform Output Example

```
BROWSE    USERID.ENCxxxxx.VIAPUNCH                    Line 00000001 Col 001 080
Command ===>                                                 Scroll ===> CSR
012800     PERFORM INITIALIZE-PGM.                                   02030000
013000     PERFORM ACCT-MAINTENANCE                                  02050000
013100        THRU ACCT-MAINTENANCE-EXIT                             02060000
013200       UNTIL TRAN-FILE-ENDS.                                   02070000
013400     PERFORM REPORT-FINAL-CNTRS.                               02090000
013600     PERFORM CLOSE-PGM.                                        02110000
015600             PERFORM INITIATE-ACTION                           02310000
016200     PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT.                02370000
016500        PERFORM EXCEPTION-REPORT-1.                            02400000
017300     PERFORM INIT-ACTION-RECORD.                               02480000
018700        PERFORM INITIATE-ACTION.                               02620000
019300        PERFORM INITIATE-ACTION                                02680000
019900        PERFORM INITIATE-ACTION.                               02740000
021600     PERFORM INITIATE-ACTION.                                  02910000
023400     PERFORM SUM-TOTAL-DAYS                                    03380000
023500             VARYING  WORK-CNT FROM 1 BY 1                     03390000
023600             UNTIL TODAY-DD = WORK-CNT.                        03400000
024400     PERFORM OPEN-FILES.                                       03480000
026200     PERFORM GET-NUM-OF-DAYS.                                  03660000
027700     PERFORM CLOSE-FILES.                                      03810000
```

The same information that displays on the Source View is copied to the top of the Punch file. The PERFORM statements indicate that they are part of the program for which the LPUNCH SUBSET PERFORM command was entered. Source statements are shown as they actually appear in the program.

PARMDEF Command

PARMDEF | PDEF

Function

The PARMDEF command displays the Options - Product Parameters pop-up (see Figure 141 on page 296). This pop-up is used to set parameters that affect the online operation of Encore.

Operands

None.

Usage Notes

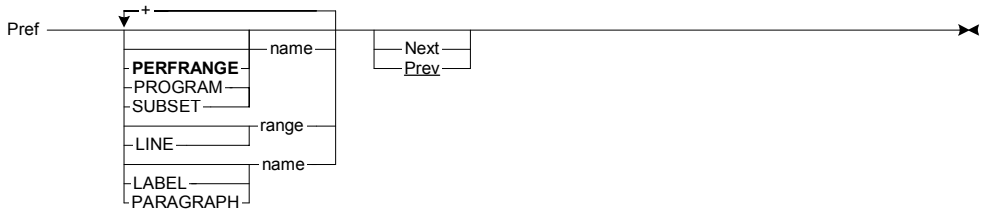
Note:

The Options - Product Parameters pop-up also displays when you select Options ▸ Product Parameters.

The PARMDEF command displays the Options - Product Parameters pop-up. Use the END command to return to the screen on which the PARMDEF command was entered.

See online help for more information about the Options - Product Parameters pop-up.

PREF Command



Bold operands are available only with Insight

Function

The PREF command is used to display the View - Paragraph Cross Reference pop-up (see Figure 51 on page 104) for the requested target. The View - Paragraph Cross Reference pop-up shows how control is transferred to or from the target paragraphs.

Operands

Operand	Description
blank	If no operand is entered, the target paragraph set is determined by the cursor location when the PREF command is issued.
PERFRANGE name	The first statement within the specified PERFORM range.
PROGRAM name	The first executable statement within the specified program.

Operand	Description

SUBSET name	These are predefined COBOL language subsets:
	ASsignment, DEbug, IO
	CAll, DEFinition, LABel
	CIcs, DIRective, MAINline
	COBOLII, DIVision, MATH
	COBOL/370, DL/I | DL/1, Output
	COMment, DML, PARagraph
	CONditional, ENtry, PERform
	COPy, EXIt | PGMExit, RETurn
	DB2/SQL, FALLthrough, SECtion
	DDL, GOto, SORTMerge
	DEAD, IDMS, STructure
	DEADCode, INClude, TAG
	DEADData, Input, TESted | UNTested
	The TESted and UNTested subsets are only available if you have applied TCA results. See the *ASG-SmartTest-TCA User's Guide* for more information.
	This is a screen subset:
	`HIghlighted | HI`
	`NONHighlighted | NONH or NHI`
	`EXcluded | EX or X`
	`NONExcluded | NONE or NX`
	A tagged line subset of tags appearing in columns 73 - 80.
	See Chapter 2, "Product Overview," on page 7 for a description of each subset.

LINE range	A single line number or range of lines. Line numbers display in columns 1 - 6 of Source View.
LABEL name	Any PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. All transfers of control to the label name are included.
PARAGRAPH name	A PROCEDURE DIVISION paragraph name or section name, or the literals PROCEDURE and PROC. The entire paragraph or section is included.

Operand	Description
Next	Generates a list of paragraphs to which the target paragraph transfers control.
Prev	Generates a list of paragraphs that transfer control to the target paragraph. This is the default for the PREF command.

Usage Notes

Note:

The View - Paragraph Cross Reference pop-up (see Figure 51 on page 104) also displays when you select View ▸ Paragraph X-ref.

The PREF command includes all paragraphs containing the target lines. For example, PREF IO displays every paragraph containing IO statements.

To concatenate targets, place a plus sign (+) between the target names. For example, IO+CALL. A concatenated set name can be used wherever a set name is valid. These sets can be concatenated:

LABEL	PROGRAM	PARAGRAPH
LINE	PERFRANGE	SUBSETS

For COBOL II Release 3 (minimum supported release level) programs, a data item, label, or program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. For example, if the label P120-READ exists in another program and also in the program VIAIDEM1, then it can be qualified as P120-READ OF VIAIDEM1.

When there is a blank target with the cursor located on excluded lines, these items apply:

- The first line of the excluded block is implicitly selected.
- If the selected line is not a label, then a backward search is done to locate the previous label. This label then becomes the target.

See online help for more information about the View - Paragraph Cross Reference pop-up.

Example

This PERF command displays the View - Paragraph Cross Reference pop-up with all paragraphs that transfer control to the WRITE-ROUTINE paragraph.

```
PREF WRITE-ROUTINE
```

PRINTLOG Command

PRINTLOG | PLOg

Function

The PRINTLOG command is used to display the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300). On this pop-up the Log, List and Punch files can be printed.

Operands

None.

Usage Notes

Note:

The Options - Log/List/Punch Definition pop-up also displays when you select Options ▸ Log/list/punch.

The PRINTLOG command can be issued from any Encore screen.

See online help for more information about the Options - Log/List/Punch Definition pop-up, and on the Log, List and Punch files.

PRINTLST Command

PRINTLST | PLIst

Function

The PRINTLST command is used to display the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300). On this pop-up the Log, List and Punch files can be printed.

Operands

None.

Usage Notes

Note:

The Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300) also displays when you select Options ▸ Log/list/punch.

The PRINTLST command can be issued from any Encore screen.

See online help for more information about the Options - Log/List/Punch Definition pop-up, and on the Log, List and Punch files.

PROCESS Command

```
PROCess ──┬──────┬──┬─(member)──────┬──model command──────────────────────────►◄
          └STEP──┘  ├─dsn(member)───┤
                    ├─'dsn'─────────┤
                    └─'dsn(member)'─┘
```

Function

The PROCESS command defines an action to be performed repetitively on a file of tokens.

The token file may contain the results list from any ESW product that can produce a file of tokens or a file of tokens you have produced. Tokens may be any text without embedded blanks or any text enclosed in quotes. Tokens must be separated by one or more spaces.

The PROCESS primary command process the first nine tokens in a file. Any tokens beyond nine are ignored. Any tokens which are referenced in the PROCESS command but not present in the token file are passed as null.

Tokens enclosed in quotes retain their quotes after substitution in the model statement.

This example illustrates a possible format for a record in the token file:

```
token1 'token 2' token3  token4  token5  'token 6'  token7  token8  token9
```

Insight, Encore, SmartTest, and Alliance can produce an output file using LPRINT, LPUNCH, or the Query Facility. This file can be edited to produce a file of tokens which can be read by the PROCESS primary command.

Operands

Operand	Description
STEP	The STEP operand is used to step through the commands in the script file. Each command in the script file displays in the command input area. The displayed command can be changed or erased from the command input area. Press Enter to execute the displayed command. Processing of the scrip file continues in this manner until all commands have displayed or executed.
(member)	A member in the default script library (defined during product installation) which contains tokens to be processed. The default script libraries may be modified for your profile using the Script file allocation action on the Options pull-down. Parentheses are required if only the member is entered.

Operand	Description
dsn(member)	A partitioned dataset member that contains tokens to be processed. This dataset must be in card image format (LRECL=80). The specified dataset name and member must be entered following TSO usage conventions in this format: `'dataset.name(member)'` Quotes are required unless the high-level qualifier is your TSO UserID.
dsn	A sequential dataset that contains tokens to be processed. This dataset must be in card image format (LRECL=80). The specified dataset must be entered following TSO usage conventions in this format: `'sequential.dataset.name'` Quotes are required unless the high-level qualifier is your TSO prefix or userID.
model command	The model command may be any ESW product command which is valid in a Script file and which can be defined using substitution variables. Tokens from the file are always substituted in the model command in numerical order from 1 through 9 for the substitution variables &1 through &9.

When the PROCESS command is included in a script, the model command may optionally reference substitution variables that were passed to the script. This is done by specifying any of the override substitution variables &&1 through &&9.

Usage Notes

The PROCESS command allows automated processing of the results from any ESW product which can produce a text file of tokens.

For example, if Alliance creates a file of records named `<userid>.PROGDATA.TEXT` in which each record contained a program name and a data name, the file can be processed with this command:

```
PROCESS PROGDATA.TEXT EXECUTE SCANPROG.SCRIPT PARMS &1 &2
```

This command reads each record of the file PROGDATA.TEXT in turn and substitutes program name from the record into substitution variable &1 and data name from the record into substitution variable &2. It then executes the named script (`<userid>.SCANPROG.SCRIPT`), passing the substituted variables &1 and &2.

The script file SCANPROG.SCRIPT might contain these commands to qualify a program for viewing and execute another script file for the program:

```
Q &1

EXECUTE PROGPROC.SCRIPT PARMS &2
```

This allows the processing of multiple programs with a single command sequence.

To accomplish this task, the PROCESS command verifies and opens the specified file. This logic is then repeated until an end-of-file occurs for the specified file:

- Read a record. If the record is null or contains an asterisk (*) in column 1, ignore the record and begin again.
- Substitute tokens from the record into the substitution variables &1 through &9.
- Construct a command using the model statement with the variable substitution.
- Execute the constructed command as if it were a script command.
- Begin again at step 1 and continue until an end-of-file.

PROCESS commands can occur within scripts and may be nested. If PROCESS commands are nested, they must refer to different datasets.

Examples

"Example 1" on page 398 shows how you might use the PROCESS command in its most basic form to process a data name list and then how this basic concept could be extended to automate the same action for a list of programs.

"Example 2" on page 398 illustrates the use of override substitution variables.

Example 1

Given a file of program names called <*userid*>.PROGRAM.LIST, you could execute this PROCESS command:

```
PROCESS PROGRAM.LIST EXEC MYSCRIPT.SCRIPT PARMS &1
```

Assume that <*userid*>.MYSCRIPT.SCRIPT contains these commands:

```
* STEP 1: OPEN THE PROGRAM
QUALIFY &1
* STEP 2: POSITION TO THE PROGRAM VIEW SCREEN (SmartTest only)
FX 1
* STEP 3: PROCESS THE DATA NAME LIST
PROCESS DATANAME.LIST HI &1
* STEP 4: SAVE THE RESULTS OF THE PROCESS COMMAND(Assumes Insight installed)
SAVE MARKS AS &1MK
* STEP 5: CLOSE THE PROGRAM
QUALIFY CANCEL
```

The set of commands in <*userid*>.MYSCRIPT.SCRIPT are executed one time for each program in <userid>.PROGRAM.LIST. <*userid*>.MYSCRIPT.SCRIPT contains a PROCESS command which executes the model command, HIGHLIGHT, one time for each record in <*userid*>.DATANAME.LIST.

The result is a MARK set, named <*pgmname*>MK, for each program in <*userid*>.PROGRAM.LIST. Each MARK set contains highlighted references to uses in the program of all data names in <*userid*>.DATANAME.LIST.

Example 2

This example illustrates the technique of passing override substitution variables &&1 and &&2 to the PROCESS command contained in the script.

```
EXEC (MYSCRIPT) PARMS ARG1 ARG2 ARG3
```

Assume the script in member MYSCRIPT in the default script libraries contains this PROCESS command:

```
PROCESS 'PROGRAM.LIST' EXEC (PGSCRIPT) PARMS &&2 &1 &2 &&1
```

The parameters from the initial EXECUTE statement and the tokens from the file are substituted in the PROCESS statement model command with this:

```
&&2     is passed the value ARG2
&1      is passed the first token of each record read from the file
'PROGRAM.LIST'
&2      is passed the second token of each record read from the file
'PROGRAM.LIST'
&&1     is passed the value ARG1
```

PRODLVL Command

PRODLVL

Function

The PRODLVL command is used to display the current Encore and Center product levels.

Operands

None.

Usage Notes

Note:

This information is requested when the ASG Customer Support group is contacted for assistance. This information also displays when you select Help ▸ About, which displays the Help - About pop-up (Figure 154 on page 320).

The PRODLVL command displays the product name, operating system, product release number, and release level on the message line, in this format:

```
ASG4250I ASG-Encore-OS(390) x.x (xxx), y.y (yyy), z.z (zzz).
```

where:

`x.x` is the Encore release number and `xxx` is the Encore release level.

`y.y` is the Encore Center release number and `yyy` is the Encore Center release level.

`z.z` is the Center release number and `zzz` is the Center release level.

See online help for more information about ASG Customer Support and the Help-About pop-up.

PROGRAM Command

Note:

This command is valid only if Insight is installed.

Function

The PROGRAM command is used to end the current Source View session and enter a new Source View session for the specified program. This is useful when you are viewing a system of multiple programs.

Operands

Operand	Description
Blank	Displays the AKR Program Selection pop-up for selection of a program.
pgm	Specifies a COBOL program or an entry point name. If the specified program or entry point name is unavailable in the current AKR, an error message displays.
Save	Saves equates and marks for this program in the Application Knowledge Repository. This operand overrides the values on the Options - Product Parameters pop-up.
Nosave	Specifies that equates and marks for this program are not saved. This operand overrides the values on the Options - Product Parameters pop-up.

Usage Notes

You can also enter a new Source View session for a new program from the File - Open Program Request pop-up. To display the File - Open Program Request pop-up, select File ▸ Open and press Enter.

Using the PROGRAM command is the easiest way of switching to another program. However, Encore does not save the calling sequence. This means the GOBACK command cannot be issued to return to a previous program.

Example

When current Source View program is VIARDEMO and this command is entered:

```
PROGRAM VIARBRWS NOSAVE
```

The VIARBRWS program displays without saving marks and equates for VIARDEMO.

QUALIFY Command

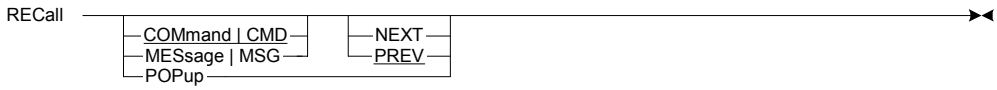
Function

The QUALIFY command is used to display the specified program on the screen for investigation. The qualified program becomes the current program, and program understanding commands operate on the qualified program. Additional programs can be qualified as needed.

Operands

Operand	Description
pgm	The program displayed on the Source View screen (see Figure 34 on page 73).
CANcel	Closes the current program.
ALL	Closes all open programs.

RECALL Command



Function

The RECALL command is used to display the previous ASG primary or internal command, message, or pop-up. The last 20 executed commands and the last 20 displayed messages are stacked. You can redisplay these commands or messages by using the RECALL command.

Operands

Operand	Description
blank	Displays the last command that was stacked. Once the RECALL command has been entered with operands, subsequently entering RECALL with no operands re-uses the same operands that were last entered.
COMmand/CMD	Displays a stacked command.
MESsage/MSG	Displays a stacked message.
NEXT	Displays the next command or message in the stack.
PREV	Displays the previous command or message in the stack. This is the default value.
POPup	Displays the last pop-up that was viewed.

Usage Notes

The RECALL command can be entered repeatedly to display any of the 20 stacked commands or messages. The NEXT and PREV operands can be used to move forward or backward through the stacked commands or messages. Once the desired command displays, it can be executed again by pressing Enter. Any command that is recalled can be changed prior to executing it.

Internal commands are issued by ESW products when certain selections are made on pull-downs and pop-ups. You can display these commands on screen before they are executed by setting Learn Mode on. See the "SET Command" on page 417 for more information.

The operands specified for the RECALL command remain in effect until one of these conditions occur:

- A different operand is specified.
- A different primary command is executed. When this occurs, the RECALL command default operands are automatically set. A message displays that indicates all stacked commands or messages were shown and the stack is redisplays.

The RECALL command is the default for PF12/24. The command and desired operand can be entered in the command input area, or the operand can be entered in the command input area and the PF key pressed.

See online help for more information about PF keys.

REDO Command

REDo ──►◄

Function

The REDO command executes the RFIND command from the cursor position, after a FIND or FINDXTND command was issued on the Source View screen (see Figure 34 on page 73) or View - Source pop-up (see Figure 35 on page 76).

Operands

None.

Usage Notes

REDO automatically executes the RFIND command.

REPEAT Command

REPeat

Function

The REPEAT command executes the last stacked ASG primary command again.

Operands

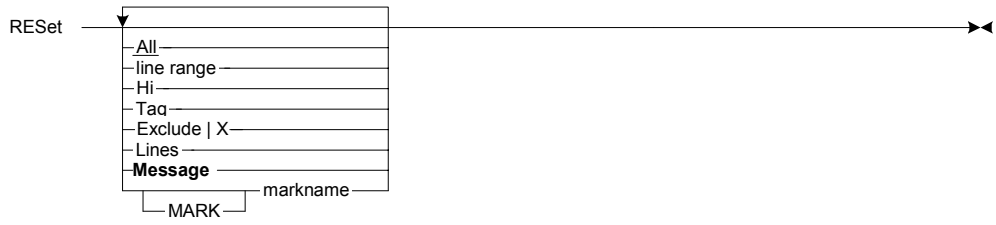
None.

Usage Notes

Enter the REPEAT command in the command input area to re-execute the last stacked primary command. The command is executed from the cursor position.

To modify the command before it is re-executed, the RECALL command must be used.

RESET Command



Bold operands are available only with Insight

Function

The RESET command, valid on any screen once a program has been selected, removes highlighting and tags, cancels pending line commands, terminates message lines, or redisplays all excluded lines.

The RESET command results depend on the view method currently in use.

Operands

Operand	Description
blank	Entering the RESET command with no operands defaults to the value ALL.
All	The ALL operand resets all conditions indicated by each of the other operands. This is the default.
line range	Entering a source line number or a range of source line numbers resets the HI, TAG, EXCLUDED, LINES, and MESSAGE conditions only for the specified line or range of lines.
Hi	Removes the highlighting from any line that has been highlighted as a result of another Encore command.
Tag	Erases tags in columns 73-80 that have been set as a result of another Encore command.
Excluded | X	Displays any lines that have been excluded.

Operand	Description
Lines	Clears any pending line commands.
Message	Erases short or long messages that display.
MARK mark name	Entering a mark name resets the HI, TAG, EXCLUDED, LINES, and MESSAGE conditions only for statements belonging to the mark set. The keyword MARK is not needed if there is no ambiguity.

Usage Notes

Note:

Features can also be reset by selecting View ▸ Reset, which displays the View - Reset Request pop-up (see Figure 54 on page 109).

The RESET command results depend on the view method currently in use. This table describes the results for each view method.

View Method	RESET Command Results
Source View	Turns off highlighting, cancels pending line commands, resets certain tags, terminates message lines, or redisplays excluded lines.
Tree View	Turns off highlighting, cancels pending line commands, resets certain tags, terminates message lines, or redisplays excluded lines.
Structure View	Turns of highlighting or terminates message lines.

See online help for more information about view methods and the View - Reset Request pop-up.

RETURN Command

Function

The RETURN command is used to terminate the current function and return to the first Encore screen displayed.

Operands

Operand	Description
blank	Entering the RETURN command with no operands uses the values specified on the Options - Product Parameters pop-up (see Figure 141 on page 296) to determine if Equates are to be saved.
Save	Saves Equates created for this program in the AKR. This operand overrides the values set on the Options - Product Parameters pop-up.
Nosave	Equates are not saved for this program. This operand overrides the values set on the Options - Product Parameters pop-up.

Usage Notes

Using the RETURN command is a quick way of ending a Encore function. RETURN simulates multiple END commands, returning directly to the first Encore screen displayed.

See online help for more information about the Options - Product Parameters pop-up.

RFIND Command

RFind

Function

The RFIND command finds the next or previous occurrence of the target of the previous FIND or FINDXTND command. RFIND is valid on the Source View screen (see Figure 34 on page 73) and View - Source pop-up (see Figure 35 on page 76).

Operands

None.

Usage Notes

The RFIND command is a convenient means of locating the next or previous occurrence of the specified target of a FIND or FINDXTND command. The search is performed in the direction indicated in the FIND or FINDXTND command. If the direction was Next, First, or All, RFIND finds the next occurrence. If the direction was Prev or Last, RFIND finds the previous occurrence.

The RFIND command is the default for PF05/17. The command and the desired operand can be entered in the command input area, or the operand can be entered in the command input area and the PF key pressed.

See online help for more information about PF keys.

RNSCOPY Command

```
RNSCOPY ──────────────────────────────────────────────────────────►◄
```

Function

The RNSCOPY command copies the generated COBOL module from a temporary file into the current Edit member.

Operands

None.

Usage Notes

The RNSCOPY command is valid only during an Edit session. RNSCOPY copies all statements from the input work file and inserts lines into the current edit member. The current Edit member must not contain any lines, (i.e., the member must be empty).

RSTRUCTURE-VIEW Command

RSTRUCTure - view | RSTView

Functions

The RSTRUCTURE-VIEW command redisplays the previous Structure View. It may display as a Structure View screen (see Figure 41 on page 87) or View - Perform Structure pop-up (see Figure 42 on page 90).

Operands

None.

Usage Notes

The Structure View screen redisplays with the exact information as previously shown. See online help for more information about the Structure View screen and View - Perform Structure pop-up.

RTREEVW Command

RTREEvw | RTV

Function

The RTREEVW command is used to redisplay the previous Tree View. It may display as a Tree View screen (see Figure 38 on page 80) or View - Tree pop-up (see Figure 39 on page 84).

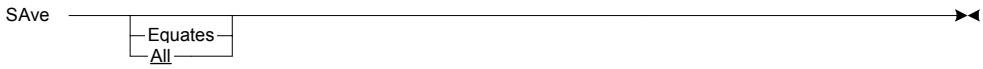
Operands

None.

Usage Notes

The Tree View screen redisplays with the exact information as previously shown. See online help for more information about the Tree View screen and View - Tree pop-up.

SAVE Command



Function

The SAVE command is used to save Equates in the Application Knowledge Repository (AKR). This command overrides the values set on the Options - Product Parameters pop-up (see Figure 141 on page 296).

Operands

Operand	Description
blank	Entering the SAVE command without operands defaults to the value ALL.
Equates	Saves all Equates entered for the current program.
All	Saves Equates entered for the current program. This is the default.

Usage Notes

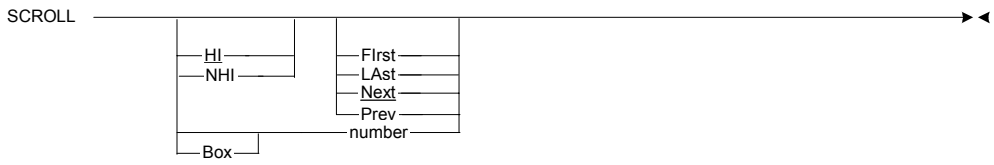
Save values established on the Options - Product Parameters pop-up can be overridden using the SAVE command.

Note:

Equates can also be saved by selecting File ▸ Save equates.

See online help for more information about the Options - Product Parameters pop-up.

SCROLL Command



Function

The SCROLL command is used to find highlighted or nonhighlighted lines on the Tree View screen (see Figure 38 on page 80), View - Tree pop-up (see Figure 39 on page 84), Structure View screen (see Figure 41 on page 87), or View - Perform Structure pop-up (see Figure 42 on page 90).

Operands

Operand	Description
blank	Entering SCROLL with no operand defaults to HI and NEXT.
Hi	Scrolls to a line or box that is highlighted. This is the default.
NHI	Scrolls to a nonhighlighted line or box.
FIrst	Searches from the top of the source file to the first occurrence of the requested target.
LAst	Searches backward from the bottom of the source file to the last occurrence of the requested target.
Next	Searches forward from the current cursor position to the next occurrence of the requested target. This is the default direction.
Prev	Searches backwards from the current cursor position to the previous occurrence of the requested target.
Box number	Identifies a scroll to a specific box number, rather than a line. This operand is only available in Structure View.

Usage Notes

The SCROLL command is only available in the View - Tree pop-up and the View - Perform Structure pop-up. In Tree View, the screen is scrolled to a specified line. In Structure View, the screen is scrolled to a specified box. See online help for more information about Tree View and Structure View.

SELECT Command

```
Select ──┬──ALL──────────────┬──────────────────────────────────────►◄
         ├──name─────────────┤
         ├──statement-number─┤
         └──program-name─────┘
```

Function

The SELECT command selects a data name, perform name, report name, statement number, or program name. This command can also be used on a View pop-up (see Chapter 4, "View Menu," on page 69).

Operands

Operand	Description
blank	On a selection panel only, an item can be indicated by positioning the cursor on the item. On a View pop-up, a similar effect may be achieved by using cursor substitution characters to supply the item name.
ALL	Allows all displayed statements to be selected. This operand is valid on the Extract - Source Ordered View pop-up (Report or Computation Variable, see Figure 67 on page 146), and the Extract - Perform Name List pop-up (see Figure 58 on page 128).
name	Can be a data name that identifies the name of a variable, a perform name that identifies the name of a perform range, or an external name that identifies the name of an external module. It is advisable to enclose an external name in single quotes.
statement number	Identifies the source statement number to select.
program name	Identifies the program to select. This operand is valid only on the File - Program Name List pop-up (see Figure 11 on page 30).

Usage Notes

If the SELECT command is used along with the S (Select) or U (Unselect) line command, there must be no conflict between the line command and the primary command. If a conflict is detected, the line command and the primary command are not performed.

When issued from a View screen during the Computation Variable or Report objectives, the SELECT command allows identification of a data name or external name, then jumps to the Extract - Source Ordered View pop-up (see Figure 67 on page 146). The applicable statement must be chosen from that screen.

Names must be sufficiently qualified to be unambiguous, but need not be fully qualified.

If a statement number operand identifies a line which contains more than one executable COBOL statement, the first statement is chosen. This action may be overridden by positioning the cursor on the same line, in which case the first statement occurring on or after the cursor is chosen.

If a statement number operand does not identify a line containing an executable statement, Encore searches forward to the next line containing an executable statement and chooses the first statement on that line.

See the "Usage Notes" on page 426 for the UNSELECT command for more information.

See online help for more information about View screens. See the *ASG-Encore User's Guide* for more information about Computation Variable and Report objectives.

SET Command

Function

The SET command is used to enable or disable the mode indicated by the specified operand.

Operands

Operand	Description
blank	Displays the Options - Processing Modes pop-up (see Figure 148 on page 308) that shows the current setting for each SET command mode. See "Usage Notes" on page 418 for more information.
LEarn	Displays the LEARN Mode - Generated Command pop-up (see Figure 158 on page 418) that specifies the internally-generated primary command when these options are requested on pop-ups. The default for this operand is OFF.
SCript	Automatically generates a script file and saves all Encore primary commands entered when this mode is ON. The script file is closed (ignored) when this mode is OFF. The default for this operand is OFF. **Note:** If the TSO PROFILE NOPREFIX option is in effect, the user ID is used as the high level qualifier for the script file (e.g., `userid.ENC`*`xxxxx`*`.VIASCRIP`). To read and execute the script file, use the EXECUTE command; or select File ▸ Execute script, which displays the File - Execute Script pop-up (see Figure 27 on page 59).

Operand	Description
RESULT	The RESULT operand can be used with the SCRIPT operand to save the results of FIND and FINDXTND commands as comments in the script file.

XMode	All lines are excluded from the screen before a primary command is executed when this mode is ON. This mode affects all primary commands with the exception of these:
	`BRANCH` `RFIND`
	`LOCATE` `SCROLL`
	The default for this operand is OFF.

Usage Notes

Note:

Any of the modes that are enabled or disabled with the SET command can also be enabled or disabled on the Options - Processing Modes pop-up (see Figure 148 on page 308). To display the Options - Processing Modes pop-up, select Options ▸ Modes, or enter the SET command with no operands on any screen.

See online help for more information about the Options - Processing Modes pop-up.

LEARN Mode - Generated Command Pop-up

The LEARN Mode - Generated Command pop-up, shown in Figure 158, automatically displays for an internally-generated primary command when actions are selected or specified on pop-ups. The LEARN mode has to be set to ON by using either the SET command or the Options - Processing Modes pop-up (see Figure 148 on page 308).

Figure 158 • LEARN Mode - Generated Command Pop-up

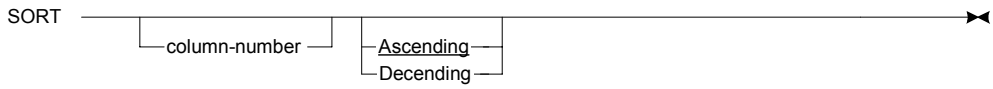
```
_                LEARN Mode - Generated Command

 To process the generated command, press Enter.  To cancel
 the command, press PF3/15 (END).

 Command  TREEVIEW 2
```

Follow the instructions on the pop-up to process or cancel the command.

SORT Command



Function

The SORT command is used to sort the entries on an Extract List pop-up (see Chapter 5, "Extract Menu," on page 113) by any of the columns on the pop-up.

Operands

Operand	Description
blank	Entering the SORT command with no operand displays a pop-up with fields to enter the column number and type of sort.
column-number	The column on which to sort. Columns are numbered consecutively from left to right on the pop-up.
Ascending	Sort the column in ascending order. This is the default.
Descending	Sort the column in descending order.

Usage Notes

Where this command is valid, a SORT PF key displays on the pop-up that SORT is available.

SOURCE-VIEW Command

SOURCE-VIEW | SView

Function

The SOURCE-VIEW command displays the Source on the Source View screen (see Figure 34 on page 73) or View -Source pop-up (see Figure 35 on page 76).

Operands

None.

Usage Notes

Use the END command to return to the screen from which the SOURCE-VIEW command was issued.

See online help for more information about the Source View screen and View - Source pop-up.

STRUCTURE-VIEW Command

```
STRUCTure-view | STView ──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──►◄
                          │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │  │
```

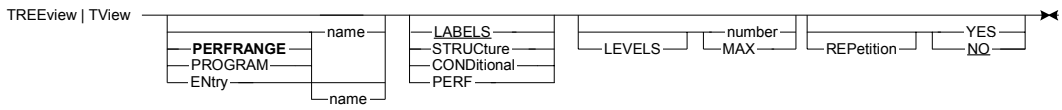
Operand	Description
LEVELs	Controls the amount of information displayed on entering Structure View. Specify the number of nested control flow structures that display. The default, 2, displays the first two levels of nested performs. MAX displays the maximum nesting level. However, if the maximum nesting level consists of only repeated perform ranges, then the nested level is truncated to the next higher level which does not consist exclusively of repeated perform ranges.
REPetition	Controls the display of duplicated perform range structures. If REPETITION is YES, all duplicate perform ranges display. If REPETITION is NO, duplicate perform range structures are only flagged as REPEATED. The default is NO.

Usage Notes

Use the END command to return to the screen where the STRUCTURE-VIEW command was entered.

See online help for more information about the Structure View screen and View - Perform Structure pop-up.

TREEVIEW Command



Bold operands are available only with Insight

Function

The TREEVIEW command displays the Tree View (see Figure 38 on page 80) and View - Tree pop-up (see Figure 39 on page 84).

Operands

Operand	Description
blank	If the TREEVIEW command is entered without operands, the defaults identified below are used.
PERFRANGE name	The starting paragraph for the Tree View. A name is any PROCEDURE DIVISION paragraph name or section name.
PROGRAM name	The main program or any of the nested programs representing all the code that is executed by entering the specified program. For COBOL II Release 3 (minimum supported release level) programs, a program name that may be ambiguous or used multiple times can be qualified by using OF followed by the program name. For example, if the label P120-READ exists in another program and also in the program VIARDEM1, then it can be qualified as P120-READ OF VIARDEM1.
ENTRY name	If ENTRY is specified without a name, the main program structure and the structure for all ENTRY points display. If a name is specified, the structure for that entry point displays.
LABELS	Displays the structure of the program using the paragraphs and section labels of the control flow statements. LABELS is the default.
STRUCture	Displays the structure of the program using control flow verbs (PERFORM, GO TO, ALTER, etc.) and program labels.
CONDitional	Displays the conditional statements that surround control flow statements, plus the control flow statements and labels.

Operand	Description
PERF	Displays the perform range hierarchy for the program.
LEVELs number	Controls the amount of information displayed on entering the Tree View. Specify the number of nested control flow structures that display. The default, 2, displays the first two levels of nested performs.
MAX	Displays the maximum nesting level. However, if the maximum nesting level consists of only repeated perform ranges, then the nested level is truncated to the next higher level which does not consist exclusively of repeated perform ranges.
REPetition	Controls the display of duplicate perform ranges. If REPETITION is YES, all duplicate perform ranges display. If REPETITION is NO, duplicate perform ranges are only flagged as REPEATED, and are not expanded unless a ZOOMIN command is entered for the perform ranges. The default is NO.

Usage Notes

Use the END command to return to the screen from which the TREEVIEW command was issued.

See online help for more information about the Tree View screen and View - Tree pop-up.

UNSELECT Command

```
Unselect ──┬──────────────┬──────────────┬──────────────┬──►◄
           ├─ALL──────────┤              │              │
           ├─name─────────┤              │              │
           └─statement-number──────────┘
```

Function

The UNSELECT command unselects a data name, perform name, external procedure name, or statement from a list of selected items. This command can also be used on any View screen.

Operands

Operand	Description
blank	On a selection panel only, an item may be indicated by positioning the cursor on the item. On a View screen, a similar effect may be achieved by using cursor substitution characters to supply the item name.
ALL	Allows you to unselect all displayed statements. This operand is valid on the Extract -Source Ordered View pop-up (Report or Computation Variable, see Figure 67 on page 146), and the Extract - Perform Name List pop-up (see Figure 58 on page 128).
name	May be a data name, identifying the name of a variable; a perform name, identifying the name of a perform range; or an external name, identifying the name of an external module. It is advisable to enclose an external name in single quotes.
statement number	Identifies the source statement number to unselect.

Usage Notes

When a data name is unselected from the Extract - Data Name List pop-up (see Figure 68 on page 148) in the Computation Variable or Report objectives, all selected statements which reference the data-name are unselected.

However, remember that for the Transaction objective the list of selected data-names merely determines the set of conditional statements that you can see on the Extract - Transaction Paths List pop-up (see Figure 75 on page 167). Unselecting a data name in the Transaction objective does not alter the selected Transaction Path.

If the UNSELECT command is used along with the S (Select) or U (Unselect) line command, there must be no conflict between the line command and the primary command. If a conflict is detected, the line command and the primary command are not performed.

Names must be sufficiently qualified to be unambiguous, but need not be fully qualified.

If a statement number operand identifies a line which contains more than one executable COBOL statement, the first statement is chosen. This action may be overridden by positioning the cursor on the same line, in which case the first statement occurring on or after the cursor is chosen.

If a statement number operand does not identify a line containing an executable statement, Encore searches forward to the next line containing an executable statement and chooses the first statement on that line.

See online help for more information about View screens. See the *ASG-Encore User's Guide* for more information about computation variable, report, and transaction objectives.

VIEW Command

Note:

This command is valid only if Insight is installed.

Function

The VIEW command is used to start a new Source View session for the specified program while retaining the current session.

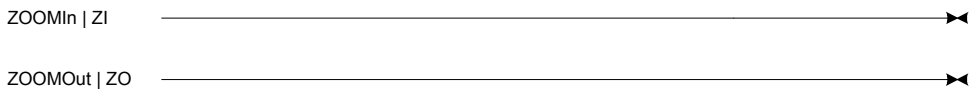
Operands

Operand	Description
Blank	Displays the AKR Program Selection pop-up for selection of a program.
pgm	Specifies a program a Source View session is to be started for.

Usage Notes

When finished with the requested program, the END or GOBACK command returns to the original session.

ZOOMIN/ZOOMOUT Commands



Function

The ZOOMIN and ZOOMOUT commands are used to display or exclude source lines or boxes according to the hierarchical levels of the program.

The ZOOMIN and ZOOMOUT command results depend on the view method currently in use.

Operands

None.

Usage Notes

The ZOOMIN and ZOOMOUT commands are only valid on the Source View (see Figure 34 on page 73), Tree View (see Figure 38 on page 80), Structure View (see Figure 41 on page 87), View - Source (see Figure 35 on page 76), View - Tree (see Figure 39 on page 84), and View - Perform Structure (see Figure 42 on page 90) screens or pop-ups.

The ZOOMIN and ZOOMOUT command results depend on the view method currently in use. The results for each view method for both commands is described below.

The target must be identified by the cursor position for Tree View and Structure View. See online help for more information about view methods.

View Method	ZOOMIN Command Results
Source View	Displays additional source lines according to the hierarchical levels of the program.
Tree View	Displays the next level of PERFORMs according to the Tree structure. If there are no remaining PERFORMs in the paragraph, the COBOL source displays.
Structure View	Displays the View - Source pop-up with a shortened action bar, showing the COBOL source lines of the Perform Range.

View Method	ZOOMOUT Command Results
Source View	Removes source lines according to the hierarchical levels of the program.
Tree View	Removes the lowest displayed level of PERFORMs according to the Tree structure.
Structure View	ZOOMOUT is not valid on Structure View.

The ZOOMIN command is the default for PF04/16. The command and desired operand can be entered in the command input area, or the operand can be entered in the command input area and the PF key pressed. See online help for more information about PF keys.

These notes apply only to Source View:

- The ZOOMIN and ZOOMOUT commands show the structure of a program and provide a means of stepping through each level or going directly into or out of a particular section of source code.
- PROCEDURE DIVISION hierarchy consists of section labels, paragraph labels, and paragraph code. DATA DIVISION hierarchy consists of sections, FDs, 01 or 77 levels, and all definitions within an 01 level.

These are the general guidelines for ZOOMIN:

- If the cursor is on an excluded block of lines, ZOOMIN displays the highest structure level within that block of lines. If there are multiple statements at that level, they also display.
- If ZOOMIN is issued repeatedly from the same cursor position, successive heading levels display.

These are the general guidelines for ZOOMOUT:

- The cursor cannot be positioned on an excluded line when ZOOMOUT is entered.
- ZOOMOUT excludes the levels in a lowest to highest order. If ZOOMOUT is issued repeatedly from the same cursor location, successive heading levels are excluded.

Note:

ZOOMIN and ZOOMOUT are also available on the View pull-down (see Figure 33 on page 71). See online help for more information.

12 Analyze Functions

This chapter describes Encore's Analyze functions and contains these sections:

Section	Page
Introduction	432
Analyzing a COBOL Program	432
Analyze Input Descriptions	432
The Analyze Process	434
Analyze Using ISPF	435
Analyze Using ISPF/PDF Edit	436
File - Analyze Submit Screen	439
Automatic JCL Modifications	444
Analyze Summary Report	449
Adding Analyze Facilities to a Standard Compile Mechanism	452
Analyze Options	453

Introduction

A program must be analyzed before ESW products can provide intelligent information about the program. This chapter describes the methods used to analyze programs.

The analyze process gathers information about the program, including program relationships, logic, data and execution paths, and stores this information in the AKR. After the analyze information is placed in the AKR, it is available to ESW products in online and batch environments, where it is accessed to provide valuable information about the design and operation of user systems.

Analyzing a COBOL Program

The analyze process is similar to a COBOL compile. The process has these three primary inputs:

- Source COBOL program (including copy books).
- JCL used to compile and link the COBOL program.
- Options and features which tailor the analyze steps.

Analyze Input Descriptions

COBOL Source Program

Like the compiler, the analyze process requires some basic program standards. These standards include:

- The COBOL language as specified in the IBM COBOL II and COBOL/370 Language Reference manuals is accepted by the analyze job. It correctly processes any program that can be compiled without warnings or errors by the IBM COBOL II compilers or COBOL for MVS and VM are accepted by the analyze job.
- COBOL II, COBOL/370, and COBOL for MVS and VM programs that receive error (E), severe (S), or unrecoverable (U) messages from the IBM compiler cannot be successfully analyzed.

These tables contain the Program Analyzer resource estimates to process COBOL programs of various sizes. Information in the associated table is the result of running Analyze under this criteria:

Version	ASG-Center V6.0
CPU Type	3090-600 running MVS/ESA
Disk Type	3390
Analyze Parms	BUFMAXK=4096KB
Compiler Parms	BUF=256K,SIZ=1024KB

Note:
For the AKR estimates in this table, a record contains 4,096 bytes.

Source Lines	Virtual Memory Size	XA Memory Size	CPU Time MM:SS	AKR Records	VIAUT3 Cyls
1000	256K	12100K	0:03	250	2
2000	256K	12800K	0:15	600	5
5000	256K	16000K	1:00	2000	15
10000	256K	20000K	5:00	5000	40
20000	256K	28000K	15:00	10000	80
50000	4096K	48000K	60:00	30000	240

Compile/Link JCL

This JCL should be the complete JCL used to compile the program. Specifically, the JCL should contain steps to fetch the source from the source manager (such as Librarian or Panvalet), execute the preprocessor, invoke the compiler with the appropriate options and COPY libraries, then invoke the linkage editor.

Analyze Options and Features

The analyze features indicate the type of analysis to be performed:

- An Encore analysis provides data uses and definitions required for Encore extract and generate functions, and also provides the information required for code extraction and execution flow capabilities.
- A SmartTest analysis provides the testing and debugging information required by SmartTest.
- An Insight analysis provides logic and execution flow capabilities.
- A SmartDoc analysis provides the information required for SmartDoc reports. An Extended SmartDoc analysis provides data flow analysis.
- A Recap analysis provides the inventory analysis data required for Visual reports.

Default options for the analyze process are established at installation time. Options that are to be overridden are specified when submitting the analyze job.

The Analyze Process

The analyze process consists of setting up and executing a batch job. There are three methods used to invoke the analyze process. The method to be used depends primarily on the environment from which the analyze process is invoked, but may depend on the access method containing the compile/link JCL. These are the three methods used to invoke the analyze process:

- Analyze Option or Command - Select the analyze option from the Primary Menu, or enter the ANALYZE command from any screen, to display the File - Analyze Submit screen (see Figure 159 on page 439). Enter the required input and output information, then submit the job.
- ISPF - From any ISPF screen, execute the VIASUBDS CLIST. This CLIST is executed by entering TSO VIASUBDS dsn parms, where dsn is a PDS member or sequential dataset containing the compile JCL, and parms represents any of the available execution parameters described in the parameter table in "Analyze Using ISPF/PDF Edit" on page 436 .

- ISPF/PDF Edit - Execute the VIASUB PDS edit macro. This edit macro is executed by entering VIASUB parms, where parms represents any of the available execution parameters described in the parameter table in "Analyze Using ISPF/PDF Edit" on page 436 .

Compile JCL is From	Method for Executing Analyze Job
PDS or sequential dataset.	Analyze Submit screen, VIASUBDS CLIST, or VIASUB edit macro
Librarian, Panvalet, or other user source manager when editing the JCL with ISPF/PDF.	VIASUB edit macro
Screen-driven submit facility that generates JCL.	VIASUBDS CLIST

Analyze Using ISPF

From any ISPF screen, the VIASUBDS CLIST can be used to submit the analyze job. This is the syntax for VIASUBDS:

```
TSO VIASUBDS input.jcl.dsn parms
```

Field	Description
input.jcl.dsn	The dataset containing the compile/link JCL. This dataset must be a sequential dataset or a member of a PDS.
parms	One or more parameters that control the operation of VIASUBDS. Typically, the PANEL parameter is entered to display the File - Analyze Submit screen (see Figure 159 on page 439) for entry of any necessary parameters. The parameters are saved in the ISPF profile and used as defaults for the next analyze submission. The parm table in the Analyze Using ISPF/PDF Edit section below contains a list of these parameters, with default parameters underlined.

Note:

Using the VIASUBDS CLIST requires the ASG CLIST library to be available through the standard SYSPROC allocations.

Analyze Using ISPF/PDF Edit

From the ISPF/PDF Edit screen, the VIASUB edit macro can be used to submit the analyze job. This is the syntax for VIASUB:

```
VIASUB parms
```

Where *parms* is one or more parameters that control the operation of VIASUB.

Note:

Using the VIASUB edit macro requires the ASG CLIST library to be available through the standard SYSPROC allocations.

Typically, the PANEL parameter is entered to display the File - Analyze Submit screen (see Figure 159 on page 439) for entry of any necessary parameters. The parameters are saved in the ISPF profile and used as defaults for the next analyze submission. These are the parameters, with the default parameters underlined:

Parameter	Description
AKR (*xxxxx*)	Indicates the AKR where the results of the analyze job are placed. The specified name must conform to the standard TSO dataset naming conventions.
AOPT (*xxxxxx*)	Specifies the options to be supplied to the analyze job. The COBOL II option is automatically added if the compiler specified in the Input JCL is COBOL II. When specifying more than one analyze option, the options should be separated by commas and enclosed in single quotes; for example, `AOPT('XMEM,RECUR,SUBSYS=D239')`.
<u>CMPL</u> NOCMPL	CMPL indicates a COBOL compile and an analysis is to be executed by the new JCL. NOCMPL indicates the new JCL is to bypass the compile step and only execute the analyze job. When NOCMPL is specified, a return code of 1000 (decimal) greater than the analyze return code is produced. This causes the subsequent job steps (e.g., a link edit) to be bypassed based on a successful compilation. NOCMPL cannot be specified if a SmartTest analysis is being executed.

Parameter	Description
DSCHK NODSCHK	DSCHK indicates datasets needed by the resulting JCL are to be verified to ensure they exist. Specifically, the AKR and the load library containing VIASMNTR are checked. When NODSCHK is specified, the AKR and the load library need not exist at the same time VIASUB or VIASUBDS is executed. NODSCHK is useful when the JCL is being prepared for submission on another system, or for delayed execution when an AKR does not yet exist. Note that the cataloged procedure libraries must exist and be accessible to VIASUBDS or VIASUB.
EDIT	EDIT specifies that the resulting JCL is not to be submitted for batch processing. The PDF editor is invoked for the resulting JCL. Any desired changes can be made and then the JCL can be submitted by entering the SUBMIT command. EDIT must be entered each time it is needed. Note that the edits made to the JCL are not saved. The CREATE command must be used to save the modified JCL elsewhere. The EDIT option is ignored if the File - Analyze Submit screen (see Figure 16 on page 39) displays. In this case, the E command must be entered to edit the JCL.
EN NOEN	EN specifies that a Encore analysis is to be performed.
INS NOINS	INS specified that a Insight analysis is to be performed.
OUTPUT (*xxxxx*)	Specifies that the resulting JCL is not to be submitted for batch processing. The JCL is written to the specified dataset. The specified name must conform to the standard TSO dataset naming conventions. A dataset is created if it does not already exist. OUTPUT must be entered each time it is needed.
PANEL NOPANEL	PANEL indicates that the File - Analyze Submit screen displays so you can enter parameters for the analyze job. The File - Analyze Submit screen displays even if a valid AKR name is specified as a parameter, or can be obtained from the ISPF profile when PANEL is specified.

Parameter	Description
PGM (*xxxxx*)	Specifies a name to be used when storing the program in the AKR. This name overrides the program name in the PROGRAM-ID paragraph.
PROCONLY	Indicates the JCL contains only a cataloged procedure rather than a complete job. PROCONLY suppresses the generation of the VIAIN DD statement. PROCONLY must be entered *each* time it is needed.
REUS NOREUS	REUS specifies that when the program is tested using SmartTest, it is dynamically loaded and is tested with RUN NOMONITOR.
SD NOSD	SD specifies that a SmartDoc analysis is to be performed.
SDR NOSDR	SDR specifies that SmartDoc reports are run.
SDX NOSDX	SDX specifies that Smartdoc Extended analysis is to be performed.
ST NOST	ST specifies that a SmartTest analysis is to be performed.
STX NOSTX	STX specifies that an Extended SmartTest analysis is to be performed. When the INS and ST parameters are specified, an Extended SmartTest analysis is automatically performed.

File - Analyze Submit Screen

The File - Analyze Submit screen, shown in Figure 159, displays when the PANEL parameter is specified when executing VIASUBDS or VIASUB, or when the NOPANEL option is used and an error condition is detected.

Figure 159 • File - Analyze Submit Screen

```
                          File - Analyze Submit
Command ===> ____________________________________________________________

           E - Edit JCL                                S - Submit JCL

Compile and link JCL (PDS or sequential):
   Data set name  'USER.TEST.CNTL(YOURJCL)'

Analyze features (Y/N):
   ASG-Encore:  Y

AKR data set name 'USER.TEST.AKR'
AKR program name  __________          (if overriding PROGRAM-ID)

Analyze options:
   ______________________________________________________________________
   ______________________________________________________________________
   ______________________________________________________________________

Compile? (Y/N) . . . . . . . . . Y     (Y if needed by features)
Link load module reusable? (Y/N) Y     (ASG-SmartTest only)
```

Options

Field	Description
E - Edit JCL	Select option E to review or change the compile/analyze JCL, if necessary. When option E is selected, the JCL to be edited is generated from the JCL specified when the VIASUBDS CLIST or VIASUB edit macro was invoked, applying the rules outlined in "Automatic JCL Modifications" on page 444 . Then, the generated JCL displays on the Edit screen. When editing is complete, the ISPF SUBMIT command must be entered to submit the edited JCL for execution. Optionally, the edited JCL can be saved in a partitioned dataset by using the ISPF CREATE command. Otherwise, any changes made at this time are not saved.
S - Submit JCL	Select option S to submit the JCL to compile/analyze the specified program. The JCL submitted is generated from the JCL specified when the VIASUBDS CLIST or VIASUB edit macro was invoked, applying the rules outlined in "Automatic JCL Modifications" on page 444 .
D - Doc Options	Displays only if SmartDoc is installed. Select option D to display the SmartDoc Options screen used to request an Extended SmartDoc analysis and to specify which reports (if any) are to be generated.

Fields

Field	Description
Compile and link JCL (PDS or sequential)	Data set name - Required. Enter the PDS member or sequential dataset containing the JCL to compile and link the program. If the JCL resides in a source manager such as Librarian or Panvalet, use the VIASUB edit macro to submit the compile/analyze job.
Analyze Features	• Understand - Optional. This field displays only if Insight is installed. This type of analysis provides the logic and program execution flow capabilities of Insight. If Insight is the only product installed, this field contains Y and cannot be changed. The default is N. • Test - Optional. This field displays only if SmartTest is installed. Y indicates that a SmartTest compile/analysis is to be performed. This type of analysis provides the testing and debugging information required by SmartTest. The default is N. • Extended Analysis - Optional. An extended analysis provides the data flow analysis for SmartDoc reports or the testing and debugging information for SmartTest. • Document - Optional. This field displays only if SmartDoc is installed. This type of analysis provides the report information generated by SmartDoc. The default is N. • Abend/Dump - Optional. This field displays only if SmartDoc is installed. This type of analysis lets you analyze the source used in diagnositng dumps. The default is N. • Re-engineer - Optional. Y (yes) specifies that a Encore analysis is to be performed. This type of analysis provides the logic and program execution flow capabilities of Encore. If Encore is the only product installed, this field contains Y and cannot be changed. The default is Y.
AKR dataset name	Required. Enter the dataset name of the AKR that is to contain the information for the analyzed program.
AKR program name	Optional. Enter an alias name to be used by the analyze job to save its results in the AKR. If a value is not entered in this field, the analyze job uses the program name from the PROGRAM-ID statement in the COBOL source as the name under which to save results in the AKR. This field is only used for the AKR program name and does not change the COBOL program name in the source.

Field	Description
Analyze options	Optional. Enter analyze options that are to be overridden. Default options for the analyze job are established at installation time. You can enter report parameters, which are used to produce analysis reports that will help you understand and diagnose analysis problems and anomalies. You can also enter additional analyze parameters to utilize the Anomaly Repair Facility to correct certain anomalies that may exist within your code. See "Analysis and Anomaly Report Parameters" on page 441 and "Anomaly Repair Facility Parameters" on page 442 for information about available analyze parameters.
Compile?	Optional. A program does not need to be compiled if Insight, Encore, or SmartDoc are the only features specified. The compile step can be suppressed by entering N in this field. This field is forced to a value of Y if SmartTest is selected.
Link load module reusable?	Optional. This field is used to test a program under SmartTest that is dynamically loaded if it is tested. It is necessary to mark the load module as reusable so that the Breakpoints are retained across calls (SmartTEst only). The default is Y.
Display this panel by default in the future?	Optional. Y causes the ISPF profile to be updated to display this screen whenever subsequent executions of VIASUBDS or VIASUB are invoked. When this field contains N, this screen does not display on subsequent executions of VIASUBDS or VIASUB unless an error condition is encountered.

Analysis and Anomaly Report Parameters

Field	Description
VIARERPT	Specifies that you want the analysis process to stop after an analysis report is generated.
VIARERPTAN	Specifies that you want the analysis process to continue after and analysis report is generated and save the information in the AKR.
NUMPRM(##)	Determines the number of parameters you allow before a call is listed on the Excessive Parameter List report.
RPTCEN	Centers the generated report (the default format for all reports is left justified).

Anomaly Repair Facility Parameters

Field	Description

VIAANOMF	Instructs the analyzer to run the Anomaly Repair Facility. This is the batch method for anomaly repair. You can also select the online method by using the Anomaly Facility - Batch Submit screen (see "Anomaly Facility - Batch Submit Pop-up" on page 62) accessed from the File pull-down menu (see Figure 9 on page 26). This parameter can be followed by these additional parameters:
	• PARM(user parm) - This parameter can be any valid user parameter (e.g., DEBUG).
	• ANOMALY - This parameter is required for anomaly fixes and is used to specify these types of anomalies to be fixed:
	— OOPGT for out of perform GOTO
	— GTD for GOTO depending
	— IPR for inconsistent performance ranges
	• COMMENT(YES or NO) - This parameter indicates whether or not you want to include the original code commented out in the generated output that includes the new repaired code. The default is YES.
	• DSN - This parameter indicates the name of the output dataset where the repaired code should be written. If this parameter is not entered, the output is written to DD VIAANOUT. If DSN is a sequential file type, do not code the MEM parameter.
	• MEM - This parameter indicates the member name of the output dataset if DSN is specified. Leave this parameter out if DSN is a sequential file.
	• HEADER(YES or NO) - This parameter indicates whether or not you want an ASG generated header at the beginning of the repaired code. The default is YES.
	For example:
	`VIAANOMF,PARM(DEBUG1),ANOMALY(OOPGT,GTD,IPR),`
	`COMMENT(YES),DSN('ASG.ANOMALY.REPAIRED.CODE'),`
	`MEM(MEM1),HEADER(YES)`

If anomalies are found during anomaly report generation, certain anomalies can be repaired by using the Anomaly Repair Facility, or by entering the anomaly parameters discussed earlier in the Analyze options section of this chapter. See "Anomaly Facility" on page 61 for more information about using the Anomaly Repair Facility.

These reports can be generated from these parameters. See online help for more information.

Report Name	Description
Dead Data (In Copy)	Lists the number of dead data (unreferenced data names) found in each copy member.
Dead Data (Not In Copy)	Lists the number of dead data found in the program.
Dead Code	Lists all dead code (unexecutable procedure division statements) found in the program.
Altered Go To	Lists the location of the ALTER statements in the program with the label being ALTERed and the ALTER target value.
Inconsistent Perform Range	Lists perform ranges that have common starting points and different ending points.
Overlapping Perform Range	Lists perform ranges whose start or end points overlap each other.
Out-of-Perform Go To	Lists perform ranges that should be re-engineered due to out-of-perform jumps or multiple entry points. The score for each perform range indicates how many paragraphs it can reach by taking the out-of-perform jumps.
Recursive Report	Lists perform ranges involved in recursive cycles. The starting line number of the perform range, the line number of the PERFORM statement, and the range being performed are shown. The first and last perform range listed in a cycle indicates the source of the recursion.

Report Name	Description
Conditional Exit	Lists lines that are program exits and if the program exit is conditional. Also shown are conditionally executed perform ranges that either directly or indirectly reach the program exit and line number of the perform statement.
Ambiguous Names	Lists level 01 data items that are either filler, blank, or duplicate.
Go To Depending	Lists the location of the GOTO DEPENDING statements in the program with the depending variable and the number of targets per statement.
CICS Handle Condition	Lists the CICS HANDLE statements contained in the program. The line number of the statement in the program is also shown.
Excessive Parameter list	This report lists all call statements using an excessive number of parameters. The default number of parameters is 8.

Automatic JCL Modifications

The analysis process automatically modifies the JCL based on the specified parameters and analyze options. If problems arise, this procedure can be used as a checklist to perform the analyze process manually until the problem can be determined and resolved. Changes can be made to the JCL, the compile procedure, or a copy of the compile procedure.

Replace the PGM= parameter in the compile step(s) with these parameters:

```
PGM= parameter:  Replace with:

PGM=IGYCRCTL     PGM=VIACOBII
PGM=CPXUPTSM     PGM=VIAOPT3
PGM=CAOTSMON     PGM=VIAOPTII
```

Add DD statements to the compile step(s) for these ESW datasets:

```
//VIAUT2     DD UNIT=SYSDA,SPACE=(vspunit,(vspprim,vspsecd))
//VIASDTOC   DD SYSOUT=*
//VIASDRPT   DD SYSOUT=*
//VIALOG     DD SYSOUT=*
//VIAMRPT    DD SYSOUT=*
//VIAPRINT   DD SYSOUT=*
//VIAAKR     DD DSN=[specified AKR name],DISP=SHR
```

Note:

The VSPUNIT, VSPPRIM, and VSPSECD parameter values are defined at installation time. See the *ASG-Center Installation Guide* for more information.

To modify the JCL

1. If the SYSIN DD statement contains FREE=CLOSE, change it to FREE=END.

2. Ensure that the ESW load libraries are available to the modified step by adding a //STEPLIB DD statement specifying the ESW load libraries, or by concatenating these libraries to an existing STEPLIB DD.

3. Ensure that the JOB and the modified STEP EXEC statements have a minimum of REGION=4096K.

4. Add a VIAIN DD statement that designates the features and options to be used during analysis. This is the format:

```
//VIAIN    DD *
* ANALYZE FEATURES:
  INS,ST,SD,SDX,SDR,EN
/*
```

This step can also be done manually by modifying the COBOL parameter string to include this ESW parameter:

```
VPARM=(vopt,vopt,vopt...)
```

Where *vopt* can be one of these:

Parameter	Description
EN	Encore analysis (no COBOL compile)
INS	Insight only analysis (no COBOL compile)
ST	SmartTest only analysis (no Extended analysis)

Parameter	Description
SD	SmartDoc analysis
SDX	Extended SmartDoc analysis
SDR	SmartDoc report generation
[analyze parms]	Valid analyze parameters (using the standard analyze options)
CMPL	COBOL compile (forces a COBOL compile and an analysis to be executed by the JCL)
NOCMPL	Suppress the COBOL compile (JCL bypasses the compile and executes only an analyze job)
(NO)SYSPRINT	Create separate compiler output file
(NO)VIADCOMP	Create SmartDoc intermediate compiler output file. The intermediate compiler output file is used to produce the SmartDoc Compiler Output.

Note:

If no ESW feature is specified (i.e., INS, ST, SD, SDX, SDR, or EN), all ESW processing is suppressed. This means the procedure executes a compile as it did before.

Figure 160 and Figure 161 on page 448 show examples of compile and analyze JCL before and after submitting the analyze job.

Figure 160 • Compile/Analyze JCL Before Modifications

```
// ASG  JOB  (ASG),'PANVALET COMPILE'
//*ROUTE PRINT DEST
//*       PANVALET EXTRACT
//*
//PANEXT     EXEC PGM=PAN#1,REGION=256K
//PANDD1     DD DSN=ASG.COBOL.PANLIB,DISP=SHR
//PANDD2     DD DSN=&&COBIN,UNIT=SYSDA,SPACE=(CYL, (1,1)),
//                 DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
++WRITE WORK,VIARDEMO
/*
//*
//*    COBOL COMPILE
//*
//COBCOMP    EXEC PGM=IKFCBL00,REGION=1024K,COND=(8,LT,PANEXT),
//                 PARM='SIZE=512K,BUF=128K,LANGLVL(2),LIB,DYNAM'
//STEPLIB    DD DSN=SYS1.VSCOBOL.COMPILER,DISP=SHR
//SYSIN      DD DSN=&&COBIN,DISP=(OLD,DELETE)
//SYSLIB     DD DSN=ASG.COBOL.COPYLIB,DISP=SHR
//SYSLIN     DD DSN=&&LINKLIN,UNIT=SYSDA,SPACE=(CYL, (1,1)),

DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLOKSIZE=3120)
//SYSPRINT   DD SYSOUT=*,DCB=(RECBM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT1     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT2     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT3     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT4     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT5     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//*
//*    LINK EDIT
//*
//LINKED     EXEC PGM=IEWL,REGION=1024K,COND=(8,LT,COBCOMP)
//SYSLIB     DD DSN=SYS1.VSCOBOL.COBLIN,DISP=SHR
//SYSLMOD    DD DSN=ASG.BAR.LOAD,DISP=OLD
//SYSPRINT   DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT1     DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSLIN     DD DSN=&LINKIN,DISP=(OLD,DELETE)
//           DD *
 NAME VIARDEMO (R )
/*
```

Figure 161 is an example of the compile JCL, shown in Figure 160 on page 447, as it would appear after the compile/analyze JCL has been generated according to the rules in this section. Statements that have been added or modified are tagged to the right with ASG NEW and ASG MOD.

Figure 161 • Compile/Analyze JCL After Modifications

```
// ASG  JOB  (ASG),'PANVALET COMPILE'
/*ROUTE PRINT DEST
//*********************************************************          ASG NEW
//*                                                       *          ASG NEW
//* THIS JCL HAS BEEN MODIFIED BY THE ASG ANALYZE         *          ASG NEW
//* SUBMIT FACILITY, WHICH CONVERTS COMPILE JCL INTO      *          ASG NEW
//* COMPILE AND ANALYZE JCL. NEW OR MODIFIED LINES        *          ASG NEW
//* CONTAIN 'ASG' IN COLUMNS 74 THROUGH 76                *          ASG NEW
//*                                                       *          ASG NEW
//*********************************************************          ASG NEW
//*
//*
//*      PANVALET EXTRACT
//*
//PANEXT      EXEC PGM=PAN#1,REGION=256K
//PANDD1      DD DSN=ASG.COBOL.PANLIB,DISP=SHR
//PANDD2      DD DSN=&&COBIN,UNIT=SYSDA,SPACE=(CYL, (1,1)),
//                 DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
++WRITE WORK,VIARDEMO
/*
//*
//*    COBOL COMPILE
//*
//COBCOMP     EXEC PGM=VIACOBVS,REGION=1024K,COND=(8,LT,PANEXT),          ASG MOD
//                 PARM='SIZE=512K,BUF=128K,LANGLVL(2),LIB,DYNAM'
//STEPLIB     DD DSN=SYS1.VSCOBOL.COMPILER,DISP=SHR
//            DD DSN=ASG.VIACENxx.LOADLIB,DISP=SHR                        ASG NEW
//SYSIN       DD DSN=&&COBIN,DISP=(OLD,DELETE)
//SYSLIB      DD DSN=ASG.COBOL.COPYLIB,DISP=SHR
//SYSLIN      DD DSN=&&LINKLIN,UNIT=SYSDA,SPACE=(CYL,(1,1)),
                 DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLOKSIZE=3120)
//SYSPRINT    DD SYSOUT=*,DCB=(RECBM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT1      DD UNIT=SYSDA,SPACE=(CYL, (5,5))                            ASG MOD
//SYSUT2      DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT3      DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT4      DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//SYSUT5      DD UNIT=SYSDA,SPACE=(CYL, (1,1))
//VIARDTOC    DD SYSOUT=*                                                 ASG NEW
```

Analyze Summary Report

Information about the analyzed program is placed in the AKR when the analyze job completes. A summary report of the run-time statistics and diagnostic messages is also produced. This report varies depending on whether the SOURCE or NOSOURCE option was specified when the analyze job was submitted.

Figure 162 and Figure 163 on page 450 shows the Analyze Summary Report with the SOURCE option used. The information shown on this summary is described below.

Figure 162 • Analyze Summary Report

```
                    (A)
00001  000100  IDENTIFICATION DIVISION.                                    00010000
00002  000200  PROGRAM-ID.    VIARDEMO.                                   00020000
00003  000300  AUTHOR.        WRITTEN BY ASG IN LANG LEVEL 2.             00030000
00004  000400*                                                            00040000
00005  000500  ENVIRONMENT    DIVISION.                                   00050000
00006  000600  INPUT-OUTPUT SECTION.                                      00060000
00007  000700  FILE-CONTROL.                                              00070000
00008  000800      SELECT  INFILE1  ASSIGN  TO  UT-S-INFILE1.             00080000
00009  000900      SELECT  INFILE2  ASSIGN  TO  UT-S-INFILE2.             00090000
00010  001000      SELECT  INFILE3  ASSIGN  TO  UT-S-INFILE3.             00100000

*STATISTICS*   SOURCE  RECORDS = 466    DATA DIVISION STATEMENTS = 120 PROCEDURE
DIVISION STATEMENTS - 220
*OPTIONS IN EFFECT*   SIZE  =  1048576,   BUF  =  262144,   LINECNT  =  54,   SPACE1,
FLAGW,   SEQ
*OPTIONS IN EFFECT*   SOURCE,  DMAP,  PMAP,  NOCLIST,  SUPMAT,  NOXREF,  NOSXREF,  LOAD,
NODECK
*OPTIONS IN EFFECT*   APOST,  NOTRUNC,  NOFLOW,  NOTERM,  NONUM,  NOBATCH,  NONAME,
COMPILE=0
*OPTIONS IN EFFECT*   NOSTATE,  RESIDENT,  DYNAM,  LIB,  NOSYNTAX,  NOOPTIMZE,  NOSYMDMP
*OPTIONS IN EFFECT*   NOTEST,  VERB,  ZWB,  SYST,  NOENDJOB,  NOMIGR,  NOLVL,  DUMP,
NOADV
*OPTIONS IN EFFECT*   NOLST,  NOFDECK,  NOCDECK,  LCOL1,  L120, NOFDECK,  NOCDECK, LCOL1
*OPTIONS IN EFFECT*   L120,  DUMP,  NOADV,  NOPRINT,  NOCOUNT,  NOVBSUM,  NOVBREF,
LANGLVL (1)
                    (B)
ASG1534I PROGRAM   VIASCOPR  STARTED
ASG1519I PROGRAM   VIASCOPR  COMPLETED  WITH  RETURN  CODE  0000
ASG1534I PROGRAM   IKFCBL00  STARTED
ASG1519I PROGRAM   IKFCBL00  COMPLETED  WITH  RETURN  CODE  0000
ASG1534I PROGRAM   VIASSYMB  STARTED
ASG1519I PROGRAM   VIASSYMB  COMPLETED  WITH  RETURN  CODE  0000
ASG1534I PROGRAM   VIASANLZ  STARTED
ASG1519I PROGRAM   VIASANLZ  COMPLETED  WITH  RETURN  CODE  0000
ASG1534I PROGRAM   VIADBTCH  STARTED
ASG1519I PROGRAM   VIADBTCH  COMPLETED  WITH  RETURN  CODE  0000
ASG1525I THE PRODUCT LEVEL FOR ASG-CENTER-OS(XA) Rx.x IS 000.
ASG1435I ASG-CENTER-OS(XA) Rx.x LVL000 - SUMMARY REPORT - PROGRAM=VIARDEMO
ASG1399I OPTIONS IN EFFECT ARE: SOURCE, NODMAP, NOPMAP
ASG1394I SUMMARY OF OS/VS COBOL SYMBOLS EXTRACTED FROM VIARDEMO.
ASG1395I        99 DATA NAME SYMBOLS PROCESSED.
ASG1396I        33 PROCEDURE SYMBOLS PROCESSED.
ASG1397I        131 TOTAL SYMBOLS.
ASG1398I        199 VERBS PROCESSED.
ASG1436I DIAGNOSTICS: 0 TOITAL - 0 WARNING, 0 ERROR, 0 SEVERE, 0 CATASTROPHE
ASDG1437I       ASG-CENTER-OS(XA) Rx.x LVL000 - END OF SYMBOL EXTRACTION FOR VIARDEMO
```

Analyze Summary Report Fields	
Field	**Description**
(A)	A complete listing of the program is produced and shows statement numbers generated by the analyze job in the first six columns. These numbers are referenced in diagnostic messages. These notations can also appear on the source listing: C Statement was inserted with a COPY statement. ** Original source statement number is out of sequence. I Statement was inserted with an INSERT statement.
(B)	This portion of the Analyze Summary is the report from the ESW monitor facility. The job steps that were executed by the monitor facility are listed along with the return codes produced.

Figure 163 • Figure 162 Analyze Summary Report (continued)

```
                    (C)
ASG-CENTER-OS(XA)       Rx.x  LVL000                        PROGRAM: VIARDEMO     DD-MM-
YYYY   HH:MM:SS       PAGE 1
                    (D)
   LINE    ERROR   MESSAGE

         ASG0237I    131  SYMBOLS  PROCESSED.
         ASG0238I    131  SYMBOLS MATCHED.
         ASG0240I    199  VERBS PROCESSED.
                    (E)
DIAGNOSTICS LINES:  0  TOTAL  -  - WARINGS,  0  CONDITIONALS,  0  ERRORS,  0 DISASTERS
                    (F)
SOURCE LINES:         466 TOTAL  -  120  DATA  DIVISION  STATEMENTS,  220  PROCEDURE
                                  DIVISION STATEMENTS
                    (G)
PARAMETERS PASSED:    NOCOBOLII, LANGLVL (1)
                    (H)
OPTIONS IN EFFECT     BUFMAXX=2000K, FEATURES=(ENCORE), FLAG(W)
                      LINECNT=60,  NORECUR,  NOSEQ,  NOSOURCE,  SPACE1,  LANGLVL(1).
                    (I)
ENTRY POINTS:         VIARDEMO
                    (J)
EXTRNAL CALLS:        VIASUB
                    (K)
END  OF  PROCESSING:  DD-MMM-YYY        HH:MM:SS
```

Analyze Summary Report Fields (continued)	
Field	**Description**
(C)	The Center (Analyze) release and product level is shown along with the date and time the analysis was performed.
(D)	LINE and ERROR MESSAGE - This information is shown only if there were error conditions encountered. If so, this area lists the line number on which the error occurred and the error message.
(E)	DIAGNOSTICS LINES - Indicates the total number of messages issued with subtotals for each type.
(F)	SOURCE LINES - Indicates the number of source lines in the program. The number of statements within the DATA DIVISION and PROCEDURE DIVISION are also shown. Each level number is counted as one statement in the DATA DIVISION. Each verb is counted as one statement in the PROCEDURE DIVISION.
(G)	PARAMETERS PASSED - Lists all of the analyze options specified for this analyze job.
(H)	OPTIONS IN EFFECT - Lists all options in effect, including default and user-specified options.
(I)	ENTRY POINTS - Lists the entry points in this program.
(J)	EXTERNAL CALLS - Lists the programs that this program CALLs.
(K)	END OF PROCESSING - Lists the day, month, year, and time the analyze job completed. This date and time also displays in the online AKR statistics.

Adding Analyze Facilities to a Standard Compile Mechanism

CLIST Compile Mechanism

The Analyze Submit Facility can be incorporated into many existing in-house compile mechanisms. This provides you with access to ESW analysis without the complications of learning a new or different compile sequence.

Most ISPF Dialogs that use combinations of panels, CLISTs and the TSO SUBMIT command are easily adapted to invoke the Analyze Submit facility, as shown in Figure 164.

Figure 164 • Adapting ISPF Dialog to Invoke the Analyze Submit Facility

The first step is to add a line in a panel to query whether a ASG-ESW Analysis should be performed. The Compile panel can be used for this purpose. The new line in the panel definition may look like:

```
*     ASG ANALYZE                ==➔ _Z+        (%Y+OR%N+)
```

The following line should be added to the) INIT section of the panel definition:

```
.ZVARS = ' (VSVANLYZ)'
```

If there is already an assignment to .ZVARS, VSVANLYZ should be added to the list in the appropriate Place. This is assuming the VSVANLYZ variable will be used to record the response to this question. If another variable name is to be used, the description below would need to change so it corresponds.

It may be appropriate to perform a 'VGET (VSVANLYZ) PROFILE' before the panel is displayed, and a 'VPUT (VSVANLYZ) PROFILE' after it is displayed.

The CLIST should then be changed to invoke VIASUBDS instead of SUBMIT, if the VSVANLYZ variable has a Y value. Assume the CLIST contains the following code:

```
IF  &E = Y THEN DO
    WRITE EDITED JCL WILL NOT BE SUBMITTED BY CLIST
    ISPEXEC EDIT DATASET (&ZUSER..CNTL(MEMBERX))
    END
IF  &E ¬= Y THEN DO
    SUBMIT &ZUSER..CNTL(MEMBERX)
    END
```

This should be changed to read as follows:

```
IF  &VSVANLYZ = Y THEN DO
    IF &E = Y THEN DO
      WRITE EDITED JCL WILL NOT BE SUBMITTED BY CLIST
         %VIASUBDS &ZUSER..CNTL(MEMBERX) EDIT
      END
    ELSE DO
         %VIASUBDS &ZUSER..CNTL(MEMBERX)
      END
    END
ELSE DO
    IF &E = Y THEN DO
      SUBMIT &ZUSER..CNTL(MEMBERX)
    END
```

Many JCL skeleton generators can have the required Analyze features easily imbedded. The JCL modifications typically done by the ESW JCL converter can be added to existing skeletons to create JCL that executes an ESW analysis. Use the Automatic JCL Modifications discussed in this section as a guide for making these additions.

ISPF Compile Mechanism

The Analyze Submit facility can also be installed to emulate the standard ISPF Compile Option, (Option 5), by adding this line to the appropriate ISPF menu panel:

```
V5, 'CMD(%VIASISP5 ISRJPA) NOCHECK'
```

This option functions exactly like the ISPF Compile Option except that the resulting JCL is not submitted directly. Instead, it is passed to the Analyze Submit facility. Review the VIASISP5 CLIST for other required modifications.

If emulation of the ISPF Compile Option is not sufficient for your site, the Analyze Submit facility can also be configured to replace the ISPF option directly. Contact ASG Customer Support for details.

Analyze Options

Use Analyze options to override default options for the Analyze job that were established when ESW products were installed.

The Analyze job uses many of the same options as the IBM COBOL II compilers. These compile-time options are available to control the output format and to describe COBOL options.

To override the installation options, enter the desired options on the File - Analyze Submit pop-up (see Figure 16 on page 39). Separate the options with a comma (,).

All analyze options are now verified before submission to eliminate erroneously entered parameters. If an option is not recognized, the product provides an opportunity to modify the option or proceed with the application analyze job submission.

Options that accept program names as parameters, with the exception of PROGRAM, accept wildcard characters. The asterisk (*) represents zero or more characters.

The question mark (?) represents a single character. For example:

Parameter	Description
DBA*	All programs that begin with DBA and end with any number of other characters.
D?A*	All programs that begin with D followed by any one character, followed by A, then followed by any number of other characters.
DBA???	All programs that begin with DBA and end with any three characters.

The analyze options are summarized in this section. In this summary, abbreviations are shown in uppercase (abbreviations comply with compiler standards). The Analyze Summary Report printed at the end of each analyze job lists the actual options in effect, and the options that were passed to it (the override options).

Buffers

```
BUF(nnnnnK)
BUF=nnnnnK
```

where:

BUF is used only as an override. The amount of main storage allocated to buffers and internal tables is dynamically allocated. Use BUF if an override is necessary. The minimum BUF value is 20K; the maximum is 20000K.

nnnnn is a number from 20 to 20000.

COBOL Level

```
COBOL370
COBOLII
COB2R3
COBOLMVSVM
NOCOBOLII
OS390
```

where:

COBOL370 overrides the LANGLVL option and processes the input program as COBOL/370.

COBOLII overrides the LANGLVL option and processes the input program as COBOL II.

COB2R3 overrides the LANGLVL option and processes the input program as COBOL II Release 3.0 or Release 3.1.

COBOLMVSVM overrides the LANGLVL option and processes the input program as COBOL for MVS and VM.

OS390 overrides the LANGLVL option and processes the input program as COBOL for OS/390.

The default is based on the compiler being used and is determined automatically by the submit process.

Note:

The Encore Extract facility does not support extracting from COBOL II Release 3 modules containing subprograms.

DB2 Load Library

```
DB2LIB=xxxxxx.xxxxx.xxxxxx
```

where:

DB2LIB specifies the load library that is used to invoke the DB2 pre-processor at your site.

xxxxxx.xxxxx.xxxxxx is the dataset name of the DB2 load library.

DB2 Application Plan

```
DB2PLAN=xxxxxxxx
```

where:

DB2PLAN specifies the ESW application plan that was created at installation time by the VIASBIND job. You can use DB2PLAN to override the default plan name.

xxxxxxxx is the name of the ASG application plan.

Dynamic CALLs

```
DYNcall
NODYNcall
```

The minimum abbreviation is DYN or NODYN.

where:

DYNCALL specifies whether Encore functions use the variable name in dynamic calls as the name of the called program.

NODYNCALL specifies that dynamic calls are not processed by the analyze and information for them is not available to ESW product functions. The default is DYNCALL.

For example, the analyze process for this code proceeds differently, depending on whether DYNCALL is specified:

```
77  MYPROG   PIC X(8).
CALL MYPROG USING PARM1, PARM2.
```

In this example, if DYNCALL is in effect, the analyze process assumes that the program being called is *MYPROG*, regardless of the data value that MYPROG contains at run-time. The analyze process looks up the analysis results of MYPROG in the AKR to determine whether PARM1 and PARM2 are used or modified.

If NODYNCALL is in effect for this example, the analyze process assumes that the program being called could be anything, and treats both PARM1 and PARM2 as used and modified on the call statement.

Note:

The DYNCALL option is unrelated to the COBOL compiler option DYNAM.

Flag Messages

```
flag
fLAGEW
fLAGE
fLAG(x)
```

where:

FLAG specifies the types of messages to be listed for the Analyze job.

FLAGEW indicates all warning and diagnostic messages are listed. This is the default.

FLAGE indicates diagnostic messages are listed; all other messages are suppressed.

FLAG(x) indicates all messages of the specified level or above are listed.

x is a specified message level. The minimum abbreviations are LAGW, LAGE, and LAG(*x*). These are the message levels:

- I - Informational
- W - Warning
- E - Error
- S - Severe
- U - Unrecoverable

Note:

Some informational messages are produced regardless of the flag setting.

Input

```
Input(x,x,...x)
Input=x
NOInput(x,x,...x)
NOInput=x
```

where:

INPUT lists the CALLed programs that contain INPUT statements. When commands that search for INPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

x is a program name. The minimum abbreviation is I or NOI, with at least one program name. Wildcard characters are allowed in the program name.

NOINPUT overrides the installation default list of CALLed programs that contain INPUT statements. The specified programs are deleted from the default list.

IO

```
IO(x,x,...x)
IO=x
NOIO
```

where:

IO lists the CALLed programs that contain INPUT and OUTPUT statements. When commands that search for INPUT and OUTPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

x is a program name. Wildcard characters are allowed in the program name.

NOIO overrides the installation default list of CALLed programs that contain INPUT and OUTPUT statements. The specified programs are deleted from the default list.

Language Level

```
LANGLVL(1)
LANGLVL(2)
```

where:

LANGLVL specifies whether to use the 1968 or 1974 American National Standard COBOL definitions when analyzing source elements with meanings that have changed.

LANGLVL(1) indicates the 1968 standard is used.

LANGLVL(2) indicates the 1974 standard (X3.23-1974) is used.

The default is based on the compiler being used and is determined automatically by the submit process.

Line Count

```
lineCNT = nn
lineCNT = 60
```

where:

LINECNT indicates the number of lines to be printed on each page of the source listing. The default is 60.

nn is a number from 01 to 99. The minimum abbreviation is CNT, with a line count number.

Main

```
MAIN
```

Where *MAIN* is used only as an override. The EXIT PROGRAM statements in COBOL programs are treated as GOBACKs by the Analyze job, because the program is treated as a CALLed subprogram. If the program is the main program, using the MAIN option treats the EXIT program as a fall-through.

Maximum Number of Errors

```
MBRERCNT=nnnn
```

where:

MBRERCNT specifies the maximum number of analysis errors allowed for a member during an analyze job. If this number of errors is exceeded, the analyze terminates processing for that member. The number specified must be between 1 and 4000. The default is set at installation.

nnnn is the number of analysis errors allowed during an analyze job.

Output

```
Output(x,x,...x)
Output=x
NOOutput(x,x,...x)
NOOutput=x
```

where:

OUTPUT lists the CALLed programs that contain OUTPUT statements. When commands that search for OUTPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

x is a program name. The minimum abbreviation is O or NOO, with at least one program name. Wildcard characters are allowed in the program name.

NOOutput overrides the installation default list of CALLed programs that contain OUTPUT statements. The specified programs are deleted from the default list.

Program

```
PROgram(xxxxxxxxxx)
PGM=xxxxxxxxxx
```

Where *xxxxxxxxxx* is a program name up to 10 characters. The minimum abbreviation is PRO, with a program name.

Analyzed programs are stored in the AKR and identified by the program name coded in the PROGRAM-ID statement. PROGRAM is used to override the name coded in the PROGRAM-ID statement.

Recursion

```
RECur
NORECur
```

The minimum abbreviation is REC or NOREC.

where:

RECUR specifies whether the recursion report should be included in the Analyze Summary Report.

If RECUR is specified and recursion is not found, a message is issued that indicates no recursion was detected. If RECUR is specified and recursion is found, a message is issued and the recursive code is printed on the report.

The default is NORECUR.

Return

```
RETurn(x,x,...x)
RETurn=x
NORETurn(x,x,...x)
NORETurn=x
```

where:

RETURN overrides the installation list of programs or entry points that do not return when CALLed. The system defaults are overridden by listing the desired programs or entry points that are to return when CALLed.

x is a program name. The minimum abbreviation is RET or NORET, with at least one program name. Wildcard characters are allowed in the program name.

NORETURN lists the additional programs or entry points that are not to return when CALLed. When any of these programs are CALLed by the program being analyzed, they are treated as non-returning CALLs. The specified programs are in addition to the system defaults for programs that do not return when CALLed.

Sequence

```
SEQ
NOSEQ
```

Where *SEQ* specifies whether the analyze job checks the source module statement number sequence. A warning message is printed if the statements are not in sequence. If the Source option is also specified, a flag (**) is placed between the Analyze job sequence numbers and the source sequence numbers. The default is SEQ.

Source

```
SOUrce
NOSOUrce
```

The minimum abbreviation is SOU or NOSOU.

Where SOURCE specifies whether the source program is to be listed. The SOURCE option is specified if a full program listing is desired at Analyze time.

The default is NOSOURCE.

Spacing

```
spACE1
spACE2
spACE3
```

The minimum abbreviations are ACE1, ACE2, and ACE3.

where:

SPACE specifies the spacing for the source listing that is generated when the SOURCE option is used.

SPACE1 specifies single spacing. This is the default.

SPACE2 specifies double spacing; that is, one blank line displays between every source line.

SPACE3 specifies triple spacing; that is, two blank lines appear between every source line.

SQL Authorization ID

```
SQLID=nnnnnnnn
```

where:

SQLID(*nnnnnnnn,nnnnnnnn,nnnnnnnn*)

SQLID specifies the authorization ID or owner that is used by the analyze process to qualify unqualified table and view references in your program.

nnnnnnnn is an 8-character name.

DB2 Subsystem

```
SUBSYS=xxxx
```

where:

SUBSYS specifies the subsystem or location that designates the DBMS in which the tables accessed by a specified program are stored. SUBSYS overrides the name provided at installation time.

xxxx is the name of the subsystem or location of the DBMS.

Anomaly Reports

```
VIARERPT
VIARERPTAN
```

where:

VIARERPT specifies that you want the analysis process to stop after the report is generated.

VIARERPTAN specifies that you want the analysis process to continue after the report is generated and save the information in the AKR.

Anomaly Facility

```
VIAANOMF
```

where:

VIAANOMF instructs the analyzer to run the Anomaly Facility. This is the batch method for anomaly repair. This parameter can be followed by these additional anomaly report parameters:

- PARM(user parm)
- ANOMALY
- COMMENT(YES or NO)
- DSN
- MEM
- HEADER(YES or NO)

See "Anomaly Facility" on page 61 for more information about using the Anomaly Facility.

Live Exit

```
XLIVE
```

Where *XLIVE* is used as an override and should only be used with programs that contain live exits. Live exits are exits from perform ranges that are left dangling by imbedded PERFORMs or GO TOs in the original performed paragraph.

If XLIVE is not used, code that is unprocessed because of the live exit is ignored. If XLIVE is used, unprocessed code is saved.

Note:

Using XLIVE can significantly increase resource usage.

Memory

```
XMEM
```

Where *XMEM* is used only as an override. If a program is extremely large (for example, 30,000 source lines) and there is insufficient memory, increase the region space. If there is still insufficient memory, enter the XMEM option. This results in more disk I/O and additional CPU usage, but less memory consumption.

13 AKR Utilities

This chapter describes Encore's AKR Utilities functions and contains these sections:

Section	Page
Introduction to AKR Management	465
AKR Structure	466
Online AKR Utilities	466
Batch AKR Utilities	467
Command Format	469
Command Syntax	470
Batch AKR Reports	481

Introduction to AKR Management

The AKR is the repository for all information used by the ESW family of components. Analyzed programs are stored in the AKR for use by ESW. A single AKR can be defined for use by all ESW users, or separate AKRs can be defined for use by departments, groups, or individual users.

ESW provides both online and batch utilities for managing the AKR. This chapter first explains the online AKR utilities (see page 466) then the batch AKR utilities (see page 467).

AKR Structure

An AKR is organized as a BDAM or VSAM file. An AKR can be defined to be shared by all users, or multiple AKRs can be defined for use by departments, groups, or individuals.

See the *ASG-Center Installation Guide* for additional information about the AKR.

Online AKR Utilities

These are some of the online AKR utility pop-ups:

- The File - AKR Utility pop-up (see Figure 17 on page 43) used to rename or delete a program, or to display the AKR Directory.
- The File - AKR Directory pop-up (see Figure 18 on page 45) used to view all programs in an AKR. This pop-up can be used to rename or delete a group of programs. Statistics about the AKR are also shown on this pop-up.
- The File - AKR Allocate/Expand pop-up (see Figure 19 on page 49) used to allocate a new AKR or to expand an existing AKR.

Note:

When you use the Allocate/Expand utility, the default AKR organization type is applied. For example, if your site's default AKR type is sequential, any new AKR allocated is created as a sequential file, and any non-sequential AKR expanded is reorganized as a sequential file.

See online help for examples of each of these pop-ups.

Batch AKR Utilities

The Batch AKR Utility is used to maintain the AKR without using ISPF. These are the commands available in the batch utility:

Command	Function
ANLZSTAT	Provides a summary of the composition and analysis for one or more AKRs.
COMMENTS	Includes comments with the command.
CONVERT	Converts selected members from a previous ESW product release level to the current release level.
COPY	Copies selected members from one AKR to another.
DELETE	Deletes selected members from the AKR.
EXPORT	Creates metrics and function point CDF files. **Note:** Export is available only for Recap users.
HELP	Prints the AKR Utility Help Report.
INT	Formats a previously-defined dataset into an AKR format.
MOVE	Copies selected members from one AKR to another and deletes them from the original AKR.
PRINT	Prints AKR directory information, or COBOL source listings for selected AKR members.
PUNCH	Produces a file that contains the AKR directory information, or the COBOL source code for selected AKR members.

Job Control Statements

These JCL statements are used by the Batch AKR Utility. The VIAAKRIN and VIAAKROT DD statements describe AKRs that are used for AKR Utility processing. The VIASYSIN DD control cards consist of the necessary Batch AKR commands described in "Command Format" on page 469. See the description for each command to determine which DD statements are affected.

Figure 165 • Batch AKR JCL Statements

```
//UTILITY  EXEC  PGM=VIASAKRU,REGION=3000K,PARM=''
//STEPLIB  DD    DISP=SHR,DSN=(ASG load library)
//VIAAKRIN DD    DISP=SHR,DSN=(Input AKR)
//VIAAKROT DD    DISP=SHR,DSN=(Output AKR)
//VIAPRINT DD    SYSOUT=A (Print file description)
//VIAPUNCH DD    SYSOUT=B (Punch file description)
//VIALOG   DD    SYSOUT=A (Log file)
//VIASYSIN DD    *
<control cards>
//
```

Control Cards

Commands are passed to the Batch AKR Utility with the control cards following the VIASYSIN DD statement. Control cards must conform to these standards:

- Command information must be contained in columns 1-72 of the control card.
- Only one command can be entered on each control card.
- Only one control card may be used per command.

All control cards with command disposition and command summaries are printed to the VIALOG AKR Utility Log file. Blank control cards are ignored.

Command Format

Commands that use member names accept special characters to signify generic names. An asterisk (*) represents zero or more characters. A question mark (?) represents one character. For example:

Parameter	Description
DBA*	This command example specifies all members that begin with DBA and that end with any other characters.
D?A*	This command example specifies all members that begin with a D followed by one character, then followed by an A, which can be followed by any other character(s).
DBA???	This command example specifies all members that begin with DBA and end with any three characters.

The LASTUSED parameter is used to provide the selection criteria for several commands. The specified number represents the number of days since the member was last referenced online, or the date the member was analyzed if it has not been referenced.

The REPLACE parameter is used to specify that members are to be replaced on the output AKR. The NOREPLACE parameter is used to prevent members from being replaced on the output AKR. NOREPLACE is the default value.

Command Syntax

Each Batch AKR Utility command is described in this section. These notational conventions and symbols are used to describe command syntax:

Item	Description
ABBREViations	Command abbreviations are shown in uppercase letters; lowercase letters in the command are optional.
lowercase	Lowercase values indicate user-supplied variable information.
Underline	The default value of an operand is underlined.
|	Vertical Bar - A vertical bar separates synonymous commands or operands.
——————▶	Right and left ending arrows indicate the end of the command syntax.
—— required ——	An operand or keyword appearing on the main command line is required.
┬──┬──┬ └─optional─┘	An operand or keyword appearing below the main command line is optional.
┬──┬──┬ ├─choice1─┤ └─choice2─┘	Stacked operands below the main line show a choice of one optional item.

Comments

Function

Includes a comment with the commands.

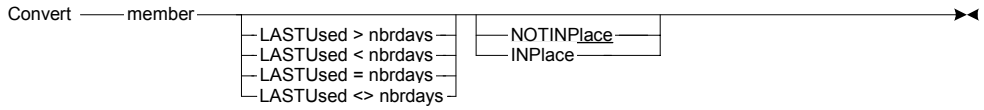
Operands

Field	Description
comment	User-supplied text.

Usage Notes

Blank control cards are ignored.

CONVERT Batch AKR Command



Function

Converts selected members that were analyzed by a prior release of ESW products to the current release level.

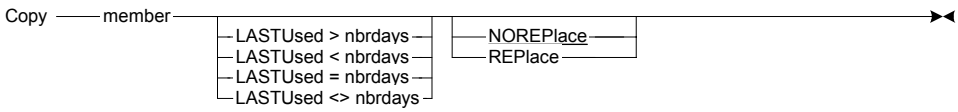
Operands

Field	Description
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 469.
NOTINPlace	Specifies that a member with the same name on the receiving AKR should be replaced with the member from the sending AKR.
INPlace	Specifies that a member is to be converted and kept within the AKR named in the VIAAKRIN DD statement. This option should be used with caution. Consult your systems programmer or ASG Customer Support.

Usage Notes

Members are copied from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement, as described in "Job Control Statements" on page 468.

COPY Batch AKR Command



Function

Copies selected members from one AKR to another.

Operands

Field	Description
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 469.
NOREPlace	Prevents existing members from being replaced by members with the same name. This is the default.
REPlace	Replaces members that have the same name on the receiving AKR.

Usage Notes

Members are copied from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement.

DELETE Batch AKR Command

```
DELete ──member──┬──────────────────────────────────────────────►◄
                 ├─LASTUsed > nbrdays─┤
                 ├─LASTUsed < nbrdays─┤
                 ├─LASTUsed = nbrdays─┤
                 └─LASTUsed <> nbrdays┘
```

Function

Erases selected members from the AKR.

Operands

Field	Description
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 469.

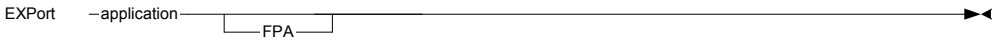
Usage Notes

Note:

Members are deleted from the AKR specified in the VIAAKRIN DD statement.

Members that begin with VIA cannot be deleted using this command. All ESW test members begin with VIA. If these members must be deleted, use the online AKR Utility function described in "Online AKR Utilities" on page 466.

EXPORT Batch AKR Command



Function

Creates metrics and function point CDF files.

Operands

Field	Description
application	Can be a specific or generic name as described in "Command Format" on page 469.
FPA	Generates only function point information. If FPA is not specified, both metrics and function point information are generated.

Usage Notes

EXPORT is available only for Recap users.

HELP Batch AKR Command

```
HELP | ? ──────────────────────────────────────────────────────────►◄
```

Function

Prints a description of the Batch AKR Utility and the commands that can be used.

Operands

None.

Usage Notes

- A question mark (?) can be used as an alternate command.
- The HELP command has no operands.

The Help Report is printed to the SYSOUT specified in the VIAPRINT DD statement.

INIT Batch AKR Command

Function

Initializes a new AKR. This internal command is used by the online AKR Utility Allocation function. See "Online AKR Utilities" on page 466 for additional information.

Operand

Field	Description
DSname(dsname)	Used to specify the dataset name for the new AKR.

Usage Notes

The AKR dataset to be initialized must be created prior to the initialization. The AKR that is initialized can be described in the VIAAKRIN DD statement. The VIAAKRIN DD statement is ignored if the DSNAME parameter is specified.

MOVE Batch AKR Command

```
MOve ──member──┬──────────────────────────┬──┬──────────────┬──┬──────────────┬──►◄
               ├─LASTUsed > nbrdays───────┤  ├──NOREPlace───┤  │              │
               ├─LASTUsed < nbrdays───────┤  └──REPlace──────┘  │              │
               ├─LASTUsed = nbrdays───────┤                     │              │
               └─LASTUsed <> nbrdays──────┘                     │              │
```

Function

Moves selected members from one AKR to another. Specified members are copied to the receiving AKR and erased from the sending AKR.

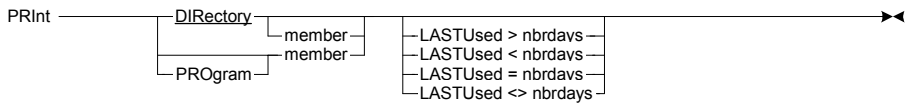
Operands

Field	Description
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 469.
NOREPlace	Prevents existing members from being replaced by members with the same name. This is the default.
REPlace	Replaces members that have the same name on the receiving AKR.

Usage Notes

Members are moved from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement.

PRINT Batch AKR Command



Function

Prints AKR directory information for the entire AKR or for a specified member, or the source code for a specified member.

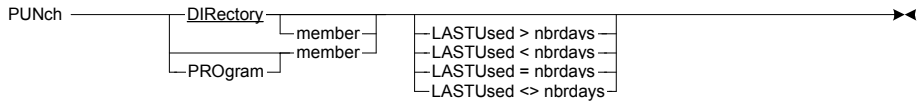
Operands

Field	Description
blank	If the PRINT batch AKR command is entered with no operand, the AKR directory information is printed.
DIRectory	Prints AKR directory information. This is the default. If a member is specified, the AKR directory information for that member only is printed.
PROgram	Prints the COBOL source for the specified AKR member. The generated COBOL source listing contains expansions of all COPYBOOKs or INCLUDEs, as well as the results of any source preprocessors such as CICS macro expansion.
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select a member based on the number of days since it was last used, as described in "Command Format" on page 469.

Usage Notes

Directory information or COBOL source is extracted from the AKR specified in the VIAAKRIN DD statement and is printed to the SYSOUT specified in the VIAPRINT DD statement.

PUNCH Batch AKR Command



Function

Produces a file that contains AKR directory information for the entire AKR or a specified member, or the source code for a specified member.

Operands

Field	Description
blank	If the PUNCH batch AKR command is entered with no operand, a file is produced containing the AKR directory information.
DIRectory	Produces a file that contains directory information. This is the default. If a member is specified, the AKR directory information for that member only is printed.
PROgram	Produces a file that contains COBOL source code for the specified AKR member. The generated COBOL source contains expansions of all COPYBOOKs or INCLUDEs, as well as the results of any source preprocessors such as CICS macro expansion.
member	Can be a specific or generic name as described in "Command Format" on page 469.
LASTUsed	Used to select a member based on the number of days since it was last used, as described in "Command Format" on page 469.

Usage Notes

Directory information or COBOL source is extracted from the AKR specified in the VIAAKRIN DD statement and is written to the file specified in the VIAPUNCH DD statement. The file that is produced is in standard IBM IEBUPDTE Utility format. ADD control cards are produced for each logical entity. The NAME parameter contains the member name for COBOL source. The NAME parameter contains AKRDIR`nn` for directory information, where *nn* is a consecutively assigned number.

This is the format of the file produced by the PUNCH DIRECTORY command:

Description	Length	Format
Member name	10	Character
Number of source lines	6	Right justified
Days since last used	4	Right justified
Analyze date	9	DDMMMYYYY
Analyze job name	8	Character
Analyze CPU	4	Character
Analyze product level	8	Character
Last reference date	9	DDMMMYYYY
Last reference user ID	8	Character
Last reference CPU	4	Character

Batch AKR Reports

Examples of these reports are provided in this section:

- AKR Utility Log (see Figure 166 on page 482).
- AKR Utility Directory Report (see Figure 167 on page 483).
- File produced by the AKR Punch Directory File command (see Figure 168 on page 483).

AKR Utility Log

The VIALOG AKR Utility Log, shown in Figure 166, provides a summary of the commands issued to the Batch AKR Utility. This log contains this information:

- Comments
- Commands
- Completion messages
- Short summary of commands processed

The heading includes the ESW product level information, and the date and time the job was executed. Comments are enclosed in a box comprised of asterisks. The second page contains the log summary.

Figure 166 • AKR Utility Log

```
ASG-CENTER-OS  RX.X LVL000       AKR UTILITY LOG              DDMMMYYYY  HH:MM:SS    Page 1

*************************************************************************000140000
*  PRODUCE A REPORT CONTAINING DIRECTORY INFORMATION FOR ALL           *000150000
*       MEMBERS OF ASG-ENCORE.AKR (VIAAKRIN) THAT HAVE                 *000160000
*       NOT BEEN REFERENCED IN THE LAST 7 DAYS.                        *000170000
*************************************************************************000180000
*                                                                        000190000
PRINT DIRECTORY * LASTUSE > 7                                           000200000

  ASG1289I  8  DIRECTORY ENTRIES SUCCESSFULLY PRINTED.


                                                                        000210000
*************************************************************************000220000
*  PRODUCE A REPORT CONTAINING DIRECTORY INFORMATION FOR ALL           *000230000
*       MEMBERS OF.ASG-ENCORE.AKR (VIAAKRIN) THAT HAVE                 *000240000
*       NOT BEEN REFERENCED IN THE LAST 7 DAYS.                        *000250000
*************************************************************************000260000
*                                                                        000270000
PUNCH DIRECTORY * LASTUSE > 7                                           000280000

  ASG1290I  8  DIRECTORY ENTRIES SUCCESSFULLY PUNCHED.


  ASG1314I  *** END OF VIASYSIN ***

ASG-CENTER-OS  RX.X LVL000       AKR UTILITY LOG - SUMMARY    DDMMMYYYY  HH:MM:SS    Page 2

  ASG1301I      8  DIRECTORY ENTRIES PRINTED    0 FAILED.
  ASG1302I      8  DIRECTORY ENTRIES PRINTED.   0 FAILED.

  ASG1315I  *** END OF SUMMARY REPORT ***.
```

AKR Utility Directory Report

The AKR Utility Directory report, shown in Figure 167, lists the results of a PRINT DIRECTORY command and is written to the VIAPRINT DD file. The title line contains the ESW product level information, title, date, and time the job was executed. The report lists the AKR dataset used, the command used to produce the report, and the directory information for the selected members AKR Utility Directory Report.

Figure 167 • AKR Utility - Directory Report Example

```
ASG-CENTER-OS  RX.X LVL000        AKR UTILITY - DIRECTORY       DDMMMYYYY  HH:MM:SS    Page 1

   AKR:  ASG.VIACENnn.AKR
Command:  PRINT  DIRECTORY  *   LASTUSE   >  7

Member        Last  ---------------  ----Analyzed ----------  -- Last Referenced ----
Name    Lines  Use Date     Time     Job   CPU  Level   Date          Time          Job        CPU

 ACTG0018  40   16 DDMMYYYY    HH:MM:SS ASGA CPUA IN030000  DDMMMYYYY    HH:MM:SS     ASG      CPUA
 PYRL0085  17   16 DDMMYYYY    HH:MM:SS ASGA CPUA IN030000  DDMMMYYYY    HH:MM:SS     ASG      CPUA
 PYRL0105  17    8 DDMMYYYY    HH:MM:SS ASGA CPUA IN030000
 SR0005A   493   8 DDMMYYYY    HH:MM:SS ASGA CPUC IN030000
 W550044   66   12 DDMMYYYY    HH:MM:SS ASGA CPUC IN030000  DDMMMYYYY    HH:MM:SS     ASG      CPUA
 XRSCL070  171412 DDMMYYYY    HH:MM:SS ASGA CPUA IN030000  DDMMMYYYY    HH:MM:SS     ASG      CPUA
 XRSCL100  15    9 DDMMYYYY    HH:MM:SS ASGZ CPUC IN030000
 XRSCL200  41   10 DDMMYYYY    HH:MM:SS ASGZ CPUC IN030000  DDMMMYYYY    HH:MM:SS     ASG      CPUA

***  End of Directory Report ***.
```

Punch Directory File

The Punch Directory, shown in Figure 168, is written to the VIAPUNCH DD file when the PUNCH DIRECTORY command is processed. The file is formatted in standard IBM IEBUPDTE Utility format. The first card, `(./ ADD)`, is an IEBUPDTE control card that indicates the cards that follow are to be added to a partitioned dataset specified in the NAME parameter. The cards that follow are in the format described in the PUNCH DIRECTORY command description (see "Usage Notes" on page 481). The last card is an IEBUPDTE control card that indicates the end of the control cards. See the "PUNCH Batch AKR Command" on page 480 for the format of the AKR Punch Directory File.

Figure 168 • AKR Punch Directory File

```
./  ADD NAME=AKRDIR1,LIST=ALL

ACTG0018   40   16DDMMYYYYASGACPUAIN030000DDMMMYYYYVIASOFT   CPUA
ACTG0018   17   16DDMMYYYYASGACPUAIN030000DDMMMYYYYVIASOFT   CPUA
ACTG0018   17    8DDMMYYYYASGACPUAIN030000
ACTG0018  493    8DDMMYYYYASGACPUCIN030000
ACTG0018   66   12DDMMYYYYASGACPUCIN030000DDMMMYYYYVIASOFT   CPUA
ACTG0018 1714   12DDMMYYYYASGACPUAIN030000DDMMMYYYYVIASOFT   CPUA
ACTG0018   15    9DDMMYYYYASGZCPUCIN030000
ACTG0018   41   10DDMMYYYYASGZCPUCIN030000DDMMMYYYYVIASOFT   CPUA
./  ENDUP
```

Allocating and Expanding AKRs without ISPF

The Batch AKR Utility can allocate and expand VSAM AKRs without using ISPF. ESW provides the VIASAKRA JCL shown in Figure 169 to expand an AKR.

Figure 169 • VIASAKRA JCL for a VSAM AKR (1 of 2)

```
// ASG  JOB  ( ),   'ALLOC / INIT AKR'
//*ROUTE  PRINT  XXXXXXX.XXXXXX
//*
//*
*********************************************************************
//*  ASG,  INC.              ASG-CENTER  RX.X          MMM, YYYY     *
//*   *                                                              *
//*   *                                                              *
//*   *  JCL  PROCEDURE TO ALLOCATE AND INITIALIZE A ASG             *
//*   *  APPLICATION  KNOWLEDGE  REPOSITORY  (AKR)                   *
//* ******************************************************************
//*
//VIASIDAP  PROC  SYSOUT='*'       PRINTED OUTPUT MESSAGE CLASS
//           VIASOFT='ASG'         HIGH LEVEL NODE FOR ASG DATA SETS
//           CENTER='VIACENxx'     MIDDLE NODE FOR ASG DATA SETS
//*
//*   ****************************************************
//*   *  DEFINE  A  NEW  ASG  AKR  FILE                   *
//*   ****************************************************
//*
//DEFAKR   EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT  DD SYSOUT=&SYSOUT
//SYSOUT    DD SYSOUT=&SYSOUT
//SYSIN     DD DDNAME=SYSIN
//*
//*   ****************************************************
//*   *  INITIALIZE THE NEWLY ALLOCATED AKR FILE          *
//*   ****************************************************
//*
//INITAKR EXEC PGM=VIASAKRU,REGION=2048K,COND=(0,LT,DEFAKR)
//STEPLIB   DD DSN=&VIASOFT..&CENTER..LOADLIB,DISP=SHR
//VIASYSIN  DD DDNAME=SYSIN
//VIALOG    DD SYSOUT=&SYSOUT
//SYSUDUMP  DD SYSOUT=&SYSOUT
//*
//*   ****************************************************
//*   *  DELETE NEW AKR (ONLY IF INITAKR FAILS)           *
//*   ****************************************************
//*
//DELETE   EXEC PGM=IDCAMS,REGION=512K,
//            COND=(EVEN, (0,LT,DEFAKR), (0, EQ,INITAKR))
//SYSPRINT  DD SYSOUT=&SYSOUT
//SYSOUT    DD SYSOUT=&SYSOUT
//SYSIN     DD DDNAME=SYSIN
//*
//          PEND
//*
//VIASAKRA  EXEC  VIASIDAP
//*
```

To allocate an AKR, replace XX in the NAME(XX), NAME(XX.DATA), and DSNAME(XX) parameters with the name of the AKR to be allocated. Then replace XX(XX) with the allocation units values and quantities for your site.

Figure 170 • VIASAKRA JCL for a VSAM AKR (2 of 2)

```
//DEFAKR.SYSIN  DD  *
 DEFINE  CLUSTER
          (NAME (XX)  -        XX  AKR  NAME  HERE  * *
          XX (XX)              XX  ALLOC  UNITS  AND  QUANTITY  HERE  *
*
          VOLUME (SRT801)
          CONTROLINTERVALSIZE (4096)
          NUMBERED  -
          RECORDSIZE (4089  4089)
          RECOVERY  -
          ERASE  -
          UNIQUE  -
          SHAREOPTIONS (3  3) )  -
       DATA  -
          (NAME (XX.DATA) )    XX  AKR  NAME  HERE  *  *
//*
//INITAKR.VIASYSIN  DD  *
 INIT  DSNAME  (XX)            XX  AKR  NAME  HERE  * *
//*
//DELETE.SYSIN  DD  *
 DELETE  XX                    XX  AKR  NAME  HERE  * *
//*
```

Figure 171, Figure 172 on page 487. and Figure 173 on page 488 show an example of how the VIASAKRX JCL should look.

Figure 171 • VIASAKRX JCL for a VSAM AKR (1 of 3)

```
// ASG  JOB  (  ), 'EXPAND AKR'
//*    INSERT  '/*ROUTE PRINT NODE.USER'  HERE  IF  NEEDED.
//*
//* ***********************************************************************
//*   *  ASG,  INC.                ASG-CENTER  RX.X                MMM, YYYY*
//*   *                                                                    *
//*   *                                                                    *
//*   *  JCL  PROCEDURE TO EXPAND AN EXISTING ASG                          *
//*   *  APPLICATION  KNOWLEDGE  REPOSITORY  (AKR).                        *
//*************************************************** ********************
//*
//VIASAKXP  PROC  SYSOUT='*'                PRINTED OUTPUT MESSAGE CLASS
//               VIASOFT='ASG'              HIGH LEVEL NODE FOR ASG DATA SETS
//               CENTER='VIACENxx'          MIDDLE NODE FOR ASG DATA SETS
//*
//*     ***************************************************
//*     *  DEFINE  A  NEW  ASG  AKR  FILE                  *
//*     ***************************************************
//*
//DEFAKR      EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT      DD SYSOUT=&SYSOUT
//SYSOUT        DD SYSOUT=&SYSOUT
//SYSIN         DD DDNAME=SYSIN
//*
//*     ***************************************************
//*     *  INITIALIZE NEW  DATA  SET  AS  ASG  AKR          *
//*     ***************************************************
//*
//INITAKR     EXEC PGM=VIASAKRU,REGION=2048K,COND=(0,LT,)
//STEPLIB       DD DSN=&VIASOFT..&CENTER..LOADLIB,DISP=SHR
//VIASYSIN      DD DDNAME=SYSIN
//VIALOG        DD SYSOUT=&SYSOUT
//SYSUDUMP      DD SYSOUT=&SYSOUT
//*
//*     ***************************************************
//*     *  COPY  OLD  AKR  TO  NEW  AKR                    *
//*     ***************************************************
//*
//REPRO       EXEC PGM=IDCAMS,REGION=512K,COND=(0,LT)
//SYSPRINT      DD SYSOUT=&SYSOUT
//SYSOUT        DD SYSOUT=&SYSOUT
//SYSIN         DD DDNAME=SYSIN
//*
//*
//*     ***************************************************
//*     *  UPDATE  INTERNAL  SIZE  OF  ASG  AKR             *
//*     ***************************************************
//*
//RESIZE      EXEC PGM=VIASAKRU,REGION=2048K,COND=(0,LT,)
//STEPLIB       DD DSN=&VIASOFT..&CENTER..LOADLIB,DISP=SHR
//VIASYSIN      DD DDNAME=SYSIN
//VIALOG        DD SYSOUT=&SYSOUT
//SYSUDUMP      DD SYSOUT=&SYSOUT
//*
//*     ***************************************************
//*     *  DELETE  OLD  AKR  AND  RENAME  NEW  AKR  TO      *
//*     *       OLD  AKR  NAME                              *
//*     ***************************************************
```

Figure 172 • VIASAKRX JCL for a VSAM AKR (2 of 3)

```
//*     ***************************************************
//*     *  RUN  THIS  STEP  ONLY  IF  ALL  ABOVE  STEPS     *
//*     *       RUN  SUCCESSFULLY                           *
//*     ***************************************************
//*
//TESTCODE    EXEC PGM=IEFBR14, COND=(0,LT,)
//SYSIN       DD DDNAME=SYSIN
//*
//*     ***************************************************
//*     *  DELETE  OLD  AKR  AND  RENAME  NEW  AKR  TO      *
//*     *       OLD  AKR  NAME                              *
//*     ***************************************************
//*
//RENAME      EXEC PGM=IDCAMS,REGION=512K,COND=(0,LT,)
//SYSPRINT    DD SYSOUT=&SYSOUT
//SYSOUT      DD SYSOUT=&SYSOUT
//SYSIN       DD DDNAME=SYSIN
//*
//*     ***************************************************
//*     *  RUN  THIS  SETP  ONLY  IF  ALL  ABOVE  STEPS     *
//*     *       RUN SUCCESSFULLY                            *
//*     ***************************************************
//*
//TESTCODE    EXEC PGM=IEFBR14, COND=(0,LT,)
//SYSIN       DD DDNAME=SYSIN
//*
//*
//*     ***************************************************
//*     *  DELETE  NEW  AKR  ONLY  IF  EXPAND  IS  NOT      *
//*     *       SUCCESSFUL                                  *
//*     ***************************************************
//*
//DELETE      EXEC PGM=IDCAMS,REGION=512K,
//                 COND=(EVEN, (0,LT,DEFAKR), (0, EQ,TESTCODE))
//SYSPRINT    DD   SYSOUT=&SYSOUT
//SYSOUT      DD   SYSOUT=&SYSOUT
//SYSIN       DD   DDNAME=SYSIN
//*
//            PEND
//*
//VIASAKRX    EXEC VIASAKXP
//*
//DEFAKR.SYSIN  DD  *
 DEFINE  CLUSTER
             (NAME (ASG.VIACENXX.AKR.EX)                 /*  AKR  NAME  */  -
             RECORDS (6000)      /*  NEW  ALLOC  UNITS  AND  QUANTITY        */  -
             VOLUME (XXXXXX)          /*  AKR  VOLSER  */  -
             CONTROLINTERVALSIZE (4096)  -
             NUMBERED  -
             RECORDSIZE (4089  4089)
             RECOVERY  -
             ERASE  -
             UNIQUE  -
             SHAREOPTIONS (3  3) )  -
        DATA  -
             (NAME (ASG.VIACENXX.AKR.EX) ) )             /*  AKR  NAME  * * -
//*
```

Figure 173 • VIASAKRX JCL for a VSAM AKR (3 of 3)

```
//INITAKR.VIASYSIN  DD  *
INIT  DSNAME  (ASG.VIACENXX.AKR.EX)                    /*  AKR  NAME  */  -
//*
//*
//REPRO.SYSIN  DD  *
REPRO  INDATASET (ASG.VIACENXX.AKR)                    /*  AKR  NAME  */  -
       OUTDATASET (ASG.VIACENXX.AKR.EX)                /*  AKR NAME  */  -
       REPLACE
//*
//*
//RESIZE.VIASYSIN  DD  *
RESIZE  DSNAME (ASG.VIACENXX.AKR.EX)                   /*  AKR  NAME  */  -
//*
//*
//RENAME.SYSIN  DD  *
DELETE  ASG.VIACENXX.AKR                               /* *  AKR  NAME  HERE  * * /
ALTER   ASG.VIACENXX.AKR.EX                            /*  AKR  NAME  */  -
        NEWNAME (ASG.VIACENXX.AKR)   /*  AKR  NAME  */  -
ALTER   ASG.VIACENXX.AKR.DATA                          /*  AKR  NAME  */  -
        NEWNAME (ASG.VIACENXX.AKR.DATA)                /*  AKR  NAME  */  -
//*
//*
//DELETE.SYSIN  DD  *
 DELETE  ASG.VIACENXX.AKR.EX                           /*  AKR  NAME  HERE  * /
//*
```

Glossary

Abend
Abnormal end of task.

action bar
The line of keywords at the top of a screen. Each keyword represents a category of actions that may be performed on that screen. An action is selected by moving the cursor to the desired keyword and pressing Enter.

AKR (Application Knowledge Repository)
A BDAM or VSAM file that contains all analysis information produced by the analyze function of the Application Analytical Engine.

AKR utility log
File that provides a summary of the commands issued to the Batch AKR Utility.

Alias Of
A field on a pop-up listing entries in the AKR. If the analyzed program contains an Entry point, Alias Of is the name of the program that contains the Entry point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias Of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.

analyze
The process by which Encore gathers information about the program, including program relationships, logic, data and execution paths, and stores this information in the AKR.

analyze options
Run-time options that control the Analyze processing. Many of these options are similar to the COBOL compiler options. Default values are established at installation time and can be overridden by editing the Analyzer JCL or by using the analyze screens.

analyze summary report
A summary of the run-time statistics and diagnostic messages that are produced when an analyze job completes.

anomalies
A deviation or departure from the normal Encore analysis process.

Anomaly Repair facility
An Encore facility that allows you to repair programs already analyzed in an AKR that contain anomalies. This is accomplished by allowing you to reanalyze the member that generated the anomaly condition using the File ▸ Anomaly facility pull-down option.

anomaly reports
Reports that contain information beneficial in understanding and diagnosing analysis problems and anomalies.

anchor
An HTML element that defines a link between Internet resources.

API
Application programming interface. A set of calling conventions defining how a service is invoked through a software package.

APPC
Advanced program-to-program communication. An implementation of the SNA LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

application
Any group of programs that a user wants to analyze/view as a whole. These programs would all be defined within the same application.

asynchronous
Without regular time relationship, unexpected or unpredictable with respect to the execution of program instructions. See "synchronous" on page 500.

Batch AKR command
Control cards input in the VIASYSIN DD dataset of the Batch AKR Utility.

Batch AKR utility
The facility used to maintain the AKR without using ISPF.

browser
An application that displays World Wide Web documents.

business group
A logical grouping of applications by business function.

business rule
A logical business function that is performed by a COBOL program.

CERN
The Couseil Europeen pour la Recherche Nucleaire (European Particle Physics Laboratory), which developed hypertext technologies.

CICS server extract
A CICS server extract locates the necessary statements, files, and data elements required to create a CICS Server Extract program. This program is then used to create a COMMAREA-based server program from a CICS pseudo-conversational program.

client
As in client/server computing, the application that makes requests to the server and, often, handles the interaction necessary with the user.

client/server computing
A form of distributed processing, in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fulfills those requests. The client and server remain transparent to each other in terms of location and platform. See client above and "server" on page 499 .

COBOL subset
COBOL verbs of a similar nature that have been grouped together. For example, READ, WRITE, OPEN, and CLOSE are grouped into the IO subset.

command
An option in a menu. A command is selected to perform an action. There are the three types of commands:

- Primary commands
- Line commands
- Batch AKR utility commands

command input area
The field on Encore screens where primary commands are entered, indicated by ===> on the fourth line of a screen, or the second line of a pop-up.

commit
An action that an application takes to make permanent the changes it has made to CICS resources.

common code extract
The process of identifying the code that is common to two or more previously-extracted Logic Segments. This includes all statements that appear in a selected Logic Segment, as well as appearing in all other selected Logic Segments.

Common Gateway Interface (CGI)
The defined standard for the communications between HTTP servers and external executable programs.

Common User Access (CUA)
A style of graphical interface that features screens, actions bars, pull-downs, and pop-ups that are designed to provide easy access to all product features.

complement extract
The process of identifying a set of COBOL statements that do not include a previous extract. This includes all statements from the original program, excluding statements of the previously-extracted Logic Segment, but retaining any statements needed by other functional paths.

complement module
A complement module contains all statements from the original program, excluding statements of the extracted Logic Segment, but retaining any statements needed by other functional paths.

computation variable extract
The process of isolating the minimum set of statements that determine the value of a particular data variable (known as the computation variable) at a particular statement in the program.

cursor substitution character
A token substitution character that may be used in some primary commands. The cursor substitution character is set on the Options - Product Parameters pop-up.

conversational
A communication model where two distributed applications exchange information by way of a conversation. Typically, one application starts (or allocates) the conversation, sends some data, and allows the other application to send some data. Both applications continue in turn until one decides to finish (or deallocate). The conversational model is a synchronous form of communication.

criteria
The information entered that was used to isolate a Logic Segment.

cursor position
The current location of the cursor on the screen, used as a starting point in certain commands.

database

- A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications.
- All data files stored in the system.
- A set of data stored together and managed by a database management system.

dataname
A standard COBOL term for fields defined in the DATA DIVISION of a COBOL program. Variable names, files, groups, array elements, and fully qualified datanames.

data usage
Defines how a data item is used.

- DEF indicates the statements in the DATA DIVISION where the data item is defined.
- USE indicates the statements where the value is used or tested.
- MOD indicates the statements where the value is set or modified.
- REF indicates any of the above conditions.

dead assignment
A statement that assigns a value to a variable, where the value is not referenced anywhere in the source code. The assignment is not necessary and may be removed from the code without affecting program execution.

delimiter
A character or sequence of characters used as a separator in text or data files.

Distributed Computing Environment (DCE)
Adopted by the computer industry as a de facto standard for distributed computing. DCE allows computers from a variety of vendors to communicate transparently and share resources such as computing power, files, printers, and other objects in the network.

distributed processing
An application or systems model in which function and data can be distributed across multiple computing resources connected on a LAN or WAN. See "client/server computing" on page 491.

Distributed Program Link (DPL)
Enables an application program executing in one CICS system to link (pass control) to a program in a different CICS system. The linked-to program executes and returns a result to the linking program. This process is equivalent to remote procedure calls (RPCs). You can write applications that issue RPCs that can be received by members of the CICS family.

Distributed Transaction Processing (DTP)
Enables a transaction running in one CICS system to communicate synchronously with transactions running in other systems. The transactions are designed and coded specifically to communicate with each other. This method is typically used by banks, for example in just-in-time stock replacement.

Double Byte Character Set (DBCS)
A character set that uses two bytes to represent each character. Various Double Byte Character Sets are used with languages such as Chinese and Japanese that cannot be represented with single byte codes.

environment
The collective hardware and software configuration of a system.

equate
A substitution name for a character string.

External Call Interface (ECI)
An application programming interface (API) that enables a non-CICS client application to call a CICS program as a subroutine. The client application communicates with the server CICS program using a data area called COMMAREA.

External Presentation Interface (EPI)
An application programming interface (API) that allows a non-CICS application program to appear to the CICS system as one or more standard 3270 terminals. The non-CICS application can start CICS transactions and send and receive standard 3270 data streams to those transactions.

File Transfer Protocol (FTP)
A protocol that defines how to transfer files from one computer to another.

forms
Parts of HTML documents that allow users to enter data.

function shipping
Enables an application program running in one CICS system to access resources owned by another CICS system. In the resource-owning system, a transaction is initiated to perform the necessary operation. For example, to access CICS files or temporary storage, and to reply to the requester. The user is unaware of these behind-the-scenes activities, and need not know where the resource actually exists.

gateway
Software that transfers data between normally incompatible applications, or between networks.

gopher
Menu-based software for exploring Internet resources.

Graphic Interchange Format (GFI)
256-color graphic format.

help
Encore provides these three levels of help:

- Long messages
- Notes
- Tutorial screens

Specific command information is available by entering a command, then pressing PF01/13. The Help facility can also be accessed from the Help pull-down or any Encore screen.

home page
The default page shown at the first connection to an HTTP server.

host

- In a computer network a computer providing services such as computation, database access, and network control functions.
- In a multiple computer installation, the primary or controlling computer.

hypertext
Text that activates connection to other documents when selected.

Hypertext Markup Language (HTML)
Standard language used to create hypertext documents.

Hypertext Transmission Protocol (HTTP)
Standard WWW client/server communications protocol.

Internet Keyed Payment Protocol (IKP)
Proposed protocol for conducting secure commercial financial transactions on the Internet.

intercommunication
Communication between separate systems by means of Systems Network Architecture (SNA), Transmission Control Protocol/Internet Protocol (TCP/IP), and Network Basic Input/Output System (NetBIOS) networking facilities.

internet
A collection of networks.

label name
Any PROCEDURE DIVISION paragraph or section name and the PROCEDURE and PROC literals. Label name specifies all transfers of control to a paragraph or section.

line command
An instruction entered in the line command area on certain screens.

list box
A dialog box option containing a list of items the user can select.

list file
The file that is allocated when a request to print is issued.

live exit
An abnormality in program control caused by out of perform range GO TOs and overlapping perform ranges.

log file
The file that is allocated by Encore and used for error messages and log commands. There is a separate log file created by the Batch AKR utility.

logic segment
A set of source statements that is the result of a perform range extract, report extract, computation variable extract, transaction extract, statement extract, complement extract, server extract, or common code extract. The criteria used to isolate a Logic Segment may be saved in the AKR.

logic segment complement
See "complement module" on page 492.

logical program unit
A PERFORMed range of code including GO TO code, or a CALLed program.

Logical Unit of Work (LUW)
An update that durably transforms a resource from one consistent state to another consistent state. A sequence of processing actions (for example, database changes) that must be completed before any of the individual actions can be regarded as committed. When changes are committed (by successful completion of the LUW and recording of the synch point on the system log), they do not need to be backed out after a subsequent error within the task or region. The end of an LUW is marked in a transaction by a synch point that is issued by either the user program or the CICS server, at the end of task. If there are no user synch points, the entire task is an LUW.

long message
A diagnostic or error message that displays on line five of a screen or line three of a pop-up. Long messages are sometimes preceded by short messages that display in the upper right corner of the screen. Pressing PF01/PF13 (HELP) after receiving a short message displays the corresponding long message.

LU type 6.2 (LU 6.2)
A type of logical unit used for CICS intersystem communication (ISC). LU 6.2 architecture supports CICS host-to-system-level products and CICS host-to-device-level products. APPC is the protocol boundary of the LU 6.2 architecture.

markup tag
Special character sequences put in text used to pass information to a tool, such as a document formatter.

member
A member in a PDS or source manager such as Panvalet or Librarian. This can be the alias name found in the AKR.

menu
A list of available commands in an application window. Menu names appear in the menu bar of the application window.

menu bar
A bar that contains the menus available for your use.

message box
A box that displays a message (error or otherwise) to inform the user of a particular condition.

Multipurpose Internet Mail Extension (MIME)
The Internet standard for mail that supports text, images, audio, and video.

NCSA Mosaic
A web browser available on multiple platforms.

online help
See "help" on page 495.

Online Transaction Processing (OLTP)
A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. An online transaction processing system supervises the sharing of resources to allow efficient processing of multiple transactions at the same time.

paragraph name
Any PROCEDURE DIVISION paragraph or section name, and the PROCEDURE and PROC literals. Paragraph name includes the entire paragraph or section.

partial statement
In a generated COBOL module, a statement that has been modified to omit certain datanames or file names that are not required for that statement in the generated module. The partial statement contains only that portion of the statement required for the generated module.

perform range
A perform range consists of the source code contained in a PERFORM statement, and includes all code that is or could be executed as a result of GO TOs, PERFORMs, etc. within that PERFORM.

perform range extract
The process of isolating all executable statements of a perform range. This includes not only the statements within the range, but also all code that is reachable using GO TO or PERFORM from within the range.

pop-up
A window that displays as the result of selecting an item on a pull-down or pop-up, or as the result of entering certain commands. It is superimposed on the screen to allow entry of information for the requested action.

postscript
The standard for presenting text and graphics in a device-independent format.

primary command
An instruction entered in the command input area of the screen.

program
Program source member name, the name specified in the IDENTIFICATION DIVISION of a COBOL program, or the CSECT name of a program that is not COBOL.

protocol

- A formal set of conventions governing the format and control of data.
- A set of procedures or rules for establishing and controlling transmissions from a source device or process to a target device or process.

proxy
A gateway that allows Web browsers to pass on a network request (a URL) to an outside agent.

pseudo conversational
A type of CICS application design that appears to the user as a continuous conversation, but consists internally of multiple tasks.

pull-down
The list that displays when an action is selected on the action bar. On a pull-down, an action followed by ellipses (...) displays a pop-up when selected and an action not followed by ellipsis (...) immediately activates internal commands.

punch file
The file that is allocated when a request to punch is issued.

recovery
The use of archived copies to reconstruct files, databases, or complete disk images after they are lost or destroyed.

recoverable resources
Items whose integrity CICS maintains in the event of a system error. These include individual files and queues.

recursion
A perform range or paragraph that performs itself.

re-engineering
The process of renovating an existing program to isolate and extract distinct functions. Encore can isolate and extract code based on a perform range, report, computation variable, transaction, or server.

report extract
The process of isolating all WRITE/GENERATE statements for a specific File Description (FD) or Report Description (RD). Individual statements may be selected. The extracted unit consists of the statements required to produce the WRITE/GENERATE statement(s).

SBCS
See "Single Byte Character Set - (SBCS)" on page 500.

screen
A full-width display of information containing an action bar as the first line. Encore screens are modeled after TSO/ISPF screens.

screen subset
The result of an interactive command.

script
An executable program invoked by HTTP servers.

script file
A dataset containing Encore commands, created when SET SCRIPT is ON, and executed with the EXECUTE primary command or the File ▸ Execute script pull-down option.

server
Any computing resource dedicated to responding to client requests. Servers can be linked to clients through LANs and WANs to perform services (such as printing, database access, fax, and image processing) on behalf of multiple clients at the same time.

server extract
Process used to create a COMMAREA-based server program from a 3270 CICS pseudo-conversational program.

shortcut key
A keyboard key or combination of keys that invokes a particular command, such as CNTL + N.

short message
A diagnostic or error message that displays in the upper right corner of Encore screens. Pressing PF1/PF13 (HELP) after receiving a short message displays the corresponding long message.

Single Byte Character Set - (SBCS)
A character set that uses one byte to represent each character. Single Byte Character Sets are used with languages, such as English, where the characters can be represented with a one-byte code.

Socket Secure (SOCKS)
The gateway that allows compliant client code (client code made socket secure) to establish a session with a remote host.

Standard Generalized Markup Language (SGML)
The standard that defines several markup languages, HTML included.

statement extract
The process of isolating all user-identified statements from a program.

status bar
The area at the bottom of a main window that lists the status of an action and gives other information, such as the meaning of a command.

Storage Management Subsystem (SMS)
An operating environment that automates and centralizes the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

subset
A grouping of source lines in a program. See "COBOL subset" on page 491, "screen subset" on page 499, and "tagged line subset" on page 501 .

synchronous

- Pertaining to two or more processes that depend on the occurrence of a specific event such as a common timing signal.
- Occurring with a regular or predictable time relationship.

synchpoint
A logical point in execution of an application program where the changes made to the databases by the program are consistent and complete and can be committed to the database. The output, which has been held up to that point, is sent to its destination. The input is removed from the message queues and the database updates are made available to other applications. When a program terminates abnormally, CICS recovery and restart facilities do not back out updates prior to the last completed synchpoint.

tagged line subset
Command results displayed in columns 73 through 80 of the Source View screen.

target
The object of a primary command.

transaction
A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction can require the initiation of one or more tasks for its execution.

transaction extract
The process of isolating all conditional statements for a specific transaction control variable. Certain program paths may then be selected or blocked, based on the transaction code value.

transaction processing
A style of computing that supports interactive applications in which requests submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

transaction routing
Enables a terminal connected to one CICS system to run a transaction in another CICS system. It is common for CICS/ESA, CICS/VSE, and CICS/MVS users to have a terminal-owning region (TOR) that owns end-user network resources.

VIASUB
An edit macro included with Encore that is used to submit an Analyze job.

VIASUBDS
A CLIST included with Encore that is used to submit an Analyze job.

view method

The manner in which a program is examined in the View facility. These are the three view methods based on source code:

- Structure view
- Tree view
- Source view

All three views are available concurrently when a program is examined.

work file

A temporary file allocated upon entry to Encore.

Appendix A
Help Facility

Introduction

A comprehensive and context sensitive Help facility, including an online Help Tutorial, is provided that answers most questions online. The Help Tutorial contains help information for several types of topics, such as screens and pop-ups, fields, commands, messages, and abends. The Help Tutorial also includes a Table of Contents that lists general information topics, and a comprehensive Index for viewing specific information.

ESW online help facilities can be reached through several means. Help can be requested by pressing PF01, pressing PF13, or by typing HELP or a question mark (?) in the command input area on any screen or pop-up.

This table shows the various online help information provided by Encore and the means of accessing them:

Help Topic	Access Methods
Screen and Pop-up. See "Screen and Pop-up Help" on page 506.	• Enter the HELP command with no messages displayed. • Press PF01/13 with no messages displayed. • Select Help ▸ Current Screen.
Field help. See "Field Help" on page 507.	• Enter a question mark (?) in the desired field.

Help Topic	Access Methods
Command help. See "Command Help" on page 508.	• Type the command in the command input area and press PF01/13. • Enter the HELP COMMANDS command. • Enter the HELP command with the command name as the operand. • Select Help ▸ All Commands. • Select Help ▸ Specific Command.
General Help. See "General Information" on page 509.	• Enter the TOC command from within the Help Tutorial. • Select Help ▸ Table of contents.
Specific help. See "Specific Information" on page 510.	• Enter the INDEX command from within the Help Tutorial. • Select option 6 on the Help Table of Contents. • Select Help ▸ Index.
Abend help. See "Help Abends" on page 511.	• Enter the HELP ABENDS command. • Select Help ▸ Common Abends.
Message help. See "Help Messages" on page 512.	• Enter the HELP command with the message displayed. • Press PF01/13 with the message displayed. • Enter the HELP command with the `msg#` operand. • Select Help ▸ Current Message with the message displayed. • Select Help ▸ Specific Message.

Help Navigational Commands

All of the online help topics listed in the previous table are contained in the Help Tutorial. Each online help topic can be reached from anywhere within the Help Tutorial by going through the Help Table of Contents or Index. Once the Help Tutorial is accessed, these commands are available for navigating within the Help Tutorial:

Help Command	Purpose
BACK	Redisplay the previous Help Tutorial screen.
END	Exit the Help Tutorial.
ENTER	Display the next screen in a continuation series.
INDEX	Display the first screen of the Help Index.
SKIP	Go directly to the next subject.
TOC	Display the Help Table of Contents.
UP	Display the next higher-level subject.
alpha character	On an Index screen, entering an alphabetic character displays the Index screen corresponding to that character.

Screen and Pop-up Help

Help for the current screen or pop-up is requested by entering the HELP primary command by:

- Pressing PF01/13 with no messages appearing on the screen.
- Entering the HELP SCREEN command.
- Selecting Current Screen on the Help pull-down (see Figure 151 on page 316). The Help Tutorial for the current screen or pop-up displays.

Figure 174 shows an example of the Help Tutorial for the Extract - Perform Range Filter pop-up. The Help Tutorial for each screen and pop-up describes all the options available on that screen or pop-up, lists descriptions of all the fields, and notes any special processing considerations.

Figure 174 • Screen and Pop-up Help Example

```
ASG-Encore R6.0 ------- Extract - Perform Range Filter  ------------------ HELP
===> _

Use the Extract - Perform Range Filter pop-up to filter the list of perform
range names that appears on the Extract - Perform Range List pop-up.

The Extract - Perform Range Filter pop-up is displayed by pressing PF5 on the
Extract - Perform Range List pop-up.  When the Extract - Perform Range pop-up
is initially displayed, each field contains default values.  If these defaults
are changed, the new values will be retained for subsequent sessions.

A ? (question mark) can be entered in text fields to display a help pop-up
about that particular field.

                                                (More...press ENTER to continue.)
```

Field Help

Help for a specific field on some screens is requested by entering a question mark (?) in the appropriate field on the screen. The Help screen for that particular field displays. Field help is only available on certain screens.

The Help screen for each field contains a description of the field and lists the valid values that can be entered in the field (if appropriate).

The field help screen also contains an input field to allow you to enter the desired value, as shown in Figure 175. Once the desired value is typed in the input field and Enter is pressed, the screen redisplays with the desired value appearing in the appropriate field. It is not necessary to enter the desired value on the field help screen. If the input field is left blank, the screen redisplays when you press Enter, with the appropriate field appearing as it did before field help was requested.

Figure 175 • Field Help Example

```
                          Help - Perform Name

Perform Name   _--------------------------------------------------

Description:

   Optional.  If left blank, all perform ranges are included.  If a
   fully-qualified perform range name is entered, a message is
   displayed indicating that the perform range has been selected.

   If the Selection Screen field is List, the perform range can be
   entered with imbedded wildcard characters (percent symbol for 1
   character; asterisk for zero or more characters).  The Extract -
   Perform Name List pop-up is displayed to allow selection of the
   perform range name.

User Action:

   Type a perform range name in the field above, or leave the field
   blank, and press Enter.  The Extract - Perform Range pop-up will
   reappear with the specified value in the PERFORM Name field.
```

Command Help

Help for a specific command is requested by:

- Typing the command in the command input area and pressing PF1/13
- Typing HELP followed by the desired command name
- Selecting Help ▸ Specific Command.

A long message displays describing the command. Pressing PF1/13 again displays the Help Tutorial screen for that command, which displays the command syntax diagram and gives a description of each operand in the command. Figure 176 shows help for the VIEW command requested from the Extract - Perform Range Filter pop-up. Entering the UP command on a command help screen displays a complete list of all ESW primary commands.

For help on all ESW commands use the HELP COMMANDS command, or select Help ▸ All Commands. This lists all of the primary command alphabetically. Pressing PF1/13 displays a list of all Encore generic help navigation commands. From this list, you can display information about a particular command by selecting the appropriate number.

Figure 176 • Command Help Example

```
ASG-Encore RX.0 -------------------  VIEW  ------------------------------  HELP
===> _

The VIEW command starts a new Source View session for the specified program
while retaining the current session.

The VIEW command syntax is:

View ----------------------------------------------------------------------><
        |-pgm-|

                                        Minimum abbreviations are in CAPS
                                        LEGEND: ---required------------------><
                                                                |-optional-|

Operand Descriptions

blank           Displays the AKR Program Selection pop-up that is used to
                select a program for viewing.

pgm             A program for which a Source View session is to be started.
```

For some commands on the Source View screen only, after the long message displays, press PF1/13 to display NOTES that give specific examples for using the command. Once the NOTES display, press PF1/13 again to display the Help Tutorial for that command. The Help NOTES displays for those commands where specific examples are helpful.

General Information

General help information, as shown in Figure 177, is requested by entering the Help Tutorial and then issuing the TOC command, or by selecting Help ▸ Table of contents.

Figure 177 • Help Table of Contents

```
ASG-Encore RX.0 -------------  Help Table of Contents  ------------------- HELP
===> _

The topics below represent general categories of information about
ASG-Encore.  To get help for a pull-down, select the Action Bar topic. This
Help Table of Contents may be redisplayed from any Help screen by
entering the TOC command.

The following topics will be presented only if explicitly selected by number:

  1  Overview of ASG-Encore

  2  Introduction to CUA

  3  The Action Bar

  4  Customer Support

  5  R6.0 Summary of Revisions

  6  Index for ASG-Encore Help
```

Specific Information

Help for specific topics is requested by entering the Help Tutorial and then entering:

- The INDEX command.
- Selecting option 5 on the Help Table of Contents.
- Selecting Help ▸ Index.

Help for a specific topic can then be viewed by selecting the appropriate Index entry. On any Index screen, as shown in Figure 178, entering an alphabetic character displays the Index screen corresponding to that character.

Figure 178 • Help Index Screen

```
ASG-Encore RX.0 ---------------  INDEX A - D  --------------------------  HELP
===> _

To select a topic, enter the two-character identifier.


A1 - Action Bar                    C1 - Commands, ASG-Encore Primary
A2 - AKR Utilities                 C2 - Computation Variable Extract Overview
A3 - ALLOCDEF Command              C3 - Copy Logic Segment
A4 - ANALYZE Command               C4 - CUA, Introduction
A5 - Analyze Options               C5 - CURRENT Command
A6 - Analyze Submit Parameters

                                   D1 - DISPLAY Command

B1 - Batch Computation - Save Logic Segment
B2 - Batch Computation - Select Logic Segment
B3 - BRANCH Command
B4 - Branch Previous Options


 Another index page can be displayed by entering its letter.
```

Help Abends

Help for ESW user abends is requested by entering:

- The HELP ABENDS command.
- ABENDS in the command input area and pressing PF1/13.
- Selecting Help ▸ Common Abends.

The ABENDS screen displays. Selecting topic 2 on this screen displays the ASG Abend Codes screen, shown in Figure 179, which lists all the ESW user abend messages, and explanations for each message.

Figure 179 • ASG Abend Codes Screen

```
ASG-Encore RX.0 ---------------  ASG ABEND CODES  ------------------------ HELP
===> _

Abend codes in the range 900 - 999 (X'384 - X'3E7') bypass ASG error
recovery, causing the abend to be handled by ISPF or by the system. If the
problem cannot be resolved, call Customer Support.


965  X'3C5'     Unable to intercept program.

967  X'3C7'     The ASG-Center AUTHORIZE password was not specified during
                installation.

968  X'3C8'     An internal error occurred during initialization.

970  X'3CA'     A package load module was called directly.

972  X'3CC'     The ASG Edit Monitor encountered a severe error.

974  X'3CE'     An invalid VIASBASE module was found. The current product
                expects a level of CE050 or greater. Enter HELP 4988 for more
                information.
                                  (continued)
```

Help Messages

ESW messages display in the long message area, which is the third line on a pop-up, or the fifth line on a screen. Long messages sometimes display in a separate window.

This is the format for messages:

```
ASGnnnnx text
```

Where *nnnn* is the message number, *x* is one of the severity levels listed in this table and `text` is the long or short message text.

Level	Description
I	Informational - no required action.
W	Warning - an error condition exists that is not critical.
E	Error - a critical error condition exists.
D	Disaster - a serious error condition exists; the product is unable to continue.
T	Termination - the product terminated with the specified error.

Short messages display when available. Long messages display if a short message does not exist, or when help is requested immediately following a displayed short message.

Help for a specific message displays by entering the HELP primary command, followed by the message number, or by selecting Help ▸ Current Message or Help ▸ Specific Message. The Help Explanation and Action Panel for that message displays, as shown in Figure 180.

Figure 180 • Help Explanation and Action Panel

```
                    HELP Explanation and Action Panel
Command ===> _____________________________________________ Scroll ===> CSR

Additional support may be found at our Web Site: www.asg.com

ASG1222  ENTER THE REQUIRED FIELD AT THE CURSOR LOCATION.

EXPLANATION:
    The field at the cursor location is a required field that must be
    entered for processing to continue.

ACTION:
    Enter the information required to continue processing or exit from
    the panel.

******************************** BOTTOM OF DATA ********************************
```

Printing Messages

All ESW messages or a range of messages can be printed using the VIASMPRT program. The VIASMPRT program produces a listing of these specified messages:

- Message number
- Short message (if available)
- Long message
- Explanation of the message
- Action (if any)

JCL to execute the VIASMPRT program is in ASG.VIACENxx.CNTL(VIASMPRT). The entire message file is printed unless a specific range is specified in the PRM parameter. For example, this parameter would print messages 300 through 499:

```
PRM='START=300,END=499'
```

The ALL keyword can be specified in the PRM parameter to print all messages. The default value for START is 1; the default value for END is 9999. If only the START value is entered, messages print starting at the message number specified and ending with 9999. If only the END value is entered, messages print starting with 1 and ending with the message number specified.

The NOTES keyword specifies that any notes associated with a message are printed. The default is NONOTES. Typically, notes are provided to illustrate primary commands, as shown in Figure 181.

Figure 181 • VIASMPRT JCL

```
//ASG  JOB (  ),'ASG-CENTER  VIASMPRT'                                00010000
//*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.                00020000
//*                                                                   00030000
//*   ************************************************************    00040000
//*   * ASG, INC.           ASG-CENTER Rx.x                     *     00050000
//*   *                                                         *     00060000
//*   *           UTILITY TO PRINT ASG MESSAGES                 *     00070000
//*   *                                                         *     00080000
//*   ************************************************************    00090000
//*                                                                   00100000
//VIASMPRT PROC VIASOFT='ASG',     ASG HI-LVL NODES                   00110000
//             CENTER='VIACENXX',  ASG MIDDLE NODES                   00120000
//             SYSOUT='*',         PRINT OUTPUT MESSAGE CLASS         00130000
//             PRM=''              PARM FOR MESSAGES TO BE PRINTED    00140000
//*                                                                   00150000
//*   ***************************************************************  00160000
//*   *                                                             *  00170000
//*   *       M E S S A G E    P R I N T   U T I L I T Y            *  00180000
//*   *                                                             *  00190000
//*   *  THIS PROGRAM WILL PRINT ALL OF THE MESSAGES IN THE ASG     *  00200000
//*   *  MESSAGE FILE AND THE HELP TEXT ASSOCIATED WITH EACH        *  00210000
//*   *  MESSAGE IT WILL PRINT THE ENTIRE FILE BY DEFAULT.  YOU MAY *  00220000
//*   *  SELECT A GIVEN RANGE OF MESSAGES BY SPECIFYING THE OPTION- *  00230000
//*   *  AL PARAMETER KEYWORDS: START AND END.  FOR EXAMPLE:        *  00240000
//*   *          PRM='START=300,END=499'                            *  00250000
//*   *  WILL PRINT MESSAGES NUMBER 300 THROUGH 499, INCLUSIVE.     *  00260000
//*   *  THE DEFAULT VALUES FOR START AND END ARE 1 AND 99999       *  00270000
//*   *  RESPECTIVELY.  CONSEQUENTLY THE PRM VALUE 'END=300' WILL   *  00280000
//*   *  PRINT MESSAGES 1 THROUGH 300, AND THE PRM VALUE            *  00290000
//*   *  'START=4000' WILL PRINT MESSAGES 4000 THROUGH 99999.       *  00300000
//*   *                                                             *  00310000
//*   *  AN OPTIONAL KEYWORD, NOTES, WILL ALSO PRINT ANY NOTES      *  00320000
//*   *  ASSOCIATED WITH A MESSAGE.                                 *  00330000
//*   *                                                             *  00340000
//*   *  ADDITIONALLY, THE KEYWORD 'ALL' WILL EXPLICITLY PRINT ALL  *  00350000
//*   *  MESSAGES.                                                  *  00360000
//*   ***************************************************************  00370000
//*                                                                   00380000
//*                                                                   00390000
//VIAMPRT  EXEC PGM=VIASMPRT,REGION=4096K,                            00400000
//           PARM='&PRM'                                              00410000
//STEPLIB   DD  DSN=&VIASOFT..&CENTER..LOADLIB,DISP=SHR               00420000
//VIAMSGS   DD  DSN=&VIASOFT..&CENTER..VIAMSGS,DISP=SHR               00430000
//SYSPRINT  DD  SYSOUT=&SYSOUT                                        00440000
//VIAPRINT  DD  SYSOUT=&SYSOUT                                        00450000
//VIALOG    DD  SYSOUT=&SYSOUT                                        00460000
//SYSUDUMP  DD  SYSOUT=&SYSOUT                                        00470000
//*                                                                   00480000
//         PEND                                                       00490000
//*                                                                   00500000
//VIASMPRT EXEC VIASMPRT          PRINT MESSAGES                      00510000
//*                                                                   00520000
```

Figure 182 shows the output from the job.

Figure 182 • VIASMPRT Job Output

```
PRINTING  MESSAGES  FROM  1222  TO  1223.
MESSAGES  PRINTED.
END  OF  MESSAGE  PRINT  PROCESSING.
ASG1222  ENTER  THE  REQUIRED  FIELD  AT  THE  CURSOR  LOCATION.

 EXPLANATION:
 The field at the cursor location is a required field that must
 be Entered for processing to continue.

 ACTION:
 Enter the information required to continue processing or exit
 from The panel.


ASG1223  VALID  UNIT  TYPES:  CYLINDERS/COL  OR  TRACKS/TRK  OR
RECORDS/REC.

 EXPLANATION:
 The space unit allocation type is invalid.

 ACTION:
 Reenter the field with one of the following:
  CYLINDERS,  CYL,  TRACKS,  TRK,  RECORDS,  REC.
```

Appendix B
Command Reference

This appendix alphabetically lists this information for each Encore primary command:

- Abbreviation
- Default PF key assignment
- Brief description

Command	Abbreviation	Default PF Key	Description
& (Retain)	none	none	Executes a primary command and keeps it displayed in the command input area for additional use.
ALLOCDEF	none	none	Displays the Options - Product Allocations pop-up (see Figure 142 on page 298).
ANALYZE	AN	none	Displays the File - Analyze Submit pop-up (see Figure 16 on page 39) used to prepare a program for testing.
BRANCH	B	none	Positions the cursor at the specified target.

Command	Abbreviation	Default PF Key	Description
CALL	Call	none	Starts a Source View session for another program without ending the current session. The GOBACK or END command can be used to return to the original program.
CANCEL	CAN	none	Terminates the current function without saving changes to marks and equates made during the session.
CURRENT	CURR	none	Saves the current location in the program source. The LOCATE &CURRENT command can then be used to reposition the screen to the saved location.
DISPLAY	DIS	none	Excludes from the screen all statements not contained in the current Logic Segment.
END	none	PF03/15	Terminates the current function and redisplays the previous screen.
EQUATE	EQ	none	Defines a name for a character string.
EXCLUDE	EX X	none	Performs a FINDXTND command on the specified target, excluding the resulting lines.
EXECUTE	EXEC	none	Reads and executes a Encore script file.

Command	Abbreviation	Default PF Key	Description
EXIT	none	none	Terminates the current function and exists Encore.
EXPORT	EXPO	none	Writes the relevant data from the Extract - Perform Name List pop-up in comma delimited format to a specified or default dataset name.
FIND	F	none	Searches for one or all occurrences of the specified character string.
FINDXTND	FINDX FX	none	Performs a COBOL intelligent search of the source code for one or all occurrences of the specified target.
GOBACK	GO	none	Reverses a CALL or VIEW command and return to a Source View session.
HELP	H ?	PF01/13	Displays information about the current screen, command, or message.
JUMP	none	none	On the Tree View screen, displays the View - Source pop-up (see Figure 35 on page 76).
KEYS	none	none	Displays the Options - PF Key Definition pop-up (see Figure 146 on page 305) used to display or modify the Encore PF key assignments.

Command	Abbreviation	Default PF Key	Description
LEVELS	none	none	Redisplays the current screen or pop-up to the specified level.
LIST	LI	none	Displays the specified List pop-up.
LOCATE	L	none	Scrolls to the selectable entry that most closely matches the specified string.
LPRINT	LP	none	Copies the lines that contain the specified target.
LPUNCH	none	none	Copies lines containing the specified target to the Punch file for subsequent processing.
PARMDEF	PDEF	none	Displays the Options - Product Parameters pop-up (see Figure 141 on page 296).
PREF	P	none	Displays the View - Paragraph Cross Reference pop-up (see Figure 51 on page 104).
PRINTLOG	PLOG	none	Displays the Options - Log/List/Punch Definition pop-up (see Figure 143 on page 300).
PRINTLST	PLI	none	Displays the Options - Log/List/Punch Definition pop-up.
PROCESS	PROC	none	Defines an action to be performed repetitively on a file of tokens.

Command	Abbreviation	Default PF Key	Description
PRODLVL	none	none	Displays the Encore and Center product levels.
PROGRAM	PRO	none	Ends the current Source View session and enters a new Source View session for the specified program.
QUALIFY	Qualify	none	Displays the specified program on the screen for investigation.
RECALL	REC	PF12/24	Displays the previous primary or internal command or message.
REDO	RED	none	Executes the RFIND command from the cursor position on Source View.
REPEAT	REP	none	Executes the last stacked primary command again.
RESET	RES	none	Removes highlighting and tags, cancels pending line commands, terminates message lines, or redisplays all excluded lines.
RETURN	none	none	Terminates the current function and returns to the primary Encore screen.
RFIND	RF	PF05/17	Finds the next or previous occurrence of the target of the last FIND or FINDXTND command.

Command	Abbreviation	Default PF Key	Description
RNSCOPY	none	none	Copies the generated COBOL module from a temporary file into the current Edit member.
RSTRUCTURE-VIEW	RSTRUCT RSTV	none	Redisplays the list Structure View.
RTREEVW	RTREE RTV	none	Redisplays the last Tree View.
SAVE	SA	none	Saves Equates in the AKR.
SCROLL	none	none	Finds highlighted or nonhighlighted lines on the Logic View or Structure View screen, or the View - Logic Segment (see Figure 43 on page 91) or View - Perform Structure pop-up (see Figure 42 on page 90).
SELECT	S	none	Selects a data name, perform name, report name, or statement number.
SET	none	none	Enables or disables the mode indicated by the specified operand.
SORT	none	none	Sorts the entries on an Extract List pop-up by any of the columns on the pop-up.
SOURCE-VIEW	SV	none	Displays the Source View.
STRUCTURE-VIEW	STRUCT STV	none	Displays the Structure View.

Command	Abbreviation	Default PF Key	Description
TREEVIEW	TREE TV	none	Displays the Tree View.
UNSELECT	U	none	Unselects a data name, perform name, external procedure name, or statement from a list of selected items.
VIEW	View	none	Starts a new Source View session for the specified program while retaining the current session.
ZOOMIN	ZOOMI ZI	PF04/16	Displays source lines according to the hierarchical levels of the program.
ZOOMOUT	ZOOMO ZO	none	Excludes source lines according to the hierarchical levels of the program.

Appendix C
Paragraph Control Transfer Methods

The View - Paragraph Cross Reference pop-up (see Figure 51 on page 104) shows how control is transferred to or from the target paragraphs. This table lists the control transfer methods:

COBOL Transfer Methods	
Method	**Abbreviations**
ALTERED-GOTO	ALTERED, AG
CALL INTERNAL	CLL INT, CI
CALL RETURN	CLL RTN, CR
FALLTHRU	FALL, F
GOTO	G
GOTO-DEPENDING	GODEP, GD
PERFORM	PERF, P
PROGRAM ENTRY	ENTRY
PROGRAM EXIT	EXIT
RETURN	RET, R
UNKNOWN	UNKN, ?
USE BEFORE RPT	USE REPORT, U
USE FOR DEBUG	USE DEBUT, U
USE ON ERROR	USE ERROR, U

COBOL Transfer Methods	
Method	**Abbreviations**
CICS Handle Strings	
ABEND	ABND, ABN
ANYKEY	ANYK, AK
CBIDERR	CBIDE,CB
CCERROR	CCERR, CC
CLEAR	CLR, CL
CLRPARTN	CLRP, CLP
DISABLED	DISA, DI
DSIDERR	DSIDE, DS
DSSTAT	DSST, DT
DUPKEY	DUPK, DK
DUPREC	DUPR, DR
ENDDATA	ENDD, ED
ENDFILE	ENDF, EF
ENDINPT	ENDI, EI
ENQBUSY	ENQB, EB
ENTER	ENT, ET
ENVDEFERR	ENVDE, ET
EOC	EC
EODS	ES
EOF	EF
ERROR	ERR, ER

COBOL Transfer Methods	
Method	**Abbreviations**
EXPIRED	EXPIR, EX
FUNCERR	FUNCE, FE
IGREQCD	IGQCD, ICD
ILLOGIC	ILLOG, IL
INBFMH	INBF, IB
INVERRTERM	INVERRT, IT
INVLDC	INVL, ILD
INVMPSZ	INVM, IM
INVPARTN	INVP, IP
INVPARTNSET	INVPART, IS
INVREQ	INVR, IR
INVTSREQ	INVTS, IT
IOERR	IOE, IE
ISCINVREQ	ISCIR, IIR
ITEMERR	ITEME, ITE
JIDERR	JIDE, JID
LENGERR	LENGE, LE
LIGTHTPEN	LPEN, LPN
MAPERROR	MAPE, MPE
MPFAIL	MAPF, MPF
NAMEERROR	NAMEE, NME
NOJBUFSP	NOJBSP, NJB

COBOL Transfer Methods	
Method	**Abbreviations**
NOVAL	NONV, NV
NOPASSBKRD	NOPASSB, NPB
NOPASSBKWR	NOPASSBW, NPW
NOSPACE	NOSA, NSA
NOSPOOL	NOSP, NSP
NOSTART	NOST, NST
NOSTG	NSTG, NTG
NOTALLOC	NALLC, NAL
NOTAUTH	NAUTH, NAH
NOTFND	NFND, NFD
NOTOPEN	NOPEN, NOP
OPENERR	OPENE, OPE
OVERFLOW	OVRFLW, OVF
OPERID	OPEI, OPI
PA1	A1, 1
PA2	A2, 2
PA3	A3, 3
PARTNFAIL	PARTNF, PNF
PF1	F1
PF2	F2
PF3	F3
PF4	F4

COBOL Transfer Methods	
Method	**Abbreviations**
PF5	F5
PF6	F6
PF7	F7
PF8	F8
PF9	F9
PF10	F10
PF11	F11
PF12	F12
PF13	F13
PF14	F14
PF15	F15
PF16	F16
PF17	F17
PF18	F18
PF19	F19
PF20	F20
PF21	F21
PF22	F22
PF23	F23
PF24	F24
PGMIDERR	PGMIDE, PDE
QBUSY	QBSY, QB

COBOL Transfer Methods	
Method	**Abbreviations**
QIDERR	QIDE, QID
QZERO	QZRO, QZ
RDATT	RDAT, RDT
RETPAGE	RETP, RTP
ROLLEDBACK	ROBACK, RBK
RTEFAIL	RTEF, RTF
RTESOME	RTES, RTS
SELNERR	SELNE, SNE
SESSBURY	SESSB, SSB
SESSIONERR	SESSIONE, SSE
SIGNAL	SIGNL, SGL
SUPPRESSED	SUPPRESS, SPS
SYSBUSY	SYSB, SYB
SYSIDERR	SYSIDE, SID
TERMERR	TERME, TME
TERMIDERR	TERMIDE, TID
TRANSIDERR	TRANSIDE, TSI
TRIGGER	TRIG, TRG
TSIOERR	TSIOE, TIO
UNEXPIN	UNEXP, UNP
USERIDERR	USERIDE, UID
WRBRK	WBRK, WBK

COBOL Transfer Methods	
Method	**Abbreviations**
WRONGSTAT	WRONGS, WST
SQL WHENEVER Statements	
SQLCONTINUE	SQLCONT, SQC
SQLWARNING	SQLWARN, SQW
SQLERROR	SQLE, SQE
SQLNOTFOUND	SQLNF, SQN

Index

Symbols

& (Retain) command 328

A

action bar
- action descriptions 10
- description 8
- overview 9
- shortened 9

AKR utilities
- AKR structure 466
- Batch AKR Utility 467
- introduction to AKR management 465
- online AKR utilities 466

Alliance
- accessing from ESW screen xxiii
- description xx
- linking xxiv

Allocating and expanding AKRs without ISPF 484

ALLOCDEF command 329

analysis options, Anomaly Repair facility parameters 42

Analysis/Anomaly report parameters
- NUMPRM(##) 41, 441
- RPTCEN 41, 441
- VIARERPT 41, 441
- VIARERPTAN 41, 441

analyze
- adding facilities to a standard compile mechanism 452
- input descriptions 432
- options and features 434
- process 434
- summary report 449
- using ISPF 435
- using ISPF/PDF edit 436
- VIASUB edit macro 40

ANALYZE command 330

analyze options
- Analysis/Anomaly report parameters 41
- Anomaly Repair facility 41
- buffers - BUF 454
- COBOL level 455
- DB2 application plan - DB2PLAN 456
- DB2 load library - DB2LIB 455
- DB2 subsystem - SUBSYS 463
- dynamic calls - DYNCALL 456
- flag messages - FLAG 457
- INPUT 457
- IO 458
- language level - LANGLVL 458
- line count - LINECNT 459
- live exit - XLIVE 464
- MAIN 459
- maximum number of errors - MBRERCNT 459
- Memory - XMEM 464
- NOINPUT 457
- OUTPUT 460
- overview 453
- program - PRO 460
- recursion - RECUR 461
- RETURN 461
- sequence - SEQ 462
- SOURCE 462
- SPACE 462
- SQL authorization ID - SQLID 463

analyzing a COBOL program 432

Anomaly Facility
- batch submit 62
- parameters 464
- program selection list 64
- user parameters 66

ANOMALY parameter
- GTD 42
- IPR 42
- OOPGT 42

Anomaly Repair facility 441

Anomaly Repair facility parameters
VIAANOMF 42, 442, 464
Anomaly report parameters
VIARERPT 463
VIARERPTAN 463
Anomaly reports 463
AutoChange
accessing from ESW screen xxiii
description xx
automatic JCL modifications 444

B

Batch AKR reports
AKR Utility Directory report 481, 483
AKR Utility Log 481
Punch Directory file 483
Batch AKR Utility
command syntax 470
control cards 468
JCL statements 468
BRANCH command 331
Bridge
accessing from ESW screen xxiii
description xx

C

CALL command 334
CANCEL command 335
Center, description xx
CICS
COMMAREA-based server program 121
pseudo-conversational program 121
RECEIVE map 121
SEND map 121
CICS and self-directed server generation 241
CICS generation 243
CICS server
endpoints 123, 189
extract java servlets 188
extract MQSeries 188
extract procedure 188
paths 191
programs 188
startpoints 123, 190
CICS server extract, description 121
CICS server module
description 241
server generation 241
COBOL
OF clause 22
syntax rules 387
COBOL subset 17
code extraction 4
command
CALL 334
CANCEL 335
END 8
FINDXTND 22
GOBACK 368
LPUNCH 381
PROGRAM 400
SCROLL 22
command format 469
command reference
& (Retain) 517
ALLOCDEF 517
ANALYZE 517
BRANCH 517
CALL 518
CANCEL 518
CURRENT 518
DISPLAY 518
END 518
EQUATE 518
EXCLUDE 518
EXECUTE 518
EXIT 519
EXPORT 519
FIND 519
FINDXTND 519
GOBACK 519
HELP 519
introduction 517
JUMP 519
KEYS 519
LEVELS 520
LIST 520
LOCATE 520
LPRINT 520
LPUNCH 520
PARMDEF 520
PREF 520
PRINTLOG 520
PRINTLST 520
PROCESS 520
PRODLVL 521
PROGRAM 521
QUALIFY 521
RECALL 521

REDO 521
REPEAT 521
RESET 521
RETURN 521
RFIND 521
RNSCOPY 522
RSTRUCTURE-VIEW 522
RTREEVW 522
SAVE 522
SCROLL 522
SELECT 522
SET 522
SORT 522
SOURCE-VIEW 522
STRUCTURE-VIEW 522
TREEVIEW 523
UNSELECT 523
VIEW 523
ZOOMIN 523
ZOOMOUT 523
command syntax
Comments 471
CONVERT Batch AKR command 472
COPY Batch AKR command 473
DELETE Batch AKR command 474
diagrams 470
EXPORT Batch AKR command 475
HELP Batch AKR command 476
INIT Batch AKR command 477
MOVE Batch AKR command 478
PRINT Batch AKR command 479
PUNCH Batch AKR command 480
command syntax diagrams 325
common code extract 491
compilers
COBOL D, E, and F 6
COBOL for MVS and VM 6
COBOL for OS/390 and VM 6
COBOL II 6
COBOL/370 6
Object Oriented conditions not supported 6
complement extract 492
computation variable extract 492
conventions page xxvii
CUA features 8
CURRENT command 336

D

data name
alias 21
DEFINITION 22
description 21
fully qualified 22
group level 22
INDIRECT 342, 360
MODIFICATION 22
REFERENCE 22
USE 22
data name type
elementary data name 21
file name 21
group name 21
special name 21
table element name 21
data, element 21
DISPLAY command 337

E

Encore
accessing from ESW screen xxiii
action bar 9
description xxi
functional components 3
screens 13
Encore analyze parameters 441
END command 8, 338
environment
client - Windows 95/98/2000 5
client - Windows NT with access to MVS 5
host - 3270 type terminals 5
host - BATCH region size 5
host - Direct Access storage 5
host - MVS ISPF 5
host - MVS/XA, MVS/ESA, or OS/390 5
host - TSO logon region size 5
host - VSAM 5
server - OS/390 5
EQUATE command 339
Estimate
accessing from ESW screen xxiii
description xxi
ESW
description xix
invoking products xxii
product integration xxiii
EX or X subset 17
EXCLUDE command 340
excluded, subset 17
EXECUTE command 346
EXIT command 350
EXPORT command 351
Extract - Self Directed Server Endpoints pop-up 193

F

File - Analyze Submit pop-up
  displaying 330
  field descriptions 40
FIND command 353
FINDXTND command 358

G

GOBACK command 368
group item 21

H

HELP command 369
help facility
  command help 508
  field help 507
  general help information 509
  help abends 511
  navigational commands 505
  overview 503
  screen and pop-up help 506
  specific help information 510
Help, messages 370
HI subset 17
highlighted subset 17

I

Insight
  accessing from ESW screen xxiii
  description xxi
  using analysis functions xxiv

J

JUMP command 371

K

KEYS command 372

L

label name, description 20
label, name set 20
LEARN Mode - Generated Command
  pop-up 418
LEVELS command 374
line
  number 21
  range 21
  set 21
line range, description 21
LIST command 375
List pop-up 376
LOCATE 377
LOCATE command 377
LPRINT command 379
LPUNCH command 381

M

migration 4

N

NONE or NX subset 17
NONExcluded subset 17
NONH or NHI subset 17
NONHighlighted subset 17

O

operational requirements 5
Options - Log/List/Punch Definition pop-up
  LPUNCH command 381
overview
  action bar 9
  screen 13
  target generation 4

P

paragraph control transfer, methods 525
paragraph name, description 21
PARMDEF command 389
pattern string, description 23
perform range extract 498
perform range name, description 20
perfrange name, description 20
pop-up
  description 11
pop-up screen
  description 8
  how to process 8
PREF command 390
preprocessor support
  command level CICS 6
  command level DL/I 6
  IDMS 6
  other preprocessed languages 6
  SQL 6
PRINTLOG command 393
PRINTLST command 394
problem solving features, solving problems
  with Encore 2
PROCESS command 395
PRODLVL command 399
product integration xxiii
PROGRAM command 400
program name, description 20
Program View screen, LPRINT
  command 380
pseudo code 387

pull-down
  description 8
  format example 11
  how to select actions 8
punch file 381

Q

QUALIFY command 401

R

RECALL command 402
Recap
  accessing from ESW screen xxiii
  description xxi
REDO command 404
REPEAT command 405
report extract, purpose 137
RESET command 406
RETURN command 408
RFIND command 409
RNSCOPY command 410
RSTRUCTURE-VIEW command 411
RTREEVW command 412

S

SAVE command 413
screen
  description 13
  overview 13
  Program View 380
  subsets 17
SCROLL command 22, 414
SELECT command 415
self-directed server
  extract procedure 192
  outputs 201
  paths 199
  startpoints 198
server extract 499
server selection, CICS server selection 188
set
  concatenate 16
  label name 20
  line set 21
  multiple 16
SET command 417
shortened action bar 9
SmartDoc
  accessing from ESW screen xxiii
  description xxi
SmartEdit
  accessing from ESW screen xxiii
  description xxii
SmartQuest
  accessing from ESW screen xxiii
  description xxii
SmartTest
  accessing from ESW screen xxiii
  description xxii
SORT command 419
SOURCE-VIEW command 420
statement extract 500
STRUCTURE-VIEW command 421
subset name, description 20
subsets 16
subsets list 17

T

table entry 21
tags 17
target
  data name 21
  description 20
  pattern 23
  screen subsets 17
  tags 17
target generation, overview 4
transaction extract 501
TREEVIEW command 423

U

UNSELECT command 425

V

VIAANOMF parameter
  ANOMALY 42, 464
  COMMENT(YES or NO) 42, 464
  DSN 42, 464
  example 42
  HEADER(YES or NO) 42, 464
  MEM 42, 464
  PARM(user parm) 42, 464
VIALOG, AKR utility log file 468
VIASUB edit macro 40
VIEW command 427

W

wildcard patterns 16
  member list 16
  selection list 16

Z

ZOOMIN and ZOOMOUT commands 428