

# ASG-Encore™ User's Guide

Version: 6.0

Publication Number: ENX0200-60

Publication Date: February 2002

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.

© 1993 - 2002 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](http://asg.com)

1333 Third Avenue South, Naples, Florida 34102 USA Tel: 941.435.2200 Fax: 941.263.3692 Toll Free: 1.800.932.5536



# ASG Documentation/Product Enhancement Fax Form

Please FAX comments regarding ASG products and/or documentation to (941) 263-3692.

Company Name	Telephone Number	Site ID	Contact name

Product Name/Publication	Version #	Publication Date
<b>Product:</b>		
<b>Publication:</b>		
<b>Tape VOLSER:</b>		

Enhancement Request:



# ASG Support Numbers

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. To expedite response time, please follow these procedures.

## 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 or displayed
- A description of the specific steps that immediately preceded the problem
- The severity code (ASG Support uses an escalated severity system to prioritize service to our clients. The severity codes and their meanings are listed below.)
- Verify whether you received an ASG Service Pack 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.

## If You Receive a Voice Mail Message:

- 1 Follow the instructions to report a production-down or critical problem.
- 2 Leave a detailed message including your name and phone number. A Support representative will be paged and will return your call as soon as possible.
- 3 Please have the information described above ready for when you are contacted by the Support representative.

## 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

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.

## ***Business Hours Support***

<b>Your Location</b>	<b>Phone</b>	<b>Fax</b>	<b>E-mail</b>
<b>United States and Canada</b>	800.354.3578	941.263.2883	support@asg.com
<b>Australia</b>	61.2.9460.0411	61.2.9460.0280	support.au@asg.com
<b>England</b>	44.1727.736305	44.1727.812018	support.uk@asg.com
<b>France</b>	33.141.028590	33.141.028589	support.fr@asg.com
<b>Germany</b>	49.89.45716.222	49.89.45716.400	support.de@asg.com
<b>Singapore</b>	65.332.2922	65.337.7228	support.sg@asg.com
<b>All other countries:</b>	1.941.435.2200		support@asg.com

## ***Non-Business Hours - Emergency Support***

<b>Your Location</b>	<b>Phone</b>	<b>Your Location</b>	<b>Phone</b>
<b>United States and Canada</b>	800.354.3578		
<b>Asia</b>	65.332.2922	<b>Japan/Telecom</b>	0041.800.9932.5536
<b>Australia</b>	0011.800.9932.5536	<b>Netherlands</b>	00.800.3354.3578
<b>Denmark</b>	00.800.9932.5536	<b>New Zealand</b>	00.800.9932.5536
<b>France</b>	00.800.3354.3578	<b>Singapore</b>	001.800.3354.3578
<b>Germany</b>	00.800.3354.3578	<b>South Korea</b>	001.800.9932.5536
<b>Hong Kong</b>	001.800.9932.5536	<b>Sweden/Telia</b>	009.800.9932.5536
<b>Ireland</b>	00.800.9932.5536	<b>Switzerland</b>	00.800.9932.5536
<b>Israel/Bezeq</b>	014.800.9932.5536	<b>Thailand</b>	001.800.9932.5536
<b>Japan/IDC</b>	0061.800.9932.5536	<b>United Kingdom</b>	00.800.3354.3578
		<b>All other countries</b>	1.941.435.2200

## ASG Web Site

Visit <http://www.asg.com>, ASG's World Wide Web site.

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 (941) 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.



---

# Contents

---

<b>Preface</b> .....	<b>vii</b>
<b>About this Publication</b> .....	<b>viii</b>
<b>Related Publications</b> .....	<b>ix</b>
<b>ASG-Existing Systems Workbench (ASG-ESW)</b> .....	<b>x</b>
<b>Invoking ESW Products</b> .....	<b>xiii</b>
<b>ESW Product Integration</b> .....	<b>xiv</b>
<b>Examples</b> .....	<b>xv</b>
<b>Publication Conventions</b> .....	<b>xvii</b>
<b>1 Introduction</b> .....	<b>1</b>
<b>Encore Advantages</b> .....	<b>1</b>
<b>Encore Overview</b> .....	<b>2</b>
<b>Encore Components</b> .....	<b>2</b>
<b>2 Product Overview</b> .....	<b>5</b>
<b>Introduction</b> .....	<b>5</b>
<b>Encore User Interface</b> .....	<b>6</b>
The Action Bar .....	<b>6</b>
<b>Help Facility</b> .....	<b>13</b>
<b>Multiple Views</b> .....	<b>13</b>
<b>Wildcard Patterns</b> .....	<b>14</b>
<b>Understanding COBOL Terms</b> .....	<b>14</b>
Sets .....	<b>14</b>
Subsets .....	<b>15</b>
COBOL Subsets .....	<b>15</b>
Targets .....	<b>18</b>

<b>3 Getting Started</b> .....	<b>23</b>
<b>Introduction</b> .....	<b>23</b>
<b>Starting an Encore Session</b> .....	<b>25</b>
<b>Defining User Options</b> .....	<b>26</b>
Setting Online Operation Parameters .....	27
Allocating Log, List, Punch, and Work Files .....	28
Setting Log/List/Punch Processing Options .....	29
Mapping PF Keys .....	31
Options - Generate Pop-Up .....	32
<b>Allocating an AKR</b> .....	<b>33</b>
Verifying AKR Allocation Results .....	35
<b>The Analyze Facility</b> .....	<b>36</b>
Program Analyze Requirements .....	36
Program Analyze Input .....	36
Analyzing a Program through Encore .....	37
Verifying Analyze Results .....	39
<b>Decomposition</b> .....	<b>40</b>
Defining Your Desired Goals .....	41
Selecting a Program for Decomposition .....	41
Program Cleanup .....	41
Choosing a Code Extraction Objective - The Logic Segment .....	42
Establishing a Project Notebook .....	44
<b>Clean-Up Tasks</b> .....	<b>45</b>
<b>Complement Module Contents</b> .....	<b>45</b>
Perform Range Extract .....	46
Report Extract .....	46
Computation Variable Extract .....	47
Transaction Extract .....	48
Statement Extract .....	48
Server Extract .....	48
<b>The Demonstration Program</b> .....	<b>49</b>
<b>4 Perform Range Extract</b> .....	<b>57</b>
<b>Introduction</b> .....	<b>57</b>
<b>The Business Scenario</b> .....	<b>58</b>
<b>Starting an Encore Session</b> .....	<b>59</b>
<b>Extracting the Perform Range</b> .....	<b>63</b>
Viewing the Logic Segment .....	67
Saving the Logic Segment .....	69

Creating the CALLable Module .....	71
<b>Creating the Complement Module.....</b>	<b>75</b>
<b>Listing of CALCDAYS .....</b>	<b>81</b>
<b>Extracting Multiple Perform Ranges.....</b>	<b>85</b>
Creating the CALLED Module with Multiple Entry Points.....	85
Creating Separate Modules from a Multiple Perform Range Extract.....	92
Reviewing the Generated Modules.....	97
Detecting and Eliminating Multiple Perform Range Common Code.....	99
Creating Complement Modules from Multiple Perform Range Extracts.....	104
<b>Understanding Perform Extracts and Common Code.....</b>	<b>108</b>
<b>Finding Common Code in CALLable Submodules and Complements ...</b>	<b>112</b>
<b>Replacing IO Statements with CALLs to an IO Module.....</b>	<b>115</b>
<b>Generating the IO Module .....</b>	<b>119</b>
<b>Perform Range Extract Compilation Issues .....</b>	<b>121</b>
Condition 1 .....	121
Condition 2 .....	121
Condition 3 .....	124
Condition 4 .....	125
Condition 5 .....	125
Condition 6 .....	126
Condition 7 .....	126
Condition 8 .....	127
Condition 9 .....	127
<b>Complement Module Program Listing .....</b>	<b>128</b>
Example 1 - Complement .....	128
Example 2 - Submodule .....	130
<b>5 Computation Variable Extract.....</b>	<b>133</b>
<b>Introduction .....</b>	<b>133</b>
<b>The Business Scenario.....</b>	<b>134</b>
<b>Starting an Encore Session .....</b>	<b>134</b>
<b>Extracting the Computation Variable.....</b>	<b>135</b>
<b>Viewing the Logic Segment .....</b>	<b>140</b>
<b>Saving the Logic Segment.....</b>	<b>143</b>
<b>Creating Pseudo Source Modules to Change Logic Segment Results.....</b>	<b>145</b>
Pseudo Source Modules .....	145
Including Non-selected Code in the Logic Segment.....	149
<b>Controlling Extract Boundaries.....</b>	<b>152</b>

Extracting Code from an Existing Logic Segment .....	153
<b>Computation Variable Extract Compilation Issues .....</b>	<b>155</b>
Condition 1 .....	155
Condition 2 .....	155
Condition 3 .....	156
Condition 4 .....	156
<b>6 Transaction Extract .....</b>	<b>157</b>
<b>Introduction .....</b>	<b>157</b>
<b>The Business Scenario .....</b>	<b>158</b>
<b>Starting an Encore Session .....</b>	<b>159</b>
<b>Extracting the Objective .....</b>	<b>160</b>
<b>Creating the Replacement Module .....</b>	<b>165</b>
<b>Changing the Start/End Points .....</b>	<b>169</b>
START/END Usage Notes .....	176
<b>Transaction Extract Compilation Issues .....</b>	<b>178</b>
Condition 1 .....	178
Condition 2 .....	179
Condition 3 .....	179
Condition 4 .....	180
Condition 5 .....	180
<b>7 Report Extract .....</b>	<b>181</b>
<b>Introduction .....</b>	<b>181</b>
<b>The Business Scenario .....</b>	<b>182</b>
<b>Starting an Encore Session .....</b>	<b>183</b>
<b>Extracting the Report .....</b>	<b>184</b>
<b>Creating the COBOL Module .....</b>	<b>188</b>
<b>Understanding the Results of the Extract .....</b>	<b>191</b>
<b>Creating the Complement Module .....</b>	<b>194</b>
<b>Complement Module - Understanding Generated Notes .....</b>	<b>198</b>
<b>Using Pseudo Source Modules to Change Logic Segments .....</b>	<b>199</b>
Pseudo Source Modules .....	199
Including Non-selected Code in the Logic Segment .....	203
<b>Controlling Extract Boundaries .....</b>	<b>207</b>
Extracting Code from a Logic Segment .....	207

<b>Report Extract Compilation Issues</b> .....	<b>210</b>
Condition 1 .....	210
Condition 2 .....	210
Condition 3 .....	211
Condition 4 .....	211
Condition 5 .....	211
<b>8 CICS Server Extract</b> .....	<b>213</b>
<b>Introduction</b> .....	<b>213</b>
<b>The Business Scenario</b> .....	<b>214</b>
<b>Conversion Process</b> .....	<b>215</b>
<b>Starting an Encore Session</b> .....	<b>216</b>
<b>Extracting the CICS Server Program</b> .....	<b>217</b>
<b>Generating the CICS Server Module</b> .....	<b>223</b>
<b>Understanding the Results of the CICS Server Module Generation</b> .....	<b>234</b>
<b>CICS Server Extract Compilation Issues</b> .....	<b>242</b>
Condition 1 .....	242
Condition 2 .....	243
Condition 3 .....	244
Condition 4 .....	244
<b>Extracting the Self Directed Server Program</b> .....	<b>245</b>
<b>Using IBM Solutions</b> .....	<b>254</b>
Approach #1: Web-to-CICS Using 3270 Bridge .....	254
Approach #2: Web-to-CICS Using Web-Aware CICS Programs .....	256
Approach #3: Web-to-CICS Using Java .....	257
Available IBM Resources .....	258
<b>Glossary</b> .....	<b>259</b>
<b>Index</b> .....	<b>273</b>



---

## Preface

---

This *ASG-Encore User's Guide* provides user information about 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.

**Note:** \_\_\_\_\_

This guide documents the Encore product. If other ASG-Existing Systems Workbench (herein called ESW) products are installed at your site, features from these products may appear in the Encore product. These features are documented in the specific ESW product guide to which they pertain. For information about these other product features, see the corresponding documentation for that product.

---

## About this Publication

This publication consists of these chapters:

- [Chapter 1, "Introduction,"](#) contains an overview of Encore program re-engineering analysis, extraction, and generation functions.
- [Chapter 2, "Product Overview,"](#) contains an overview of the Encore product, including the user interface, help facility, sets, subsets, and target processing.
- [Chapter 3, "Getting Started,"](#) contains an overview of how to initiate an Encore session, define user options, allocate an Application Knowledge Repository (AKR), analyze a COBOL program, code extraction objectives, and Complement Module contents.
- [Chapter 4, "Perform Range Extract,"](#) contains a practical demonstration of how to perform a Perform Range extract to create a called submodule and a Complement Module.
- [Chapter 5, "Computation Variable Extract,"](#) contains a practical demonstration of how to perform a Computation Variable extract to compute the value of a variable.
- [Chapter 6, "Transaction Extract,"](#) contains a practical demonstration of how to perform a Transaction Extract to create a replacement standalone program.
- [Chapter 7, "Report Extract,"](#) contains a practical demonstration of how to perform a Report extract to:
  - Create a program that performs all of the month-end calculations for the different types of checking accounts without producing the exception report.
  - Create a standalone program that produces the exception report.
- [Chapter 8, "CICS Server Extract,"](#) contains a practical demonstration of how to perform a CICS Server extract to create a COMMAREA-based server program from a 3270 CICS pseudo-conversational program.

## 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 all ASG-ESW 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.

**Note:** \_\_\_\_\_

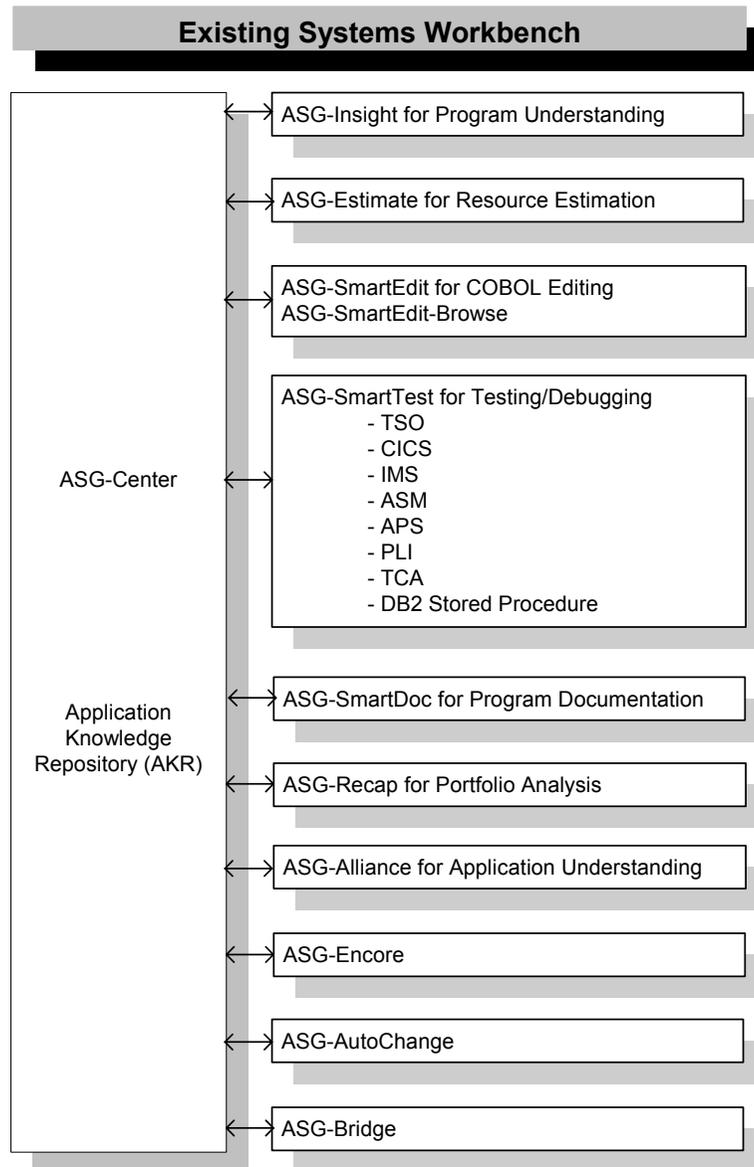
To obtain a specific version of a publication, contact the ASG Service Desk.

---

## 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-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 xiv](#) 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	Shortcut	ESW Pull-down Options
Alliance	AL	Understand ▶ Application
AutoChange	CC	Change ▶ Conversion Set
Bridge	BR	Change ▶ ASG-Bridge
Encore (Re-engineer)	EN	Re-engineer ▶ Program
Estimate	ES	Measure ▶ ASG-Estimate
Insight (Understand)	IN	Understand ▶ Program
Recap (Portfolio Analysis)	RC	Measure ▶ Portfolio
SmartDoc (Document)	DC	Document ▶ Program
SmartEdit	SE	Change ▶ Program <b>Or</b> Change ▶ Program with Options
SmartTest	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, you might see different fields, values, action bar options, and pull-down options 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 2.** [Figure 4](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest directly. [Figure 5](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest through ESW.

Notice that the Analyze features field in [Figure 5](#) lists additional ESW products than shown on [Figure 4](#). This field is automatically customized to contain the ESW products you have installed on your system.

The actions shown on these screens also vary. For example, the D action (ASG-SmartDoc Options) is available on the File - Analyze Submit screen if the SmartDoc product is installed on your system. In [Figure 4](#), the ASG-SmartDoc Options action is not available.

**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 'USER12.REL.CNTL(UIAPCOBC)'

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

AKR data set name 'USER12.GENERAL.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
```

**Figure 5 • File - Analyze Submit Screen (Accessed through ESW)**

```
                                File - Analyze Submit
Command ==> -----
                E - Edit JCL   S - Submit JCL   D - ASG-SmartDoc Options

Compile and link JCL (PDS or sequential):
  Data set name 'USER12.REL.CNTL(HTEST)'

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

AKR data set name 'USER12.GENERAL.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)
```

## Publication Conventions

ASG uses these conventions in technical publications:

<b>Convention</b>	<b>Represents</b>
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).
<i>lowercase italic monospace</i>	Information that you provide according to your particular situation. For example, you would replace <i>filename</i> 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  <u>N</u> ).



---

# 1

## Introduction

---

This chapter describes Encore program re-engineering analysis, extraction, and generation functions, and contains these sections:

Section	Page
<a href="#">Encore Advantages</a>	<a href="#">1</a>
<a href="#">Encore Overview</a>	<a href="#">2</a>

### Encore Advantages

Corporations are increasingly under pressure to perform these functions:

- 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.
- Reengineer 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, and 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, the extracted Logic Segment can be viewed and modified using the COBOL editor.

Encore is designed to support these features:

- 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 Components

Encore is comprised of these four functional components:

- Reengineering analysis
- Code extraction
- Migration
- Target generation

## Reengineering Analysis Overview

Encore re-engineering analysis displays the source code in various levels. The re-engineering analysis component consists of these three displays:

View Methods	Contents of Views
Structure	Displays the hierarchical relationships within the program in a graphical mode. You can zoom in on the actual COBOL source represented by the graphical display.
Tree	Displays the program in logical execution order. This view allows straightforward analysis of how the program works.
Source	Displays the program in source code order and offers COBOL-intelligent browsing capabilities.

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.

## Extract Overview

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:

Objective Type	Purpose
Perform Range	Isolates and extracts a specific named PERFORMed range of code within the COBOL program. See <a href="#">Chapter 4, "Perform Range Extract," on page 57</a> for more details about Perform Range extracts.
Transaction	Isolates and extracts 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 values of a data variable). See <a href="#">Chapter 6, "Transaction Extract," on page 157</a> for more details about Transaction extracts.
Computation Variable	Isolates and extracts all code that calculates a specific data variable from the beginning of the program to a specific statement in the program. See <a href="#">Chapter 5, "Computation Variable Extract," on page 133</a> for more details about Computation Variable extracts.
Report	Isolates and extracts all code necessary to produce a set of identified output (WRITE) statements. See <a href="#">Chapter 7, "Report Extract," on page 181</a> for more details about Report extracts.

Objective Type	Purpose
Statement	Isolates and extracts a user-selected set of COBOL source statements. See <a href="#">Chapter 3, "Getting Started," on page 23</a> for more details about Statement extracts.
Complement	Extracts the original program, excluding statements from a previous extract. See <a href="#">Chapter 4, "Perform Range Extract," on page 57</a> for more details about Complement extracts.
Common Code	Identifies code common to two or more previous extracts. See <a href="#">Chapter 4, "Perform Range Extract," on page 57</a> for more details about Common Code extracts.
CICS Server	Isolates and extracts all code required to generate a CICS server program. See <a href="#">Chapter 8, "CICS Server Extract," on page 213</a> for more details about CICS server extracts.

## 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 Overview

Target generation addresses the disposition of the segment created in the code extraction phase. For target generation, Encore provides these capabilities:

- 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 with 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.

---

# 2

## Product Overview

---

This chapter describes the user interface, help facility, multiple views, wildcard patterns, and COBOL terms, and contains these sections:

Section	Page
<a href="#">Introduction</a>	<a href="#">5</a>
<a href="#">Encore User Interface</a>	<a href="#">6</a>
<a href="#">Help Facility</a>	<a href="#">13</a>
<a href="#">Multiple Views</a>	<a href="#">13</a>
<a href="#">Wildcard Patterns</a>	<a href="#">14</a>
<a href="#">Understanding COBOL Terms</a>	<a href="#">14</a>

### 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 perform these functions:

- Extract the knowledge and investment made in COBOL systems over the years.
- 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. These business functions, or business rules, can then be saved for later use (e.g., enhancement or code reuse).

## 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. An action is selected by moving the cursor to the desired keyword and pressing 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. There are two ways to select an item on a pull-down:
  - Move the cursor to the desired keyword and press Enter.
  - 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.-

**Note:** \_\_\_\_\_

Use the END command (default PF3 or PF15) to exit a pull-down or a pop-up without processing any actions. If the cursor is on the action bar when PF03/15 is pressed, the cursor is moved 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 these features of other products see the corresponding product user's guide or online help.

---

Selecting an action is achieved by moving the cursor with the tab, mouse, or arrow keys to the desired keyword and pressing Enter.



Action	Description
Logic	<p>Displays the Logic pull-down, which is used to follow the execution of a program, searching for a specific target.</p> <p><b>Note:</b> _____ The appearance of the Logic menu option on the action bar depends on whether you have installed other ESW products, such as Insight. If other products have not been installed, Logic does not appear on the action bar.</p> _____
List	<p>Displays the List pull-down, which is used to access prompt list pop-ups that list information about calls, equates, performs, programs, subsets, and segments.</p> _____
Options	<p>Displays the Options pull-down, which is used to customize the Encore environment.</p> _____
Help	<p>Displays the Help pull-down, which is used to access the online help facility.</p> _____

See the *ASG-Encore Reference Guide* for more information about the functions available on these pull-downs.



## 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 [Figures 3](#).

**Figure 3 • Screen Format (Source View Screen)**

```
File View Extract Generate Search List Options Help (A)
-----
Command ==> (D) Source View (B) (C) Program: VIARDEMO
(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	The action bar is displayed at the top of the screen.
B	The name of the screen.
C	VIARDEMO - The name of the active program. A program must be open for this field to be displayed. This area of the screen is also used to temporarily display short informational or error messages.
D	The command input area (command line), used to enter Encore primary commands.
E	<p>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:</p> <ul style="list-style-type: none"> <li>• 1-9999 A number from 1 to 9999 can be entered to specify the number of lines or columns to scroll.</li> <li>• CSR Specifies a scroll value of one page from the cursor location.</li> <li>• DATA Specifies a scroll value of one less than a page.</li> <li>• HALF Specifies a scroll value of one half page.</li> <li>• MAX Specifies a scroll value of the top, bottom, right or left margin.</li> <li>• PAGE Specifies a scroll value of one page.</li> </ul>
F	The long message area contains descriptive, information, or error messages. Long messages are displayed when there are no corresponding short messages, or when HELP is entered in the command input area while a short message is displayed.

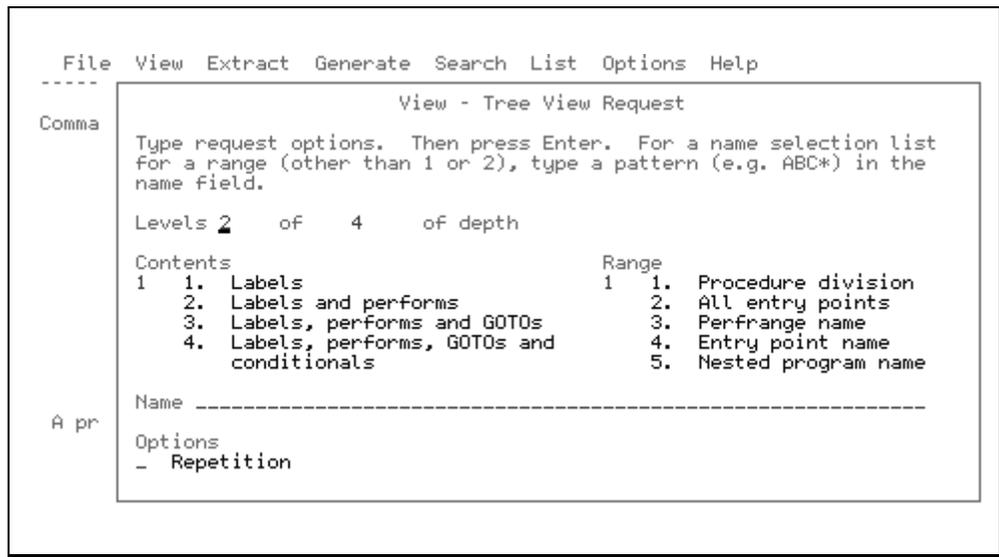
See the online help for more information about Encore commands.

## Pop-Up Screens

[Figures 4](#) show an example of a pop-up screen that displays when you select an item on a pull-down or pop-up, or as the result of entering certain commands. 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. Use the END command (default PF03/15) to exit from a pop-up without processing the actions you have performed.

**Figure 4 • Pop-Up Format (View - Tree View Request Pop-Up)**



## 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 <a href="#">Figure 4 on page 12</a> .
Selection Fields	
Numbered Selection Fields	Allows selection of one item from a list. Enter 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 <a href="#">Figure 4 on page 12</a> .

Field	Description
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 <a href="#">Figure 4 on page 12</a> .
Field Input Area	A field input area is designated by an underline. When a pop-up is first displayed, each field contains its default value. On some pop-ups, if the default is changed the new value is retained for subsequent sessions.

## 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 are displayed, 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. The desired value can be entered on this help screen. When the product screen is redisplayed, the entered value displays in the proper field.

## Multiple Views

The View facility 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 the three available view methods:

View Methods	Contents of Views
Structure	Displays the hierarchical relationships within the program in a graphical mode. You can zoom in on the actual COBOL source represented by the graphical display.
Tree	Displays the program in logical execution order. This view allows straightforward analysis of how the program works.
Source	Displays the program in source code order and offers COBOL-intelligent browsing capabilities.

## Wildcard Patterns

For some Encore screens, there are fields that require the entry of names. If the name field is left blank, a selection list is displayed. Encore provides the ability to enter wildcard patterns to reduce the selection list.

These are the wildcard pattern matching characters:

- The question mark (?) character matches any one character.
- The 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. In addition, sets can be concatenated 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 types of subsets:

- COBOL language subsets
- Screen subsets
- Tagged lines subsets

Use the LIST SUBSETS command or select List ► Subsets to display the List - COBOL Subset Names pop-up, which describes each subset.

## 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 [Tagged Line Subsets](#).

## Screen Subsets

Screen subsets are generally the result of an interactive search request. To specify one of these subsets, enter the entire name or the minimum abbreviation, as indicated by these upper case letters:

```

HIGHLIGHTed = HI
NONHIGHLIGHTed = NONH or NHI
EXCLUDED = EX or X
NONEXCLUDED = NONE or NX

```

## Tagged Line Subsets

Tagged line subsets are displayed in columns 73 through 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.

This tagged line subset, which refers to all lines that have information tags on them, can be used as a subset in commands that accept subsets as targets:

```
TAGged
```

This table contains COBOL subsets that accept tagged line subsets are 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.
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.

COBOL Subsets	Description
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.
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 range that return control.
SORTMerge	SORT/MERGE statements and related IO procedures.

COBOL Subsets	Description
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. The name of any section contained in the DECLARATIVES can also be specified.

### *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 15](#).

### *Line Range*

A line range can be a single line or a group of lines. Line ranges are specified by placing a - (hyphen) between the first and last line numbers in the range (e.g., 214-376).

Line numbers are shown 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 items:

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

Any legal COBOL reference for a data element can be specified as a data name. If a variable is redefined 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.

Fully-qualified data names can be specified by following them with a standard COBOL OF clause, followed 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

Multiple data names can be located at the same time by concatenating 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: <ul style="list-style-type: none"> <li>• P'=' Any character</li> <li>• P'-' Any nonblank character</li> <li>• P'.' Any nondisplay character</li> <li>• P'#' Any numeric character</li> <li>• P'- ' Any non-numeric character</li> <li>• P'@' Any alphabetic character (upper or lowercase)</li> <li>• P'&lt;' Any lowercase alphabetic character</li> <li>• P'&gt;' Any uppercase alphabetic character</li> <li>• P'\$' Any special character (not alphabetic or numeric)</li> </ul>
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

## Getting Started

---

This chapter describes how to initiate an Encore session, define user options, allocate an AKR, analyze a COBOL program, code extraction objectives, and Complement Module contents, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">23</a>
<a href="#">Starting an Encore Session</a>	<a href="#">25</a>
<a href="#">Defining User Options</a>	<a href="#">26</a>
<a href="#">Allocating an AKR</a>	<a href="#">33</a>
<a href="#">The Analyze Facility</a>	<a href="#">36</a>
<a href="#">Decomposition</a>	<a href="#">40</a>
<a href="#">Clean-Up Tasks</a>	<a href="#">45</a>
<a href="#">Complement Module Contents</a>	<a href="#">45</a>
<a href="#">The Demonstration Program</a>	<a href="#">49</a>

### Introduction

This chapter is intended for new Encore users, and describes fundamental tasks, such as invoking an Encore session, and the preliminary tasks you need to perform prior to using Encore.

The steps necessary to invoke Encore vary by site, so check with the systems administrator at your facility for startup and user profile information. Before following along with the sample Encore session presented in this section, you should review [Chapter 1, "Introduction," on page 1](#) and [Chapter 2, "Product Overview," on page 5](#) to become familiar with the terminology used by Encore.

You must define some of Encore's parameters the first time you activate the product. The first part of this section describes these functions:

- Activating an Encore session.
- Setting and verifying these user options from the Options pull-down:
  - Online operations parameters on the Options - Parameters Definition pop-up.
  - Log, List, Punch, and Work file allocations on the Options - Product Allocations pop-up.
  - Log, List, and Punch file processing options and provide a Job card on the Options - Log/List/Punch Definition pop-up.
  - PF key values.
- Allocating an AKR.
- Analyzing a program.

See ["Starting an Encore Session" on page 25](#).

The second part of this section deals with pre and post decomposition tasks and contains a listing of VIARDEMO, one of the demonstration program used in the other task oriented sections of the *ASG-Encore User's Guide*. See ["Decomposition" on page 40](#).

See the online help for detailed information about the screens and fields that are used in these sections.



## Defining User Options

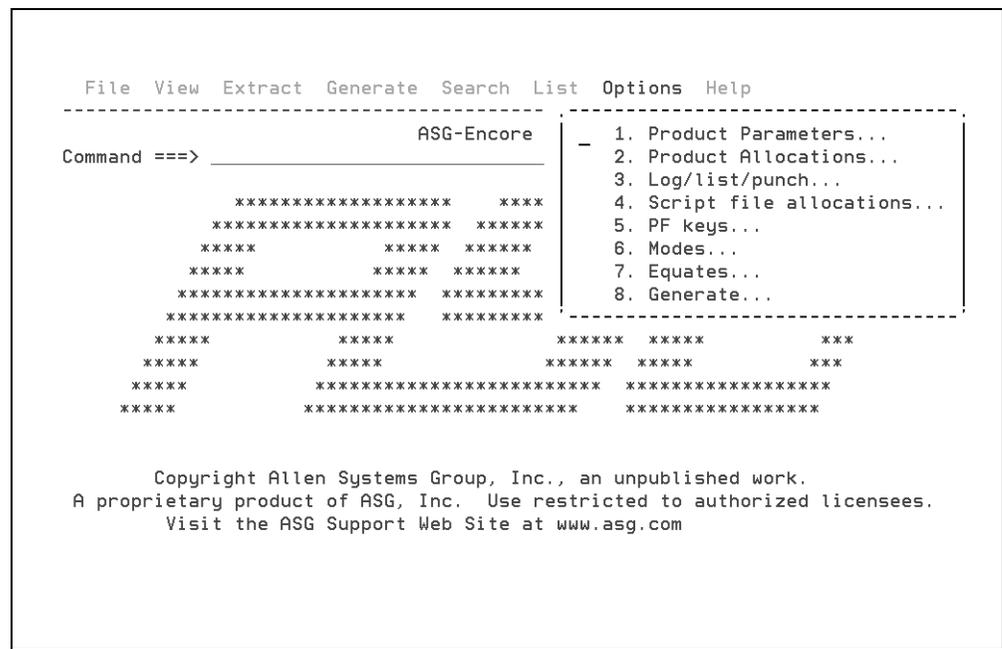
The first time you enter Encore, you should customize the options to reflect the appropriate settings for your environment. Some options are initially set to default values established during product installation by the systems administrator.

[Figure 6](#) shows the Options pull-down, which is used to customize the Encore environment. Customization includes these tasks:

- Setting and verifying online operations parameters on the Options - Product Parameters pop-up (see [Figure 7 on page 27](#)).
- Setting and verifying the Log, List, Punch, and Work file allocations in the Options - Product Allocations pop-up (see [Figure 8 on page 28](#)).
- Setting and verifying Log, List, and Punch file processing options and providing a Job card on the Options - Log/List/Punch Definition pop-up (see [Figure 9 on page 29](#)).
- Setting and verifying PF key values.
- Setting the default view.
- Setting and verifying output and formatting options for generated COBOL modules.

Select Options on the action bar to display the Options pull-down. See the online help for more information about the Options available in Encore.

**Figure 6 • Options Pull-down**





## Allocating Log, List, Punch, and Work Files

Use the Options - Product Allocations pop-up (see [Figure 8](#)) to set or verify the DASD volumes for the Log, List, Punch, and Work files. Some options are initially set to default values established during product installation by the systems administrator.

### To define the Log, List, Punch, and Work file DASD allocations

- 1 Select Options ► Product Allocations and press Enter to display the Options - Product Allocations pop-up.

Figure 8 • Options - Product Allocations Pop-up without SMS

```
File View Extract Generate Search Logic List Options Help
-----
C Command ==> Options - Product Allocations
Log file:
  Generic unit . . . SYSDA      (generic group name or unit address)
  Volume serial . . . _____ (blank for authorized default volume)
List file:
  Generic unit . . . SYSDA      (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 . . . 1        (space units)
  Secondary space . . . 1      (space units)
```

- 2 Type the appropriate information in the fields and press Enter.

See the online help for more information about the fields on this pop-up.

## Setting Log/List/Punch Processing Options

Use the Options - Log/List/Punch Definition pop-up (see [Figure 9](#)) to set the processing values for the Encore Log, List, and Punch files. These files are used for system message logging, error handling, and holding the results of several Encore commands. It is not necessary to exit Encore to process these files. Some options are initially set to default values established during product installation by the systems administrator.

### To specify the Log/List/Punch Definition options

- 1 Select Options ► Log/List/Punch and press Enter to display the Options - Log/List/Punch Definition pop-up.

Figure 9 • 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              PK              K
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:
//NAME JOB (ACCOUNT),'(NAME)',
//      MSGCLASS=A
//*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
-----

```

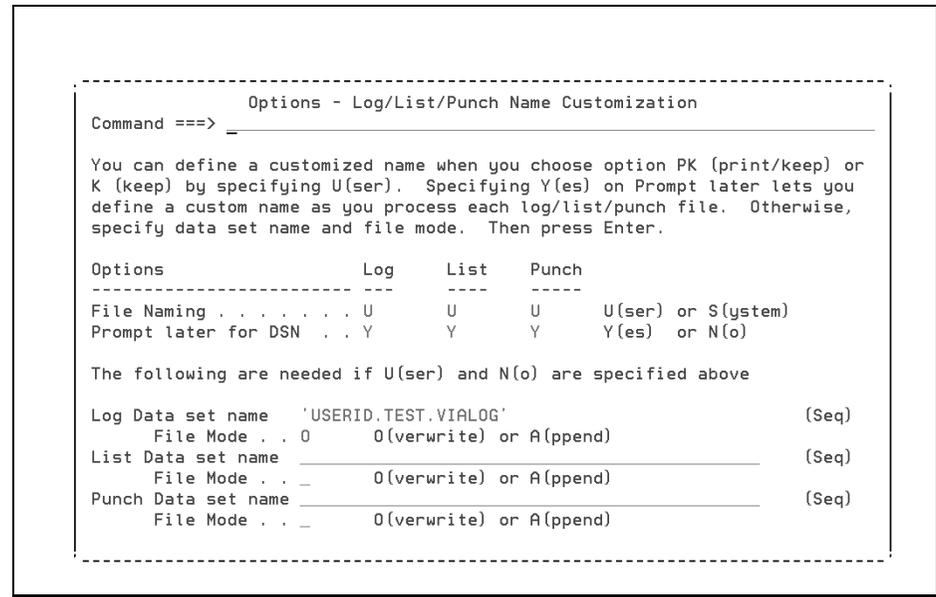
#### Note:

When the PK or PD process option is specified, you must enter a valid job card in the Job statement information field prior to processing any Log, List, or Punch files.

- 2 If you specify the K or PK process option, you can customize the dataset where the log, list, or punch file is allocated. By default, Encore allocates the Log, List, and Punch files as USERID.ENTnnnnn.VIAxxxxxx, where nnnnn is a sequential number from 00001 to 99999 and xxxxxx is LOG for Log, LIST for List, and PUNCH for Punch files. If you have specified a TSO Prefix, the prefix is appended to the beginning of the file name allocated for the Log, List, and Punch files.

- 3 Type 4 on the command line and press Enter to display the Options - Log/List/Punch Name Customization pop-up, as shown in [Figure 10](#).

**Figure 10 • Options - Log/List/Punch Name Customization Pop-up**



- 4 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 Data set name field and specify O (overwrite) or A (append) in the File Mode field.

If you specify Y in the Prompt later for DSN field, Encore prompts you for the dataset name during file processing.

See the online help for more information about the fields on this pop-up.

## Mapping PF Keys

### To display or map your PF Keys

- 1 Select Options ► PF Keys and press Enter to display the Options - PF Key Definition pop-up, as shown in [Figure 11](#).

Figure 11 • Options - PF Key Definition

```

File View Extract Generate Search List Options Help
-----
Options - PF Key (01-12) Definition
C 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 AFIND
PF06 FX %
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 LEFT
PF11 RIGHT
PF12 RECALL

```

- 2 Type the appropriate PF key values in the fields. PF keys 1 through 12 are displayed initially. Press Enter to display PF keys 13 through 24.

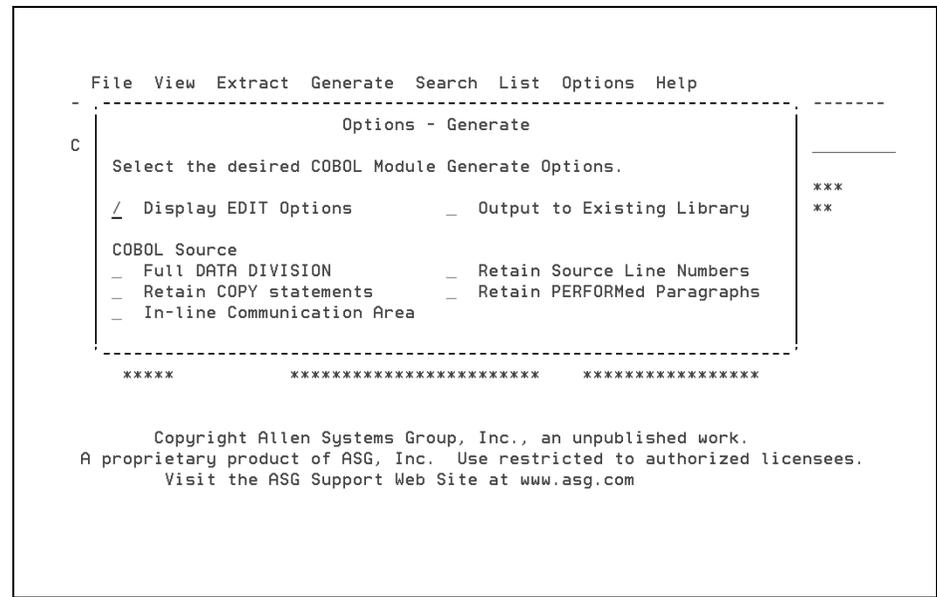
See the online help for more information about the fields on this pop-up.

## Options - Generate Pop-Up

To specify output and format settings for generated COBOL modules

- 1 Select Options ► Generate and press Enter to display the Options - Generate pop-up, as shown in [Figure 12](#).

Figure 12 • Options - Generate Pop-Up



- 2 Type a forward slash (/) in the appropriate fields and press Enter.

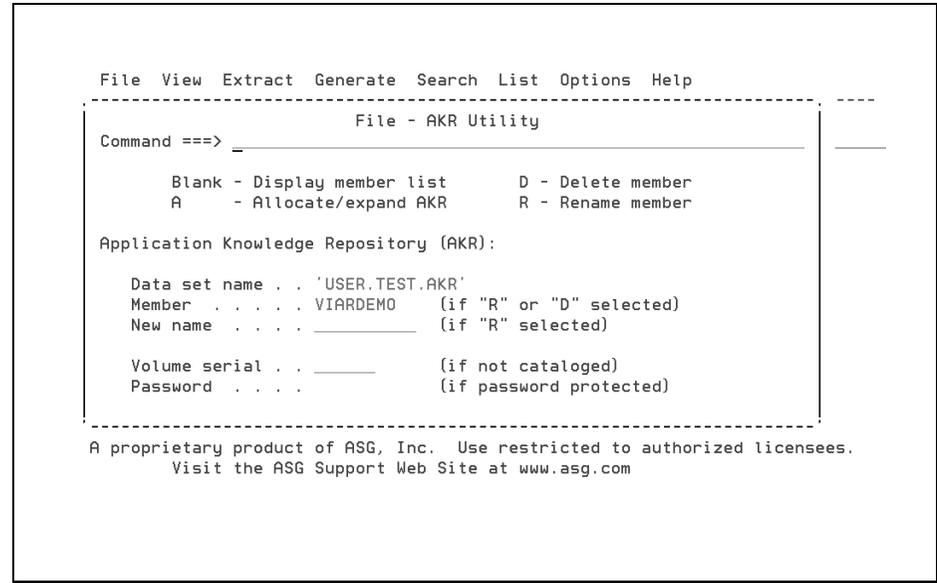
See the online help for more information about the fields on this pop-up.

## Allocating an AKR

### To allocate an AKR

- 1 Select File ► AKR Utility and press Enter to display the File - AKR Utility pop-up, as shown in [Figure 13](#).

Figure 13 • File - AKR Utility Pop-Up



- 2 Specify the dataset name of an AKR in the Data set name field.
- 3 Type A in the command input area and press Enter to display the File - AKR Allocate/Expand pop-up (see [Figure 14 on page 34](#)).

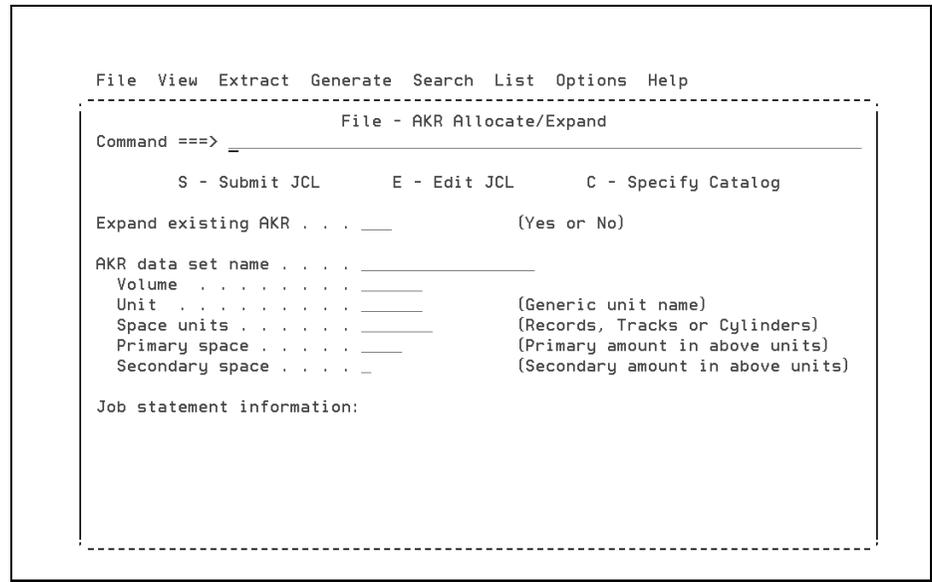
**Note:** \_\_\_\_\_

Four possible pop-ups may display because both BDAM and VSAM AKRs are supported. The pop-up that displays depends on whether you use BDAM or VSAM AKRs and whether SMS is used at your site.

\_\_\_\_\_

- 4 Enter the appropriate information in these required fields (see [Figure 14](#)):
  - a Type No in the Expand existing AKR field.
  - b Specify the AKR name in AKR data set name field.
  - c Specify a valid job card in the Job statement information field.

**Figure 14 • File - AKR Allocate/Expand Pop-up**



See the online help for more information about the fields on this pop-up.

- 5 Submit the JCL to allocate the AKR using one of these methods:
  - a Submit the JCL by typing S on the command line and pressing Enter.

**Or**

  - b Edit the JCL by typing E on the command line and pressing Enter. While in the editor, make the appropriate modifications and enter the TSO command SUBMIT (or SUB). Return to the File - AKR Allocate/Expand pop-up by entering the END primary command or pressing PF3.
- 6 Verify the AKR allocation results by examining the job output.

## Verifying AKR Allocation Results

After the AKR allocation batch job has completed, review the job output to verify successful allocation and initialization. [Figure 15](#) and [Figure 16](#) show output excerpt examples with messages that indicate successful VSAM AKR allocation and initialization.

**Figure 15 • AKR Utility Log - Initialization Message Output**

```
ASG-CENTER-OS (ESA) R n.n LVL # # # AKR UTILITY LOG DD-MMM-YYYY HH: MM: SS PAGE 1
INIT DSNAME (USER.TEST.AKR)
      ASG1316I AKR "USER.TEST.AKR" INITIALIZED.

ASG1314I *** END OF VIASYSIN ***
```

**Figure 16 • AKR Utility Log - Summary Message Output**

```
ASG-CENTER-OS (ESA) R n.n LVL # # # AKR UTILITY LOG - SUMMARY DD-MMM-YYYY HH: MM: SS PAGE 2
      ASG13001      1 AKR (S) INITIALIZED  0 FAILED
      ASG1315I *** END OF SUMMARY REPORT ***
```

## The Analyze Facility

You must analyze a program before Encore can provide intelligent information about it.

The analyze process gathers information about a 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 Encore in an online environment, where it is accessed to provide valuable information about the design and operation of the program.

### Program Analyze Requirements

A Program Analyze is similar to a COBOL compile. Like a compile, these basic program standards are required:

- The correct COBOL language as specified in the *IBM COBOL II Language Reference* and the *IBM COBOL for MVS and VM Guides*.
- Programs that can be compiled without errors by the IBM COBOL II compiler.
  - COBOL II programs that receive error (E), severe (S), or unconditional (U) messages from the IBM compiler cannot be successfully analyzed.

### Program Analyze Input

Input to the Program Analyze function includes these items:

- JCL to compile the program.

The JCL should be the complete JCL used to compile the program, and should perform these steps:

- Retrieve the source from the source manager (such as Librarian or Panvalet)
- Execute any pre-processors
- Invoke the compiler

**Note:** \_\_\_\_\_

Encore does not require the program to be compiled or linked for analysis.  
\_\_\_\_\_

- Program Analyze features that indicate the type of analysis to be performed.
- An initialized AKR to receive Program Analyze output.
- Program Analyze options.

## Analyzing a Program through Encore

The File - Analyze Submit pop-up is used to specify the analyze information and to submit a program to be analyzed through Encore.

**Note:** \_\_\_\_\_

If the compile JCL resides in a source manager such as Librarian or Panvalet, you cannot use this method. See the online help for additional information.

### To analyze a program

- 1 Select File ► Analyze and press Enter to display the File - Analyze Submit pop-up, as shown in [Figure 17](#).

Figure 17 • File - Analyze Submit Pop-up

```

File View Extract Generate Search List Options Help
-----
File - Analyze Submit      Program: VIARDEMO
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)
-----

```

- 2 Enter this information in the required fields:
  - a Specify the PDS member or sequential dataset containing the compile JCL in the Data set name field.
  - b Specify the analyze feature by typing Y beside Encore in the Analyze features (Y/N) fields.

**Note:** \_\_\_\_\_

If Encore is the only product installed, this field may not be changeable and is automatically set to Y. Other analyze features are available on the Analyze Submit pop-up when additional ESW products are installed at your site.

Optionally, specify analysis report parameters that are used to produce analysis reports that help you understand and diagnose analysis problems and anomalies. You can also enter a different analyze parameter to use the Anomaly Repair Facility to correct certain anomalies that may exist within your code. These are the available parameters:

- 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 save the information in the AKR.
  - NUMPRM(##) - Determines the number of parameters to 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 parameter:
  - 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 accessed from the File pull-down menu. This parameter can be followed by additional anomaly report parameters, as shown in this example:

```
VIAANOMF , PARM (DEBUG1) , ANOMALY (OOPGT , GTF , IPR) ,  
COMMENT (YES) , DSN ( ' ASG . ANOMALY . REPAIRED . CODE ' ) ,  
MEM (MEM1) , HEADER (YES)
```

See the online help and the *ASG-Encore Reference Guide* for more information about these parameters and the Analyze option field.

**c** Specify the AKR dataset name in the AKR data set name field.

**3** Submit the JCL to compile and/or analyze by using one of these methods:

**a** Typing **S** in the command input area and pressing Enter.

**Or**

**b** Edit the compile JCL by typing **E** in the command input area and pressing Enter. While in the editor, make the appropriate modifications and issue the TSO command SUBMIT (or SUB). Press PF3 to return to the File - Analyze Submit pop-up.

**4** Verify the analyze results by examining the job output.

## Verifying Analyze Results

After the Analyze batch job has completed, review the job output to verify results. The output should be checked for these results:

- Acceptable compiler results, if requested.
- Messages indicating the program has been successfully analyzed and stored in the AKR. Storage in the AKR does not occur if the program does not analyze successfully.

[Figure 18](#) is an example of an output excerpt indicating a successful analyze.

**Figure 18 • Example of an Output Excerpt from a Successful Analyze Job**

```

ASG-CENTER-OS (ESA) Rn.n LVL###          PROGRAM: VIARDEMO          DD-MMM-YYYY HH: MM: SS PAGE 1

LINE ERROR MESSAGE

(A) ASG0248I PROGRAM 'VIARDEMO' WAS STORED IN AKR 'USER.TEST.AKR'.

(B)

DIAGNOSTICS LINES:  0 TOTAL - 0 WARNINGS, 0 ERRORS, 0 SEVERE ERRORS, 0 UNRECOVERABLE ERRORS
SOURCE LINES:      434 TOTAL - 133 DATA DIVISION STATEMENTS, 108 PROCEDURE DIVISION STATEMENTS
PARAMETERS PASSED: COBOLII, COB2R3, FEATURES=(EN)
OPTIONS IN EFFECT: BUFMAXR=4096K, COBOLII, FEATURES=(ASG-ENCORE), FLAG (W), LINECNT=55, NORECUR, NOSEQ,
NOSOURCE, SPACE1

ENTRY POINTS:      VIARDEMO

EXTERNAL CALLS:

END OF PROCESSING: DD-MMM-YYYY   HH: MM: SS

```

where:

Field	Description
A	A message (ASG0248I) indicating that program VIARDEMO was successfully stored in the AKR.
B	The area that contains any associated program diagnostics.

## Decomposition

Decomposition is the process of separating an applications code into smaller, related subgroups that are more manageable. These subgroups are fully functional and retain limited interaction with other decomposed subgroups.

This section explains the preparation process for using Encore. The explanation also describes the types of Logic Segments that can be extracted and the contents of Complement Modules.

Before you can begin decomposition, you must perform these preliminary tasks:

- ["Defining Your Desired Goals" on page 41](#).
- ["Selecting a Program for Decomposition" on page 41](#).
- ["Program Cleanup" on page 41](#).
- ["Choosing a Code Extraction Objective - The Logic Segment" on page 42](#).
- ["Establishing a Project Notebook" on page 44](#).

You should be familiar with these terms when using Encore:

- Logic Segment
- Complement Module
- Data name references
- Data name aliases
- Subsets

See [Chapter 2, "Product Overview" on page 5](#) or the online help for a complete definition of these terms.

### ***Defining Your Desired Goals***

You must define your desired goals for re-engineering. These are some common goals:

- Extracting common code into a standalone, shared module.
- Splitting programs into smaller, easier to maintain modules.
- Reducing or eliminating GOTOs.
- Eliminating obsolete code from programs.
- Building a library of common or reusable code.

Defining an objective promotes a common vision of what is to be accomplished by all project participants. This common vision guarantees project acceptance at all levels and helps to propagate subsequent decomposition activity.

### ***Selecting a Program for Decomposition***

For your initial decomposition exercise, it is recommended that you select a medium sized program. The best type of program would be one that is of strategic value to your company, in need of re-engineering, and contains multiple functions. Ideally, the program should also contain code common to several other programs for the purpose of eventually isolating that common code into a standalone or callable module.

There is a learning curve for understanding Encore, so you want to avoid complex programs that may overwhelm the project team and not show success in a relatively short period of time. Expertise comes with repetition and each program decomposition serves as a building block of knowledge for the next one.

### ***Program Cleanup***

Before analyzing Encore, you need to resolve existing program abnormalities whenever possible. These would consist of live exits, out of PERFORM transfers, and dead code or dead data. Also give 01 Fillers in the Data Division meaningful names so that data names used for CALLs between complement and segment modules can be more easily reconciled later on.

## Choosing a Code Extraction Objective - The Logic Segment

In decomposition, it is sometimes difficult to be able to determine which extract objective to use. One method you can use is to determine the best way to focus on the code you want to change. You can perform these types of code extracts:

- Perform Range extract
- Transaction extract
- Computation Variable extract
- Report extract
- Statement extract
- CICS server extract

### Perform Range Extract

A Perform Range extract is all of the code that could be executed by a PERFORM statement. It includes all of the statements in the paragraph range of the PERFORM, plus any subordinate code that is PERFORMed or reachable by GOTOs within this paragraph range.

If the program is already well structured, the logic in question could already be inside one or more PERFORM ranges. A good candidate for the PERFORM range extract should possess these characteristics:

- Contains or invokes a large amount of code.
- Performs many times or from many other units.
- Contains all of the IO statements for a certain file.
- Contains many nested PERFORMs that are not shared by others, so that it is well isolated from the rest of the program.
- Contains a small number of 01-level structures that are serviced only by this code.

A Perform Range extract can be used to reduce the size of a program, dividing it into separately called subroutines that can be separately compiled.

See [Chapter 4, "Perform Range Extract," on page 57](#) for more information about Perform Range extracts.

### Transaction Extract

A Transaction extract contains all of the code that can be reached from a designated starting point to a designated ending point when selected conditions are true or false as you have stated they would be. A typical example of its use is to extract the code that is executed when a certain transaction code variable has a specific value. You would first select the variable and Encore would then display the conditional statements associated with that variable. The logic path extracted would depend on whether you specify those conditional statements to be only true, only false, or true or false.

You can also use a transaction extract to simplify a large program that does not use PERFORMs, in which case the code falls through from top to bottom. If you can identify both a starting point and an ending point for the portion of the program that you are interested in, you can isolate all of the code that is executed within those boundaries.

See [Chapter 6, "Transaction Extract," on page 157](#) for more information about transaction extracts.

### Computation Variable Extract

A Computation Variable extract locates the minimum set of statements that contribute to the value of a data variable at a point in the program, while excluding all other code in the program. For example, [Figure 19](#) shows an extract of the ACTION-CODE variable at line 80.

**Figure 19 • Computation Variable Extract Example**

```

0010 ACCEPT ACTION-PARM-A.
0020 ACCEPT ACTION-PARM-B.
0030 ACCEPT ACTION-PARM-C.
0040 IF ACTION-PARM-A = 7 ADD 1 TO ACTION-PARM-C.
0050 IF ACTION-PARM-A = 8 SUBTRACT 2 FROM ACTION-PARM-B.
0060 COMPUTE ACTION-PARM-A = ACTION-PARM-A + 1.
0070 COMPUTE ACTION-CODE = ACTION-CODE -C + 2
0080 DISPLAY ACTION-CODE.
0090      . .
0100      . .

```

Lines 10, 30, 40, 70 and 80 are extracted because of these reasons:

- The value of ACTION-CODE at line 80 is determined by the computation at line 70, which depends on the variable ACTION-PARM-C.
- The value of ACTION-PARM-C depends on lines 30 and 40.
- The value of ACTION-PARM-C at line 40 depends on the value of ACTION-PARM-A, which is determined at line 10.

A Computation Variable extract can also be used to create a called module that calculates the value of a data variable, or determines the actual set of computations used to set a variable.

See [Chapter 5, "Computation Variable Extract," on page 133](#) for more information about Computation Variable extracts.

### **Report Extract**

A Report extract locates the minimum set of statements that contribute to the content of an output file once you have identified the IO statements that produce the file. It is ideally suited to extract IO statements and associated data to create a standalone program that creates only that file, or to create a standalone program that omits the creation of the extracted file.

See [Chapter 7, "Report Extract," on page 181](#) for more information about Report extracts.

### **Statement Extract**

A Statement extract should be used only when no other type of extract objective is suitable. This would be the case if the code in a program is so convoluted and intertwined that it makes other extract objectives impossible. The Statement extract allows you to select an arbitrary set of unrelated statements at your discretion.

### **CICS Server Extract**

A 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.

See [Chapter 8, "CICS Server Extract," on page 213](#) for more information about CICS server extracts.

## **Establishing a Project Notebook**

At this point, you should establish a project notebook for step-by-step documentation of the ongoing process. This documentation becomes a flexible, working document for future Encore decomposition projects within your organization. The project notebook serves as an excellent source for preparing internal decomposition presentations that are likely to follow project completion. If properly maintained, a project notebook provides substantial time savings for future decomposition projects.

These are other sections to include in the project notebook:

- The Troubleshooting Log should contain entries for all technical problems encountered and the steps taken to resolve them.
- The Issues Log should chronologically list all functional or procedural issues resulting from decomposition activity. If properly maintained, these logs substantially reduce the required completion time of subsequent decomposition requests.

## Clean-Up Tasks

Once the extract portion of the decomposition project is completed, check to see that the extracts were successful by verifying these results:

- The resolution of any execution and compilation issues.
- The revision of JCL to reflect the changes for both the affected programs and the system flow.
- Verification that thorough system integration testing has been performed.

## Complement Module Contents

A Complement Module is the original program with the Logic Segment removed. The purpose of the Complement Module Generation facility is to accurately remove logic from a program.

Logic is usually removed from a program for one of these reasons:

- The function is no longer necessary.
- The function is processed elsewhere. In this case, the logic may be in a completely separate program, or may be processed in a CALLED module.

In either case, the code can be removed from the original program.

The accurate definition of a Complement Module depends on the extract objective selected. The next section describes the complement produced for each extract objective type.

## Perform Range Extract

[Figure 20](#) shows a complement that contains all of the statements from the original program, excluding all of the statements of the Perform Range that are not required by other logic paths. For example, the original program contains this code:

**Figure 20 • Complement Module - Perform Range Extract Example**

```

PERFORM A.
PERFORM B.
. . .
A.
PERFORM B.
B.
PERFORM C.
C.
. . .
    
```

If A was the Perform Range extracted, the Complement Module would contain all code except paragraph A, since B and C are also required for a separate execution path.

In addition, the Complement Module replaces a PERFORM of the extracted range with a CALL statement. A PERFORM VARYING is replaced with a PERFORM of an Encore-generated SECTION containing the CALL statement.

## Report Extract

[Figure 21](#) shows a Complement Module for a report extract objective containing all statements of the original program, excluding statements of the Logic Segment not required by other logic paths. For the report extract objective, all identified WRITE/CALL statements are removed. The resulting program is capable of performing all functions except the function defined by the Logic Segment. For example, a program could contain this code:

**Figure 21 • Complement Module - Report Extract Example**

```

0010  MOVE VAR-1 TO B.
0020  MOVE A TO DETAIL1-FIELD-A.
0030  MOVE B TO DETAIL2-FIELD-B.
0040  MOVE B TO DETAIL1-FIELD-B.
0050  WRITE DETAIL1.
0060  WIRTE DETAIL2.
    
```

In this example, lines 10, 20, 40 and 50 represent the Logic Segment for DETAIL1. The complement of this Logic Segment would be lines 10, 30, and 60. Note that line 10 is common to another path within the program and, therefore, cannot be removed.

## Computation Variable Extract

The Complement Module for a Computation Variable extract contains all statements of the original program, excluding statements of the positive Logic Segment not required by any other logic paths. Because a Computation Variable derivation may be started at any point in the program, each of these examples and the associated paths are possible. Each example is based on the type and context of the statement selected as the end point of the variable.

### Example 1

The selected statement is a logical termination of the selected data variable value. Typically, these are OUTPUT statements and no further uses of the variable are dependent on the derived value. The resulting Complement Module excludes all code (from and including the selected statement) back to the beginning of the program that contributes to or influences the value of the Computation Variable, but retains any code needed for other functional paths. For example:

```
DISPLAY data name
```

The resulting Complement Module excludes all code from, and including, the DISPLAY statement to the beginning of the program that contributes to or influences the value of the data name variable, but retains any code needed for other functional paths.

### Example 2

The selected statement is a modification or assignment of the data variable selected as the Computation Variable.

An Encore-generated comment indicates that the value of the Computation Variable needs to be available at this point in the program so that subsequent dependent statements (if any) that depend on the variable execute properly. For example:

```
COMPUTE data name = A + B.  
  
DISPLAY data name.
```

The resulting Complement Module excludes all code from, and including, the COMPUTE statement to the beginning of the program that contributes to or influences the value of the data name variable, but retains any code needed for other functional paths. The COMPUTE statement is retained as a comment.

### Example 3

The selected statement is a use of the data variable selected as the Computation Variable in an assignment statement. An Encore-generated comment indicates that the use of the Computation Variable is no longer applicable in the statement, and that the statement should be altered to no longer reference the Computation Variable. For example:

```
COMPUTE VAR-1 = A + data name.
```

The resulting Complement Module excludes all code from, and including, the COMPUTE statement back to the beginning of the program that contributes to or influences the value of the data name variable, but retains any code needed for other functional paths. The COMPUTE statement is retained as a comment.

### Example 4

The selected statement is a conditional statement that references the data variable selected as the Computation Variable. An Encore-generated comment is inserted indicating that the condition being tested must be altered so that it no longer depends on a value of the Computation Variable so that the IF statements are correctly invoked. For example:

```
IF data name EQUAL A THEN...
```

The resulting Complement Module excludes all code from, and including, the IF statement back to the beginning of the program that contributes to or influences the value of the data name variable, but retains any code needed for other functional paths. The IF statement is retained as a comment.

### Transaction Extract

The Complement Module for the Transaction extract is defined by identifying the conditional paths that definitely are, and those that definitely are not, part of the transaction complement. The transaction complement should be defined to include the code required to process all transactions other than the one(s) defined for the transaction objective.

### Statement Extract

The Complement Module for the statement selection is defined as the physical complement. The complement contains all statements not selected on the Statement Selection screen.

### Server Extract

The Complement Module for the CICS server extract is defined as the business logic component that can be invoked from a web server or other types of clients. The complement contains all statements required to define a server-side application.

## The Demonstration Program

VIARDEMO is one of the Encore demonstration programs that is used in various examples to demonstrate the extraction types. It does the month-end calculations for each type of checking account carried by the Universal Bank.

**Note:** \_\_\_\_\_

VIARBRWS is the Encore demonstration program used for CICS server extraction and generation. See [Chapter 8, "CICS Server Extract," on page 213](#) for more information.

VIARDEMO processes these two input files:

- The Checking Account master file, a keyed random access file that contains the records for all checking account customers.
- The Transaction file, which is a sequential file used to update the Checking Account master.

VIARDEMO generates these two output files:

- An Action file created from the processed transactions that is used as input to another program.
- The Exception report file that reports all errors encountered during processing.

[Figure 22](#) shows an example of the type of file output generated from the VIARDEMO program.

**Figure 22 • Example of File Output from the Demonstration Program**

```

000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID, VIARDEMO.
000003 AUTHOR, ASG 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
000017 ENVIRONMENT DIVISION.
000018 CONFIGURATION SECTION.
000019 SOURCE-COMPUTER, IBM-370.
000020 OBJECT-COMPUTER, IBM-370.
000021 INPUT-OUTPUT SECTION.
000022
000023 FILE-CONTROL.
000024 SELECT ACCT-FILE ASSIGN TO ACCT
000025 ORGANIZATION IS INDEXED
000026 ACCESS IS RANDOM
000027 RECORD KEY IS ACCT-KEY.
000028 SELECT EXCEPTION-FILE ASSIGN TO UT-S-EXCEPTS.
000029 SELECT TRAN-FILE ASSIGN TO UT-S-TRAN.
000030 SELECT ACTION-FILE ASSIGN TO UT-S-ACTIONS.

```

[Figure 23](#) shows how VIARDEMO is called from another program that passes the control parameters needed for processing.

**Figure 23 • Example of Control Parameter Definitions for Demonstration Program Processing**

```

LINKAGE SECTION.

01 CONTROL-PARAMETERS.
   05 FREE-SERVICE-MIN          PIC S9(11)V9(4) COMP-3.
   05 CD-REVIEW-MIN             PIC S9(11)V9(4) COMP-3.
01 BRANCH-BALANCE              PIC S9(11)V9(4) COMP-3.
01 BANK-BALANCE                PIC S9(11)V9(4) COMP-3.
    
```

[Figure 24](#) shows how the PROCEDURE DIVISION is written in a structured format, with all of the processing initiated by PERFORM statements.

**Figure 24 • Example of the PROCEDURE DIVISION for the Demonstration Program**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARWxx.DEVL.CNTL(VIARDEMO) - xx.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
019800 012754 PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,
019900 012760                                     BANK-BALANCE.
020000 012770
020100 012780     OPEN OUTPUT EXCEPTION-FILE.
020200 012790
020300 012800     PERFORM INITIALIZE-PGM.
020400 012900
020500 013000     PERFORM ACCT-MAINTENANCE
020600 013100             THRU ACCT-MAINTENANCE-EXIT
020700 013200             UNTIL TRAN-FILE-ENDS.
020800 013300
020900 013400     PERFORM REPORT-FINAL-CNTRS.
021000 013500
021100 013600     PERFORM CLOSE-PGM.
021200 013700
021300 013800     GOBACK.
021400 013900
021500 014000
021600 014100 ACCT-MAINTENANCE.
    
```

[Figure 25](#) shows how the INITIALIZE-PGM paragraph opens the files and verifies the parameters passed to VIARDEMO. It also shows how it performs paragraph GET-NUM-OF-DAYS, which determines the number of days in the current accounting period.

**Figure 25 • Example of the INITIALIZE-PGM Paragraph**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARNxx.DEVL.CNTL(VIARDEMO) - xx.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
034600 024200 INITIALIZE-PGM.
034700 024300
034800 024400      PERFORM OPEN-FILES.
034900 024500
035000 024600      MOVE  +0          TO ACCT-IO-CNT
035100 024700                      TRAN-CNT
035200 024800                      ACTION-CNT
035300 024900                      WORK-RETURN-CODE.
035400 025000
035500 025100      IF  FREE-SERVICE-MIN NOT NUMERIC
035600 025200      MOVE +0          TO FREE-SERVICE-MIN.
035700 025300      IF  CD-REVIEW-MIN   NOT NUMERIC
035800 025400      MOVE +0          TO CD-REVIEW-MIN.
035900 025500      IF  BRANCH-BALANCE  NOT NUMERIC
036000 025600      MOVE +0          TO BRANCH-BALANCE.
036100 025700      IF  BANK-BALANCE   NOT NUMERIC
036200 025800      MOVE +0          TO BANK-BALANCE.
036300 025900
036400 026000      MOVE ZEROES      TO WORK-ACTION-REASON.

```

[Figure 26](#) shows how the ACCT-MAINTENANCE paragraph matches the Checking Account master and the transaction file, and how it applies maintenance to the Checking Account master by performing paragraph UPDATE-ACCT THRU UPDATE-ACCT-EXIT. Processing errors encountered are written to the Exception report.

**Figure 26 • Example of the ACCT-MAINTENANCE Paragraph**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARNxx.DEVL.CNTL(VIARDEMO) - xx.01          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
021600 014100 ACCT-MAINTENANCE.
021800 014300     READ TRAN-FILE
021900 014400         AT END
022000 014500             MOVE EOF-LITERAL TO TRAN-FILE-FLAG
022100 014600             GO TO ACCT-MAINTENANCE-EXIT.
022300 014800     ADD +1 TO TRAN-CNT.
022500 015000     MOVE TRAN-ACCT-KEY TO ACCT-KEY.
022700 015200     READ ACCT-FILE KEY IS ACCT-KEY
022800 015300         INVALID KEY
022900 015400             MOVE ACTION-MISSING-ACCT TO WORK-ACTION-REASON
023000 015500             MOVE SPACES TO ACCT-RECORD
023100 015600             PERFORM INITIATE-ACTION
023200 015700             MOVE +4 TO WORK-RETURN-CODE
023300 015800             GO TO ACCT-MAINTENANCE-EXIT.
023600 016100     ADD +1 TO ACCT-IO-CNT.
023700 016200     PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT.
023900 016400     IF ACCT-MIN-BALANCE IS LESS THAN ACCT-BALANCE
024000 016500         PERFORM EXCEPTION-REPORT-1.
024200 016700 ACCT-MAINTENANCE-EXIT.
024300 016800     EXIT.

```

[Figure 27](#) shows how the UPDATE-ACCT paragraph calculates the interest payment to be applied on interest bearing checking accounts. It also shows how it applies the monthly service charge for each account, if applicable, and outputs a record to the Action file. The conditional IF ACCT-TYPE OF ACCT-RECORD = 'MMA' is used to illustrate the Transaction extract described in ["Transaction Extract" on page 48](#).

**Figure 27 • Example of the UPDATE-ACCT Paragraph**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARNxx.PROD.CNTL(VIARDEMO) - xx.00          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
024600 017100 UPDATE-ACCT.
024700 017200
024800 017300          PERFORM INIT-ACTION-RECORD.
024900 017400
025000 017500          IF ACCT-TYPE OF ACCT-RECORD = 'INT'
025100 017600              MOVE CHK-INTEREST-RATE TO INTEREST-RATE
025200 017700          ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
025300 017800              MOVE ZERO TO INTEREST-RATE
025400 017900          ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
025500 018000              MOVE MMA-INTEREST-RATE TO INTEREST-RATE
025600 018100          ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
025700 018200              MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
025800 018300
025900 018400          IF ACCT-MULTI-ACCT = 'YES'
026000 018500              ADD CHK-MULTI-ACCT-RATE TO INTEREST-RATE
026100 018600              MOVE ACTION-MULTI-ACCT TO WORK-ACTION-REASON
026200 018700              PERFORM INITIATE-ACTION.
026300 018800
026400 018900          IF ACCT-MIN-BALANCE < FREE-SERVICE-MIN
026500 019000              MOVE SERVICE-INTEREST-RATE TO INTEREST-RATE

```

[Figure 28](#) shows how the code is used to illustrate the Computation Variable extract described in "[Computation Variable Extract](#)" on page 47.

**Figure 28 • Example of the Computational Variable Extract**

```
IF ACCT-MIN-BALANCE < FREE-SERVICE-MIN
  MOVE SERVICE-INTEREST-RATE TO INTEREST-RATE
  COMPUTE ACCT-SERVICE-CHARGE = SERVICE-CHARGE
  MOVE ACTION-BELOW-MIN TO WORK-ACTION-REASON
  PERFORM INITIATE-ACTION
ELSE
  MOVE ZEROS TO ACCT-SERVICE-CHARGE.

IF ACCT-MIN-BALANCE < 0
  MOVE ACTION-NEG-BALANCE TO WORK-ACTION-REASON
  PERFORM INITIATE-ACTION.

COMPUTE WK-INTEREST =
  (ACCT-AVG-BALANCE / DAYS-IN-YEAR) *
  DAYS-IN-PERIOD * (INTEREST-RATE / 100).

MOVE WK-INTEREST TO ACCT-INTEREST.
ADD ACCT-INTEREST TO ACCT-BALANCE.
ADD ACCT-INTEREST TO ACCT-INT-YTD.

SUBTRACT SERVICE-CHARGE FROM ACCT-BALANCE.
SUBTRACT ACCT-INTEREST FROM BRANCH-BALANCE.
SUBTRACT ACCT-INTEREST FROM BANK-BALANCE.

ADD SERVICE-CHARGE TO BRANCH-BALANCE.
ADD SERVICE-CHARGE TO BANK-BALANCE.

PERFORM INITIATE-ACTION.

MOVE TODAY-DATE TO ACCT-LAST-UPDATE.

REWRITE ACCT-RECORD.

UPDATE-ACCT-EXIT.
EXIT.
```

[Figure 29](#) shows how the EXCEPTION-REPORT-1 paragraph lists all accounts with a minimum balance that is less than the current balance. The highlighted code is used to illustrate the report extract described in ["Report Extract" on page 46](#).

**Figure 29 • Example of the EXCEPTION-REPORT-1 Paragraph**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARNxx.PROD.CNTL(VIARDEMO) - xx.x          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
030000 021870 EXCEPTION-REPORT-1.
030100 021880*
030200 021890*   EXCEPTION REPORT 1 LIST ALL ACCOUNTS WITH A MINIMUM BALANCE
030300 021900*   THAT IS LESS THAN THE CURRENT BALANCE.
030400 022000*
030500 022100   IF EX-RPT-LINE-CNT IS GREATER THAN 60
030600 022200   ADD +1 TO EX-RPT-PAGE-CNT
030700 022300   MOVE EX-RPT-PAGE-CNT TO EXCEP1-REPORT-PAGE-CNT.
030800 022400   WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE1.
030900 022401   WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE2.
031000 022402   MOVE 0 TO EX-RPT-LINE-CNT.
031100 022403   ADD +1 TO EX-RPT-LINE-CNT.
031200 022404   MOVE ACCT-BALANCE TO EX-CUR-BAL.
031300 022405   MOVE ACCT-MIN-BALANCE TO EX-REPORT-MIN-BAL.
031400 022406   WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE1.
031500 022407   WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE2.
031600 022408

```

[Figure 30](#) shows how the GET-NUM-OF-DAYS paragraph calculates the number of days in the current accounting period. This code is used to illustrate the paragraph extraction described in ["Perform Range Extract" on page 46](#).

**Figure 30 • Example of the GEN-NUM-OF-DAYS Paragraph**

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          VIARNxx.PROD.CNTL(VIARDEMO) - xx.x          Columns 00001 00072
Command ==>                                         Scroll ==> CSR
033100 022700 GET-NUM-OF-DAYS.
033200 022800*
033300 022900   ACCEPT TODAY-DATE FROM DATE.
033400 023000
033500 023100   COMPUTE DAYS-IN-PERIOD =
033600 023200   MONTH (TODAY-MM) - TODAY-DD.
033700 023300   MOVE +0 TO DAYS-IN-YEAR.
033800 023400   PERFORM SUM-TOTAL-DAYS
033900 023500   VARYING WORK-CNT FROM 1 BY 1
034000 023600   UNTIL TODAY-DD = WORK-CNT.
034100 023700*
034200 023800 SUM-TOTAL-DAYS.
034300 023900   COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ).
034400 024000*

```



---

# 4

## Perform Range Extract

---

This chapter contains a practical demonstration of how to use a Perform Range extract to create a CALLED submodule and a Complement Module, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">57</a>
<a href="#">The Business Scenario</a>	<a href="#">58</a>
<a href="#">Starting an Encore Session</a>	<a href="#">59</a>
<a href="#">Extracting the Perform Range</a>	<a href="#">63</a>
<a href="#">Creating the Complement Module</a>	<a href="#">75</a>
<a href="#">Listing of CALCDAYS</a>	<a href="#">81</a>
<a href="#">Extracting Multiple Perform Ranges</a>	<a href="#">85</a>
<a href="#">Understanding Perform Extracts and Common Code</a>	<a href="#">108</a>
<a href="#">Finding Common Code in CALLable Submodules and Complements</a>	<a href="#">112</a>
<a href="#">Replacing IO Statements with CALLs to an IO Module</a>	<a href="#">115</a>
<a href="#">Generating the IO Module</a>	<a href="#">119</a>
<a href="#">Perform Range Extract Compilation Issues</a>	<a href="#">121</a>
<a href="#">Complement Module Program Listing</a>	<a href="#">128</a>

### Introduction

In addition to the Perform Range extract demonstration, this chapter also discusses execution issues and compilation issues.

The Encore demonstration program VIARDEMO is used for the examples in this chapter. For a partial listing of VIARDEMO, see ["The Demonstration Program" on page 49](#).

## The Business Scenario

Your manager at the Universal Bank tells you that some new programs are being written for a loan calculation and they need to call a routine to determine the number of days in the interest accrual period. Since this calculation is probably going to be used in future programs, you are to make this routine a CALLable submodule.

The Interest Calculation Program (ICP) contains the calculations needed. Using Encore, the subroutine is extracted from the program and a CALLable subroutine is created. You need to remove the subroutine from the ICP by using Encore's Complement Module Generation facility. If the calculations in the interest accrual period routine are changed in the future, maintenance is only done to the submodule.

By doing some preliminary analysis, you determine that the routine you want is located in the ICP program that contains the GET-NUM-OF-DAYS paragraph, as shown in [Figure 31](#), and it's structured as a performed subroutine. In this case, the best extract objective to use is the Perform Range extract.

**Figure 31 • Example Program Containing the GET-NUM-OF-DAYS Paragraph**

```
000358 GET-NUM-OF-DAYS.  
000359 *  
000360     ACCEPT TODAY-DATE FROM DATE.  
000361  
000362     COMPUTE DAYS-IN-PERIOD =  
000363         MONTH (TODAY-MM) - TODAY-DD.  
000364     MOVE +0 TO DAYS-IN-YEAR.  
000365     PERFORM SUM-TOTAL-DAYS  
000366         VARYING WORK-CNT FROM 1 BY 1  
000367         UNTIL TODAY-DD = WORK-CNT.  
000368 *  
000369     SUM-TOTAL-DAYS.  
000370     COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ).  
000371 *
```

After performing your initial analysis, you build a checklist that contains these tasks, which are involved in extracting the Perform Range and generating the CALLED submodule and the corresponding Complement Module:

- Identify the Perform Range name
- Verify that the Perform Range contains the desired logic
- Extract the desired Perform Range
- Generate the CALLable submodule
- Generate the Complement Module
- Verify the results

## Starting an Encore Session

To start the Encore session, you need to access the ESW Primary screen and open the Encore application.

**Note:**

Logon procedures and AKR information are unique to your programming environment. If necessary, contact your systems administrator or your Encore coordinator for the correct dataset names.

### *To open Encore*

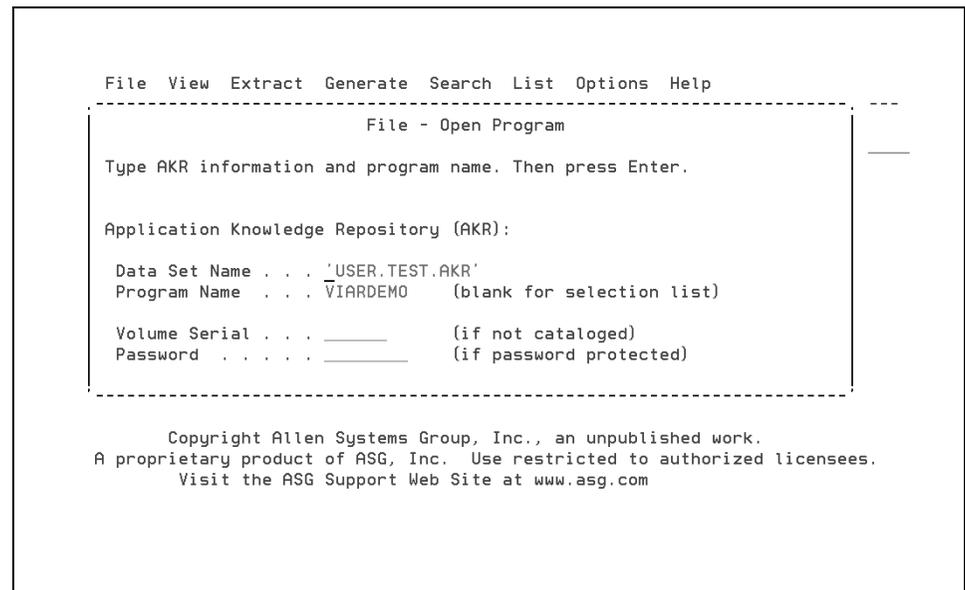
- 1 Logon to the ESW Primary screen.
- 2 Select Re-engineer ▶ Program and press Enter to display the Encore Primary screen.

Or

Type EN on the command line and press Enter to display the Encore Primary screen.

- 3 Select File ▶ Open and press Enter to display the File - Open Program pop-up, as shown in [Figure 32](#).

**Figure 32 • File - Open Program Pop-up**



- 4 Type the AKR dataset name in the Data Set Name field and VIARDEMO in the Program Name field and press Enter.

**Note:**

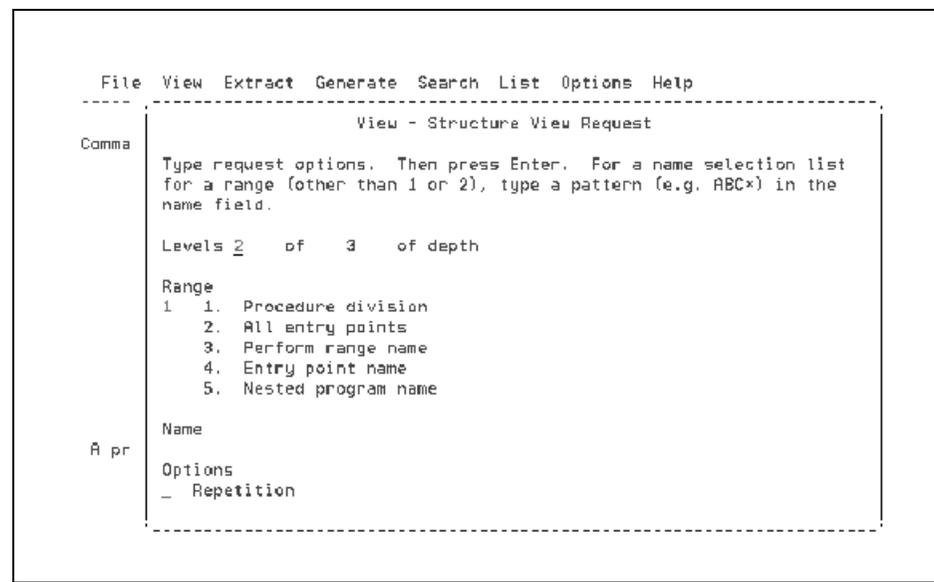
A program must be analyzed and placed in the AKR before you can use Encore. For more information about analyzing a program and placing it in the AKR, see ["Getting Started" on page 23](#).

The program is now available to Encore and your next step is to identify the Perform Range name. This is done by using a Structure View, which is a graphical representation of logical program units.

**To use Structure View**

- 1 Select View ▶ Structure and press Enter to display the View - Structure View Request pop-up, as show in [Figure 33](#).

**Figure 33 • View - Structure View Request Pop-up**



- 2 To view the entire program, type 3 on the Levels option of the View - Structure View Request pop-up and press Enter to display the Structure View screen (see [Figure 34 on page 61](#)). It is not necessary to type anything in the Range field (the default is Procedure Division).

See the online help for more detailed information about the options in the structure view request pop-up.

[Figure 34](#) shows that box 007 has the paragraph name GET-NUM-OF-DAYS, which is the Perform Range you are looking for. You can verify this by examining the code in the Perform Range from the Structure View screen.

- 3 Select View ► Zoom In and press Enter.
- 4 Place your cursor in box 007 GET-NUM-OF-DAYS and press Enter to zoom in on the selected data.

**Figure 34 • Structure View Screen**

```

File View Extract Generate Search List Options Help
-----
                          Structure View          Program: VIARDEMO
Command ==>
ASG1582I 9 OF 9 LEVELS DISPLAYED IN THE VIEW.          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*|
+-----+ +-----+ +-----+ +-----+
|          |          |          | |          |
+-----+ +-----+ +-----+ +-----+

```

**Note:**

If your PF keys are set to the Encore defaults, you can perform the Zoom In function by placing your cursor in the graphics box and pressing PF4. See the online help for more information about PF keys.

- The Zoom In request displays the source lines contained in the GET-NUM-OF-DAYS paragraph, as shown in [Figure 35](#). Press PF3 to return to the Structure View screen.

**Figure 35 • View - Source Pop-up**

```

View Search List Options Help
-----
Command ==> View - Source SCREEN POSITIONED
Scroll ==> CSR

000358 GET-NUM-OF-DAYS. PERF RNG
----- 1 LINE NOT DISPLAYED
000360 ACCEPT TODAY-DATE FROM DATE. PERF RNG
----- 1 LINE NOT DISPLAYED
000362 COMPUTE DAYS-IN-PERIOD = PERF RNG
000363 MONTH (TODAY-MM) - TODAY-DD.
000364 MOVE +0 TO DAYS-IN-YEAR. PERF RNG
000365 PERFORM SUM-TOTAL-DAYS PERF RNG
000366 VARYING WORK-CNT FROM 1 BY 1 PERF RNG
000367 UNTIL TODAY-DD = WORK-CNT. RETURN
----- 1 LINE NOT DISPLAYED
000369 SUM-TOTAL-DAYS. PERF RNG
000370 COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ). PERF
----- 64 LINES NOT DISPLAYED
***** BOTTOM OF DATA *****

```

The previous Structure View redisplay and the boxes containing the code for the Perform Range extract are highlighted, as shown in [Figure 36](#).

**Figure 36 • Structure View Screen**

```

File View Extract Generate Search List Options Help
-----
Command ==> Structure View Program: VIARDEHO
Scroll ==> CSR
More: - < >

+--- 007+ +--- 008+ +--- 009+
| GET-NUM-OF-DAYS | | INITIATE | | UPDATE-A | |
| OF-DAYS | | -ACTION | | CCT THRU | |
| | | | | | |
+-----+ +-----+ +-----+
| | | | | |
+--- 012+ +--- 013+ +--- 014+ +--- 015+ +--- 016+ +--- 017+
| SUM-TOTAL-DAYS | | INIT-ACT | | INITIATE | | INITIATE | | INITIATE | | | |
| | | | ION-RECO | | -ACTION | | -ACTION | | -ACTION | |
| | | | RD | | (8) | | (8) | | (8) | | (8) |
+-----+ +-----+ +-----+ +-----+ +-----+ +-----+
***** BOTTOM OF DATA *****

```

- Press PF3 to return to the Encore Primary screen.



- 2 Press Enter again to display the Extract - Perform Name List pop-up, which contains a list of all Perform Ranges in the program, as show in [Figure 38](#).

**Figure 38 • 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

PERFORM Range Name      Size      Times PERF  Direct Subord
                        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
-----

```

Use the Filter feature to narrow the list of selectable statements and simplify the statement selection.

- 3 Type `Filter` on the command line and press Enter (or press PF5) to display the Extract - Perform Range Filter pop-up (see [Figure 39 on page 65](#)).

**Note:**

You can type `EXPORT` on the command line to write selected Perform Range name data from the Extract - Perform Name List pop-up to a specified, or default, dataset name. See the *ASG-Encore Reference Guide* for a detailed description of the `EXPORT` command and associated operands.

- 4 Type GET-NUM-OF-DAYS in the PERFORM Name field, as show in [Figure 39](#), to filter the list of selectable perform ranges to include only this Perform Range. For this example, use the default values for the Use Context field.

**Figure 39 • Extract - Perform Range Filter Pop-up**

```

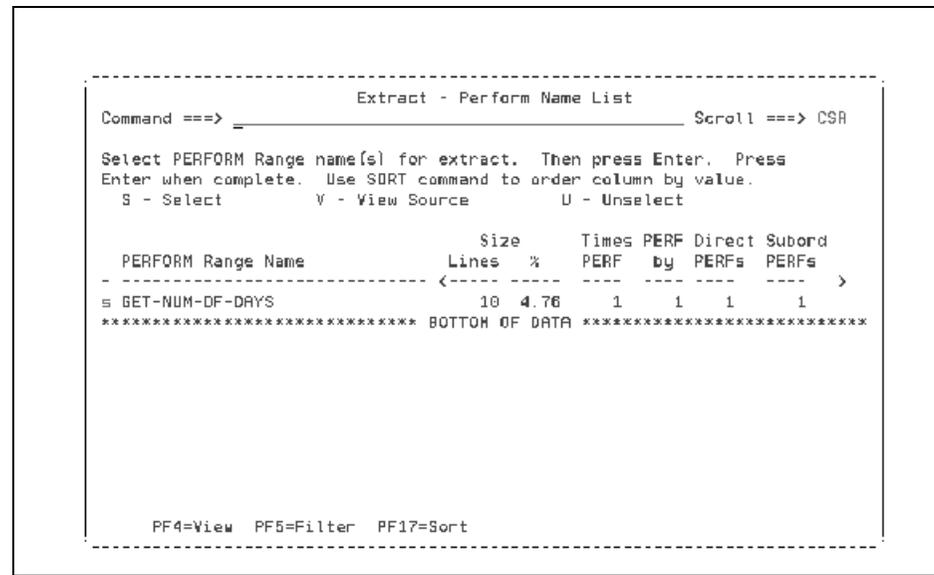
C -----
S                               Extract - Perform Name List
E                               Extract - Perform Range Filter
E Type information for filtering PERFORM Range Name list.
E PERFORM Name  GET-NUM-OF-DAYS
E Data Name    . . .
E Use Context  Options
E 1 1. References / Aliasing
E 2 2. Uses
E 3 3. Modifications
E Line Range  . . .
E COBOL Subset
E Text Pattern
E -----
E REPORT-FINAL-CNTRS          5 2.98  1  1  0  0
E PF4=View PF5=Filter PF17=Sort
E -----

```

- 5 Press Enter to redisplay the Extract - Perform Name List pop-up, as show in [Figure 40](#). The only selection is GET-NUM-OF-DAYS.

Type S to the left of GET-NUM-OF-DAYS and press Enter. Press Enter again to return to the Encore Primary screen, which displays with a message indicating that the criteria for the extract were successfully selected.

**Figure 40 • Extract - Perform Name List Pop-up (filtered)**



See the online help for more detailed information about Extract features.

## Viewing the Logic Segment

After extracting the Logic Segment, review the output to verify that you have selected the correct code.

### To view the Logic Segment created by the extract

- 1 Select View ► Logic Segment and press Enter to display the View - Select Logic Segments pop-up, as show in [Figure 41](#).

The View - Select Logic Segment pop-up lists all Logic Segments created for the program. The last active Logic Segment is highlighted and selected for viewing. For this example, view the PERFORM-RANGE-0001 Logic Segment.

**Figure 41 • View - Select Logic Segments Pop-up**

```

-----
Command ==> _          View - Select Logic Segment          1 LOGIC SEGMENTS
                        Scroll ==> CSR
S - Select
-----
Program   Logic Segment Name   Description
-----
S VIARDEMO PERFORM-RANGE-0001   Perform Range
***** BOTTOM OF DATA *****
-----

```

**Note:**

To select another segment on the View - Select Logic Segments pop-up, (if applicable), blank out the default S and select the Logic Segment you want to view and press Enter.

- 2 Press Enter to display the Source View screen, as show in [Figure 42](#), which contains the Logic Segment. The Logic Segment contains all of the code in the selected Perform Range and all of the data items that are referenced by that code.

Figure 42 • Source View Screen - View Logic Segment

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

- - - - - 210 LINES NOT DISPLAYED
000358 GET-NUH-OF-DAYS. LOGICSEG
- - - - - 1 LINE NOT DISPLAYED
000360 ACCEPT TODAY-DATE FROM DATE. LOGICSEG
- - - - - 1 LINE NOT DISPLAYED
000362 COMPUTE DAYS-IN-PERIDD = LOGICSEG
000363 MONTH (TODAY-MM) - TDDAY-DD. LOGICSEG
000364 MOVE +0 TO DAYS-IN-YEAR. LOGICSEG
000365 PERFORM SUM-TOTAL-DAYS LOGICSEG
000366 VARYING WORK-CNT FROM 1 BY 1 LOGICSEG
000367 UNTIL TDDAY-DD = WORK-CNT. LOGICSEG
- - - - - 1 LINE NOT DISPLAYED
000369 SUM-TOTAL-DAYS. LOGICSEG
000370 COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ), LOGICSEG
- - - - - 64 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
```

## Saving the Logic Segment

After viewing the Logic Segment, save it in the AKR for future processing and access.

### To save a Logic Segment in the AKR

- 1 Select File ► Save Segment and press Enter to display the File - Save Logic Segments pop-up, as show in [Figure 43](#).

Figure 43 • 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 *****

```

- 2 Type S to the left of the Logic Segment to be saved (PERFORM-RANGE-0001 in this example) and press Enter to display the File - Save Segment pop-up (see [Figure 44 on page 70](#)).

**Note:** \_\_\_\_\_

To generate the Logic Segment using a batch job, type B and press Enter to select the desired Logic Segment. See the online help for more information about batch generation.

\_\_\_\_\_

- 3 Perform these actions on the File-Save Segment pop-up (see [Figure 44](#)):
  - a Type the name to save the segment as in the Name field.
  - b Optionally enter a short description in the Description field.
  - c Optionally enter a long description of the Long Description field.

**Figure 44 • 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 PERFORM-RANGE-EXTRACT _____  
  
Long Description:  
TEST SEGMENT FOR INSTALLATION. _____  
_____  
_____  
_____  
_____  
_____  
_____
```

- 4 Press Enter to return to the File-Save Logic Segments pop-up.
- 5 Press PF3 once to return to the Source View Screen and once again to return to the Encore Primary screen.

## Creating the CALLable Module

After the Logic Segment has been extracted, viewed, and saved, you need to create the CALLable submodule.

### To create a CALLable submodule from the Logic Segment

- 1 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment pop-up, as show in [Figure 45](#).

Figure 45 • Generate - Select Logic Segment Pop-up

```

-----
Command ==> _____ Generate - Select Logic Segment      1 LOGIC SEGMENTS
                               Scroll ==> CSR
S - Select
Program   Logic Segment Name      Description
-----
S VIARDEHD  GET-NUM-OF-DAYS        CALCULATE NUMBER OF DAYS
*****
***** BOTTOM OF DATA *****
-----

```

**Note:** \_\_\_\_\_

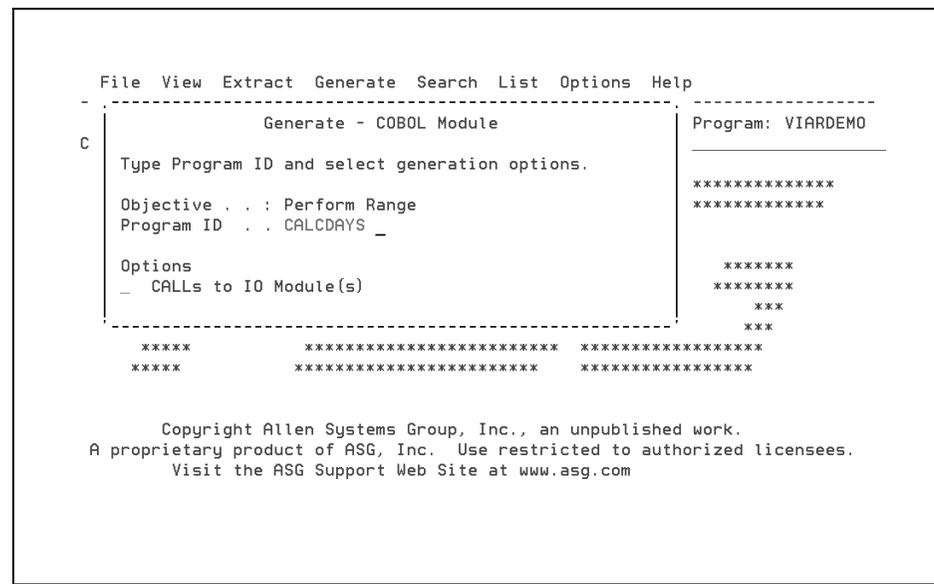
CALLable modules can also be generated using a batch job. See the online help for more information about batch generation.

\_\_\_\_\_

The Generate - Select Logic Segment pop-up lists all of the Logic Segments that you have extracted during your current session, including any other Logic Segments that have been saved in the AKR. The last referenced Logic Segment is highlighted and preselected. Since the Logic Segment that you have extracted was saved as GET-NUM-OF-DAYS, this is the name that is highlighted.

- 2 Press Enter to display the Generate - COBOL Module pop-up, as show in [Figure 46](#).

**Figure 46 • Generate - COBOL Module Pop-up**



- 3 A default name is assigned in the Program ID field, but can be changed as required. In this case, change the name to CALCDAYS and press Enter to display the Generate - Specify Perform Range ENTRY Names pop-up.

**Note:**

\_\_\_\_\_

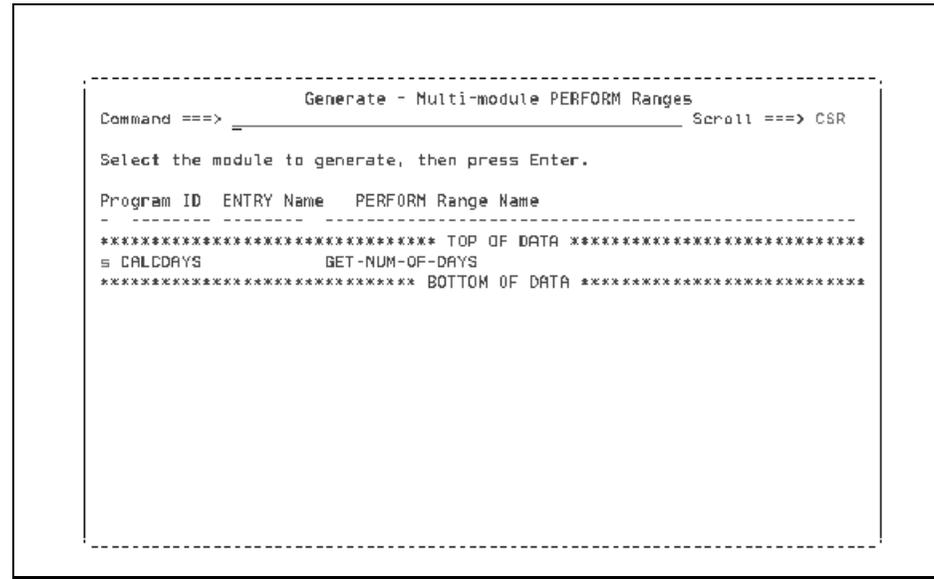
To set format and content options for the COBOL module being generated, select Options ► Generate. See the online help for more information about Generate options.

\_\_\_\_\_

- 4 Press Enter to display the Generate - Multi-module PERFORM Ranges pop-up (see [Figure 47 on page 73](#)).

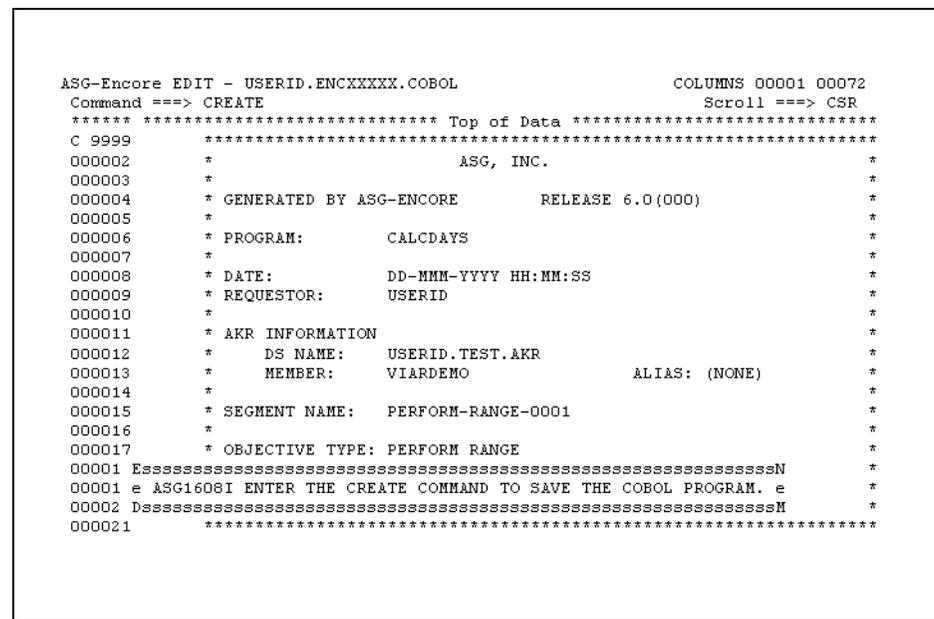
- To select the Perform Range to be generated from the Generate - Multi-module PERFORM Ranges pop-up (see [Figure 47](#)), type S in the Program ID area for the GET-NUM-OF-DAYS Perform Range and press Enter to display the Edit Screen (see [Figure 48](#)).

Figure 47 • Generate - Multi-Module PERFORM Ranges Pop-up



- When the Edit screen displays, type CREATE on the command line and type C 9999 on the first line of code. Press Enter to save the module in your source library and display the Edit/View - Create pop-up (see [Figure 49 on page 74](#)).

Figure 48 • Edit Screen Display of CALCDAYS



- 7 Type the name you want to save the program under in your source library, as shown in [Figure 49](#), verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter to create the member and return to the Edit screen. Press PF3 until you return to the Encore Primary screen.

**Figure 49 • Edit/View - Create Pop-up**

```
                                Edit/View - Create
Command ==>

"Current" Data Set: USERID.ENCXXXXX.COBOL

To ISPF Library:
Project . . . ASG
Group . . . CE60XXXX
Type . . . CNTL
Member . . . CALCDAYS

To Other Partitioned Data Set Member:
Data Set Name . . .
Volume Serial . . . (If not cataloged)

Data Set Password . . (If password protected)

Enter "/" to select option
Specify pack option for "CREATE" Data Set

Press ENTER key to create. Enter END command to cancel create.
```

**Note:** \_\_\_\_\_

CREATE may only be issued against an empty member. If CALCDAYS already exists in your library, you would need to use the REPLACE command. See the online help and the *ASG-Encore Reference Guide* for more information about these commands.

---



- 3 To specify the Logic Segment from which to generate a Complement segment, select GET-NUM-OF-DAYS and press Enter, as show in [Figure 51](#).

**Figure 51 • Extract - Complement Pop-up**

```

-----
Command ==> _                Extract - Complement                1 LOGIC SEGMENTS
                               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
-----
S VIARDEMO  GET-NUM-OF-DAYS      CALCULATE NUMBER OF DAYS
*****
***** BOTTOM OF DATA *****
-----

```

- 4 Press Enter to return to the Encore Primary screen.
- 5 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment pop-up, as show in [Figure 52](#). Select the Logic Segment that is used to create a Complement Module (COMPLEMENT-0002 in this example) and press Enter to display the Generate - COBOL Module pop-up (see [Figure 53 on page 77](#)).

**Figure 52 • Generate - Select Logic Segment Pop-up**

```

-----
Command ==> _                Generate - Select Logic Segment                3 LOGIC SEGMENTS
                               Scroll ==> CSR

S - Select

Program  Logic Segment Name      Description
-----
_ VIARDEMO  COMPLEMENT-0002      Complement Perform Range
S VIARDEMO  TRANSACTION-0001      Transaction
_ VIARDEMO  TRANSACTION-0002      Transaction
*****
***** BOTTOM OF DATA *****
-----

```

- 6 Accept the default name (VIARDEMV in this example, as shown in [Figure 53](#)), and press Enter to display the Generate - Specify Perform Range CALL Names pop-up (see [Figure 54](#)).

Figure 53 • 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 . . : Complement                |
  | 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                      |

```

- 7 Type CALCDAYS (if not automatically displayed, see [Figure 54](#)) as the ENTRY Name for the GET-NUM-OF-DAYS Perform Range and press Enter to accept the name.

Figure 54 • Generate - Specify Perform Range CALL Names Pop-up

```

-----
Command ==> =          Generate - Specify Perform Range CALL Names          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 *****
          CALCDAYS    GET-NUM-OF-DAYS
***** BOTTOM OF DATA *****

```

- 8 Press Enter to display the Edit screen with the generated module. Type CREATE on the Command Line and then type C 9999 on the first line of code, as show in [Figure 55](#). Press Enter to display the Edit/View - Create pop-up (see [Figure 56 on page 79](#)).

**Figure 55 • Edit Screen - Complement Module**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==> CREATE                                     Scroll ==> CSR
***** ***** Top of Data *****
C 9999 *****
000002 *                ASG, INC.                *
000003 *                *                        *
000004 * GENERATED BY ASG-ENCORE          RELEASE 6.0(000) *
000005 *                *                        *
000006 * PROGRAM:          VIARDEMV          COMPLEMENT *
000007 *                *                        *
000008 * DATE:            DD-MMM-YYYY HH:MM:SS *
000009 * REQUESTOR:      USER                *
000010 *                *                        *
000011 * AKR INFORMATION *                        *
000012 * DS NAME:        USERID.TEST.AKR *
000013 * MEMBER:        VIARDEMO          ALIAS: (NONE) *
000014 *                *                        *
000015 * SEGMENT NAME:   COMPLEMENT-0003 *
000016 *                *                        *
000017 * OBJECTIVE TYPE: COMPLEMENT (COMPUTATION VARIABLE) *
00001 E*****N
00001 e ASG1608I ENTER THE CREATE COMMAND TO SAVE THE COBOL PROGRAM. e *****
00002 D*****M
000021 *

```

- 9 When the Edit/View - Create screen displays, type the name you want to save the program under (VIARDEMV in this example, as show in [Figure 56](#)), verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter.

**Figure 56 • Edit /View - Create Screen - Complement Module**

```

                                     Edit/View - Create
Command ==>

"Current" Data Set: USERID.ENCXXXXX.COBOL

To ISPF Library:
Project . . . ASG
Group . . . . CE60XXXX
Type . . . . CNTL
Member . . . . VIARDEMV

To Other Partitioned Data Set Member:
Data Set Name . . .
Volume Serial . . .           (If not cataloged)

Data Set Password . .         (If password protected)

Enter "/" to select option
Specify pack option for "CREATE" Data Set

Press ENTER key to create. Enter END command to cancel create.
```

**Note:**

CREATE may be issued only against an empty member. If VIARDEMV already exists in your library, use the REPLACE command. See the online help and the *ASG-Encore Reference Guide* for more information about these commands.

Encore has now created program VIARDEM<sub>V</sub>, which contains all of the logic of the original program, except for the extracted Logic Segment.

If you examine the Complement Module, part of which is shown in [Figure 57](#), notice that the statement PERFORM GET-NUM-OF-DAYS has been replaced with a CALL to CALCDAYS.

**Figure 57 • VIARDEM<sub>V</sub> Complement Module**

```
ASG-Encore      EDIT - USERID.ENCXXXXXX.COBOL      COLUMNS 0001 00072
Command ==>    *                                  Scroll ==> CSR
000377          *                                  ASG-ENCORE-CALL
000378          CALL 'CALCDAYS' USING
000379                      DAYS-IN-PERIOD
000380                      DAYS-IN-YEAR
000381                      TODAY-DATE.
000382
000383
000384          INIT-ACTION-RECORD.
000385          MOVE ACCT-CONTROL-CODE TO ACT-CONTROL-CODE.
000386          MOVE ACCT-KEY TO ACT-ACCT-KEY.
000387          MOVE ACCT-NUM OF ACCT-RECORD TO
000388                      ACCT-NUM OF ACTION-RECORD.
000389          MOVE ACCT-TYPE OF ACCT-RECORD TO
000390                      ACCT-TYPE OF ACTION-RECORD.
000391          MOVE ACCT-STATUS-CODE TO ACT-STATUS-CODE.
000392          MOVE ACCT-BALANCE TO ACT-CUR-BALANCE.
000393
000394
000395          CLOSE-PGM.
000396
000397          PERFORM CLOSE-FILES.
000398
```

- 10 Press PF3 until you are returned to the Encore Primary screen.
- 11 Select File ► Close and press Enter to display the Close Program pop-up.
- 12 Select Close, discard segments. All unsaved logic segments are discarded when you exit Encore by selecting the Exit option. For this example, you do not need to save any generated data.

**Note:** \_\_\_\_\_

The Exit pop-up displays if you attempt to exit Encore using PF3. From this pop-up, options are available to save logic segments, discard logic segments, or cancel the exit and return to Encore.

---

## Listing of CALCDAYS

[Figure 58](#) shows the first section of the Encore-generated module CALCDAYS. Samples of the CALL Encore generated from the module VIARDEMO to CALCDAYS, listing the data elements it is passing as parameters, are contained in this section.

**Figure 58 • Member Containing Sample CALCDAYS Generated CALL**

```

ASG-Encore EDIT - ASG.VIACENXX.CNTL(CALCDAYS)          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:           CALCDAYS                                       *
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-001                                 *
000016 *                               *                                       *
000017 *   OBJECTIVE TYPE: PERFORM RANGE                                     *
000020 *   ASG-ENCORE NOTES:
000021 *     GET-NUM-OF-DAYS
000019 *****

```

[Figure 59](#) shows the sample CALL containing input and output code.

**Figure 59 • Sample CALL Showing Inputs and Outputs**

```

ASG-Encore      EDIT - ASG.VIACENXX.CNTL(CALCDAYS)          COLUMNS 00001 00072
Command ==>                                          Scroll ==> CSR
000022 *   ASG-ENCORE NOTE:
000023 *     THE FOLLOWING ARE SAMPLE CALLS SHOWING INPUTS AND OUTPUTS:
000024 *
000025 *     CALL 'CALCDAYS' USING
000026 *       DAYS-IN-PERIOD <<OUTPUT>>
000027 *       DAYS-IN-YEAR  <<OUTPUT>>
000028 *       TODAY-DATE   <<OUTPUT>> .
000029 *
000030 *   ASG-ENCORE NOTES:
000031 *
000032 *   1. "GOBACK" WAS INSERTED TO TERMINATE THE PROGRAM NORMALLY WHEN
000033 *     ITS INTENDED FUNCTION IS COMPLETED. PLEASE EXAMINE EACH
000034 *     INSERTION TO DETERMINE IF THIS ACTION IS APPROPRIATE.
000035 *
000036 *
000037 *   IDENTIFICATION DIVISION.
000038 *   PROGRAM-ID. CALCDAYS.
000039 *   AUTHOR. ASG DEMONSTRATION.
000040 *
000041 * *****
000042 *     THIS PROGRAM DOES THE MONTH END CALCULATIONS FOR EACH TYPE *
000043 *     OF CHECKING ACCOUNT CARRIED BY THE UNIVERSAL BANK. IT      *

```

When Encore generates a CALLable module, it creates notes that you need to examine to verify that the module performs the way you expect it to, as show in [Figure 60](#).

**Figure 60 • Encore Workbench Notes**

```

ASG-Encore      EDIT - ASG.VIACENXX.CNTL(CALCDAYS)      COLUMNS 00001 00072
Command ==>                                         Scroll ==> CSR
000030      * ASG-ENCORE NOTES:
000031      *
000032      * 1. "GOBACK" WAS INSERTED TO TERMINATE THE PROGRAM NORMALLY WHEN
000033      * ITS INTENDED FUNCTION IS COMPLETED. PLEASE EXAMINE EACH
000034      * INSERTION TO DETERMINE IF THIS ACTION IS APPROPRIATE.
000035      *
000036      *
000037      IDENTIFICATION DIVISION.
000038      PROGRAM-ID. CALCDAYS.
000039      AUTHOR. ASG DEMONSTRATION.
000040
000041      *****
000042      * THIS PROGRAM DOES THE MONTH END CALCULATIONS FOR EACH TYPE *
000043      * OF CHECKING ACCOUNT CARRIED BY THE UNIVERSAL BANK. IT *
000044      * HANDLES THE FOLLOWING TYPES OF ACCOUNTS. *
000045      * CHECKING WITH INTEREST *
000046      * CHECKING WITH NO INTEREST *
000047      * MONEY MARKET ACCOUNT (MMA) *
000048      * SAVINGS ACCOUNT *
000049      * IT ALSO UPDATES THE ACCOUNT BALANCE, BRANCH BALANCE AND *
000050      * BANK BALANCE. *
000051      *****

```

[Figure 61](#) shows that Encore generated all of the necessary divisions when it created the CALLable submodule. In the DATA DIVISION, Encore has included all data items referenced by the Perform Range GET-NUM-OF-DAYS.

**Figure 61 • Divisions Generated**

```

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-370.
OBJECT-COMPUTER. IBM-370.

DATA DIVISION.

WORKING-STORAGE SECTION.
77 WORK-CNT PIC 9(4).
01 MONTH-TABLE-INIT.

```

[Figure 62](#) shows that any data elements contained in the Working-Storage section of the original program that had Value clauses that were placed in the LINKAGE SECTION of the CALLable module had the value clauses removed and shown as comments. COBOL does not allow Value clauses in the LINKAGE SECTION.

**Figure 62 • Value Clauses Removed and Shown as Comments**

```

LINKAGE SECTION.

* ASG-ENCORE NOTES:
*
* 1. VALUE CLAUSES OMITTED FROM LINKAGE ARE SHOWN AS COMMENTS.
* 2. USED, MODIFIED AND REFERENCED (USE/MOD) SYMBOLS ARE
*    NOTED.
* 3. ANNOTATED COPYBOOK SYMBOLS ARE REPRODUCED AS COMMENTS.
*
  77 DAYS-IN-PERIOD          PIC S9(9)V9(4) COMP-3.
*                               ...MOD
  77 DAYS-IN-YEAR           PIC S9(9)V9(4) COMP-3.
*                               ...REF
  01 TODAY-DATE.
* .....MOD
   05 TODAY-YY              PIC 9(02).
   05 TODAY-MM             PIC 9(02).
*                               .....REF
   05 TODAY-DD             PIC 9(02).
*                               .....REF

PROCEDURE DIVISION USING

```

[Figure 63](#) shows that any statements generated by Encore are preceded by a comment. In the case of the module CALCDAYS, Encore indicated that it inserted a GOBACK to terminate the program normally when its intended function is completed. You need to verify that this is what you want. If not, you need to change it to continue processing.

**Figure 63 • GOBACK Comment**

```
— PERFORM GET-NUM-OF-DAYS.  
  GOBACK.  
  
*                               ASG-ENCORE= GOBACK  
  
GET-NUM-OF-DAYS.  
*  
  ACCEPT TODAY-DATE FROM DATE.  
  
  COMPUTE DAYS-IN-PERIOD =  
    MONTH (TODAY-MM) - TODAY-DD.  
  MOVE +0 TO DAYS-IN-YEAR.  
  PERFORM SUM-TOTAL-DAYS  
    VARYING WORK-CNT FROM 1 BY 1  
    UNTIL TODAY-DD = WORK-CNT.  
  
*  
  SUM-TOTAL-DAYS.  
  COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ).  
*  
  
* ASG-ENCORE NOTE:  ***** END OF GENERATED MODULE *****
```

## Extracting Multiple Perform Ranges

By default, when you create a Logic Segment containing multiple Perform Ranges, the CALLable module has multiple entry points. You can assign unique names to the entry points. These entry point names are included in the CALLable module and the CALLing module has CALLs generated for each Perform Range using the entry point names that you assigned.

As an additional option, you can generate more than one module from a multiple Perform Range extract. What this means is that rather than generating one module with an entry point for each Perform Range extracted, you have the ability to create a separate module for one or more of the Perform Ranges selected.

### ***Creating the CALLED Module with Multiple Entry Points***

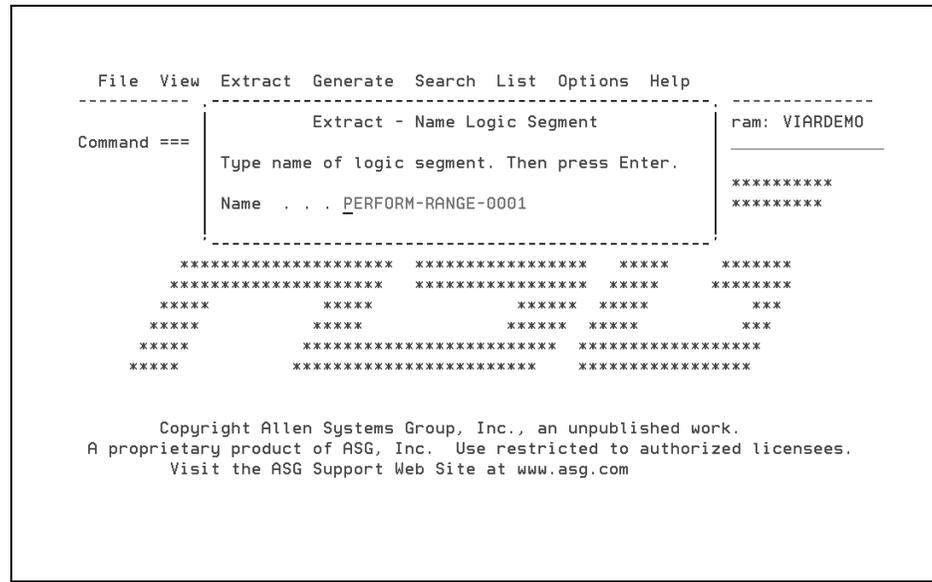
Assume that you created a Logic Segment containing three Perform Ranges that are used to create a CALLable module and the complement or CALLing program. The CALLable module contains 3 entry points that are named PROGENT1, PROGENT2, and PROGENT3.

#### ***To create the Logic Segment***

- 1 Start an Encore session and, if necessary, open program VIARDEMO.
- 2 Select Extract ► Perform range and press Enter to display the Extract - Name Logic Segment pop-up (see [Figure 64 on page 86](#)).

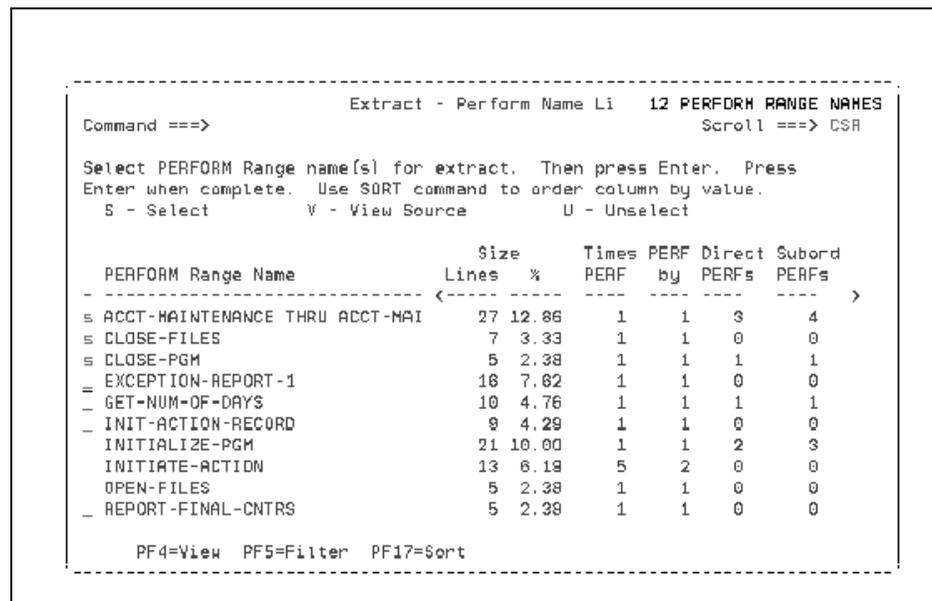
- 3 Accept the default name shown in [Figure 64](#) and press Enter to display the Extract - Perform Name List pop-up (see [Figure 65](#)).

**Figure 64 • Extract - Name Logic Segment**



- 4 Select the first three Perform Ranges and press Enter twice to return to the Encore Primary screen.

**Figure 65 • Extract - Perform Name List Pop-up**



- 5 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment pop-up, as show in [Figure 66](#). Press Enter to select the Logic Segment you just created and to display the Generate - COBOL Module Pop-up (see [Figure 67 on page 88](#)).

Figure 66 • Generate - Select Logic Segment Pop-up

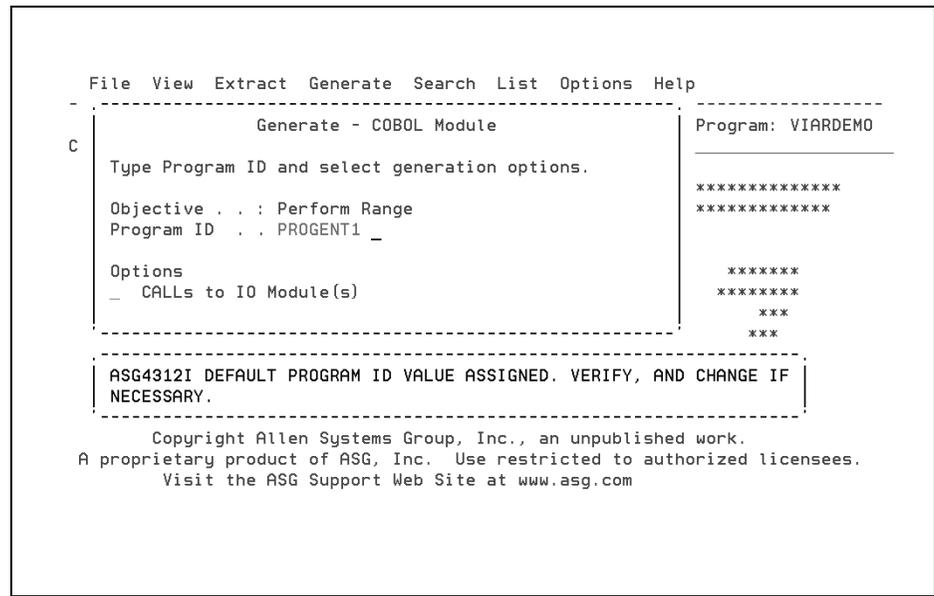
```
-----  
Command ==> _____ Generate - Select Logic Segment 2 LOGIC SEGMENTS  
Scroll ==> CSR  
  
S - Select  
  
Program Logic Segment Name Description  
-----  
VIARDEM GET-NUM-OF-DAYS CALCULATE NUMBER OF DAYS  
S VIARDEM PERFORM-RANGE-00D1 Perform Range  
***** BOTTOM OF DATA *****
```

- 6 Type the first Program Entry name (PROGENT1 in this example, as shown in [Figure 67](#)) in the Program ID field and press Enter to display the Generate - Specify Perform Range ENTRY Names pop-up (see [Figure 68 on page 89](#)).

**Note:**

For multiple entry points, there is no specific order to the use of entry point names. Any entry point name can be used for the Program ID.

**Figure 67 • Generate - COBOL Module Pop-up**



[Figure 68 on page 89](#) shows the Generate - Specify Perform Range ENTRY Names pop-up, which is used to provide the ENTRY statement names for the CALLable module (PROGENT1, the name you specified on the Generate - COBOL Module pop-up. Encore assigns the Program ID name to one of the extracted Perform Ranges.

In this case, ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT was assigned as the program name. You need to add the other two ENTRY names.

- 7 Type PROGENT2 as the ENTRY name for CLOSE-PGM.
- 8 Type PROGENT3 as the ENTRY name for CLOSE-FILES.
- 9 Press Enter twice to display the Generate - Multi-module PERFORM Ranges pop-up (see [Figure 69 on page 90](#)).

**Note:**

The name of the Selected Perform Range displays to the right of the entry point name column. Any Perform Ranges performed by the selected Perform Range are indented directly beneath. The list of Perform Range names can be condensed by using the Collapse line command. To remove the subordinate Perform Ranges from the list, enter '-' in the line command field next to the desired Perform Range. To expand the list to include the subordinate Perform Ranges, enter a plus sign (+) in the line command field next to the collapsed entry.

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.

**Figure 68 • Generate - Specify Perform Range ENTRY Names Pop-up**

```

-----
Generate - Specify Perform Range ENTRY Names
Command ==> _____ Scroll ==> CSR

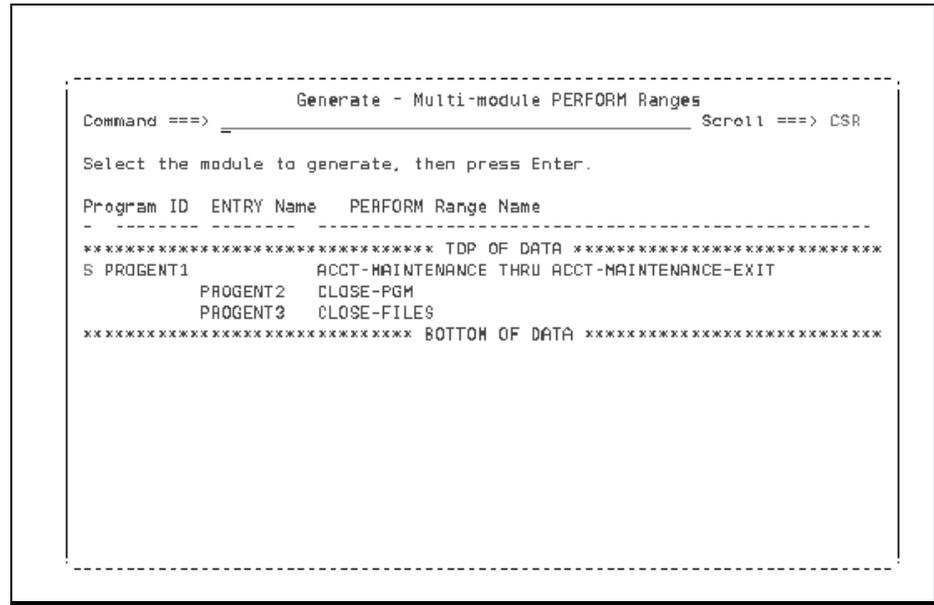
Specify the ProgramID and ENTRY names of the modules to be generated.
S - Split  H - Move  I - Into  + - Expand  - - Collapse

Program  ENTRY      PERFORM Range Name
-----
***** TOP OF DATA *****
PROGENT1      *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
              PERFORM INITIATE-ACTION
              PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT
              PERFORM INIT-ACTION-RECORD
-             PERFORM EXCEPTION-REPORT-1
-             PROGENT2 *CLOSE-PGM
              * PERFORM CLOSE-FILES
              PROGENT3 *CLOSE-FILES
***** BOTTOM OF DATA *****

```

- 10 Press Enter twice when you have finished typing the ENTRY statement names to display the Generate - Multi-module PERFORM Ranges pop-up, as show in [Figure 69](#).

**Figure 69 • Generate - Multi-module PERFORM Ranges Pop-up**



This pop-up is used to select the CALLable module to be generated. The contents of this pop-up depend on how you complete the Generate - Specify Perform Range ENTRY Names pop-up (see [Figure 68 on page 89](#)). In this example, three Perform Ranges were extracted to generate one module with three ENTRY names. For that reason, only one program name is displayed in this pop-up.

- 11 Type S in the selection field next to the Program ID PROGENT1 and press Enter to display the generated module in the Edit screen (see [Figure 70 on page 91](#)).

[Figure 70](#) shows the multiple ENTRY statements generated by Encore.

**Figure 70 • CALLable Module With Multiple ENTRY Statements**

```

ASG-Encore      EDIT - USERID.ENCXXXXX.COBOL      COLUMNS 00001 00072
Command ==>          Scroll ==> CSR
000319
000320          PERFORM ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT.
000321          GOBACK.
000322
000323          *          ASG-ENCORE- GOBACK
000324          ENTRY 'PROGENT2' USING
000325          WORK-RETURN-CODE.
000326          *          ASG-ENCORE- ENTRY
000327
000328          PERFORM CLOSE-PGM.
000329          GOBACK.
000330
000331          *          ASG-ENCORE- GOBACK
000332          ENTRY 'PROGENT3'.
000333          *          ASG-ENCORE- ENTRY
000334
000335          PERFORM CLOSE-FILES.
000336          GOBACK.
000337
000338          *          ASG-ENCORE- GOBACK
000339
000340

```

- 12 Press PF3 until you return to the Encore Primary screen.

## Creating Separate Modules from a Multiple Perform Range Extract

For this example, assume that these Perform Ranges are extracted from VIARDEMO:

- ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
- CLOSE-PGM
- CLOSE FILES

Perform Range ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT performs the Perform Ranges shown in [Figure 71](#).

**Figure 71 • Perform Range ACCT-MAINTENANCE**

```
ACCT-MAINTENANCE.  
  READ TRAN-FILE  
  AT END  
    MOVE EOF-LITERAL TO TRAN-FILE-FLAG  
    GO TO ACCT-MAINTENANCE-EXIT.  
  ADD +1 TO TRAN-CNT.  
  MOVE TRAN-ACCT-KEY TO ACCT-KEY.  
  READ ACCT-FILE KEY IS ACCT-KEY  
  INVALID KEY  
    MOVE ACTION-MISSING-ACCT TO WORK-ACTION-REASON  
    MOVE SPACES TO ACCT-RECORD  
    PERFORM INITIATE-ACTION  
    MOVE +4 TO WORK-RETURN-CODE  
    GO TO ACCT-MAINTENANCE-EXIT.  
  ADD +1 TO ACCT-IO-CNT.  
  PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT.  
  IF ACCT-MIN-BALANCE IS LESS THAN ACCT-BALANCE  
    PERFORM EXCEPTION-REPORT-1.  
  
ACCT-MAINTENANCE-EXIT.  
EXIT.
```

Perform Range CLOSE-PGM performs the Perform Range shown in [Figure 72](#). Perform Range CLOSE-FILES does not perform any other Perform Ranges, but is performed by CLOSE-PGM.

**Figure 72 • Structure of Perform Range CLOSE-PGM**

```

CLOSE-PGM.

    PERFORM CLOSE-FILES.

    MOVE WORK-RETURN-CODE TO RETURN-CODE.

CLOSE-FILES.

    CLOSE EXCEPTION-FILE.

    CLOSE TRAN-FILE
        ACCT-FILE
        ACTION-FILE.

```

The default result of an extract of multiple Perform Ranges is a CALLable module with an ENTRY name for each Perform Range extracted. You have determined that UPDATE-ACCT THRU UPDATE-ACCT-EXIT, performed by ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT, should be a separate module. The result you want would be these two modules:

- One module containing three ENTRY names, PROGENT1 for ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT, PROGENT2 for CLOSE-PGM, and PROGENT3 for CLOSE-FILES.
- One module containing the Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT along with any other Perform Ranges executed by it.

This example shows the steps involved in creating multiple modules from a multiple Perform Range extract:

- 1 Start an Encore session and open VIARDEMO if necessary.
- 2 Select Extract ► Perform range and press Enter to display the Extract - Name Logic Segment pop-up.
- 3 Press Enter to accept the default Logic Segment name on the Extract - Name Logic Segment pop-up.
- 4 Select the first 3 Perform Ranges from the Extract - Perform Name List pop-up and press Enter twice.

- 5 Select Generate ► COBOL module.
- 6 Press Enter to select the Logic Segment just extracted.
- 7 On the Generate - COBOL Module pop-up, type PROGENT1 in the Program ID field and press Enter to display the Generate - Specify Perform Range ENTRY Names pop-up, as show in [Figure 73](#).

**Figure 73 • Generate - Specify Perform Range ENTRY Names Pop-up**

```

-----
Command ==>          Generate - Specify Perform Range ENTRY Names          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 *****
-  PROGENT1          *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
-                   PERFORM INITIATE-ACTION
S                   PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT
-                   PERFORM INIT-ACTION-RECORD
-                   PERFORM EXCEPTION-REPORT-1
-                   PROGENT2 *CLOSE-PGM
-                   * PERFORM CLOSE-FILES
-                   PROGENT3 *CLOSE-FILES
***** BOTTOM OF DATA *****

```

[Figure 73](#) shows that PROGENT1, the ENTRY name for ACCT-MAINTENANCE THRU ACCT-MAINTENANCE EXIT, is the default Program ID. PROGENT2 is assigned to Perform Range CLOSE-PGM, and PROGENT3 is assigned to Perform Range CLOSE-FILES. Also, Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT, which is performed by ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT, is generated as a separate module.

- 8 Type PROGENT2 as the ENTRY name for CLOSE-PGM.
- 9 Type PROGENT3 as the ENTRY name for CLOSE-FILES.
- 10 To generate UPDATE-ACCT THRU UPDATE-ACCT-EXIT as a separate module, type S in the line command column next to PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT and press Enter to display the updated Generate - Specify Perform Range ENTRY Names pop-up (see [Figure 74 on page 95](#)).

[Figure 74](#) shows that the Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT has been split out as a separate program by Encore. The Perform statement within ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT has been converted to a CALL.

**Figure 74 • 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 *****
_ PROGENT1 *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
-          *PERFORM INITIATE-ACTION
-          + CALL "UPDATE-ACCT THRU UPDATE-ACCT-EXIT"
-          *PERFORM EXCEPTION-REPORT-1
-          PROGENT2 *CLOSE-PGM
-          *PERFORM CLOSE-FILES
-          PROGENT3 *CLOSE-FILES
-          *UPDATE-ACCT THRU UPDATE-ACCT-EXIT
-          *PERFORM INIT-ACTION-RECORD
***** BOTTOM OF DATA *****

```

- 11 Type UPDTACCT as a Program ID for the Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT, as shown in [Figure 75](#), and press Enter twice to display the Generate - Multi-module PERFORM Ranges pop-up (see [Figure 76 on page 96](#)).

**Figure 75 • Generate - Specify Perform Range ENTRY Names Pop-up - Assign Program ID**

```

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 *****
_ PROGENT1 *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
-          *PERFORM INITIATE-ACTION
-          + CALL "UPDATE-ACCT THRU UPDATE-ACCT-EXIT"
-          *PERFORM EXCEPTION-REPORT-1
-          PROGENT2 *CLOSE-PGM
-          *PERFORM CLOSE-FILES
-          PROGENT3 *CLOSE-FILES
-          *UPDATE-ACCT THRU UPDATE-ACCT-EXIT
-          *PERFORM INIT-ACTION-RECORD
_ UPDTACCT
***** BOTTOM OF DATA *****

```

- 12 To generate a module and display it in Edit, type S on the selection field next to the desired Program ID name (PROGENT1 in this example, as show in [Figure 76](#)) and press Enter to generate the module and display it on the Edit screen.

This pop-up is used to select each module to be generated. Only one module can be selected at a time.

**Figure 76 • Generate - Multi-module PERFORM Ranges - Select Module to Generate**

```

-----
Generate - Multi-module PERFORM Ranges
Command ==> _____ Scroll ==> CSA

Select the module to generate, then press Enter.

Program ID  ENTRY Name  PERFORM Range Name
-----
***** TOP OF DATA *****
S PROGENT1          ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
                PROGENT2          CLOSE-PGM
                PROGENT3          CLOSE-FILES
= UPDTACCT          UPDATE-ACCT THRU UPDATE-ACCT-EXIT
***** BOTTOH OF DATA *****
    
```

- 13 Save the generated module by typing CREATE on the command line and then C 9999 on the first line of code. Press Enter to save the module in your source library and display the Edit/View - Create pop-up.
- 14 On the Edit/View - Create pop-up, type the name you want to save the program under in your source library, verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter to create the member and return to the Edit screen.
- 15 Press PF3 to return to the Generate - Multi-module PERFORM Ranges pop-up. You can repeat this procedure to generate the remaining modules. When finished, press PF3 until you return to the Encore Primary screen.

## Reviewing the Generated Modules

The results of this exercise have produced two modules; PROGENT1 and UPDTACCT.

[Figure 77](#) shows that PROGENT1 is the Program ID and is made up of the Perform Ranges ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT, CLOSE-PGM, and CLOSE-FILES.

**Figure 77 • PROGENT1**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          Member PROGENT1 created
Command ==>                                     Scroll ==> CSR
***** ***** Top of Data *****
000001 *****
000002 *                                     ASG, INC. *
000003 *                                     *
000004 * GENERATED BY ASG-ENCORE          RELEASE 6.0(000) *
000005 *                                     *
000006 * PROGRAM:          PROGENT1 *
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 *               ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT *
000020 *               CLOSE-PGM *
000021 *               CLOSE-FILES *

```

[Figure 78](#) shows that PROGENT2 is the ENTRY name for CLOSE-PGM Perform Range.

**Figure 78 • PROGENT2**

```

*                                     ASG-ENCORE- GOBACK
*   ENTRY 'PROGENT2' USING
*                                     WORK-RETURN-CODE.
*                                     ASG-ENCORE- ENTRY
*
* PERFORM CLOSE-PGM.
*   GOBACK.

```

[Figure 79](#) shows that PROGENT3 is the ENTRY name for CLOSE-FILES.

**Figure 79 • PROGENT3**

```
*                                     ASG-ENCORE- GOBACK
ENTRY 'PROGENT3'.
*                                     ASG-ENCORE- ENTRY
PERFORM CLOSE-FILES.
GOBACK.
*                                     ASG-ENCORE- GOBACK
```

[Figure 80](#) shows that the Perform for UPDATE-ACCT THRU UPDATE-ACCT-EXIT has been changed to a CALL to program UPDTACCT.

**Figure 80 • Call to UPDTACCT**

```
*                                     ASG-ENCORE- CALL
CALL 'UPDTACCT' USING
TRAN-RECORD
INTEREST-RATE
DAYS-IN-PERIOD
DAYS-IN-YEAR
WORK-ACTION-REASON
ACTION-CNT
TODAY-DATE
CONTROL-PARAMETERS
BRANCH-BALANCE
BANK-BALANCE.

IF ACCT-MIN-BALANCE IS LESS THAN ACCT-BALANCE
PERFORM EXCEPTION-REPORT-1.

ACCT-MAINTENANCE-EXIT.
EXIT.
```

## **Detecting and Eliminating Multiple Perform Range Common Code**

This section discusses the techniques you can use to eliminate replication of common code within multiple Perform Range extracts.

Common code can occur within a multiple Perform Range extract when the Perform Range selected to be converted to a CALLable module contains a performed subroutine that is executed by additional Perform Ranges in the extract. Encore identifies this condition and gives you the option of eliminating replication of common code.

For this discussion, assume that the two Perform Ranges in VIARDEMO, ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT and UPDATE-ACCT THRU UPDATE-ACCT-EXIT are to be selected for extract. Additionally, Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT is to be generated as a CALLable module.

ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT and UPDATE-ACCT THRU UPDATE-ACCT-EXIT both perform subroutine INITIATE-ACTION. When both Perform Ranges are generated as CALLable modules, the code in INITIATE-ACTION is present in both modules. Encore detects this condition and allows you to circumvent it.

### ***To select the Perform Ranges***

These steps select Perform Ranges ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT and UPDATE-ACCT THRU UPDATE-ACCT-EXIT:

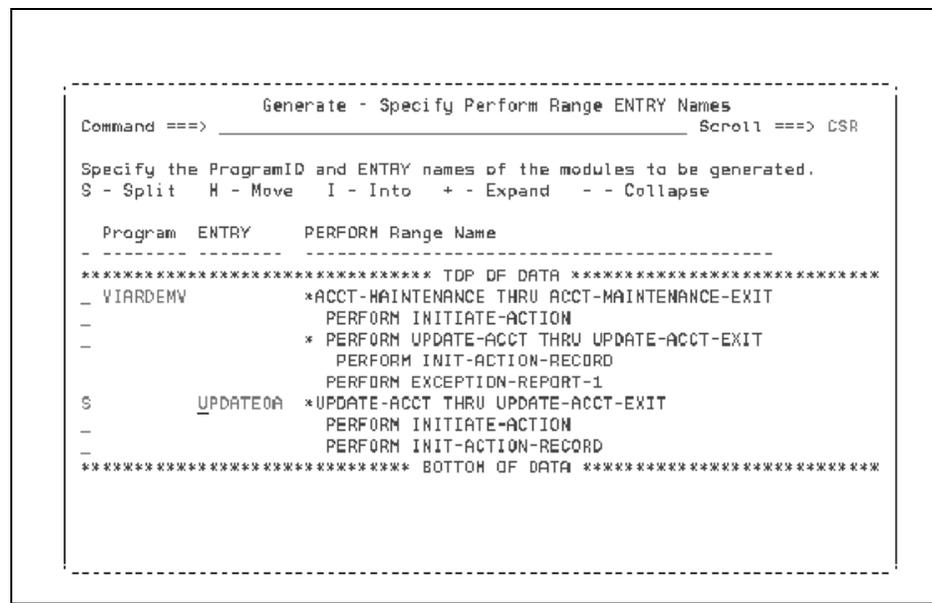
- 1** Select Extract ► Perform range.
- 2** Accept the default Logic Segment name on the Extract - Name Logic Segment pop-up by pressing Enter.
- 3** On the Extract - Perform Name List pop-up, select the Perform Ranges ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT and UPDATE-ACCT THRU UPDATE-ACCT-EXIT (you may have to page down to see the last Perform Range). Press Enter to confirm the selection and press Enter again to return to the Encore Primary screen.

**To generate the COBOL module**

The object of this extract is to generate two CALLable modules. Currently, module VIARDEM V contains two ENTRY points; VIARDEM V as the Program ID, and another ENTRY point that represents Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT. These steps provide the option to convert a Perform Range to a CALLable module.

- 1 Select Generate ► COBOL module.
- 2 Select the Logic segment just extracted or except the highlighted selection.
- 3 Press Enter to accept the default Program ID on the Generate - COBOL Module pop-up. The Generate - Specify Perform Range ENTRY Names pop-up displays and is used to provide the ENTRY statement names for the CALLable module, as shown in [Figure 81](#).

**Figure 81 • Generate - Specify Perform Range ENTRY Names Pop-up**



- 4 To convert UPDATE-ACCT THRU UPDATE-ACCT-EXIT to a CALLable module, initiate the split option by typing S to the left of Perform Range UPDATE-ACCT THRU UPDATE-ACCT-EXIT and press Enter.

[Figure 82](#) shows that the ENTRY field for UPDATE-ACCT THRU UPDATE-ACCT-EXIT now becomes the Program ID field, indicating that the Perform Range is generated as a CALLable module. Note that the words DUP (002) appear to the right of the Perform statements PERFORM INITIATE-ACTION. Encore has detected that the code in this Perform Range is duplicated in both modules. The number within the parentheses indicate the number of times the Perform Range is executed in this Logic Segment.

**Figure 82 • Using the Split Option to Generate a Callable Module**

```

-----
Generate - Specify Perform Range ENTRY Names
Command ==> _____ Scroll ==> CSA

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 *****
VIARDEM  *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
          PERFORM INITIATE-ACTION                                DUP (002)
          +* CALL "UPDATE0A"
          PERFORM EXCEPTION-REPORT-1
- UPDATE0A *UPDATE-ACCT THRU UPDATE-ACCT-EXIT
S          PERFORM INITIATE-ACTION                                DUP (002)
          PERFORM INIT-ACTION-RECORD
-----
***** BOTTOM OF DATA *****

```

### *To change executed Perform statement code to a CALLable module*

- 1 Type S to the left of the Perform statement PERFORM INITIATE-ACTION and press Enter.

**Note:** \_\_\_\_\_

You can eliminate duplicate code that can occur when converting Perform Ranges to modules by using the Split option to convert the duplicated code into a CALLable submodule. By doing this, the Perform statements in the two modules are changed to CALLs.

\_\_\_\_\_

A Program ID field is now displayed next to Perform statement INITIATE-ACTION. Note that the PERFORMs have been changed to CALLs, as shown in [Figure 83](#).

**Figure 83 • Using the Split Option**

```

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 *****
- VIARDEM          *ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
                  CALL "INITIATE-ACTION"
                  +* CALL "UPDATE0A"
                  PERFORM EXCEPTION-REPORT-1
UPDTACCT          *UPDATE-ACCT THRU UPDATE-ACCT-EXIT
                  CALL "INITIATE-ACTION"
                  PERFORM INIT-ACTION-RECORD
- INITACT_         INITIATE-ACTION
***** BOTTOM OF DATA *****

```

- 2 Change the Program ID for UPDATE-ACCT THRU UPDATE-ACCT-EXIT and enter a Program ID for INITIATE-ACTION.
- 3 Replace UPDATE0A with UPDTACCT in the PROGRAM field next to UPDATE-ACCT THRU UPDATE-ACCT-EXIT.
- 4 Type INITACT in the PROGRAM field next to INITIATE-ACTION.

- 5 Press Enter twice to display the Generate - Multi-module PERFORM Ranges pop-up, as show in [Figure 84](#).

This pop-up is used to select which COBOL module is generated. Only one module can be selected at a time. Select a module and press Enter to generate the module and display the source using your default editor. When you have processed the generated module (saving it, for example), press PF3 to redisplay the Generate - Multi-module PERFORM Ranges pop-up. You can select another Program ID or you can press PF3 to return to the Encore Primary screen.

**Figure 84 • Selecting a Module for Generating a COBOL Module**

```

                                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
  UPDTACCT          UPDATE-ACCT THRU UPDATE-ACCT-EXIT
  _ INITACT         INITIATE-ACTION
***** BOTTOM OF DATA *****

```

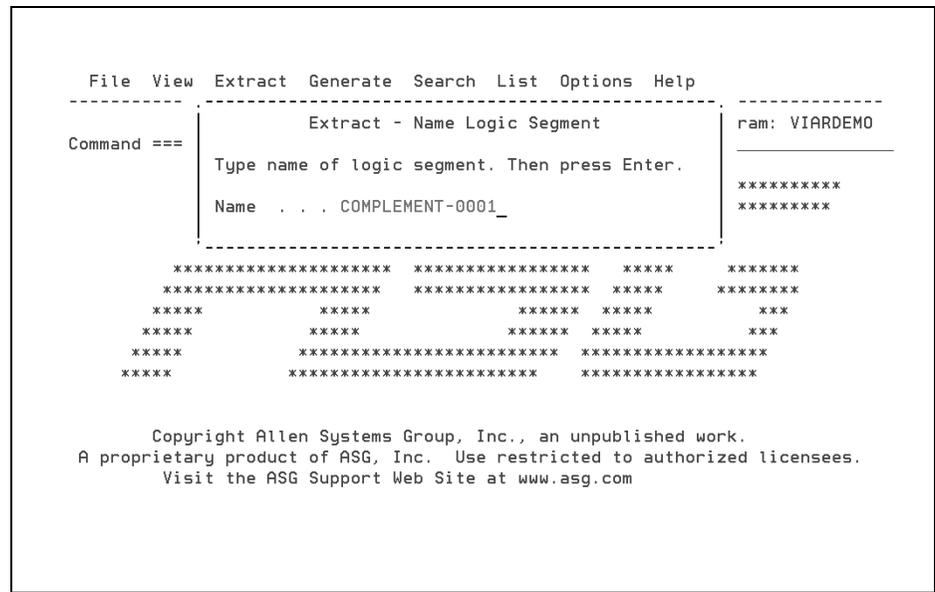
## Creating Complement Modules from Multiple Perform Range Extracts

When creating the Complement Module from a multiple Perform Range extract, Encore replaces the PERFORM statements for each Perform Range with a CALL to the ENTRY name that you assigned to each Perform Range. Once you have created the CALLable submodule, you then need to create the CALLing program (complement).

### To create a Complement Module from a Multiple Perform Range extract

- 1 Select Extract ► Complement to display the Extract - Name Logic Segment pop-up, as show in [Figure 85](#).

Figure 85 • Extract - Name Logic Segment Pop-up



- 2 Accept the default Logic Segment name and press Enter to display the Extract - Complement pop-up (see [Figure 86 on page 105](#)).

- 3 Select the multiple Perform Range Logic Segment previously generated, as show in [Figure 86](#). Press Enter, then PF3 twice to return to the Encore Primary screen.

**Figure 86 • 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
-----
VIADEMO     GET-NUM-OF-DAYS                  CALCULATE NUMBER OF DAYS
- VIADEMO     PERFORM-RANGE-0001                Perform Range
- VIADEMO     PERFORM-RANGE-0002                Perform Range
S VIADEMO     PERFORM-RANGE-0003                Perform Range
***** BOTTOM OF DATA *****

```

- 4 From the Encore Primary screen, select Generate ► COBOL Module and press Enter to display the Generate - Select Logic Segment pop-up (see [Figure 87 on page 106](#)).

- 5 Select the Complement Logic Segment, as show in [Figure 87](#), that was created in [step 3 on page 105](#) and press Enter to display the Generate - COBOL Module pop-up (see [Figure 88](#)).

**Figure 87 • Generate - Select Logic Segment Pop-up**

```

-----
Command ==> _____ Generate - Select Logic Segment      5 LOGIC SEGMENTS
                                Scroll ==> CSR

S - Select

Program      Logic Segment Name      Description
-----
S VIARDEMO  COMPLEMENT-0004            Complement Perform Range
VIARDEMO    GET-NUM-DF-DAYS              CALCULATE NUMBER OF DAYS
VIARDEMO    PERFORM-RANGE-0001          Perform Range
VIARDEMO    PERFORM-RANGE-0002          Perform Range
_ VIARDEMO  PERFORM-RANGE-0003          Perform Range
***** BOTTOM OF DATA *****
    
```

- 6 Type the name of the CALLing program in the Program ID field and press Enter to display the Generate - Specify Perform Range CALL Names pop-up (see [Figure 89 on page 107](#)).

**Figure 88 • 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 . . : Complement
Program ID . . : VIARDEM

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
    
```

- 7 Encore remembers the ENTRY statement names you assigned when extracting the multiple Perform Ranges and lists them on the Generate - Specify Perform Range CALL Names pop-up, as show in [Figure 89](#). Verify that the names are correct and press Enter to display the Edit screen (see [Figure 90](#)).

**Figure 89 • 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 *****
          PROGENT1  ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
          PROGENT2  CLOSE-PGM
          PROGENT3  CLOSE-FILES
***** BOTTOM OF DATA *****

```

The Edit screen shows an example of one of the CALLs to the multiple program ENTRYs generated by Encore at the appropriate place (line 241).

**Figure 90 • Encore-Generated CALLs to Multiple Program ENTRYs**

```

ASG-Encore  EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==> _____ Scroll ==> CSR

000234      PERFORM ASGGEN-PROGENT1
000235      *
000236          UNTIL TRAN-FILE-ENDS.
000237
000238      PERFORM REPORT-FINAL-CNTRS.
000239
000240      *
000241      CALL 'PROGENT2' USING
000242          WORK-RETURN-CODE.
000243
000244
000245      GOBACK.
000246
000247
000248
000249      GET-NUM-OF-DAYS.
000250      *
000251          ACCEPT TODAY-DATE FROM DATE.
000252
000253      COMPUTE DAYS-IN-PERIOD =
000254          MONTH (TODAY-MM) - TODAY-DD.
000255      MOVE +0 TO DAYS-IN-YEAR.

```

- 8 After you have reviewed the output, press PF3 until you return to the Encore Primary screen.

- 9 Optional. After entering the edit session, type CREATE on the command line and then C 9999 on the first line of code. Press Enter to display the Edit/View - Create pop-up and save the compliment module (if desired).

## Understanding Perform Extracts and Common Code

When you extract a Perform Range Logic Segment, you need to be aware of common code within the Perform Range. Common code occurs when a Perform subroutine is performed by two or more subroutines in a program. An example of common code is shown in [Figure 91](#).

**Note:**

This section shows examples that are not contained in VIARDEMO. The programs and associated figures used in this section are provided as specific examples of how to review Perform extracts and common code.

- Box 003, AML-MORT-TYPE, performs Box 008, CALCULATE-RATE.
- Box 009, VERIFY-ACCT also performs CALCULATE-RATE.

**Figure 91 • Structure View of Common Code**

```

File      View      Extract      Generate      Search      List      Options      Help
-----
Command==> _____ Structure View _____ Program: MOR00090
                                           Scroll==> CSR
                                           More: - + >

+----- 002 +          +----- 003 +
| INITIALI |          | AML-MORT |
| ZE-PGM   |          | -TYPETH | ← Paragraph AML-MORT-TYPE
|          |          | RU AML.* | performs CALCULATE-RATE
+-----+          +-----+

+-----+ |-----+ |-----+ |-----+
| OPEN-FIL | | GET-NUM- | | CALCULAT | | VERIFY.A |
| ES       | | OF-DAYS | | E-RATE  | | CCT THRU | ← Paragraph VERIFY-ACCT
|          | |          | |         | | VERIFY * | performs CALCULATE-RATE
+-----+ +-----+ +-----+ +-----+

|          |          |-----+ |-----+ |-----+ |-----+
+----- 012 +          +----- 013 + +----- 014 + +----- 015 + +----- 016 +
| SUB-TOTA |          | APPLY-TR | | CALCULAT | | CALCULAT | | CALCULAT |
| L-DAYS   |          | ANS       | | E-RATE  | | E-RATE  | | E-RATE  |
|          |          |          | |         | |         | |         |
+-----+          +-----+ (3) +-----+ (3) +-----+ (3) +-----+
+-----+          +-----+ REPEATED + +-----+ REPEATED+ +-----+ REPEATED+

```

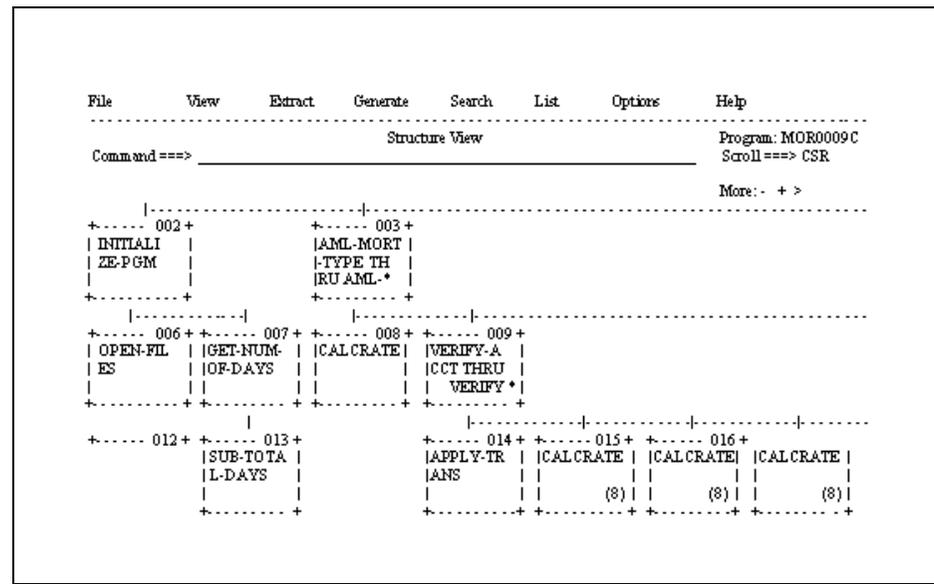
If you were to extract paragraph VERIFY-ACCT and create a CALLable submodule, it would contain paragraphs VERIFY-ACCT, APPLY-TRANS, and CALCULATE-RATE. Normally, a Complement Module would not contain code present in the extracted Logic Segment(s). However, in this case the Complement Module would also contain paragraph CALCULATE-RATE because it is also performed outside of the extracted Logic Segment, i.e, from AML-MORT-TYPE. This means that the logic in CALCULATE-RATE would exist in both the CALLable submodule and the Complement Module.

**To prevent the replication of common code**

- 1 Since paragraph CALCULATE-RATE is common to several Perform Ranges, you would extract it first and generate a CALLable submodule.
- 2 After generating the submodule, generate the Complement Module.

Notice that the common code in paragraph CALCULATE-RATE (as shown in [Figure 91 on page 108](#)) has been replaced by a CALL to module CALCRATE in [Figure 92](#), the CALLable submodule you just created.

**Figure 92 • Structure View of Complement Module MOR0009C**



**Note:** The Complement Module (MOR0009C) created in [step 2](#) must be Analyzed and placed in the AKR before the extraction in [step 3 on page 110](#) is done.

- 3 Extract the Perform Range VERIFY-ACCT from the Complement Module MOR0009C created in [step 2 on page 109](#) and generate a CALLable submodule. The generated submodule (after VERACCNT was analyzed) is shown [Figure 93](#).

**Figure 93 • Structure View of Submodule VERACCNT**

```

File      View      Extract  Generate  Search  List  Options  Help
-----
Command ==> _____ Structure View          Program: VERACCNT
                                                Scroll ==> CSR
                                                More: +

+----- 001 +
| VERACCNT |
|         |
+-----+
|
+----- 002 +
| VERIFY-A |
| CCT THRU |
| VERIFY * |
+-----+

|-----|-----|-----|-----|
+----- 003 +----- 004 +----- 005 +----- 006 +----- 007 +
| APPLY-TR | |'CALCRAT | |'CALCRAT | |'CALCRAT | |'CALCRAT |
| ANS      | |E'      | |E'      | |E'      | |E'      |
|         | |ASG1582I 2 OF 2 LEVELS DISPLAYED IN THE VIEW |
+-----+ +-----+ +-----+ +-----+ +-----+

```

Note that the common code in paragraph CALCULATE-RATE in the original module has been replaced by a CALL to module CALCRATE, created in [step 1 on page 109](#).





- 3 Select the Logic Segments to be checked for common code. In this example, Logic Segments COMPLEMENT-0004 and GET-NUM-OF-DAYS are selected, as show in [Figure 96](#). Press Enter twice to return to the Encore Primary screen.

**Figure 96 • Extract - Common Code Pop-up**

```

-----
Command ==>                               Extract - Common Code           3 LOGIC SEGMENTS
                                           Scroll ==> CSR

Select segments for common code extract.  When selections are complete,
press Enter.
$ - Select           U - Unselect

Program  Logic Segment Name      Description
-----
S VIARDEMO  COMPLEMENT-0004      Complement Perform Range
S VIARDEMO  GET-NUM-OF-DAYS         CALCULATE NUMBER OF DAYS
= VIARDEMO  PERFORM-RANGE-0002   Perform Range
***** BOTTOM OF DATA *****

```

- 4 Select View ► Logic Segment to display the View - Select Logic Segment pop-up, as show in [Figure 97](#).

**Figure 97 • View - Select Logic Segments Pop-up**

```

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

$ - Select

Program  Logic Segment Name      Description
-----
S VIARDEMO  COMMON-CODE-0001           Common Code
_ VIARDEMO  COMPLEMENT-0004           Complement Perform Range
_ VIARDEMO  GET-NUM-OF-DAYS         CALCULATE NUMBER OF DAYS
_ VIARDEMO  PERFORM-RANGE-0002     Perform Range
***** BOTTOM OF DATA *****

```

- 5 Select the Logic Segment that was created from the Common Code extract and press Enter to display the View - Common Code PERFORM Ranges pop-up, as show in [Figure 98](#).

**Figure 98 • View - Common Code PERFORM Range Pop-up**

```

View - Common Code PERFORM Ranges
Command ==> _____ Scroll ==> CSR

PERFORM Range Name
-----
VIARDEMO
INITIALIZE-PGM
  OPEN-FILES
  GET-NUM-OF-DAYS
  SUM-TOTAL-DAYS
ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
  INITIATE-ACTION
  UPDATE-ACCT THRU UPDATE-ACCT-EXIT
  INIT-ACTION-RECORD
EXCEPTION-REPORT-1
REPORT-FINAL-CNTRS
CLOSE-PGM
CLOSE-FILES
INITIALIZE-PGM
  OPEN-FILES

```

If common code exists between a Perform Range Logic Segment and the Complement this pop-up is displayed, showing all Perform Ranges in the program. Perform Ranges containing code that would appear in both the CALLable submodule and the CALLing program are highlighted. Indented Perform Range names are subordinate to the preceding non-indented Perform Range names.

You can display the common code within the highlighted Perform Range(s) in a Source View, Tree View, or Structure View from this pop-up by entering the commands SV, TV, or STV, respectively, on the command line.

Once common code is identified, it can be eliminated by following the procedures described in ["Detecting and Eliminating Multiple Perform Range Common Code" on page 99](#).

**Note:** \_\_\_\_\_

If no common code exists, a No Statements message displays when you attempt to view the Common Code Logic Segment.

\_\_\_\_\_

## Replacing IO Statements with CALLs to an IO Module

Encore provides an automated solution to the COBOL constraint against file references across program boundaries. It involves replacing IO statements contained in the Perform Range with Encore-generated CALLs to an IO module. Encore can also generate the IO Module.

**Note:**

The CALLs to IO Module(s) option from the Generate - COBOL Module pop-up is available from any extract method except for Common Code.

### To replace IO statements with CALLs to IO modules

- 1 Select Generate ► COBOL Module to display the Generate - Select Logic Segment pop-up, as show in [Figure 99](#).

Figure 99 • Generate - Select Logic Segment Pop-up

```

Generate - Select Logic Segment      3 LOGIC SEGMENTS
Command ==> _                        Scroll ==> CSR
S - Select
-----
Program      Logic Segment Name      Description
-----
S VIARDEMO   COMPLEMENT-0004          Complement Perform Range
VIARDEMO     GET-NUM-OF-DAYS                CALCULATE NUMBER OF DAYS
_ VIARDEMO   PERFORM-RANGE-0002              Perform Range
***** BOTTOM OF DATA *****

```

- 2 Type S next to the Logic Segment name you want to generate and press Enter to display the Generate - COBOL Module pop-up (see [Figure 100 on page 116](#)).

- 3 Type the Program ID and enable the CALLs to IO Module(s) option by typing a forward slash (/) next to the option, as shown in [Figure 100](#).

**Figure 100 • 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 . . : Computation Variable                                             | *****
  | 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
  
```

- 4 Press Enter to display the Select FD Names pop-up, as shown in [Figure 101](#).

**Figure 101 • Select FD Names Pop-up**

```

-----
Command ==> _                               Select FD Names                               4 FD NAMES
-----
Select the FD names(s) for which CALLs to an I/O module should be generated
Then press Enter. '*' indicates IO Modules have been previously generated.

IO Parm Block Prefix . . ASG

  FD Name                                     ENTRY Name
  -----                                     -----
  - ACCT-FILE                                 ACCTOFIL
  - ACTION-FILE                               ACTIONOF
  - EXCEPTION-FILE                           EXCEPTIO
  - TRAN-FILE                                 TRANOFIL
  ***** BOTTOM OF DATA *****
  
```

- 5 Verify the Encore-generated prefix to the IO Parm block that is created in the LINKAGE section, to select one or more FD names, and to verify the ENTRY names that are used in the generated CALL to the selected files. You can use the Encore-generated prefix and ENTRY name, or assign your own by typing over the respective fields.

**Note:**

An asterisk (\*) next to the FD name signifies that an IO module has been previously generated for that file.

- 6 Press Enter to display the Generate - Specify Perform Range CALL Names pop-up, as show in [Figure 102](#).

**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 *****
          PROGENT1    ACCT-MAINTENANCE THRU ACCT-MAINTENANCE-EXIT
          PROGENT2    CLOSE-PGM
          PROGENT3    CLOSE-FILES
***** BOTTOM OF DATA *****
-----

```

The Complement Module includes CALLs to extracted PERFORM Range(s). If not automatically displayed, you need to specify the ENTRY name of the program to be CALLED.

- 7 In the ENTRY Name field for the Perform Range, enter a name for the program to be CALLED, if needed.

- 8 Press Enter to accept the name and press Enter again to generate the module.

When Encore generates the module, it substitutes CALLs to the IO module(s) selected in the Select FD Names pop-up.

The CALLs to the selected IO module are shown in [Figure 103](#).

**Figure 103 • Encore-Generated CALLs to IO Module**

```
ASG-Encore      EDIT - USERID.ENCXXXXX.COBOL      COLUMNS 0001 00072
Command ==>                               Scroll ==> CSR
000368          CALL 'ACCTOFIL' USING VIA-IOBLOCK, ACCT-RECORD
000369          *                               ASG-ENCOREEND I/O STMT
000370          OPEN  INPUT  TRAN-FILE
000371          OUTPUT ACTION-FILE.
000372
000373
000374
000375          VIAGEN-PROGENT1 SECTION.
000376
000377          CALL 'PROGENT1' USING
000378          TRAN-FILE-FLAG
000379          INTEREST-RATE
000380          DAYS-IN-PERIOD
000381          DAYS-IN-YEAR
000382          WORK-ACTION-REASON
000383          ACCT-IO-COUNT
000384          TRAN-CNT
000385          EX-RPT-LINE-CNT
000386          EX-RPT-PAGE-CNT
000387          ACTION-CNT
000388          WORK-RETURN-CODE
000389          TODAY-DATE
```

- 9 Press PF3 until you return to the Encore Primary screen.

At some point, it becomes necessary to generate the IO module that is CALLED by the CALLable submodule generated in the previous steps. There is no required sequence for creating the IO module or the CALLing module. The only requirement is that the IO module Program-ID be the same as the CALL Name assigned on the Select FD Names pop-up.

## Generating the IO Module

The IO module supports physical sequential, VSAM sequential, VSAM relative, and VSAM indexed files. All variations of IO statements for these files are supported, including declaratives. RD, SD, and CD files are not supported. It is not necessary to perform an extract prior to generating an IO module, however, an analyzed program must be opened.

### To generate an IO module

- 1 Select Generate ► File IO module to display the Generate - FD Name List pop-up, as show in [Figure 104](#).

This pop-up is used to select the file to be used in the IO module. All FD names defined in the program are listed in the column labeled FD Name. An asterisk to the left of the FD name indicates that an IO module has been previously generated for this FD. Information about each file is listed on the pop-up, such as access method, organization, number of READs to the file, and number of WRITES to the file.

**Figure 104 • Generate FD Name List Pop-up**

```

-----
Command ==> _____ Generate - FD Name List _____ 4 FD NAMES
                               Scroll ==> CSR

Select the name of the FD for the IO module generation.  Then press Enter.
'x' indicates names have been previously generated.

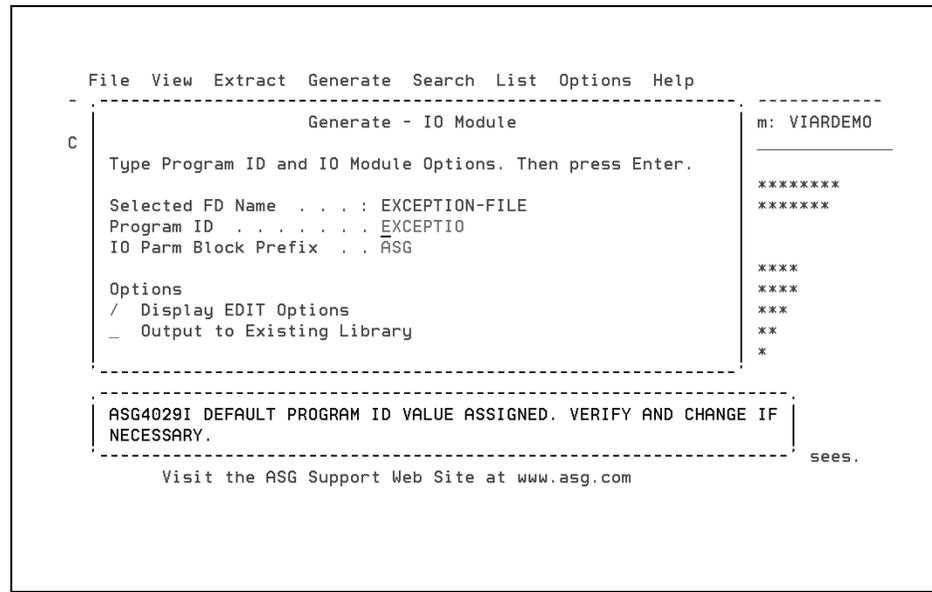
  FD Name                Access  Organization  READs  WRITES
-----
  ACCT-FILE              I/O    INDEXED        1      1
  ACTION-FILE           OUTPUT SEQUENTIAL  0      1
  * EXCEPTION-FILE      OUTPUT SEQUENTIAL  0      4
  * TRAN-FILE           INPUT  SEQUENTIAL  1      0
  ***** BOTTOM OF DATA *****

```

- 2 Type S next to the FD name to be selected. Only one FD can be selected per IO module. Press Enter to display the Generate - IO Module pop-up (see [Figure 105 on page 120](#)).

- 3 Press Enter to display the Generate - IO Module pop-up, as show in [Figure 105](#), and generate the IO module.
- 4 Verify that the Program ID matches the ENTRY name assigned to the CALL to this module.

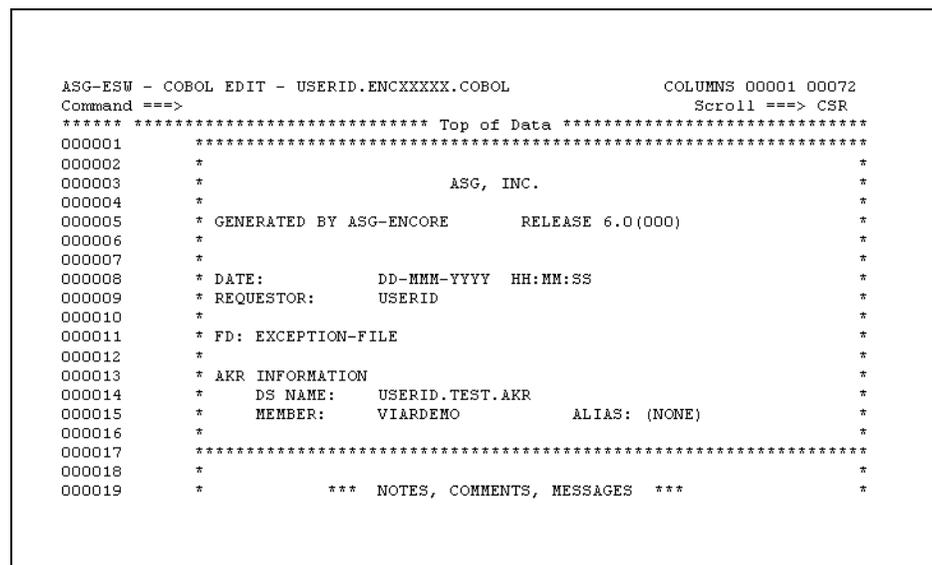
**Figure 105 • Generate - IO Module Pop-up**



Once the IO module has been created, as show in [Figure 106](#), you have the option of saving it using the CREATE command (for an example, see [step 9 on page 108](#)).

- 5 Press PF3 until you return to the Encore Primary screen.

**Figure 106 • Generated IO Module Screen**



## Perform Range Extract Compilation Issues

When Encore generates CALLable modules from a Perform Range extract, certain conditions may occur that cause compile errors or cause the program not to run at all. As shown in this section, Encore inserts comment messages in the generated modules when these conditions occur.

### Condition 1

Encore-generated CALLs from the Complement Module to the extracted module.

Cause	Action
The Complement Module produced for a Perform Range objective contains Encore-generated CALLs to the extracted module.	None.

### Condition 2

Encore generated a CALL that replaced a PERFORM VARYING construct.

Cause	Action
A PERFORM with the VARYING or TIMES option cannot be replaced by a CALL statement. In this case, Encore generates a SECTION to contain the generated CALL, and PERFORMs that SECTION using the original VARYING or TIMES option.	If the replaced PERFORM used the VARYING option, verify or alter SECTION name or the extracted module.

**Example - Condition 2**

If the generated submodule was originally PERFORMed by the Complement Module using the VARYING or TIMES option, Encore cannot replace the PERFORM with a CALL. Instead, Encore creates a SECTION to contain the generated CALL and PERFORMs that SECTION using the original VARYING or TIMES option. [Figure 107](#) shows the PERFORM VARYING option to an extracted subroutine in the original program.

**Figure 107 • PERFORM VARYING Example 1**

```

000371
000372     GET-NUM-OF-DAYS.
000373     *
000375         ACCEPT TODAY-DATE FROM DATE.
000376
000377         COMPUTE DAYS-IN-PERIOD =
000378             MONTH (TODAY-DD) - TODAY-DD.
000379     MOVE +0 TO DAYS-IN-YEAR.
000380     PERFORM SUM-TOTAL-DAYS
000381         VARYING WORK-CNT FROM 1 BY 1
000382             UNTIL TODAY-DD = WORK-CNT.
    
```

[Figure 108](#) shows the message created by Encore in the Complement Module.

**Figure 108 • PERFORM VARYING Example 2**

```

000028 * 2. A PROGRAM SECTION HAS BEEN GENERATED FOR CALLING EACH
000029 *     EXTRACTED PERFORM RANGE THAT HAS PERFORMED USING THE
000030 *     'VARYING' OPTION. EXAMINE EACH SECTION TO DETERMINE
000031 *     IF THE VARIED DATA ITEM IS HANDLED APPROPRIATELY.
000032 *
    
```

[Figure 109](#) shows the PERFORM - VARYING to the SECTION generated by Encore.

**Figure 109 • PERFORM VARYING Example 3**

```
000372      GET-NUM-OF-DAYS.
000373      *
000374          ACCEPT TODAY-DATE FROM DATE.
000375
000376          COMPUTE DAYS-IN-PERIOD =
000377              MONTH (TODAY-DD) - TODAY DD.
000378          MOVE +0 TO DAYS-IN-YEAR.
000379      *
000380          PERFORM ASGGEN-VIARDEMV
000381              VARYING WORK-CNT FROM 1 BY 1
000382              UNTIL TODAY-DD = WORK-CNT.
000383      *
000384      ASGGEN-VIARDEMV SECTION.
000385
000386          CALL 'VIARDEM' USING
000387              DAYS-IN-YEAR
000388              WORK-CNT.
000389
000390      ASGGEN-VIARDEM-EXIT.
000391      EXIT.
000392
000393
000394      *      ***** END OF ASG-ENCORE GENERATED MODULE. *****
```

### Condition 3

An unnamed 01 level data item, moved from WORKING-STORAGE to LINKAGE, must be assigned a valid COBOL data item name.

Cause	Action
<ul style="list-style-type: none"><li>• An unnamed 01 level was moved to the LINKAGE SECTION. When Encore generated the LINKAGE SECTION for a subprogram, it did not rename data items that were moved from the WORKING-STORAGE SECTION.</li></ul>	Assign a valid COBOL data item name to each LINKAGE SECTION data item with the name '_FILLER'.
<ul style="list-style-type: none"><li>• COBOL syntax allows a nonreferenced 01 level data item to go unnamed in WORKING-STORAGE, but not in LINKAGE.</li></ul>	
<ul style="list-style-type: none"><li>• Encore assigns the name '_FILLER' to the data item and issues this message. The subprogram cannot successfully compile until the data item is given a valid name.</li></ul>	
<ul style="list-style-type: none"><li>• A similar situation occurs when two 01 levels are defined with the same name and are not referenced at the 01 level. In this case, the compiler detects a duplicate name.</li></ul>	

---

**Condition 4**

An unnamed 01 level data item is contained in the USING list of an Encore-generated CALL from the Complement Module to the extracted module.

Cause	Action
<ul style="list-style-type: none"> <li>An unnamed 01 level was moved to the LINKAGE SECTION. When Encore generated the LINKAGE SECTION for a subprogram, it did not rename data items that were moved from the WORKING-STORAGE SECTION.</li> <li>COBOL syntax allows a nonreferenced 01 level data item to go unnamed in WORKING-STORAGE, but not in LINKAGE.</li> <li>Encore assigns the name '_FILLER' to the data item and issues this message. The subprogram cannot successfully compile until the data item is given a valid name.</li> <li>A similar situation occurs when two 01 levels are defined with the same name and are not referenced at the 01 level. In this case, the compiler detects a duplicate name.</li> </ul>	<p>Assign a valid COBOL data item name to each generated CALL USING entry with the name '_FILLER'.</p>

**Condition 5**

Encore-generated statements that cause normal exits from the program.

Cause	Action
<p>Encore generates program exit statements (STOP RUN, GOBACK, EXEC CICS RETURN) to ensure that the program exits properly. A normal program exit is generated when the intended function of the program is complete.</p>	<p>Review the conditions surrounding each generated exit statement to determine if the default exit statement should be augmented or replaced.</p>

## Condition 6

Encore has determined that a COPY statement cannot be used. Portions of the copybook are reproduced inline.

Cause	Action
<ul style="list-style-type: none"> <li>Under certain conditions, the original COPY statement cannot be used in a generated module.</li> <li>When data elements in the COPY statement contain definitions split between WORKING STORAGE and LINKAGE sections.</li> <li>When COPY statements contain dead code.</li> <li>When the contents of a COPY statement used in the LINKAGE SECTION contain the VALUE clause.</li> <li>In addition to this message, Encore generates comments indicating why the COPY statement could not be used as originally coded.</li> </ul>	<p>Determine why the COPY member was not usable. No action is required, however, the COPY member contents may be modified in order to use it in this context.</p> <p>Example:</p> <p>A copybook contains both FD and data item definitions. One or more of the data item definitions must be used in the LINKAGE SECTION.</p>

## Condition 7

A generated subprogram references a file, but not all of the references to that file are contained within the subprogram. COBOL does not support file references across program boundaries.

Cause	Action
<p>If the Perform Range code contains some references to a file, but does not contain all references to it, this message is generated.</p>	<p>Consider regenerating the module with the CALLS to IO Module option on the Generate - COBOL Module pop-up (see <a href="#">Figure 88 on page 106</a>).</p> <p>Example:</p> <p>One Perform Range opens FILE-X, a second Perform Range reads FILE-X, and a third Perform Range closes FILE-X. If the Logic Segment fails to select all three Perform Ranges, this message is generated.</p>

**Condition 8**

Encore has generated ENTRY statements to provide alternate entries into a subprogram.

Cause	Action
When multiple Perform Ranges are selected for the Perform Range objective, each Perform Range is assumed to be a separate ENTRY of the generated module. This message is produced to document the generation of multiple ENTRY statements. Each ENTRY was assigned a name from the Generate - Specify Perform Range Entry Names pop-up (see <a href="#">Figure 89 on page 107</a> ).	None, information only message.

**Condition 9**

Encore has generated the following variables to pass INDEX values between subprograms. Verify that appropriate names have been generated.

Cause	Action
The Perform Range requires table index variables as input parameters. Index Names are not true COBOL variables, and their values cannot be communicated through linkage. To accomplish execution equivalency, Encore adds additional code to both the Complement Module and the submodule.	Review the generated SET and 77 level statements.

## Complement Module Program Listing

This section shows a Complement Module that was generated from a program that contained an indexed table. These examples include the Complement Module and the submodule that remain after the perform extract.

### Example 1 - Complement

[Figure 110](#) shows the Encore generated note that describes the variables that have been generated for the Complement Module.

**Figure 110 • Compliment Module Note**

```
*      ASG-ENCORE NOTE:  THE FOLLOWING VARIABLES HAVE BEEN
*      GENERATED TO PASS 'INDEX' VALUES BETWEEN SUBPROGRAMS
*      VERIFY THAT APPROPRIATE NAMES HAVE BEEN GENERATED.
```

[Figure 111](#) shows that Encore created a 77 level variable with a USAGE IS INDEXED clause. This figure also shows that Encore has set the generated index variable to the local table index value.

**Figure 111 • Complement Module 77 Level Variable & Generated Index Variable**

```
77      ASGRNS-TABLE-INDEX          USAGE IS INDEX.

PROCEDURE DIVISION.
  ACCEPT INPUT-PROCESS-CODE.
  MOVE ZEROS TO DATA-TABLE.
  MOVE 0 TO NUM-ROWS.
  SET TABLE-INDEX TO 1.
  PERFORM LOAD-TABLE THRU LOAD-TABLE-EXIT.
  SET TABLE-INDEX TO 1.
```

[Figure 112](#) shows that Encore CALLs the submodule and passes the table along with the index variable.

**Figure 112 • Complement Module Submodule CALL**

```
CALL 'INDEXDEV' USING
      DATA-TABLE
      ASGRNS-TABLE-INDEX
      INPUT-PROCESS-CODE
      VALID-ITEM-SW
```

[Figure 113](#) shows that on the return from the CALL, Encore generated a SET statement to set the value of the table index to the value of the index variable that was passed to the submodule.

**Figure 113 • Encore Generated SET Statement**

```
*                               ASG-ENCORE-SET

      SET TABLE-INDEX TO ASGRNS-TABLE-INDEX.

      PERFORM DISPLAY-TABLE THRU DISPLAY-TABLE-EXIT.
      STOP RUN

LOAD-TABLE.
  ACCEPT INPUT-VAR.
  IF INPUT-VAR = 0, GO TO LOAD-TABLE-EXIT.
  MOVE INPUT-VAR TO TABLE-ENTRY (TABLE-INDEX).
  ADD 1 TO NUM-ROWS.
  IF NUM-ROWS = 20, GOT 0 LOAD-TABLE-EXIT.
  SET TABLE-INDEX UP BY 1,
  GO TO LOAD-TABLE.
LOAD-TABLE-EXIT.
  EXIT.

DISPLAY-TABLE.
  IF VALID-ITEM-SW = 'Y'
    DISPLAY 'PROCESS CODE IS VALID'
    DISPLAY TABLE-ENTRY (TABLE-INDEX)
  ELSE
    DISPLAY 'PROCESS CODE INVALID'.
  DISPLAY-TABLE-EXIT.

*                               ***** END OF ASG-ENCORE GENERATED MODULE. *****
```







---

# 5

## Computation Variable Extract

---

This chapter contains a practical demonstration of how to perform a Computation Variable extract to compute the value of a variable, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">133</a>
<a href="#">The Business Scenario</a>	<a href="#">134</a>
<a href="#">Starting an Encore Session</a>	<a href="#">134</a>
<a href="#">Extracting the Computation Variable</a>	<a href="#">135</a>
<a href="#">Viewing the Logic Segment</a>	<a href="#">140</a>
<a href="#">Saving the Logic Segment</a>	<a href="#">143</a>
<a href="#">Creating Pseudo Source Modules to Change Logic Segment Results</a>	<a href="#">145</a>
<a href="#">Controlling Extract Boundaries</a>	<a href="#">152</a>
<a href="#">Computation Variable Extract Compilation Issues</a>	<a href="#">155</a>

### Introduction

This chapter contains a scenario that demonstrates the technique of using a Computation Variable extract to determine how the value in a variable gets computed.

The Encore demonstration program VIARDEMO is used for the examples in this chapter. For a partial listing of VIARDEMO, see "[The Demonstration Program](#)" on page 49.

## The Business Scenario

You are a senior programmer at the MIS department at Universal Bank. Accounting asks you how the Interest Calculation Program (ICP) calculates interest for the different types of customer accounts. The ICP program has not been looked at in years and you don't remember the algorithm for calculating the interest.

By doing some preliminary analysis, you determine that the interest calculation variable is WK-INTEREST. Using Encore, you can identify and select the variable used to compute the interest and isolate all of the statements that contribute to or influence the value in that variable. The extract method used to do this is the Computation Variable extract.

## Starting an Encore Session

To start the Encore session, you need to access the ESW Primary screen and open the Encore application.

**Note:** \_\_\_\_\_

Logon procedures and AKR information are unique to your programming environment. If necessary, contact your systems administrator or your Encore coordinator for the correct dataset names.

\_\_\_\_\_

### *To open Encore*

- 1 Logon to the ESW Primary screen.
- 2 Select Re-engineer ► Program and press Enter to display the Encore Primary screen.

**Or**

Type EN on the command line and press Enter to display the Encore Primary screen.

- 3 Select File ► Open and press Enter to display the File - Open Program pop-up.
- 4 Type the AKR dataset name in the Data Set Name field and VIARDEMO in the Program Name field and press Enter.

See ["Starting an Encore Session" on page 59](#) for more information.



- 2 Press Enter to display the Extract - Data Name List pop-up, as shown in [Figure 120](#).

This pop-up displays the name of all variables in the program. The list can be scrolled up and down. By default, the data names are displayed in alphabetic sequence with the name of the group level in which the variable is defined. To sort the variables in a different order, press PF17.

You can also display the variable name profile instead of the Qualification by typing 2 in the Data Name Information field.

**Figure 120 • Extract - Data Name List Pop-up**

```

-----
Extract - Data Name List                               59 DATA NAMES
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-CONTRDL-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
-----

```

Use the Filter option to simplify the variable selection process.

- 3 Type `Filter` on the command line of the Extract - Data Name List pop-up and press Enter (or press PF5) to display the Extract - Computation Variable Filter pop-up (see [Figure 121 on page 137](#)).

- 4 To shorten the list of selectable variables to those that begin with W, type W\* on the Data Name line, as shown in [Figure 121](#), and press Enter to redisplay the Extract - Data Name List pop-up.

**Figure 121 • Extract - Computation Variable Filter Pop-up**

```

D -----
S Extract - Data Name List
S
S Extract - Computation Variable Filter
S Type information for filtering Data Variable Name list.
Data Name . . W*_
Use Context          Options          Level Number . . _
  1. References      - Aliasing
  2. Uses
  3. Modifications

- COPYBOOK Name
- Line Range . . _____
- Label Name . . _____
- CDSOL Subset

- Options
- - Limit to within an existing Segment
-

PF4=View PF5=Filter PF17=Sort

```

**Note:**

Use the Limit to within an existing Segment option on the Extract - Computation Variable Filter pop-up to limit or bound the current extract definition. See ["Extracting Code from an Existing Logic Segment" on page 153](#) for an example of how, and when, this option is used.

After filtering, the Extract - Data Name List pop-up displays all data names beginning with W, as shown in [Figure 122](#).

**Figure 122 • Extract - Data Name List Pop-up After Filtering**

```

-----
Extract - Data Name List          4 DATA NAMES
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. Qualification
  2. Profile
S - Select      V - View Source      U - Unselect

LV Data Name          Qualification
-----
S 77 WK-INTEREST
_ 77 WORK-ACTION-REASON
_ 77 WORK-CNT
_ 77 WORK-RETURN-CODE
***** BOTTOM OF DATA *****

```

- 5 Type S on the selection line showing WK-INTEREST and press Enter to display the Extract - Source Ordered View pop-up (see [Figure 123 on page 139](#)).

**Note:** \_\_\_\_\_

If the fully qualified data name is typed in the Data Name field on the Extract - Computation Variable Filter screen, [Figure 122](#) does not display. The Encore Primary screen redisplay with the information message ASG4289I VARIABLE(S) "WK-INTEREST" SELECTED. SELECT VIEW OR GENERATE.

\_\_\_\_\_

- 6 Type S on lines 303 and 307, as shown in [Figure 123](#), and press Enter. Press Enter again to redisplay the Extract - Data Name List pop-up (see [Figure 124](#)).

This pop-up shows the statements associated with WK-INTEREST.

**Figure 123 • Extract - Source Ordered View Pop-up**

```

-----
Command ==> Extract - Source Ordered View Scroll ==> CSR
Select statements for Computation Variable Extract, press Enter. When
complete press Enter.
Referenced Data Name: WK-INTEREST
Aliasing: YES

S - Select      V - View Source      U - Unselect
-----
S 000303      COMPUTE WK-INTEREST =
000304          (ACCT-AVG-BALANCE / DAYS-IN-YEAR) *
000305          DAYS-IN-PERIOD * (INTEREST-RATE / 100).
S 000307      MOVE WK-INTEREST TO ACCT-INTEREST.
***** BOTTOM OF DATA *****
    
```

The Extract - Data Name List pop-up redisplayes with the WK-INTEREST data name highlighted.

**Figure 124 • Extract Data Name List Pop-up - Variable Name Selected**

```

-----
Command ==> Extract - Data Name List 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
-----
- 77 WK-INTEREST
- 77 WORK-ACTION-REASON
- 77 WORK-CNT
- 77 WORK-RETURN-CODE
***** BOTTOM OF DATA *****

ASG4044I COMPUTATION VARIABLE STATEMENTS SUCCESSFULLY SELECTED.
    
```

- 7 Press Enter. The Encore Primary screen displays the message ASG4044I COMPUTATION VARIABLE STATEMENTS SUCCESSFULLY SELECTED, indicating that the variable was selected.

## Viewing the Logic Segment

Viewing the selected Logic Segment allows you to determine how the value of WK-INTEREST is derived.

### *To view the Logic Segment*

- 1 Select View ► Logic Segment and press Enter to display the View - Select Logic Segment pop-up, as shown in [Figure 125](#).

This pop-up displays all of the Logic Segments extracted during the current session, as well as the logic segments that are saved in the AKR. The most recently extracted Logic Segment is highlighted and preselected for convenience.

**Figure 125 • View - Select Logic Segment Pop-up**

```
-----
Command ==> _____ View - Select Logic Segment      2 LOGIC SEGMENTS
                                      Scroll ==> CSR
S - Select
Program   Logic Segment Name      Description
-----
S VIARDEM COMPUTATION-VARIABLE-0001  Computation Variable
VIARDEM  GET-NUH-QF-DAYS          CALCULATE NUMBER OF DAYS
***** BOTTOM OF DATA *****
```

- 2 Type 'FIND COMPUTE WK-INTEREST =' on the command line and press Enter to view the results. [Figure 126](#) shows the Logic Segment extracted for variable WK-INTEREST. From this segment we can see the algorithm for computing the account interest, including the component that calculates the interest rate (INTEREST-RATE / 100).

**Figure 126 • Search View - Logic Segment for WK-INTEREST Extraction**

```

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

000303 COMPUTE WK-INTEREST =                      PATTERN
000304 (ACCT-AVG-BALANCE / DAYS-IN-YEAR) *
000305 DAYS-IN-PERIOD * (INTEREST-RATE / 100).
----- 1 LINE NOT DISPLAYED
000307 MOVE WK-INTEREST TO ACCT-INTEREST.
----- 16 LINES NOT DISPLAYED
000324 UPDATE-ACCT-EXIT.
----- 33 LINES NOT DISPLAYED
000358 GET-NUH-OF-DAYS.
----- 1 LINE NOT DISPLAYED
000360 ACCEPT TODAY-DATE FROM DATE.
----- 1 LINE NOT DISPLAYED
000362 COMPUTE DAYS-IN-PERIOD =
000363 MONTH (TODAY-MM) - TODAY-DD.
000364 MOVE +0 TO DAYS-IN-YEAR.
000365 PERFORM SUM-TOTAL-DAYS
000366 VARYING WORK-CNT FROM 1 BY 1
000367 UNTIL TODAY-DD = WORK-CNT.                RETURN
----- 1 LINE NOT DISPLAYED

```

***To use Logic Segments to discover interest rate values***

- 1 Select Search ▶ Data to display the Search - Data Name pop-up.
- 2 Type INTEREST-RATE in the Data Name field.
- 3 Select All from the Direction option.
- 4 Select Mods from the Reference option and press Enter to find the statements that modify the interest rate. Leave the default values for all other fields.

The results of the search are displayed in the Source View, as shown in [Figure 127](#). All statements that modify the interest rate are highlighted. You now know that the interest rate is controlled by the setting of ACCT-TYPE of ACCT-RECORD.

**Figure 127 • Search View - Statements that Modify Interest Rate**

```

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

000278      MOVE CHK-INTEREST-RATE TO INTEREST-RATE          DATA MOD
000279      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
000280      MOVE ZERO TO INTEREST-RATE                        DATA MOD
000281      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
000282      MOVE MMA-INTEREST-RATE TO INTEREST-RATE          DATA MOD
000283      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
000284      MOVE SAV-INTEREST-RATE TO INTEREST-RATE.          DATA MOD
-----
000286      IF ACCT-MULTI-ACCT = 'YES'                        1 LINE NOT DISPLAYED
000287      ADD CHK-MULTI-ACCT-RATE TO INTEREST-RATE          DATA MOD
-----
000291      IF ACCT-MIN-BALANCE < FREE-SERVICE-MIN          3 LINES NOT DISPLAYED
000292      MOVE SERVICE-INTEREST-RATE TO INTEREST-RATE      DATA MOD
-----
000296      ELSE
-----
000303      C ASG0443I 6 DATA MODS FOUND FOR INTEREST-RATE.
000304
000305      DAYS-IN-PERIOD * (INTEREST-RATE / 100).

```

## Saving the Logic Segment

It is not necessary to actually save the Logic Segment for the Computation Variable to demonstrate this scenario. However, since this computation is essentially a company business rule, you should save the Logic Segment in the AKR for future reference.

### *To save the segment*

- 1 Select File ► Save segment and press Enter to display the File - Save Logic Segments pop-up, as shown in [Figure 128](#). This pop-up lists all Logic Segments created during an Encore session.

**Figure 128 • File - Save Logic Segment Pop-up**

```

-----
Command ==> _                               File - Save Logic Segments          1 LOGIC SEGMENTS
                                           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  COMPUTATION-VARIABLE-0001    Computation Variable
*****
***** BOTTOM OF DATA *****
-----

```

- 2 Type S next to the Logic Segment COMPUTATION-VARIABLE-0001 and press Enter to display the File - Save Segment pop-up (see [Figure 129 on page 144](#)).

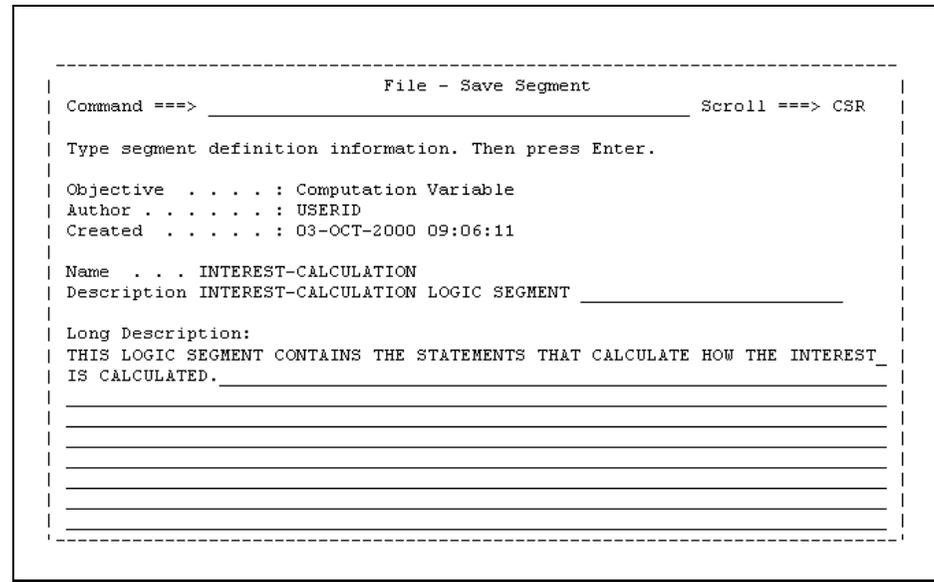
**Note:** \_\_\_\_\_

To generate the Logic Segment using a batch job, use the B line command to select the desired Logic Segment. See the online help for more information about batch generation.

\_\_\_\_\_

- 3 On the File - Save Segment pop-up, type the segment name, a short description, and if desired, a long description, as shown in [Figure 129](#).

**Figure 129 • File - Save Segment Pop-up**



- 4 Press Enter to return to the File - Save Logic Segments pop-up where a message displays in the short message field indicating the segment was successfully added to the AKR.
- 5 Press F3 until you return to the Encore Primary screen.

**Note:** \_\_\_\_\_

You can save the Logic Segment under the default name. However, it is recommended that you save them under meaningful names.

\_\_\_\_\_

## Creating Pseudo Source Modules to Change Logic Segment Results

If the program being analyzed contains a CALL to a program that has not been analyzed, you should consider creating pseudo source to obtain better results in an extracted program.

### *Pseudo Source Modules*

These conditions may require creating pseudo source for programs that have not been analyzed:

- Encore assumes all parameters are both used and modified. If this is not the case, then pseudo source should be written.
- If the CALLED program is not COBOL.
- You choose not to analyze the CALLED program.

The purpose of the pseudo source module is to identify the inputs and outputs of the CALLED routine. The pseudo source module must be saved in the AKR.

Each CALL statement parameter is input only, output only, or both input and output.

- Input-only parameters are used but not modified by the CALLED module.
- Output-only parameters are modified but not used by the CALLED module.
- Input and output parameters are both used and modified by the CALLED module.

[Figure 130](#) shows an example of a program being re-engineered that contains this code.

**Figure 130 • Creating Pseudo Source Modules - Program Re-engineering Code Example**

```
MOVE 0 TO PAYMENT.  
MOVE LOAN-AMOUNT TO PARM-1.  
CALL 'INTEREST' USING PARM-1, PAYMENT.  
DISPLAY PAYMENT
```

If a Computation Variable extract request is made for the variable PAYMENT at the DISPLAY statement, you should assume that the preceding CALL statement and the MOVE LOAN-AMOUNT statement are both required. This assumption can be made because it appears that the CALL to the INTEREST module is modifying the PAYMENT parameter. If this assumption is true, the CALL is required; however, if this assumption is false, the CALL statement is unnecessary (although the MOVE 0 TO PAYMENT statement would be required).

The report and Computation Variable extract objectives assume that all parameters are both input and output, which may not always be an accurate assumption. In some cases, the extract contains more than the minimum set of code because of this assumption. To obtain completely accurate results, the actual type of each parameter must be made known to the extract objective process. This can be accomplished in either of the ways described in these conditions.

- Analyze the CALLED program and save it in the AKR containing the CALLING program. In some cases, this may not be feasible, e.g., the program source is not accessible, the program may be so large that sufficient resources are unavailable, or the program is non-COBOL.
- Write a pseudo source module, identifying the type of each parameter. Analyze this pseudo source module instead of the actual CALLED program.

See ["The Analyze Facility" on page 36](#) for more information about using Encore to analyze a program.

Defining precise parameter types in a pseudo source module results in more accurate extract results for the report and Computation Variable extracts, because unnecessary code cannot be extracted.

These are the rules for writing pseudo source modules:

- For input parameters, use a *MOVE data-item TO ws-data-item* statement; where *ws-data-item* is a WORKING-STORAGE data item that is the same data type as *data-item*.
- For output parameters, use a *MOVE SPACES TO data-item statement*. A *MOVE ZEROES* statement may be used if appropriate.
- For output parameters that are only possibly modified (as opposed to definitely modified) by the CALLED routine, code the *MOVE SPACES TO data-item* statement in an IF statement that tests a WORKING-STORAGE data item.
- Code all inputs before outputs. If a data item is both an input and an output, code it first as an input and then as an output.
- The LINKAGE SECTION declarations should match the declarations in the calling program.

[Figure 131](#) and [Figure 132](#) are examples of a program that CALLs a non-COBOL subroutine that calculates an interest amount, and the pseudo source module that may be written for that subroutine.

**Figure 131 • Program Being Re-engineered**

```

IDENTIFICATION DIVISION.
PROGRAM-ID. VIARDEMO.
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
77     WORK-LOAN          PIC 9999.
77     WORK-RATE         PIC 9999.
77     WORK-AMOUNT       PIC 9999.
. . .
MOVE 10000 TO WORK-LOAN.
MOVE 9.5 TO WORK-RATE.
MOVE 10 TO WORK-AMOUNT.
CALL 'SUBRTN' USING WORK-LOAN.
                    WORK-RATE, WORK-AMOUNT.
DISPLAY WORK-LOAN WORK-AMOUNT.

```

**Figure 132 • Pseudo Source Module**

```

IDENTIFICATION DIVISION.
PROGRAM-ID. SUBRTN.
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
77     WS-LOAN-AMT       PIC 9999.
77     WS-INTEREST-RATE PIC 9999.
LINKAGE SECTION.
77     LOAN-AMOUNT       PIC 9999.
77     INTEREST-RATE     PIC 9999.
77     INTEREST-AMT     PIC 9999.
PROCEDURE DIVISION USING LOAN-AMT.
                    INTEREST-RATE, INTEREST-AMT.
*
*INPUTS
*
                    MOVE LOAN-AMT TO WS-LOAN-AMT.
                    MOVE INTEREST-RATE TO WS-INTEREST-RATE.
*
*OUTPUTS
*
                    MOVE ZEROES TO INTEREST-AMT.
                    GOBACK.

```

In [Figure 132](#), the pseudo source subroutine SUBRTN contains these three parameters:

- LOAN-AMT
- INTEREST-RATE
- INTEREST-AMT

The first MOVE statement of the pseudo source module specifies a use of LOAN-AMT, indicating that LOAN-AMT is an input parameter.

The second MOVE statement indicates that INTEREST-RATE is also an input parameter.

The third MOVE statement specifies a modification to INTEREST-AMT, indicating that it is an output parameter.

Given that the report and Computation Variable extract objectives assume that all parameters are both input and output, these scenarios explain the example (see the code of the program being re-engineered in [Figure 131 on page 147](#) and [Figure 132 on page 147](#)):

### Scenario 1

A Computation Variable extract is performed for the variable WORK-AMOUNT at the DISPLAY statement.

- Without a pseudo source module, the `MOVE 10 TO WORK-AMOUNT` statement would be extracted because it is assumed that WORK-AMOUNT is both input and output.
- With a pseudo source module, the MOVE statement would not be extracted.

### Scenario 2

A Computation Variable extract is performed for the variable WORK-LOAN at the DISPLAY statement.

- Without a pseudo source module, the CALL statement would be extracted.
- With a pseudo source module, the CALL statement would not be extracted.

## Including Non-selected Code in the Logic Segment

Under certain conditions, Encore produces a result that does not include all required code. In [Figure 133](#), assuming that only the READ statement was determined to be part of the result, Encore would not have extracted line 250 (ADD +1 TO TRAN-CNT).

**Figure 133 • Logic Segment Non-selected Code Example**

```

000242
000243 ACCT-MAINTENANCE.
000244
000245 READ TRAN-FILE
000246 AT END
000247 MOVE EDF-LITERAL TO TRAN-FILE-FLAG
000248 GO TO ACCT-MAINTENANCE-EXIT.
000249
000250 ADD +1 TO TRAN-CNT.
000251
000252 MOVE TRAN-ACCT-KEY TO ACCT-KEY.
000253
000254 READ ACCT-FILE KEY IS ACCT-KEY

```

For Computational Variable extracts, Encore allows for the inclusion of omitted lines of code from the resulting Logic Segment.

- 1 Select View ► Logic Segment and press Enter to display the View - Select Logic Segment pop-up, as shown in [Figure 134](#).

**Figure 134 • View - Select Logic Segment Pop-up**

```

-----
Command ==> _ View - Select Logic Segment 2 LOGIC SEGMENTS
Scroll ==> CSR
S - Select
-----
Program Logic Segment Name Description
-----
VIARDEM GET-NUM-OF-DAYS CALCULATE NUMBER OF DAYS
S VIARDEM INTEREST-CALCULATION INTEREST-CALCULATION LOGIC SEG
***** BOTTOM OF DATA *****
-----

```

- 2 Type S next to the Logic Segment you saved in the last exercise and press Enter to display the Source View screen (see [Figure 135 on page 150](#)).

- 3 Type LOCATE 243 on the command line and press Enter to position line 243 at the top of the screen, as shown in [Figure 135](#).

**Figure 135 • Expanded Source View Screen - Zoom In Command**

```

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

000243 ACCT-MAINTENANCE.                          LOGICSEG
-----
000245 READ TRAN-FILE                              LOGICSEG
000246 AT END                                      LOGICSEG
000247 MOVE EOF-LITERAL TO TRAN-FILE-FLAG         LOGICSEG
000248 GO TO ACCT-MAINTENANCE-EXIT.               LOGICSEG
-----
000252 MOVE TRAN-ACCT-KEY TO ACCT-KEY.             LOGICSEG
-----
000254 READ ACCT-FILE KEY IS ACCT-KEY             LOGICSEG
000255 INVALID KEY                                LOGICSEG
-----
000259 MOVE +4 TO WORK-RETURN-CODE                LOGICSEG
000260 GO TO ACCT-MAINTENANCE-EXIT.              LOGICSEG
-----
000264 PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT. LOGICSEG
-----
000269 ACCT-MAINTENANCE-EXIT.                     LOGICSEG
-----

```

- 4 Place the cursor immediately below line 248 and press PF4 (Zoom in) to expand the view to include non-selected lines, as shown in [Figure 136](#).

Notice that the expanded lines (249 - 251) are not part of the Logic Segment. They are not highlighted and they have not been tagged with LOGICSEG.

**Figure 136 • Expanded Source View Screen - Zoom In Results**

```

File View Extract Generate Search List Options Help
-----
Source View                                     3 DETAIL LINES
Command ==>                                     Scroll ==> CSR

000243 ACCT-MAINTENANCE.                          LOGICSEG
-----
000245 READ TRAN-FILE                              LOGICSEG
000246 AT END                                      LOGICSEG
000247 MOVE EOF-LITERAL TO TRAN-FILE-FLAG         LOGICSEG
000248 GO TO ACCT-MAINTENANCE-EXIT.               LOGICSEG
000249 ADD *1 TO TRAN-CNT.                          LOGICSEG
000250
000251
000252 MOVE TRAN-ACCT-KEY TO ACCT-KEY.             LOGICSEG
-----
000254 READ ACCT-FILE KEY IS ACCT-KEY             LOGICSEG
000255 INVALID KEY                                LOGICSEG
-----
000259 MOVE +4 TO WORK-RETURN-CODE                LOGICSEG
000260 GO TO ACCT-MAINTENANCE-EXIT.              LOGICSEG
-----
000264 PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT. LOGICSEG
-----

```

- Type I on line 250 and press Enter, as shown in [Figure 137](#).

**Figure 137 • Including Source Line in Logic Segment**

```

File View Extract Generate Search List Options Help
-----
Source View                                     3 DETAIL LINES
Command ==>                                     Scroll ==> CSR

000243 ACCT-MAINTENANCE.                          LOGICSEG
----- 1 LINE NOT DISPLAYED
000245     READ TRAN-FILE                          LOGICSEG
000246     AT END                                  LOGICSEG
000247     MOVE EOF-LITERAL TO TRAN-FILE-FLAG     LOGICSEG
000248     GO TO ACCT-MAINTENANCE-EXIT.          LOGICSEG
000248
000250     ADD +1 TO TRAN-CNT.
000251
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.          LOGICSEG
----- 1 LINE NOT DISPLAYED
000254     READ ACCT-FILE  KEY IS ACCT-KEY         LOGICSEG
000255     INVALID KEY                              LOGICSEG
----- 3 LINES NOT DISPLAYED
000259     MOVE +4 TO WORK-RETURN-CODE            LOGICSEG
000260     GO TO ACCT-MAINTENANCE-EXIT.          LOGICSEG
----- 3 LINES NOT DISPLAYED
000264     PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT. LOGICSEG
----- 4 LINES NOT DISPLAYED

```

[Figure 138](#) shows that the Source code at line 250 is highlighted and has been included in the Logic Segment.

**Figure 138 • Source Line Included in Logic Segment**

```

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

000243 ACCT-MAINTENANCE.                          LOGICSEG
----- 1 LINE NOT DISPLAYED
000245     READ TRAN-FILE                          LOGICSEG
000246     AT END                                  LOGICSEG
000247     MOVE EOF-LITERAL TO TRAN-FILE-FLAG     LOGICSEG
000248     GO TO ACCT-MAINTENANCE-EXIT.          LOGICSEG
000248
000250     ADD +1 TO TRAN-CNT.
000251
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.          LOGICSEG
----- 1 LINE NOT DISPLAYED
000254     READ ACCT-FILE  KEY IS ACCT-KEY         LOGICSEG
000255     INVALID KEY                              LOGICSEG
----- 3 LINES NOT DISPLAYED
000259     MOVE +4 TO WORK-RETURN-CODE            LOGICSEG
000260     GO TO ACCT-MAINTENANCE-EXIT.          LOGICSEG
----- 3 LINES NOT DISPLAYED
000264     PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT. LOGICSEG
----- 4 LINES NOT DISPLAYED

```

Because the Logic Segment has been changed, you must save the file before returning to the Encore Primary screen.

- Select File ► Save Segment and press Enter to display the File - Save Logic Segments pop-up.

- 7 Type S next to the Logic Segment to be saved and press Enter to display the File - Save Segment pop-up. Type the appropriate descriptive information and press Enter to return to the File - Save Logic Segments pop-up. Press PF3 until you return to the Encore Primary screen.

**Note:** \_\_\_\_\_

In addition to including the identified statement, Encore ensures that all of the code necessary to execute the statement is also included in the generated module.

---

## Controlling Extract Boundaries

There are instances when each extract objective gives you more code than you want, e.g., a program processes multiple transactions. In these instances, you need to be able to specify the extract boundaries to better control the amount of extractable code.

In this example, you want to extract all of the compute statements for a variable associated with a particular transaction type and generate a CALLable module. First, you need to assess which type of extract you should use by considering these options:

- A Perform Range extracts the desired statements mentioned in the example, however, it also extracts all statements that can be executed by the Perform statement, including other Perform Ranges and GO TOs.
- A Transaction extract with altered START and END points cannot sufficiently narrow the scope of the extract unless the statements are adjacent.
- A Report extract isolates the output statements required, however, it also extracts all contributing or influencing statements from the selected statements to the beginning of the program, giving you extra code that is not required. Also, the Report extract does not generate CALLable modules.
- A Computation Variable extract isolates the compute statements for the variable mentioned in the example. It also extracts all contributing or influencing statements from the compute statement selected, to the beginning of the program, again selecting extra statements that are not required. As with the Report extract, it does not generate CALLable modules.

The Computation Variable extract objective has an option that allows you to extract code from an existing Perform Range or Transaction Logic Segment. Normally, within the Report and Computation Variable extract objectives, code is extracted from the selected statements to the beginning of the program. When you perform a Report or Computation Variable extract from a Logic Segment, the extract boundaries extend from the selected statements to the beginning of the Logic Segment.

## Extracting Code from an Existing Logic Segment

### To extract code from an existing Logic Segment

- 1 Follow the same procedure you perform for a conventional Computation Variable extract (see ["Extracting the Computation Variable" on page 135](#)), with this exception:

When you select the Filter option on the Extract - Data Name List screen (see [Figure 120 on page 136](#)) and the Extract - Computation Variable Filter pop-up displays, as shown in [Figure 139](#), select the Limit to within an existing Segment option by typing a forward slash (/) next to the option. Selecting this option indicates that the extract is performed against an existing Logic Segment. In addition to typing data in the various fields as you would for a conventional extract, use this option to indicate that the extract is to be done within a Logic Segment.

**Figure 139 • Selecting the Limit to within an Existing Segment Option**

```

C .-----
S      Extract - Data Name List
S      Extract - Computation Variable Filter
S      Type information for filtering Data Variable Name list.
      Data Name . . . _
      Use Context          Options
      1. References       / Aliasing      Level Number . .
      2. Uses
      3. Modifications

      CDPYBOOK Name
      Line Range . .
      Label Name . .
      COBOL Subset _____

      Options
      / Limit to within an existing Segment
  
```

- 2 Press Enter to display the Select Extract-within Logic Segment pop-up, as shown in [Figure 140](#).
- 3 Select the Logic Segment for code extraction. Only Perform Range or Transaction Logic Segments are displayed in the list of Logic Segments. Only one Logic Segment can be selected.

**Figure 140 • Select Extract-Within Logic Segment Pop-up**

```
-----
Select Extract-within Logic Segment      1 LOGIC SEGMENTS
Command ==> _____ Scroll ==> CSA

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

-----
Program      Logic Segment Name      Description
-----
VIARDEMD    GET-NUM-OF-DAYS          CALCULATE NUMBER OF DAYS
*****
***** BOTTOM OF DATA *****
-----
```

- 4 Press Enter until you return to the Extract - Data Name List pop-up.

From this point, the extract procedures are the same as for a conventional Computation Variable extract (see ["Extracting the Computation Variable" on page 135](#)). Only those statements that lie within the selected Logic Segment are eligible for extract.

**Note:**

\_\_\_\_\_

If the Logic Segment contains IO statements, you may want to consider selecting the CALLS to IO Module(s) option when you generate the program to avoid the COBOL constraint of referencing files across program boundaries.

\_\_\_\_\_

## Computation Variable Extract Compilation Issues

When Encore generates CALLable modules from a Computation Variable extract, certain conditions may occur that cause compile errors or cause the program not to run. As shown in this section, comment messages are inserted in the generated modules when these conditions occur:

### Condition 1

Encore has generated statements that cause normal exits from the program.

Cause	Action
Encore generates program exit statements (GOBACK, EXEC CICS RETURN) to ensure that the program exits properly.	Review the conditions surrounding each generated exit statement to determine if the default exit statement should be augmented or replaced.

### Condition 2

Encore has determined that a COPY statement cannot be used. Portions of the copybook are reproduced inline.

Cause	Action
Under certain conditions, the original COPY statement cannot be used in a generated module, such as when COPY statements contain dead code.  In addition to this, Encore generates comments indicating why the COPY statement could not be used as originally coded.	Determine why the COPY member was not usable. No action is required, however, the COPY member contents may be modified to use it in this context.  Example: A copy book is used in many programs. Some statements are not referenced in this program and, therefore, would be considered dead code.

### Condition 3

Encore has tailored OPEN or CLOSE statements.

Cause	Action
Encore ensures that all necessary file-related statements are extracted, including OPEN statements and CLOSE statements. When an OPEN or CLOSE statement includes references to files that are not required by the Logic Segment, Encore generates a partial statement to reference only the desired files.	None.

---

### Condition 4

STOP RUN in empty Exception clauses (i.e., ON SIZE, ON OVERFLOW)

Cause	Action
COBOL does not permit a NEXT SENTENCE to appear in an exception clause. Encore inserts a STOP RUN in an exception clause that is part of a Computation Variable extract.	Revise to include a more meaningful error handling routine.

---

---

# 6

## Transaction Extract

---

This chapter contains a practical demonstration of how to perform a Transaction extract to create a replacement standalone program, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">157</a>
<a href="#">The Business Scenario</a>	<a href="#">158</a>
<a href="#">Starting an Encore Session</a>	<a href="#">159</a>
<a href="#">Extracting the Objective</a>	<a href="#">160</a>
<a href="#">Creating the Replacement Module</a>	<a href="#">165</a>
<a href="#">Changing the Start/End Points</a>	<a href="#">169</a>
<a href="#">Transaction Extract Compilation Issues</a>	<a href="#">178</a>

### Introduction

This chapter contains a scenario that demonstrates the technique of using a Transaction extract to create a replacement standalone program called INTCALC, which does not contain processing logic for money market accounts.

The Encore demonstration program VIARDEMO is used for the examples in this chapter. For a partial listing of VIARDEMO, see ["The Demonstration Program" on page 49](#).

## The Business Scenario

You have been assigned the task of removing all processing logic for money market accounts in the Interest Calculation Program (ICP).

After performing your initial analysis, you build a checklist that contains these tasks, which are involved in extracting the code you require:

- Identify the variable that contains the identifying code for money market accounts.
- Extract all of the logic for money market account processing.
- Generate a replacement module without the money market account processing logic.

You determine that the money market accounts are identified on the input file by the MMA code in the data field ACCT-TYPE of ACCT-RECORD. The conditional that drives the transaction processing is in paragraph UPDATE-ACCT in the VIARDEMO demonstration program, as shown in [Figure 141](#). Since the logic you need to extract is driven by the MMA transaction code, you determine that the best extraction method would be a Transaction extract.

**Figure 141 • Money Market Accounting Example**

```
000273 UPDATE-ACCT.
000274
000275     PERFORM INIT-ACTION-RECORD.
000276
000277     IF ACCT-TYPE OF ACCT-RECORD = 'INT'
000278         MOVE CHK-INTEREST-RATE TO INTEREST-RATE
000279     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
000280         MOVE ZERO TO INTEREST-RATE
000281     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
000282         MOVE MMA-INTEREST-RATE TO INTEREST-RATE
000283     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
000284         MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
000285
000286     IF ACCT-MULTI-ACCT = 'YES'
000287         ADD CHK-MULTI-ACCT-RATE TO INTEREST-RATE
000288         MOVE ACTION-MULTI-ACCT TO WORK-ACTION-REASON
000289         PERFORM INITIATE-ACTION.
000290
```

**Note:** \_\_\_\_\_

The choice of a particular extract method does not necessarily mean it is the best or only method. Preliminary analysis helps narrow the available choices. The final choice is up to you.

\_\_\_\_\_

## Starting an Encore Session

### *To start the Encore session*

**Note:** \_\_\_\_\_

Logon procedures and AKR information are unique to your programming environment. If necessary, contact your systems administrator or your Encore coordinator for the correct dataset names.

---

- 1 Logon to the ESW Primary screen.
- 2 Select Re-engineer ▶ Program and press Enter to display the Encore Primary screen.

**Or**

Type EN on the command line and press Enter to display the Encore Primary screen.

- 3 Select File ▶ Open from the Encore Primary screen action bar and press Enter to display the File - Open Program pop-up.
- 4 Type the AKR dataset name in the Data Set Name field and VIARDEMO in the Program Name field and press Enter.

See ["Starting an Encore Session" on page 59](#) for more information.

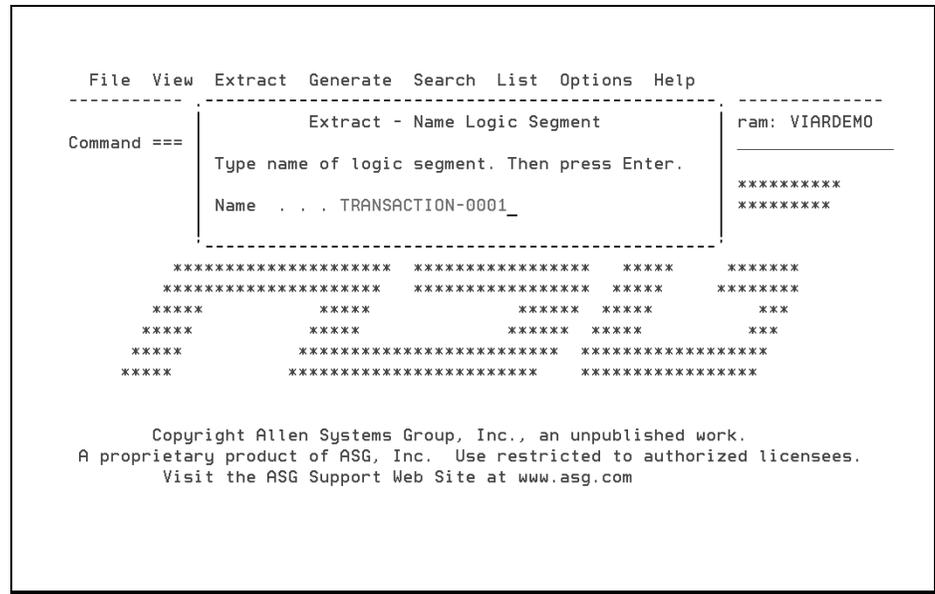
## Extracting the Objective

*To perform a Transaction Extract using the ACCT-TYPE variable*

- 1 Select Extract ▶ Transaction and press Enter to display the Extract - Name Logic Segment pop-up, as shown in [Figure 142](#).

The Logic Segment Name field contains a default name, generated by Encore for the Logic Segment that you extract. You can assign your own name to the Logic Segment or you can use the default name. For this example, use the default name TRANSACTION-0001.

**Figure 142 • Extract - Name Logic Segment Pop-up**



- 2 Press Enter to display the Extract - Transactions Paths List screen, which lists all transaction paths in the program, as shown in [Figure 144](#).

**Figure 143 • Extract - Transaction Paths List Screen**

```

-----
                          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
-----
= 000268      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 = 'HMA'
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
-----

```

The first step in the process is to filter the transaction lists to select those paths referencing ACCT-TYPE.

- 3 Type `Filter` on the command line and press `Enter` (or press `PF5`) to display the Extract - Data Name List Filter pop-up, as shown in [Figure 144](#).

This pop-up displays all the data names associated with the transaction paths that are listed for the program. You can select a data name from the list, or you can use the Filter option to reduce the list or select a specific data name. For this exercise, use the filter option to select the data name `ACCT-TYPE`.

**Figure 144 • Extract - Data Name List Filter Pop-up**

```

-----
Command ==> _          Extract - Data Name List Filter          17 DATA NAMES
                               Scroll ==> CSA

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
-----

```

- 4 Type `Filter` again on the command line and press `Enter` (or press `PF5`) on the Extract - Data Name List Filter pop-up to display the Extract - Transaction Variable pop-up (see [Figure 145 on page 163](#)).

See the online help for more detailed information about Transaction extract features.

- 5 Type ACCT-TYPE in the Data Name field to select all transaction paths using this variable name, as shown in [Figure 145](#).

You can also enter a pattern for the variable name containing wildcards (? for one character; \* for zero or more characters) to reduce the list of available selections.

**Figure 145 • Extract - Transaction Variable Pop-up**

```

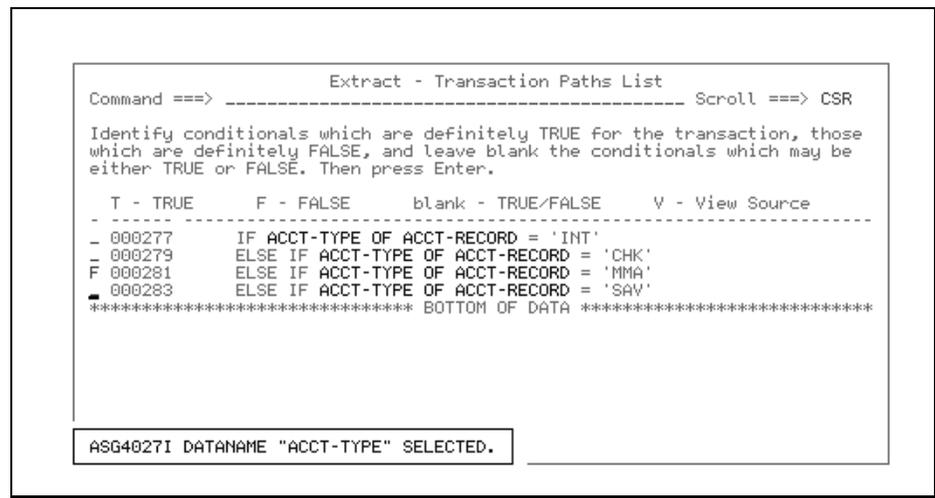
C -----
S                               Extract - Data Name List Filter
c                               -----
                               Extract - Transaction Variable
                               -----
c  Type information for filtering Data Variable Name list.
                               -----
Data Name . . . ACCT-TYPE
_ Aliasing   Level Number . . . . .
                               -----
COPYBOOK Name _____
Line Range . . _____
- Label Name . . _____
- COBOL Subset  COND
-
- -----
- 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
-----

```

- 6 Press Enter to redisplay the Extract - Transaction Paths List screen, as shown in [Figure 146](#).

This screen redisplay with all paths referencing the variable ACCT-TYPE highlighted and the message ASG4027I DATA NAME "ACCT-TYPE" SELECTED in the long message area. You can indicate which conditionals (IF statements) can be true (T) for the replacement module, which conditionals can be false (F), and which conditionals can be true or false (blank). For this extract, you want to remove the logic associated with the MMA transaction code, so the statement associated with that transaction must be false.

**Figure 146 • Extract - Transaction Paths List Screen with Variable Name Selected**



- 7 Type F on line 281, which disables the True path for this conditional in the replacement module. The remaining IFs for ACCT-TYPE are left blank, since they can be either true or false. Press Enter.
- 8 Press Enter twice to return to the Encore Primary screen.

## Creating the Replacement Module

### *To create the replacement module*

- 1 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment screen, as shown in [Figure 147](#).

This screen lists all of the Logic Segments you have extracted during your current session, as well as all Logic Segments that have been saved in the AKR. The last Logic Segment extracted (in this case TRANSACTION-0001) is highlighted and tagged for selection. TRANSACTION-0001 is the Logic Segment you use to generate the COBOL program.

**Figure 147 • Generate - Select Logic Segment Screen**

```

-----
Command ==>  _          Generate - Select Logic Segment          2 LOGIC SEGMENTS
                                Scroll ==> CSR
$ - Select
Program      Logic Segment Name      Description
-----
VIARDEMO    GET-NUH-OF-DAYS           CALCULATE NUMBER OF DAYS
S VIARDEMO  TRANSACTION-0001                   Transaction
*****
***** BOTTOM OF DATA *****
-----

```

**Note:** \_\_\_\_\_

The Generate - Select Logic Segment pop-up is scrollable, but you must first blank out the S select code.

\_\_\_\_\_





**Note:**

CREATE may be issued only against an empty member. If INTCALC already exists in your library, use the REPLACE command. See the online help and the *ASG-Encore Reference Guide* for more information about these commands.

[Figure 151](#) shows the Encore-generated notes added to the INTCALC standalone program indicating that STOP-RUN was inserted into the code.

**Figure 151 • Encore Blocked False Path - INTCALC - 1**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==>                                     Scroll ==> CSR
000031      * ASG-ENCORE NOTES:
000032      *
000033      * 1. "STOP RUN" WAS INSERTED TO ABNORMALLY TERMINATE THE PROGRAM
000034      *   FOR CONDITIONS WHICH SHOULD NOT OCCUR. PLEASE EXAMINE EACH
000035      *   INSERTION TO DETERMINE IF THIS ACTION IS APPROPRIATE.
000036      *
000037      *

```

[Figure 152](#) show the generated code blocking the false path in the INTCALC standalone program. This is the result of marking the conditional for ACCT-TYPE = MMA false. Encore generates a NO PATH and a STOP RUN if the conditional is ever true, but you may want to change these entries to something more meaningful.

**Figure 152 • Encore Blocked False Path - INTCALC - 2**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==>                                     Scroll ==> CSR
000280
000281      PERFORM INIT-ACTION-RECORD.
000282
000283      IF ACCT-TYPE OF ACCT-RECORD = 'INT'
000284      MOVE CHK-INTEREST-RATE TO INTEREST-RATE
000285      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
000286      MOVE ZERO TO INTEREST-RATE
000287      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
000288
000289      STOP RUN
000290      *
000291      *
000292      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
000293      MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
000294

```

**Note:**

Rather than having Encore display an error message, you can direct Encore to issue a CALL to your shop standard error abend program (if you have one) by specifying the name of your abend program on the Options - Product Parameters pop-up (see [Figure 7 on page 27](#)).

***To close Encore***

- 1 Press PF3 until you are returned to the Encore Primary screen.
- 2 Select File ► Close and press Enter to display the Close Program pop-up.
- 3 Select Close, discard segments. All unsaved Logic Segments are discarded when you exit Encore by selecting the Exit option. For this exercise, you do not need to save any generated data.

**Note:** \_\_\_\_\_

The Exit pop-up displays if you attempt to exit Encore using PF3. From this pop-up, options are available to save logic segments, discard logic segments, or cancel the exit and return to Encore.

---

## Changing the Start/End Points

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 extract to be performed, they can be changed.

After initiating Encore and opening program VIARDEMO (see "[Starting an Encore Session](#)" on page 159), use these steps to change the default START and END points.

***To change START and END point defaults***

- 1 Select Extract ► Transaction and then press Enter to display the Extract - Name Logic Segment pop-up.

- 2 Press Enter again to display the Extract - Transactions Path List pop-up, as shown in [Figure 153](#).

**Figure 153 • 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
-----
= 000268      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 = 'HMA'
000283      ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAY'
- 000288      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
-----
    
```

- 3 Press PF6 to display the Extract - START/END Statements pop-up, as shown in [Figure 154](#).

**Figure 154 • Extract - START/END Statements Pop-up - Deselect START and END Points**

```

-----
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
-----
u 000225  PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE START
u 000226  BANK-BALANCE, START
u 000240  GOBACK. END
***** BOTTOM OF DATA *****
    
```

- 4 Type U on lines 225, 226, and 240 and press Enter to de-select the current START and END points.

[Figure 155](#) shows the Extract - START/END Statements pop-up redisplayed with the message NO STARTING STATEMENT in the upper right corner of the screen.

**Figure 155 • Extract - START/END Statements Pop-up - Information Message Display**

```

-----
Extract - START/END Stateme      NO STARTING STATEMENT
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 ENTRY
_ 000226                                BANK-BALANCE.                      ENTRY
_ 000240 GOBACK.                                                                EXIT
***** BOTTOM OF DATA *****

```

- 5 Type 2 on the command line, as shown in [Figure 156](#), and press Enter to expand the display to include all the Procedure Division statements (see [Figure 157 on page 172](#)).

**Figure 156 • Extract - START/END Statements Pop-up - List All Statements**

```

-----
Extract - START/END Statements
Command ==> 2_                      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
-----
000225 PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE ENTRY
000226                                BANK-BALANCE.                      ENTRY
_ 000240 GOBACK.                                                                EXIT
***** BOTTOM OF DATA *****

```

[Figure 157](#) shows that the Extract - START/END Statements pop-up now displays the expanded information. You are now ready to select a START statement.

**Figure 157 • Extract - START/END Statements Pop-up - Expanded View**

```

-----
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
-----
- 000225 PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE ENTRY
- 000226                                BANK-BALANCE.                ENTRY
- 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.
-----

```

**To select a new START statement**

- 1 Scroll to line 273 on the redisplayed Extract - START/END Statements pop-up, as shown in [Figure 158](#).

**Figure 158 • Extract - START/END Pop-up - Scrolling to Line 273**

```

-----
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
-----
- 000273 UPDATE-ACCT.
- 000274
- 000275 PERFORM INIT-ACTION-RECORD.
- 000276
- 000277 IF ACCT-TYPE OF ACCT-RECORD = 'INT'
- 000278 MOVE CHK-INTEREST-RATE TO INTEREST-RATE
- 000279 ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
- 000280 MOVE ZERO TO INTEREST-RATE
- 000281 ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
- 000282 MOVE MMA-INTEREST-RATE TO INTEREST-RATE
- 000283 ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
- 000284 MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
-----

```

- 2 Type S next to the new START line UPDATE-ACCT, as shown in [Figure 159](#), and press Enter.

**Figure 159 • Extract - START/END Pop-up - Selecting START Statement**

```

-----
Command ==> Extract - START/END Statements 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
-----
S 000273 UPDATE-ACCT.
- 000274
- 000275     PERFORM INIT-ACTION-RECORD.
000276
000277     IF ACCT-TYPE OF ACCT-RECORD = 'INT'
000278         MOVE CHK-INTEREST-RATE TO INTEREST-RATE
- 000279     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
- 000280         MOVE ZERO TO INTEREST-RATE
- 000281     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
000282         MOVE MMA-INTEREST-RATE TO INTEREST-RATE
000283     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
000284         MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
-----

```

[Figure 160](#) shows that the new START line (line 273) now has the START tag on the far right of the screen. (A START point includes an entire COBOL statement. In this case, the statement is on line 273.) You are now ready to select an END statement.

**Figure 160 • Extract - START/END Statement Pop-up - Results of Start Change**

```

-----
Command ==> Extract - START/END Statement NO END STATEMENT(S) 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
-----
000273 UPDATE-ACCT. START
000274
000275     PERFORM INIT-ACTION-RECORD.
- 000276
000277     IF ACCT-TYPE OF ACCT-RECORD = 'INT'
000278         MOVE CHK-INTEREST-RATE TO INTEREST-RATE
000279     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'
- 000280         MOVE ZERO TO INTEREST-RATE
- 000281     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'MMA'
000282         MOVE MMA-INTEREST-RATE TO INTEREST-RATE
- 000283     ELSE IF ACCT-TYPE OF ACCT-RECORD = 'SAV'
000284         MOVE SAV-INTEREST-RATE TO INTEREST-RATE.
-----

```

*To select a new END statement*

- 1 Scroll to line 325, type E next to the new END line, as shown in [Figure 161](#), and press Enter.

**Figure 161 • Extract - START/END Statement Pop-up - Select END Statement**

```

-----
Command ==> Extract - START/END Statements 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
-----
- 000321
- 000322 REWRITE ACCT-RECORD.
- 000323
- 000324 UPDATE-ACCT-EXIT.
E 000325 EXIT.                                PERF-END
- 000326
- 000327 EXCEPTION-REPORT-1.
- 000328 *
- 000329 * EXCEPTION REPORT 1 LIST ALL ACCOUNTS WITH A MINIMUM BALANCE
000330 * THAT IS LESS THAN THE CURRENT BALANCE.
000331 *
000332 IF EX-RPT-LINE-CNT IS GREATER THAN 60
-----

```

[Figure 162](#) shows that the display is now highlighted with the END tag on the far right of the screen. The highlighting indicates all statements that are reachable from the set START point.

**Figure 162 • Extract - START/END Statement Pop-up - Highlighted Display**

```

-----
Command ==> _ Extract - START/END Statements 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
-----
000321
000322 REWRITE ACCT-RECORD.
000323
- 000324 UPDATE-ACCT-EXIT.
- 000325 EXIT.                                END
000326
000327 EXCEPTION-REPDR1-1.
- 000328 *
- 000329 * EXCEPTION REPORT 1 LIST ALL ACCOUNTS WITH A MINIMUM BALANCE
- 000330 * THAT IS LESS THAN THE CURRENT BALANCE.
000331 *
000332 IF EX-RPT-LINE-CNT IS GREATER THAN 60
-----

```

- 2 Press Enter to return to the Extract - Transaction Paths List pop-up and press Enter again to return to the Encore Primary screen. You are now ready to generate a CALLable submodule.

**To generate a CALLable submodule**

- 1 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment pop-up.
- 2 Press Enter to display the Generate - COBOL Module pop-up, type in the required information, and press Enter.
- 3 Encore generates a CALLable submodule that executes the code you selected by setting the new START and END points (see [Figure 163](#)).

**Figure 163 • Generate - Encore-Generated Callable SubModule**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==>                                     Scroll ==> CSR
000205      PROCEDURE DIVISION USING
000206          TRAN-RECORD
000207          PGM-SIGNATURE
000208          INTEREST-RATE
000209          DAYS-IN-PERIOD
000210          DAYS-IN-YEAR
000211          WORK-ACTION-REASON
000212          ACTION-CNT
000213          TODAY-DATE
000214          CONTROL-PARAMETERS
000215          BRANCH-BALANCE
000216          BANK-BALANCE.
000217
000218
000219
000220      UPDATE-ACCT.
000221
000222          PERFORM INIT-ACTION-RECORD.
000223
000224          IF ACCT-TYPE OF ACCT-RECORD = 'INT'
000225              MOVE CHK-INTEREST-RATE TO INTEREST-RATE
000226          ELSE IF ACCT-TYPE OF ACCT-RECORD = 'CHK'

```

- 4 After saving the module, press PF3 until you return to the Encore Primary screen.
- 5 Select File ► Close and press Enter to display the Close Program pop-up.
- 6 Select Close, discard segments. All unsaved Logic Segments are discarded when you exit Encore by selecting the Exit option. For this tutorial, you do not need to save any generated data.

**Note:**

The Exit pop-up displays if you exit Encore using the PF3 key. From this pop-up, options are available to save logic segments, discard logic segments, or cancel the exit and return to Encore.

## START/END Usage Notes

When you select Option 2, List ALL statements on the Extract - START/END Statement pop-up (see [step 5 on page 171](#) and [Figure 156 on page 171](#)), all PROCEDURE DIVISION statements in the program are displayed on the screen. Selection of START and END statements may be based on the cursor location.

**Figure 164 • Start/End Usage Notes Example**

```

000359 *
000360 ACCEPT TODAY-DATE FROM DATE.
000361
000362 COMPUTE DAYS-IN-PERIOD =
000363 MONTH (TODAY-MM) - TODAY-DD.
000364 MOVE +0 TO DAYS-IN-YEAR.
000365 PERFORM SUM-TOTAL-DAYS
000366 VARYING WORK-CNT FROM 1 BY 1
000367 UNTIL TODAY-DD = WORK-CNT. RETURN
000368 *
000369 SUM-TOTAL-DAYS.
000370 COMPUTE DAYS-IN-YEAR = DAYS-IN-YEAR + MONTH ( WORK-CNT ). RETURN
000371 *
000372
000373 INITIALIZE-PGM.
000374
000375 PERFORM OPEN-FILES.
000376
000377 MOVE +0 TO ACCT-IO-CNT

```

The rules for cursor location are explained in these steps, using the code in [Figure 164](#) as an example. In the rules for selecting a statement, the E line command (to select an END statement) is used as an example.

- If the E line command is used on a line containing a single COBOL statement, e.g., line 364, that statement is selected as an END statement, highlighted, and tagged.
- If the E line command is used on a line containing a COBOL statement that spans more than one line, e.g., lines 362 and 363, the entire statement is selected. The E line command could be entered on either line 362 or line 363. The statement is highlighted and tagged.
- If the E line command is used on a line containing more than one full or partial COBOL statement (not shown in this example), use this procedure:
  - Type E on the selection line and move the cursor to the desired statement and press Enter. The statement is selected as an END statement, highlighted, and tagged.

When selecting a START statement, type S next to an executable COBOL statement. If the selected statement is an unexecutable COBOL statement, these general rules apply:

- Comment lines and compiler directives occurring at the end of an executable code block are considered part of the paragraph or section that follows. The starting point begins at the paragraph or section name that follows.
- A line followed by one or more executable statements prior to the succeeding section, or a line that is part of the last section, is considered part of the preceding section. The starting point is the end of the statement, before the comment or directive.
- Blank lines that occur at the end of an executable code block, and that are not preceded by a comment or compiler directive, are considered part of the preceding paragraph or section. The starting point is at the end of the last executable statement in the preceding paragraph.
- A blank line that follows all executable statements of the preceding paragraph, and that is preceded by a comment or compiler directive, is considered part of the following paragraph. The starting point is the paragraph or section name in the following paragraph.
- A blank line that is followed by one or more executable statements, or that is part of the last paragraph or section of the program, is considered part of the preceding paragraph. The starting point is the end of the preceding paragraph.

The statements that compose the current transaction are highlighted. The highlighting changes if the START or END statements are redefined, or if conditional paths are blocked or unblocked on the Extract - Transaction Paths List screen.

**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, or without passing through a blocked conditional path.

---

## Transaction Extract Compilation Issues

When Encore generates modules from a Transaction extract, certain conditions may occur that cause compile errors or cause the program not to run. As shown in this section, Encore inserts comment messages in the generated modules when these conditions occur.

### Condition 1

An unnamed 01 level data item, moved from WORKING-STORAGE to LINKAGE, must be assigned a valid COBOL data item name (sub-program request).

Cause	Action
<ul style="list-style-type: none"><li>An unnamed 01 level was moved to the LINKAGE SECTION. When Encore generated the LINKAGE SECTION for a subprogram, it did not rename data items that were moved from the WORKING-STORAGE SECTION.</li></ul>	Assign a valid COBOL data item name to each LINKAGE SECTION data item with the name '_FILLER'.
<ul style="list-style-type: none"><li>COBOL syntax allows an unreferenced 01 level data item to go unnamed in WORKING-STORAGE, but not in LINKAGE.</li></ul>	
<ul style="list-style-type: none"><li>Encore assigns the name '_FILLER' to the data item and issues this message. The subprogram cannot successfully compile until the data item is given a valid name.</li></ul>	
<ul style="list-style-type: none"><li>A similar situation occurs when two 01 levels are defined with the same name and are not referenced at the 01 level. In this case, the compiler detects a duplicate name.</li></ul>	

---

**Condition 2**

Encore has generated statements to cause normal and abnormal exits from the program.

Cause	Action
Encore generates program exit statements (STOP RUN, GOBACK, EXEC CICS RETURN, EXEC CICS ABEND) to ensure that the program exits properly. A normal program exit is generated when the intended function of the program is complete. For a transaction objective, an abnormal program exit is generated for conditional clauses that have been blocked using TRUE/FALSE.	Review the conditions surrounding each generated exit statement to determine if the default exit statement should be augmented or replaced. For a transaction objective, ensure that the proper program paths were selected.

**Condition 3**

Encore has determined that a COPY statement cannot be used. Portions of the copybook are reproduced inline.

Cause	Action
<p>Under certain conditions, the original COPY statement cannot be used in a generated module:</p> <ul style="list-style-type: none"> <li>• When data elements in the COPY statement contain definitions split between the WORKING-STORAGE and LINKAGE sections.</li> <li>• When COPY statements contain dead code.</li> <li>• When the contents of a COPY statement used in the LINKAGE SECTION contain the VALUE clause.</li> </ul> <p>In addition to this message, Encore generates comments indicating why the COPY statement could not be used as originally coded.</p>	<p>Determine why the COPY member was not usable. No action is required, however, the COPY member contents may be modified in order to use it in this context.</p> <p>Example:</p> <p>A copybook contains both FD and data item definitions. One or more of the data item definitions must be used in the LINKAGE SECTION.</p>

### Condition 4

Encore has generated a LABEL target used to transfer control to a transaction START statement.

Cause	Action
For a transaction objective Logic Segment, the start point of the transaction was not the first statement of the program module. Encore generated a label, with a GO TO statement to branch to that label, to create a path to the transaction code.	Review and verify.

---

### Condition 5

Encore has generated a DISPLAY statement prior to each abnormal program exit.

Cause	Action
Encore has generated DISPLAY statements just prior to a generated abnormal program exit.	Review the conditions surrounding each generated DISPLAY statement to determine if it should be augmented or replaced. Ensure that a DISPLAY is an appropriate action for the condition that occurred.

---

**Note:** \_\_\_\_\_

A DISPLAY statement is not generated for a CICS program, or when a subroutine is specified in the COBOL Module Exit field on the Options - Product Parameters pop-up (see [Figure 7 on page 27](#)).

---

---

# 7

## Report Extract

---

This chapter provides a practical demonstration of how to perform a Report extract to create sample programs that perform month-end calculations and produce exception reports, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">181</a>
<a href="#">The Business Scenario</a>	<a href="#">182</a>
<a href="#">Starting an Encore Session</a>	<a href="#">183</a>
<a href="#">Extracting the Report</a>	<a href="#">184</a>
<a href="#">Creating the COBOL Module</a>	<a href="#">188</a>
<a href="#">Understanding the Results of the Extract</a>	<a href="#">191</a>
<a href="#">Creating the Complement Module</a>	<a href="#">194</a>
<a href="#">Complement Module - Understanding Generated Notes</a>	<a href="#">198</a>
<a href="#">Using Pseudo Source Modules to Change Logic Segments</a>	<a href="#">199</a>
<a href="#">Controlling Extract Boundaries</a>	<a href="#">207</a>
<a href="#">Report Extract Compilation Issues</a>	<a href="#">210</a>

### Introduction

This chapter demonstrates how to use Encore to create these programs:

- A program that performs all of the month-end calculations for the different types of checking accounts without producing the exception report.
- A standalone program that produces an exception report.

The Encore demonstration program VIARDEMO is used for the examples in this chapter. For a partial listing of VIARDEMO, see ["The Demonstration Program" on page 49](#).

## The Business Scenario

The Universal Bank has a program that does the month-end calculations for each type of checking account that is run at the close of business at month-end. The process cycle requires the maximum time allowed to run in the available process window. The program produces an exception report that is produced after month-end (see [Figure 165](#)). Management has requested that the report be run standalone, thereby extending the process window. Extracting the report from the existing program would be time consuming and carries the risk of corrupting the current program.

**Figure 165 • Report Extract Business Scenario Example**

```
EXCEPTION-REPORT-1.
*
* EXCEPTION REPORT 1 LIST ALL ACCOUNTS WITH A MINIMUM BALANCE
* THAT IS LESS THAN THE CURRENT BALANCE.
*
    IF EX-RPT-LINE-CNT IS GREATER THAN 60
        ADD +1 TO EX-RPT-PAGE-CNT
        MOVE EX-RPT-PAGE-CNT TO EXCEP1-REPORT-PAGE-CNT.
        WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE1.
        WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE2.
        MOVE 0 TO EX-RPT-LINE-CNT.

    ADD +1 TO EX-RPT-LINE-CNT.
    MOVE ACCT-BALANCE TO EX-CUR-BAL.
    MOVE ACCT-MIN-BALANCE TO EX-REPORT-MIN-BAL.
    WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE1.
    WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE2.
```

Encore can be used to automate the decomposition process of removing the report from the month-end calculation program without changing the rest of the program's functions. In addition, Encore does not change the original program so there is no danger of inadvertently changing the calculation functionality.

The object of this decomposition exercise is to remove a file and all the data statements and logic statements that contribute to the output file in the program. The Report extract locates the minimum set of statements that contribute to the content of an output file, when the IO statements that produce the file are identified. For this reason, the logical extract choice is the Report extract.

## Starting an Encore Session

### *To start the Encore session*

**Note:** \_\_\_\_\_

Logon procedures and AKR information are unique to your programming environment. If necessary, contact your systems administrator or your Encore coordinator for the correct dataset names.

---

- 1 Logon to the ESW Primary screen.
- 2 Select Re-engineer ▶ Program and press Enter to display the Encore Primary screen.

**Or**

Type EN on the command line and press Enter to display the Encore Primary screen.

- 3 Select File ▶ Open and press Enter to display the File - Open Program pop-up.
- 4 Type the AKR dataset name in the Data Set Name field and VIARDEMO in the Program Name field and press Enter.

See ["Starting an Encore Session" on page 59](#) for more information.



- 3 Type S next to the FD name EXCEPTION-FILE and press Enter to display the Extract - Source Ordered View pop-up (see [Figure 168 on page 186](#)).

[Figure 167](#) contains a list of all report names in the program.

**Figure 167 • 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

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

PF4=View PF5=Filter PF17=Sort
-----

```

**Note:**

You can reduce the list of selectable names by pressing PF5, which displays the Extract - Report Name Filter pop-up. This pop-up is used to specify filtering criteria. See the online help for more information about the Extract - Report Name Filter pop-up.

- 4 Type S next to line numbers 335, 336, 341, and 342 and press Enter, as shown in [Figure 168](#). This pop-up displays all of the IO statements associated with EXCEPTION-FILE. You need to extract all statements that contribute to the output data.

**Figure 168 • Extract - Source Ordered View Pop-up - Statement Selected**

```
-----  
Extract - Source Ordered View  
Command ==> _                               Scroll ==> CSR  
  
Select statements for Report Extract, press Enter. When complete press  
Enter.  
  
Report Name: EXCEPTION-FILE  
  
S - Select      V - View Source      U - Unselect  
-----  
S 000335      WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE1.  
S 000336      WRITE EXCEPT-BUF FROM EXCEPT-HEAD-LINE2.  
S 000341      WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE1.  
S 000342      WRITE EXCEPT-BUF FROM EXCEPT-DETAIL-LINE2.  
*****  
***** BOTTOM OF DATA *****  
*****
```

- 5 Press Enter again to return to the Extract - Report Name List pop-up (see [Figure 169 on page 187](#)).

The message in the long message area shown in [Figure 169](#) indicates that the Report Statements for the EXCEPTION-FILE report was successfully selected.

- 6 Press Enter to return to the Encore Primary screen.

**Figure 169 • Extract - Report Name List Pop-up**

```

                                Extract - Report Name List
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

Name                Type      Organization      OPEN      Num
-----            -
- 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 *****

```

ASG4043I REPORT STATEMENTS SUCCESSFULLY SELECTED.

See the online help for more detailed information about Extract features.

## Creating the COBOL Module

Now you need to create a standalone module to produce the Exception report. This is done by using the Logic Segment created in ["Extracting the Report" on page 184](#). The module contains all of the statements required to produce the Exception Report along with the necessary file and working storage elements.

- 1 Select Generate ► COBOL module and press Enter to display the Generate - Select Logic Segment pop-up, as shown in [Figure 170](#).

**Figure 170 • Generate - Select Logic Segment Pop-up**

```
-----
Generate - Select Logic Segment      3 LOGIC SEGMENTS
Command ==> _____ Scroll ==> CSA
$ - Select
-----
Program      Logic Segment Name      Description
-----
_ VIARDEMO   GET-NUM-OF-DAYS                   CALCULATE NUMBER OF DAYS
_ VIARDEMO   INTEREST-CALCULATION              INTEREST-CALCULATION LOGIC SEG
S VIARDEMO   REPORT-0001                        Report
*****
***** BOTTOM OF DATA *****
*****
```

- 2 Type S next to the Logic Segment name REPORT-0001 and press Enter to display the Generate - COBOL Module pop-up (see [Figure 171 on page 189](#)).



- 5 Type the name you want to save the program under in your source library (REPTPRGM in this example, as shown in [Figure 173](#)), verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter. You are returned to the Edit screen where message Member REPTPRGM created is displayed in the upper right corner of the screen confirming that the member has been successfully saved (see [Figure 174 on page 191](#)).

**Figure 173 • Edit/View - Create Pop-up**

```

                                         Edit/View - Create
Command ==>

"Current" Data Set: USERID.ENCXXXXX.COBOL

To ISPF Library:
Project . . . ASG
Group . . . CE60XXXX
Type . . . CNTL
Member . . . REPTPRGM

To Other Partitioned Data Set Member:
Data Set Name . . .
Volume Serial . . . (If not cataloged)

Data Set Password . . (If password protected)

Enter "/" to select option
Specify pack option for "CREATE" Data Set

Press ENTER key to create. Enter END command to cancel create.
```



[Figure 175](#) shows how Encore flags any code changes it has made with comments imbedded within the program.

**Figure 175 • Report Module - Screen 2**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                           Scroll ==> CSR
000172      * ASG-ENCORE NOTES:
000173      *
000174      *      1. VALUE CLAUSES OMITTED FROM LINKAGE ARE SHOWN AS COMMENTS.
000175      *      2. USED, MODIFIED AND REFERENCED (USE/MOD) SYMBOLS ARE
000176      *      NOTED.
000177      *      3. ANNOTATED COPYBOOK SYMBOLS ARE REPRODUCED AS COMMENTS.
000178      *
000179      01 CONTROL-PARAMETERS.
000180      05 FREE-SERVICE-MIN                PIC S9(11)V9(4) COMP-3.
000181      *
000182      05 CD-REVIEW-MIN                    PIC S9(11)V9(4) COMP-3.
000183      01 BRANCH-BALANCE                  PIC S9(11)V9(4) COMP-3.
000184      01 BANK-BALANCE                    PIC S9(11)V9(4) COMP-3.
000185
000186      PROCEDURE DIVISION USING CONTROL-PARAMETERS, BRANCH-BALANCE,
000187      BANK-BALANCE.
000188
000189      OPEN OUTPUT EXCEPTION-FILE.
000190
000191      PERFORM INITIALIZE-PGM.
000192
000193      PERFORM ACCT-MAINTENANCE

```

Encore has included only those statements and files that contribute to the creation of the Exception Report. Notice in [Figure 176](#) that ACTION-FILE was not extracted from VIARDEMO, since it does not contribute to the creation of the Exception Report file.

**Figure 176 • Report Module - Screen 3**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                           Scroll ==> CSR
000043      ENVIRONMENT DIVISION.
000044      CONFIGURATION SECTION.
000045      SOURCE-COMPUTER. IBM-370.
000046      OBJECT-COMPUTER. IBM-370.
000047      INPUT-OUTPUT SECTION.
000048
000049      FILE-CONTROL.
000050      SELECT ACCT-FILE                ASSIGN TO ACCT
000051      ORGANIZATION IS INDEXED
000052      ACCESS IS RANDOM
000053      RECORD KEY IS ACCT-KEY.
000054      SELECT EXCEPTION-FILE          ASSIGN TO UT-S-EXCEPTS.
000055      SELECT TRAN-FILE                ASSIGN TO UT-S-TRAN.
000056
000057      DATA DIVISION.
000058
000059      FILE SECTION.
000060      FD ACCT-FILE
000061      LABEL RECORDS ARE STANDARD.
000062      01 ACCT-RECORD.
000063      05 ACCT-CONTROL-CODE                PIC 9(6).
000064      05 ACCT-KEY.

```

[Figure 177](#) shows that the OPEN and CLOSE statements in VIARDEMO reference TRAN-FILE, ACCT-FILE, EXCEPTION-FILE, and ACTION-FILE. Since ACTION-FILE was not extracted, Encore modified the OPEN and CLOSE statements in REPTPRGM that referenced ACTION-FILE. The original OPEN and CLOSE statements remain in the program, but they are commented out.

After reviewing the extract results, press PF3 until you return to the Encore Primary screen.

**Figure 177 • Report Module - Screen 4**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                           Scroll ==> CSR
000329      OPEN-FILES.
000330
000331      *   OPEN  INPUT  TRAN-FILE
000332      *   I-O   ACCT-FILE
000333      *   OUTPUT ACTION-FILE.
000334      *                                           ASG-ENCORE - PARTSTMT
000335      OPEN  INPUT  TRAN-FILE
000336      I-O   ACCT-FILE.
000337
000338
000339      CLOSE-FILES.
000340
000341      CLOSE EXCEPTION-FILE.
000342
000343      *   CLOSE TRAN-FILE
000344      *   ACCT-FILE
000345      *   ACTION-FILE.
000346      *                                           ASG-ENCORE - PARTSTMT
000347      CLOSE TRAN-FILE
000348      ACCT-FILE.
000349
000350      * ASG-ENCORE NOTE: ***** END OF GENERATED MODULE *****

```

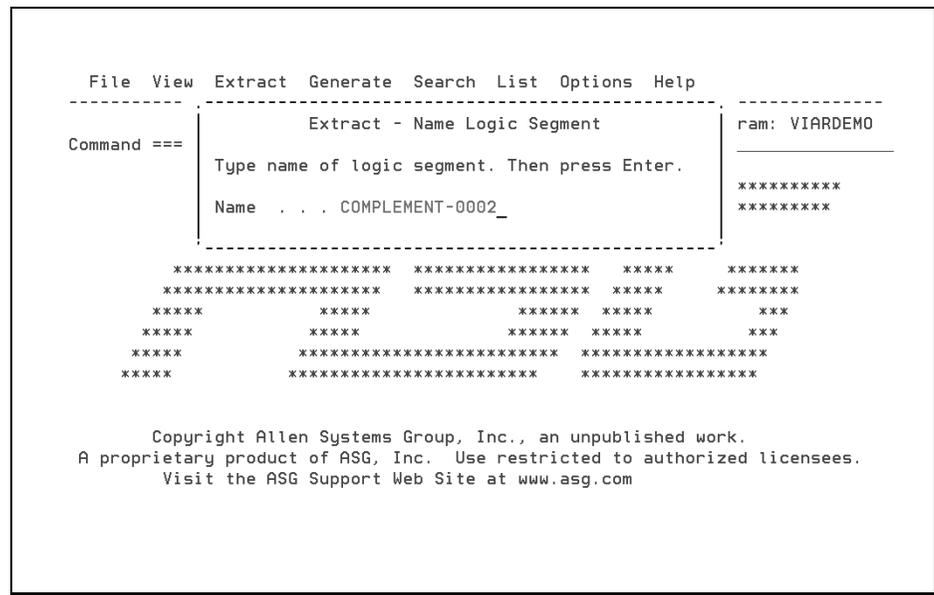
## Creating the Complement Module

After generating the standalone program, you can create the Complement Module to the standalone. The Complement Module is the original program with the statements defined by the extracted logic Segment removed. The program does the month-end processing but does not generate the Exception Report. The Complement Module Generation facility is used to create the complement of the original program.

### To create the Complement Module

- 1 Select Extract ► Complement and press Enter to display the Extract - Name Logic Segment pop-up, as shown in [Figure 178](#).

Figure 178 • Extract - Name Logic Segment Pop-up



- 2 Accept the default name and press Enter to display the Extract - Complement pop-up (see [Figure 179 on page 195](#)).

- 3 Type S next to Logic Segment name REPORT-0001, as shown in [Figure 179](#), and press Enter.

This pop-up is used to select the Logic segments to be excluded from the original program. Since you have extracted the logic for creating the exception report as a standalone program, the Complement Module would not create the Exception report. For this reason, you need to exclude Logic Segment REPORT-001.

**Figure 179 • Extract - Complement Pop-up**

```

-----
Command ==> _                               Extract - Complement          3 LOGIC SEGMENTS
                                           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 GET-NUH-OF-DAYS              CALCULATE NUMBER OF DAYS
_ VIARDEMO INTEREST-CALCULATION    INTEREST-CALCULATION LOGIC SEG
S VIARDEMO REPORT-0001             Report
***** BDTTDM OF DATA *****

```

- 4 Press Enter again to return to the Encore Primary screen.

- 5 Select Generate ► COBOL Module and press Enter to display the Generate - Select Logic Segment pop-up, as shown in [Figure 180](#).

**Figure 180 • Generate - Select Logic Segment Pop-up**

```

-----
Generate - Select Logic Segment      4 LOGIC SEGMENTS
Command ==> _____ Scroll ==> CSR

S - Select

Program      Logic Segment Name      Description
-----
S VIARDEMD  COMPLEMENT-0002              Complement Report
VIARDEMD    GET-NUM-OF-DAYS          CALCULATE NUMBER OF DAYS
_ VIARDEMD  INTEREST-CALCULATION      INTEREST-CALCULATION LOGIC SEG
_ VIARDEMD  REPORT-0001               Report
***** BOTTOM OF DATA *****

```

- 6 Type S next to COMPLEMENT-0002 and press Enter to select it and display the Generate - COBOL Module pop-up, as shown in [Figure 181](#).

**Figure 181 • 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 . . : Complement
Program ID . . : CALCPGM_

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

```

- 7 Type CALCPGM in the Program ID name field and press Enter to display the Edit screen, as shown in, as shown in [Figure 182](#).
- 8 Type CREATE on the command line and type C 9999 on the first line of code. Press Enter to display the Edit/View - Create pop-up to save the member (see [Figure 183](#)).

Figure 182 • Edit Screen

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==> CREATE                               Scroll ==> CSR
***** ***** Top of Data *****
C 9999 *****
000002      *                               ASG, INC.                               *
000003      *                               *                               *
000004      * GENERATED BY ASG-ENCORE      RELEASE 6.0(000)                    *
000005      *                               *                               *
000006      * PROGRAM:          CALCPGM      COMPLEMENT                          *
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:    COMPLEMENT-0002                                  *
000016      *                               *                               *
000017      * OBJECTIVE TYPE:  COMPLEMENT (REPORT)                              *
00001 EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEN
00001 e ASG1608I ENTER THE CREATE COMMAND TO SAVE THE COBOL PROGRAM. e *****
00002 DEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEM
000021      *

```

- 9 Type the name you want to save the program under (CALCPGM in this example), verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter.

Figure 183 • Edit/View - Create Pop-Up - Save the Complement Module

```

                                Edit/View - Create
Command ==>

"Current" Data Set: USERID.ENCXXXXX.COBOL

To ISPF Library:
Project . . . ASG
Group . . . CE60XXXX
Type . . . CNTL
Member . . . CALCPGM

To Other Partitioned Data Set Member:
Data Set Name . . .
Volume Serial . . . (If not cataloged)

Data Set Password . . . (If password protected)

Enter "/" to select option
Specify pack option for "CREATE" Data Set

Press ENTER key to create. Enter END command to cancel create.

```

Program CALCPGM is created. This program contains all of the logic of the original program, except for the extracted Logic Segment, and is stored in your COBOL source library.

## Complement Module - Understanding Generated Notes

When Encore creates the Complement Module, remarks are generated in the module concerning any statements modified by Encore.

Note 1, as shown in [Figure 184](#), is generated whenever a Complement Module is created in anything other than a Perform Range extract. A Complement Module may include a CALL to a submodule. Since the CALL is not automatically generated, as in a Perform Range extract, you must place any required CALLs to submodules wherever appropriate. In this case, no CALLable submodule was created, so no action is required for this message.

Figure 184 • Complement Module Descriptive Notes

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                           Scroll ==> CSR
000020      * ASG-ENCORE NOTES:
000021      *
000022      * 1. THIS MODULE IS THE FUNCTIONAL COMPLEMENT OF THE MODULE
000023      *   DEFINED BY THE LOGIC SEGMENT SELECTION CRITERIA. YOU MUST
000024      *   DETERMINE THE APPROPRIATE POINT(S) TO PLACE A CALL TO THE
000025      *   LOGIC SEGMENT MODULE.
000026      *
000027      *
000028      * IDENTIFICATION DIVISION.
000029      * PROGRAM-ID. CALCPGM.
000030      * AUTHOR. VIASOFT DEMONSTRATION.
000031      *
000032      * *****
000033      * THIS PROGRAM DOES THE MONTH END CALCULATIONS FOR EACH TYPE *
000034      * OF CHECKING ACCOUNT CARRIED BY THE UNIVERSAL BANK. IT *
000035      * HANDLES THE FOLLOWING TYPES OF ACCOUNTS. *
000036      * CHECKING WITH INTEREST *
000037      * CHECKING WITH NO INTEREST *
000038      * MONEY MARKET ACCOUNT (MMA) *
000039      * SAVINGS ACCOUNT *
000040      * IT ALSO UPDATES THE ACCOUNT BALANCE, BRANCH BALANCE AND *
000041      * BANK BALANCE. *
    
```

## Using Pseudo Source Modules to Change Logic Segments

### Pseudo Source Modules

If the program being analyzed contains a CALL to a non-COBOL program, or to a program that has not been analyzed, you should consider creating pseudo source to obtain better results in an extracted program under these conditions:

- If all parameters are not used and/or modified, pseudo source should be written.
- If the CALLED program is not COBOL.
- You choose not to analyze the CALLED program.

The purpose of the pseudo source module is to identify the inputs and outputs of the CALLED routine. The pseudo source module must be saved in the AKR.

Each CALL statement parameter is input-only, output-only, or both input and output.

- Input-only parameters are used but not modified by the CALLED module.
- Output-only parameters are modified but not used by the CALLED module.
- Input and output parameters are both used and modified by the CALLED module.

For example, [Figure 185](#) shows a program being re-engineered that uses this code:

**Figure 185 • Using Pseudo Source Modules - Program Re-engineering Code Example**

```
MOVE 0 TO PAYMENT.  
MOVE LOAN-AMOUNT TO PARM-1.  
CALL 'INTEREST' USING PARM-1, PAYMENT.  
DISPLAY PAYMENT
```

If a Report extract request is made for the variable PAYMENT at the DISPLAY statement, it may be assumed that the preceding CALL statement and the MOVE LOAN-AMOUNT statement are both required. This assumption can be made because it appears that the CALL to the 'INTEREST' module is modifying the PAYMENT parameter. If this assumption is true, the CALL is required; however, if this assumption is false, the CALL statement is unnecessary (although the MOVE 0 TO PAYMENT statement would be required).

The report and Computation Variable extract objectives assume that all parameters are both input and output, which may not always be an accurate assumption. In some cases, the extract contains more than the minimum set of code because of this assumption. To obtain completely accurate results, the actual type of each parameter must be made known to the extract objective process. This can be accomplished in either of the these two ways:

- Analyze the CALLED program and save it in the AKR containing the CALLING program. In some cases, this may not be feasible, e.g., the program source is not accessible, the program may be so large that sufficient resources are unavailable, or the program is non-COBOL.
- Write a pseudo source module, identifying the type of each parameter. Analyze this pseudo source module instead of the actual CALLED program.

See ["The Analyze Facility" on page 36](#) for more information about using Encore to analyze a program.

Precisely defining the parameter types in a pseudo source module results in more accurate extract results for the report and Computation Variable extracts, because unnecessary code is not extracted.

These are the rules for writing pseudo source modules:

- For input parameters, use a *MOVE data-item TO ws-data-item* statement.

where:

*ws-data-item* represents a WORKING-STORAGE data item that is the same data type as *data-item*.

- For output parameters, use a *MOVE SPACES TO data-item* statement. A *MOVE ZEROES* statement may be used if appropriate.
- For output parameters that are only possibly modified (as opposed to definitely modified) by the called routine, code the *MOVE SPACES TO data-item* statement in an IF statement that tests a WORKING-STORAGE data item.
- Code all inputs before outputs. If a data item is both an input and an output, code it first as an input and then as an output.
- The LINKAGE SECTION declarations should match the declarations in the CALLING program.

[Figure 186](#) and [Figure 187 on page 202](#) are examples of a program that CALLs a non-COBOL subroutine that calculates an interest amount, and the pseudo source module that may be written for that subroutine.

**Figure 186 • Program Being Re-engineered**

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. VIARDEMO.  
. . .  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77      WORK-LOAN      PIC 9999.  
77      WORK-RATE      PIC 9999.  
77      WORK-AMOUNT    PIC 9999.  
. . .  
MOVE 10000 TO WORK-LOAN.  
MOVE 9.5 TO WORK-RATE.  
MOVE 10 TO WORK-AMOUNT.  
CALL 'SUBRTN' USING WORK-LOAN.  
                WORK-RATE, WORK-AMOUNT.  
DISPLAY WORK-LOAN WORK-AMOUNT.
```

In [Figure 187](#), the pseudo source subroutine SUBRTN contains these parameters: LOAN-AMT, INTEREST-RATE, and INTEREST-AMT. The first MOVE statement of the pseudo source module specifies a use of LOAN-AMT, indicating that LOAN-AMT is an input parameter. The second MOVE statement indicates that INTEREST-RATE is also an input parameter. The third MOVE statement specifies a modification to INTEREST-AMT, indicating that it is an output parameter.

**Figure 187 • Pseudo Source Module**

```

IDENTIFICATION DIVISION.
PROGRAM-ID. SUBRTN.
. . .
DATA DIVISION.
WORKING-STORAGE SECTION.
77     WS-LOAN-AMT           PIC 9999.
77     WS-INTEREST-RATE     PIC 9999.
LINKAGE SECTION.
77     LOAN-AMOUNT          PIC 9999.
77     INTEREST-RATE        PIC 9999.
77     INTEREST-AMT         PIC 9999.
PROCEDURE DIVISION USING LOAN-AMT.
        INTEREST-RATE, INTEREST-AMT.

*
*INPUTS
*
        MOVE LOAN-AMT TO WS-LOAN-AMT.
        MOVE INTEREST-RATE TO WS-INTEREST-RATE.

*
*OUTPUTS
*
        MOVE ZEROES TO INTEREST-AMT.
        GOBACK.

```

Because both the report and Computation Variable extract objectives assume that all parameters are both input and output, these scenarios explain the example (see the code of the program in [Figure 186 on page 201](#) and [Figure 187](#)).

### Scenario 1

A Computation Variable extract is performed for the variable WORK-AMOUNT at the DISPLAY statement.

- Without a pseudo source module, the MOVE 10 TO WORK-AMOUNT statement would be extracted because it is assumed that WORK-AMOUNT is both input and output.
- With a pseudo source module, the MOVE statement would not be extracted.

## Scenario 2

A Computation Variable extract is performed for the variable WORK-LOAN at the DISPLAY statement.

- Without a pseudo source module, the CALL statement would be extracted.
- With a pseudo source module, the CALL statement would not be extracted.

## Including Non-selected Code in the Logic Segment

Under certain conditions, Encore produces a result that does not include all required code.

**Figure 188 • Logic Segment - Non-selected Code Example**

```

000242
000243 ACCT-MAINTENANCE.
000244
000245     READ TRAN-FILE
000246         AT END
000247             MOVE EOF-LITERAL TO TRAN-FILE-FLAG
000248             GO TO ACCT-MAINTENANCE-EXIT.
000249
000250     ADD +1 TO TRAN-CNT.
000251
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.
000253
000254     READ ACCT-FILE  KEY IS ACCT-KEY

```

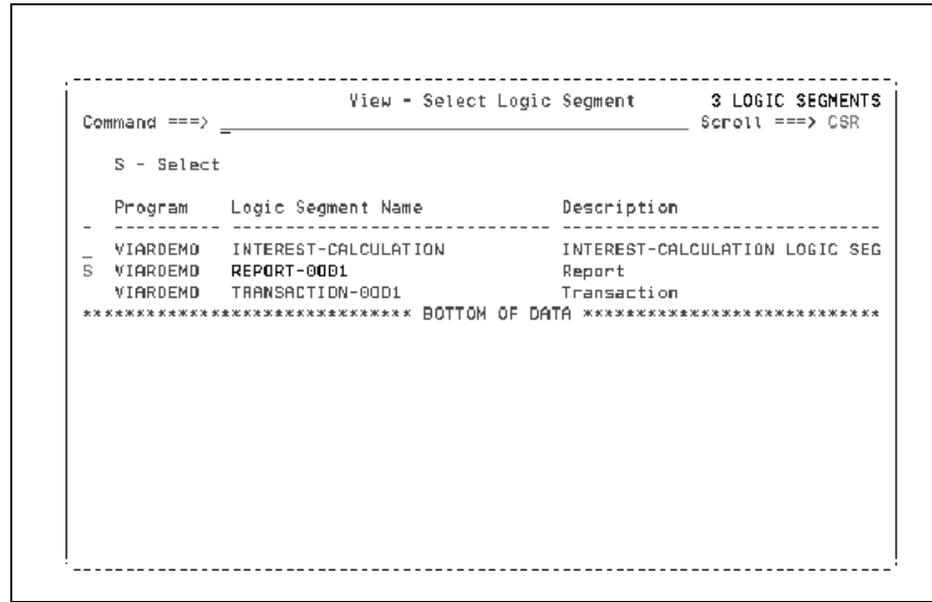
In [Figure 188](#), assuming that only the READ statement was determined to be part of the result, Encore would not have extracted line 250 relating to TRAN-CNT.

For Reports extracts, Encore allows for the inclusion of omitted lines of code from the resulting Logic Segment.

**To include lines in a Logic Segment**

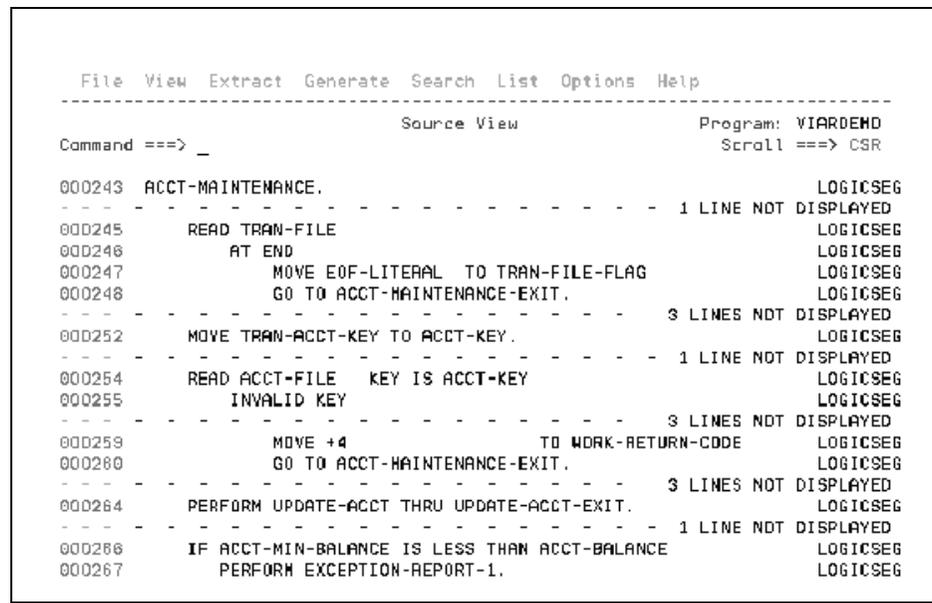
- 1 Select View ► Logic segment and press Enter to display the View - Select Logic Segment pop-up, as shown in [Figure 189](#).

**Figure 189 • View - Select Logic Segments Pop-up**



- 2 Type S to the left of Logic Segment REPORT-0001 and press Enter to display the Source View screen, as shown in [Figure 190](#).

**Figure 190 • Source View Screen - Non-Expanded**



- 3 Position line 243 to the top of the screen.
- 4 Place the cursor immediately below line 248 and press PF4 (Zoom In) to expand the view to include non-selected lines, as shown in [Figure 191](#). Notice that the expanded lines (249 - 251) are not part of the Logic Segment. They are not highlighted, nor are they tagged with LOGICSEG.

**Figure 191 • .Source View Screen - Expanded**

```

File View Extract Generate Search List Options Help
-----
Source View                                13 DETAIL LINES
Command ==>                               Scroll ==> CSR

000243 ACCT-MAINTENANCE.                    LOGICSEG
000244                                     LOGICSEG
000245     READ TRAN-FILE                    LOGICSEG
000246     AT END                          LOGICSEG
000247     MOVE EOF-LITERAL TO TRAN-FILE-FLAG LOGICSEG
000248     GO TO ACCT-MAINTENANCE-EXIT.     LOGICSEG
000249                                     LOGICSEG
000250     ADD +1 TO TRAN-CNT.                LOGICSEG
000251                                     LOGICSEG
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.    LOGICSEG
000253                                     LOGICSEG
000254     READ ACCT-FILE KEY IS ACCT-KEY     LOGICSEG
000255     INVALID KEY                       LOGICSEG
000256     MOVE ACTION-MISSING-ACCT TO WORK-ACTION-REASON
000257     MOVE SPACES TO ACCT-RECORD
000258     PERFORM INITIATE-ACTION
000259     MOVE +4 TO WORK-RETURN-CODE        LOGICSEG
000260     GO TO ACCT-MAINTENANCE-EXIT.     LOGICSEG
000261

```

- 5 Type I on line 250, as shown in [Figure 192](#), and press Enter to refresh the Source View screen (see [Figure 193 on page 206](#)).

**Figure 192 • Source View Screen - Including Source Line in Logic Segment**

```

File View Extract Generate Search List Options Help
-----
Source View                                3 DETAIL LINES
Command ==>                               Scroll ==> CSR

000243 ACCT-MAINTENANCE.                    LOGICSEG
000244                                     LOGICSEG
000245     READ TRAN-FILE                    LOGICSEG
000246     AT END                          LOGICSEG
000247     MOVE EOF-LITERAL TO TRAN-FILE-FLAG LOGICSEG
000248     GO TO ACCT-MAINTENANCE-EXIT.     LOGICSEG
000249                                     LOGICSEG
I000250     ADD +1 TO TRAN-CNT.                LOGICSEG
000251                                     LOGICSEG
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.    LOGICSEG
- - - - - 1 LINE NOT DISPLAYED
000254     READ ACCT-FILE KEY IS ACCT-KEY     LOGICSEG
000255     INVALID KEY                       LOGICSEG
- - - - - 3 LINES NOT DISPLAYED
000259     MOVE +4 TO WORK-RETURN-CODE        LOGICSEG
000260     GO TO ACCT-MAINTENANCE-EXIT.     LOGICSEG
- - - - - 3 LINES NOT DISPLAYED
000264     PERFORM UPDATE-ACCT THRU UPDATE-ACCT-EXIT. LOGICSEG
- - - - - 1 LINE NOT DISPLAYED

```

The Source code at line 250 is highlighted and has been included in the Logic Segment, as shown in [Figure 193](#).

**Figure 193 • Source View Screen - Source Line Included in Logic Segment**

```

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

000243 ACCT-MAINTENANCE.                                LOGICSEG
000244
000245     READ TRAN-FILE                                LOGICSEG
000246     AT END                                        LOGICSEG
000247     MOVE EOF-LITERAL TO TRAN-FILE-FLAG           LOGICSEG
000248     GO TO ACCT-MAINTENANCE-EXIT.                 LOGICSEG
000249
000250     ADD +1 TO TRAN-CNT.
000251
000252     MOVE TRAN-ACCT-KEY TO ACCT-KEY.                LOGICSEG
000253
000254     READ ACCT-FILE  KEY IS ACCT-KEY                LOGICSEG
000255     INVALID KEY                                     LOGICSEG
000256     MOVE ACTION-MISSING-ACCT TO WORK-ACTION-REASON
000257     MOVE SPACES TO ACCT-RECORD
000258     PERFRM INITIATE-ACTION
000259     MOVE +4 TO WORK-RETURN-CODE                    LOGICSEG
000260     GO TO ACCT-MAINTENANCE-EXIT.                 LOGICSEG
000261

```

Because the Logic Segment has been changed, you must save the file before returning to the Encore Primary screen.

- 6 Select File ► Save Segment and press Enter to display the File - Save Logic Segments pop-up.
- 7 Type S next to the Logic Segment to be saved and press Enter to display the File - Save Segment pop-up. Type the appropriate descriptive information and press Enter to return to the File - Save Logic Segments pop-up. Press PF3 until you return to the Encore Primary screen.

**Note:**

In addition to the identified statement, Encore ensures that all the code necessary to execute the statement is also included in the generated module.

## Controlling Extract Boundaries

There are instances when each extract objective gives you more code than you want. For example, when the output statements for a particular file are imbedded within a Perform Range. You want to extract only those output statements imbedded in the Perform Range and generate a CALLable module.

A Perform Range extracts the desired statements. However, it also extracts all statements that can be executed by the Perform statement, including other Perform Ranges and GO TOs. A Transaction extract with altered START and END points does not sufficiently narrow the scope of the extract unless the statements are adjacent.

A Report extract isolates the output statements mentioned in the first example. It also extracts all contributing or influencing statements from the selected statements to the beginning of the program, giving you more code than you want. Also, the Report extract does not generate CALLable modules.

The Report extract objectives allow you to extract code from an existing Perform Range or Transaction Logic Segment. Normally, within the Report extract objective, code is extracted from the selected statements to the beginning of the program. When you perform a Report extract from a Logic Segment, the extract boundaries are now from the selected statements to the beginning of the Logic Segment.

### Extracting Code from a Logic Segment

When extracting code from an existing Logic Segment, follow the same procedure as you do when you perform a conventional Report extract (see ["Extracting the Report" on page 184](#)) with the exception noted in [step 3](#) below and [step 4 on page 208](#).

#### *To extract code from a Logic Segment*

- 1 Select Extract ▶ Report and press Enter to display the Extract - Name Logic Segment pop-up.
- 2 Accept the default name and press Enter to display the Extract - Report Name List pop-up.
- 3 Type `Filter` on the command line and press Enter or press PF5 to display the Extract - Report Filter pop-up (see [Figure 194 on page 208](#)). When the Extract - Report Filter pop-up displays, enter an asterisk (\*) in the Report Name field, which allows you to perform the extract against an existing Logic Segment.

- 4 Select the Limit to within an existing segment option by typing a forward slash (/) next to the option, as shown in [Figure 194](#). In addition to typing data in the various fields as you would for a conventional extract, the option Limit to within an existing Segment is used to indicate that the extract is to be done within a Logic Segment.

**Figure 194 • Extract - Report Filter Pop-up**

```

C      Extract - Report Name List
S      Extract - Report Filter
c      Type information for filtering Report Name list.
      Report Name   x_____
      Data Name    . . _____
      COPYBOOK Name
      Line Range   . .
      Options
      / Limit to within an existing Segment
      ***** BOTTOM OF DATA *****
      PF4=View PF5=Filter PF17=Sort
```

- 5 Press Enter to display the Select Extract-within-Logic Segment pop-up, as shown in [Figure 195](#).

This pop-up is used to select the Logic Segment for code extraction. Only Perform Range or Transaction Logic Segments are displayed in the list of Logic Segments. Only one Logic Segment can be selected.

**Figure 195 • 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 *****
-----

```

- 6 After selecting the Logic Segment, press Enter to highlight the selection. Press Enter until you to return to the Extract - Report Name List pop-up.

From this point on, the extract procedures are the same as for a conventional Report extract (see ["Extracting the Report" on page 184](#)). Only those statements that lie within the selected Logic Segment are eligible for extraction.

**Note:**

Because the Logic Segment contains IO statements, you may want to consider selecting the CALLs to IO Module(s) option when you generate the program to avoid the COBOL constraint of referencing files across program boundaries.

## Report Extract Compilation Issues

When Encore generates standalone replacement and Complement Modules from a Report Range extract, certain conditions may occur that could cause compile errors or the program not to run at all. As shown in this section, Encore inserts comment messages in the generated modules when these conditions occur.

### Condition 1

Encore has generated statements that cause normal exits from the program.

Cause	Action
Encore generates program exit statements (GOBACK, EXEC CICS RETURN) to ensure that the program exits properly.	Review the conditions surrounding each generated exit statement to determine if the default exit statement should be augmented or replaced.

### Condition 2

Encore has determined that a COPY statement cannot be used. Portions of the copybook are reproduced inline.

Cause	Action
Under certain conditions, the original COPY statement cannot be used in a generated module, such as when COPY statements contains dead code.  In addition to this, Encore generates comments indicating why the COPY statement could not be used as originally coded.	Determine why the COPY member was not usable. No action is required, however, the COPY member contents may be modified to use it in this context.  Example: A copybook is used in multiple programs. When it is copied into one program, it contains statements that are not referenced by this particular program. Those statements are considered dead code.

**Condition 3**

Encore has generated a DISPLAY statement prior to each abnormal program exit.

Cause	Action
Encore has generated DISPLAY statements just prior to a generated abnormal program exit.	Review the conditions surrounding each generated DISPLAY statement to determine if it should be augmented or replaced. Ensure that a DISPLAY is an appropriate action for the condition that occurred.

**Note:**

A DISPLAY is not generated for a CICS program, or when a subroutine is specified on the COBOL Module Exit field on the Options - Product Parameters pop-up (see [Figure 7 on page 27](#)).

**Condition 4**

Encore has tailored OPEN or CLOSE statements.

Cause	Action
For Computation Variable and report objectives, Encore ensures that all necessary file-related statements are extracted, including OPENs and CLOSEs. When an OPEN or CLOSE statement includes references to files that are not required by the Logic Segment, Encore tailors the statement to reference only the desired files.	None.

**Condition 5**

STOP RUN in empty Exception clauses (i.e., ON SIZE, ON OVERFLOW).

Cause	Action
COBOL does not permit a NEXT SENTENCE to appear in an exception clause. Encore inserts a STOP RUN in an exception clause that is part of a Report Variable extract.	Revise to include a more meaningful error handling routine.



---

# 8

## CICS Server Extract

---

This chapter contains a practical demonstration of how to perform a CICS server extract to create a COMMAREA-based server program from a 3270 CICS pseudo-conversational program, and contains these sections:

<b>Section</b>	<b>Page</b>
<a href="#">Introduction</a>	<a href="#">213</a>
<a href="#">The Business Scenario</a>	<a href="#">214</a>
<a href="#">Conversion Process</a>	<a href="#">215</a>
<a href="#">Starting an Encore Session</a>	<a href="#">216</a>
<a href="#">Extracting the CICS Server Program</a>	<a href="#">217</a>
<a href="#">Generating the CICS Server Module</a>	<a href="#">223</a>
<a href="#">Understanding the Results of the CICS Server Module Generation</a>	<a href="#">234</a>
<a href="#">CICS Server Extract Compilation Issues</a>	<a href="#">242</a>
<a href="#">Extracting the Self Directed Server Program</a>	<a href="#">245</a>
<a href="#">Using IBM Solutions</a>	<a href="#">254</a>

### Introduction

In addition to demonstrating the technique of using a CICS server extract to create a COMMAREA-based server program from a 3270 CICS pseudo-conversational program, this chapter also discusses the various ways to use a server program to accomplish the goal discussed in [The Business Scenario](#), including execution and compilation issues.

**Note:** \_\_\_\_\_

The Encore demonstration program VIARBRWS is used for the examples in this chapter.

---

## The Business Scenario

You have a CICS application that tracks the 401(K) contributions made by your employees. The application allows new contributions to be made, past contributions to be updated to correct data entry errors, and all past contributions to be viewed. Application users have requested that the view function be made available through a web browser. You have tried web access to CICS through 3270 Bridge, but you know that this is a short-term solution because Bridge depends on the screen layout, which may change from time to time. You would also like the solution to be adaptable to new ways of interfacing to the external world, e.g., through a Java servlet that performs additional processing before generating an HTML page.

The implementation of web access for existing CICS applications should be viewed as a re-engineering process. For applications where presentation logic and business logic are intermixed, migration to another presentation medium using the existing presentation logic may be inefficient or impossible depending on your current environment. Encore allows you to separate presentation logic and business logic in both conversational and pseudo-conversational CICS programs, resulting in a CICS server program containing business logic. Once the CICS server program is isolated from the presentation logic, it becomes a business logic component that can be invoked from a web server, as well as from other client types, (e.g., Java servlets, MQSeries, and CICS clients on other platforms).

This chapter demonstrates how this business scenario can be handled by Encore through its server extract and generation facilities.

**Note:** \_\_\_\_\_

This process also includes steps that are performed outside of Encore.  
\_\_\_\_\_

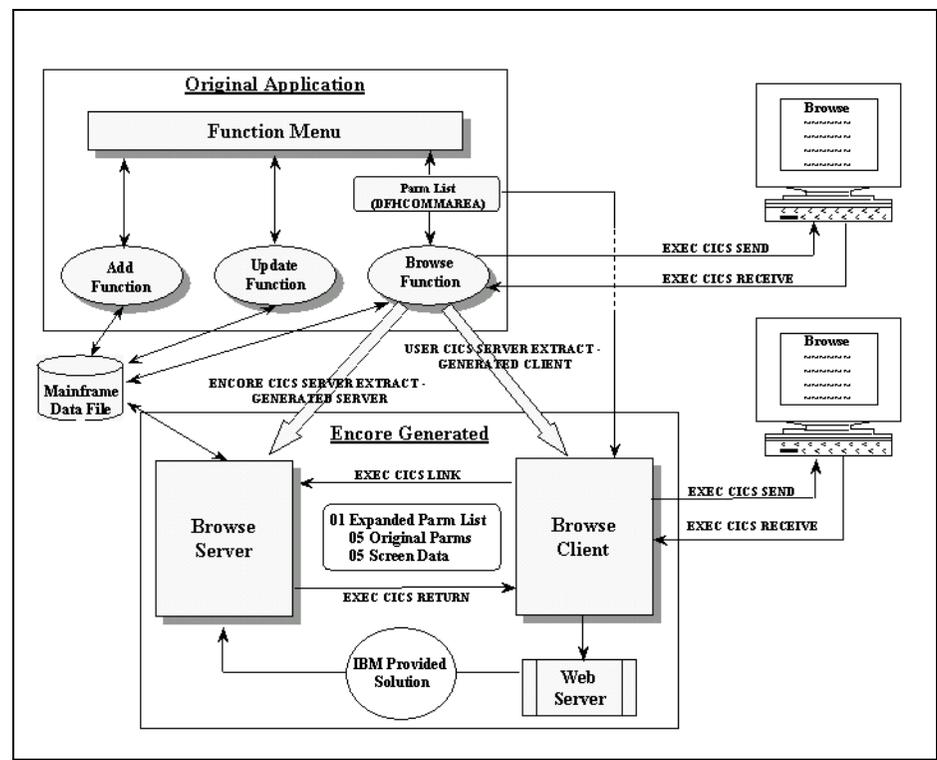
## Conversion Process

Figure 196 shows the process of converting the browse function in the application to support web access.

### To convert the application browse function

- 1 Analyze the browse program for Encore. See ["The Analyze Facility" on page 36](#) for more information about using Encore to analyze a program.
- 2 Extract the logic responsible for record retrieval. The resulting program is known as the Next-Page server module. For this demonstration, extract the logic responsible for retrieving records in the backward direction, with the assumption that employees would want to view the most recent records first.
- 3 Generate the server program containing the extracted logic.

Figure 196 • CICS Server Extract/Generation Business Scenario Example



## Starting an Encore Session

### *To start the Encore session*

**Note:** \_\_\_\_\_

Logon procedures and AKR information are unique to your programming environment. If necessary, contact your systems administrator or your Encore coordinator for the correct dataset names.

---

- 1 Logon to the ESW Primary screen.
- 2 Select Re-engineer ▶ Program and press Enter to display the Encore Primary screen.

**Or**

Type EN on the command line and press Enter to display the Encore Primary screen.

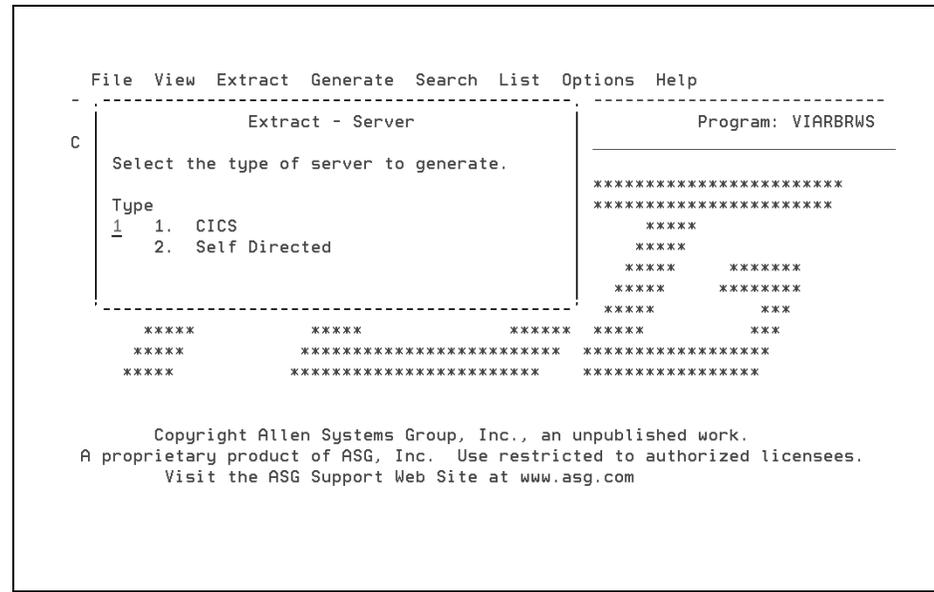
- 3 Select File ▶ Open and press Enter to display the File - Open Program pop-up.
- 4 Type the AKR dataset name in the Data Set Name field and VIARBRWS in the Program Name field and press Enter.

See ["Starting an Encore Session" on page 59](#) for more information.



- 3 Type 1 to select CICS, as shown in [Figure 198](#), and press Enter to display the Extract - CICS server Endpoints pop-up (see [Figure 199 on page 219](#)).

**Figure 198 • Extract -Server Pop-up**



**Note:**

If you select the Self Directed server name you can reduce the list of selectable names by pressing PF5. The Filter - Self Directed Server pop-up displays, which allows you to specify filtering criteria. See ["Extracting the Self Directed Server Program" on page 245](#) and the online help for more information about the Filter - Self Directed Server pop-up.

- 4 To ensure that you are selecting the code you want to extract, type V next to the CICS EXEC you want to view, as shown in [Figure 199](#). Press Enter to display the View Source pop-up (see [Figure 200 on page 220](#)).

This pop-up contains all of the statements associated with the CICS member. You need to extract all statements that contribute to the output screen.

**Figure 199 • Extract - CICS Server Endpoints Pop-up - View Option**

```

-----
Command ==>                                Extract - CICS Server Endpoints                                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.
V 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('VIARHNU') ERASE END-EXEC.
***** BDTDM OF DATA *****
-----

```

- Verify that the CICS EXEC server code selected represents the type of code you want to extract, as shown in [Figure 200](#). After verifying the code, press PF3 to return to the Extract - CICS server Endpoints pop-up (see [Figure 201](#)). If necessary, repeat this process until you find the code you want to extract.

**Figure 200 • View - Source Pop-up - CICS Extract**

```

View Search List Options Help
-----
View - Source                               Prd  2 LINES FOUND
Command ==> _                               Scroll ==> CSR

000609      MOVE NUMB TO NUMBER10.
000610      MOVE NAME TO NAME10.
000611      MOVE AMOUNT TO AMOUNT10.
000612 *
000613 *      DISPLAY MAP
000614 *
000615      EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRW')          LINE RNG
000616      ERASE END-EXEC.                                          LINE RNG
000617 *
000618 *      RETURN WITH A COMM. AREA.
000619 *
000620      EXEC CICS RETURN TRANSID(EIBTRNID) COMHAREA(COHMAREA)
000621      LENGTH(19) END-EXEC.                                     PGM EXIT
000622 *
000623 *      AFTER THE "RECEIVE" COMMAND EXECUTES, THE PROGRAM TESTS THE
000624 *      OPERATORS RESPONSE. ONLY "CLEAR", "PF1", "PF2", "F" OR "B"
000625 *      KEYS ARE TAKEN AS VALID INPUT.
    
```

- Type S next to the line you want to select and press Enter to display the Extract - CICS Server Startpoints pop-up (see [Figure 202 on page 221](#)).

**Figure 201 • Extract - CICS Server Endpoints Pop-up - Select Option**

```

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.
S 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 *****
    
```

- 7 Type S in the indicated command area or press Enter to accept the selection (all highlighted text is preselected, see [Figure 202](#)). Press Enter to return the Extract - CICS Server Endpoints pop-up, (see [Figure 203](#)).

**Figure 202 • 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 *****
-----

```

- 8 The message displayed in the long message area confirms that the CICS statements for the EXEC CICS SEND code were successfully selected. Press Enter to display the Extract - CICS Server Paths pop-up (see [Figure 204 on page 222](#)).

**Figure 203 • Extract - CICS Server Endpoints Pop-up - Selection Confirmation Message**

```

-----
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.
*****
ASG4102I CICS STATEMENTS SUCCESSFULLY SELECTED.
-----

```



## Generating the CICS Server Module

Now you are ready to create a standalone CICS server module that contains all of the statements required to produce the CICS server.

### *To generate the CICS server module*

- 1 Select Generate ► Server and press Enter to display the Generate - Select Logic Segment pop-up, as shown in [Figure 206](#).

**Figure 206 • 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 *****

```

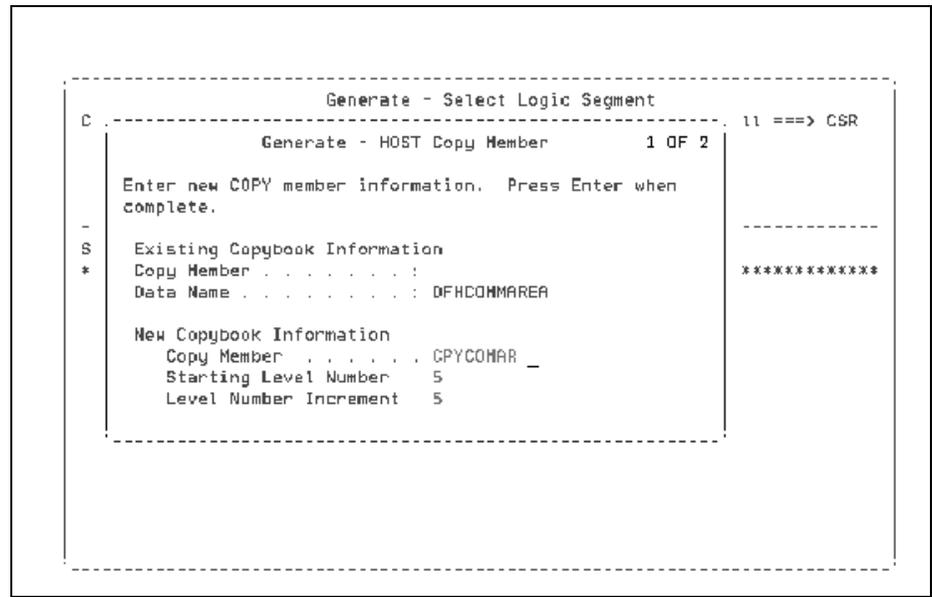
- 2 Type S next to the Logic Segment name you want, or accept the default selection, and press Enter to display the first Generate - HOST Copy Member pop-ups (see [Figure 207 on page 224](#)).

**Note:**

The number of Generate - HOST Copy Member pop-ups displayed during this procedure depends on the ENDING and STARTING statements selected on the corresponding Extract - CICS Server Endpoints (see [Figure 201 on page 220](#)) and Extract - CICS Server Startpoints pop-ups (see [Figure 202 on page 221](#)). A minimum of two Generate - HOST Copy Member pop-ups always display for a generated server.

- 3 Type the new name you want to assign to the DFHCOMMAREA copy member in the Copy Member field (CPYCOMAR in this example, see [Figure 207](#)). Assign the starting level number and level number increment you want or accept the defaults (the default record level is 5). Press Enter to display the second Generate - HOST Copy Member pop-up (see [Figure 208 on page 225](#)).

Figure 207 • Generate - HOST Copy Member Pop-up Number 1



- 4 Accept the new copy member defaults, as shown in [Figure 208](#), or change the name and level numbers, and press Enter to display the Generate - HOST Server Module pop-up (see [Figure 209](#)).

Figure 208 • Generate - HOST Copy Member Pop-up - Number 2

```

Generate - Select Logic Segment
C ----- 11 ==> CSR
      Generate - HOST Copy Member      2 OF 2
Enter new COPY member information. Press Enter when
complete.
-----
S Existing Copybook Information
* Copy Member . . . . . : VIARBRWC      *****
  Data Name . . . . . : BROWSED
New Copybook Information
  Copy Member . . . . . : VIARBRCS
  Starting Level Number : 5
  Level Number Increment : 5

```

- 5 Type 1 to select Generate Options and press Enter to display the Options - Generate pop-up (see [Figure 210 on page 226](#)).

Figure 209 • Generate - HOST Server Module Pop-up

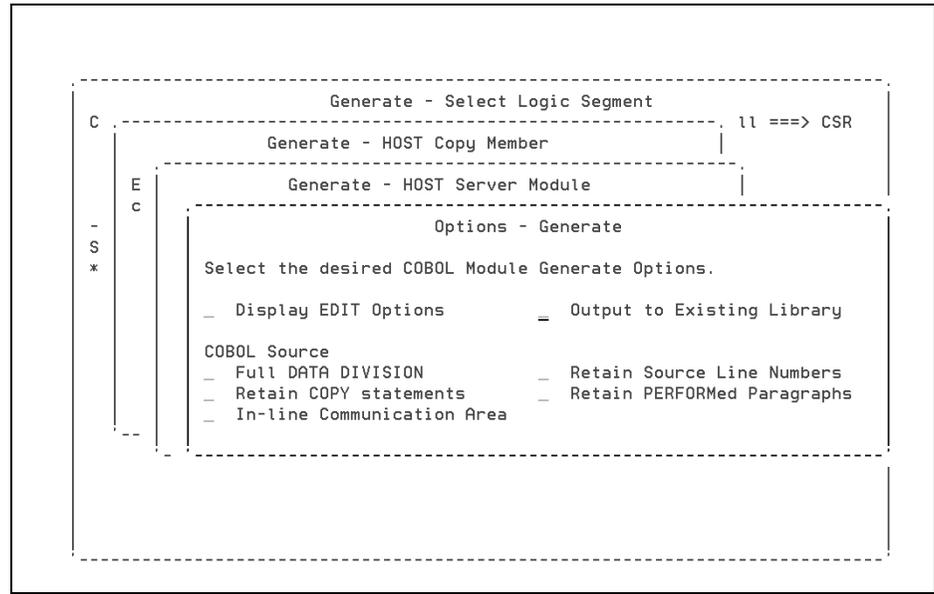
```

Generate - Select Logic Segment
C ----- 11 ==> CSR
      Generate - HOST Copy Member
      Generate - HOST Server Module
Type Program ID and select generation options.
-----
S Objective . . . : Client Server      *****
* Program ID . . . : SERVER
Functions
1= 1. Generate options...
   2. Generate COPY members...
   3. Generate server module
   4. Exit

```

- 6 Indicate the options you want for the COBOL Module generation, as shown in [Figure 210](#), and press Enter. Press PF3 to return to the Generate - HOST Server Member pop-up (see [Figure 211 on page 227](#)).

**Figure 210 • Options - Generate Pop-up**



The COBOL Generation Options pop-up allows you to select these generation options:

Option	Description
Display EDIT Options	Displays the ASG-SmartEdit Edit Options screen.
Output to Existing Library	Allows Encore to insert the generated output directly into the 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.



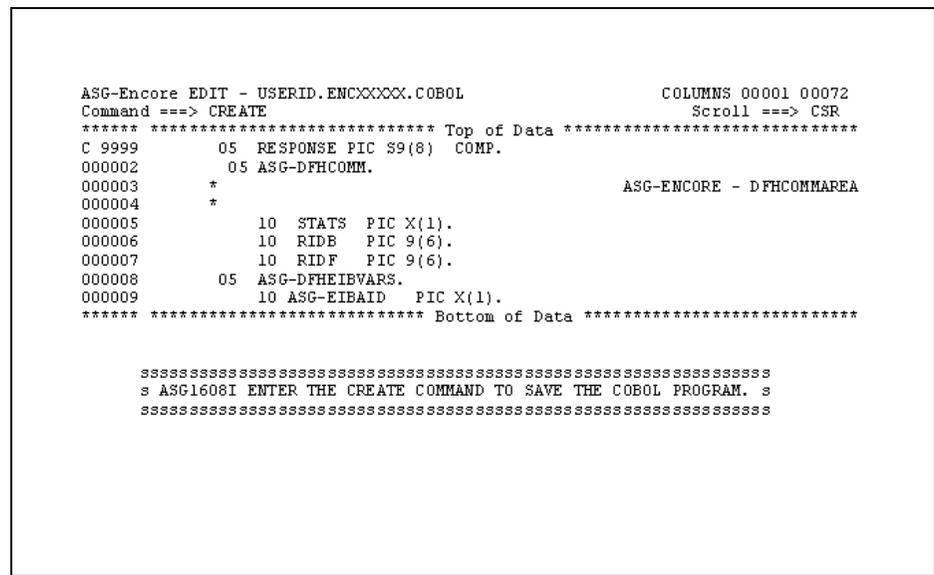
- 8 Type S next to the COPY member you want to select, as shown in [Figure 212](#), and press Enter to display the Edit screen (see [Figure 213](#)).

**Figure 212 • Generate - COPY Members Pop-up**



- 9 Type CREATE on the command line and type C 9999 on the first line of code. Press Enter to display the Edit/View - Create pop-up to save the module in your source library (see [Figure 214 on page 229](#)).

**Figure 213 • Edit Screen - CREATE Command**



- 10 Type the name you want to save the member as in your source library, as shown in [Figure 214](#), verify or clear any data in the Data Set Name and Volume Serial fields, and press Enter to return to the Edit screen (see [Figure 215](#)).

**Figure 214 • Edit/View - Create Pop-up - CPYCOMAR Member**

```

                                Edit/View - Create
Command ==>

"Current" Data Set: USERID.ENC00033.COBOL

To ISPF Library:
Project . . . . ASG
Group . . . . CE60XXXX
Type . . . . CNTL
Member . . . . CPYCOMAR

To Other Partitioned Data Set Member:
Data Set Name . . .
Volume Serial . . .      (If not cataloged)

Data Set Password . .      (If password protected)

Enter "/" to select option
Specify pack option for "CREATE" Data Set

Press ENTER key to create. Enter END command to cancel create.

```

- 11 The Edit screen redisplay with a message confirming that the member has been created. Press PF3 to return to the Generate - Copy Members pop-up (see [Figure 216 on page 230](#)).

**Figure 215 • Edit Screen - CPYCOMAR Member Created**

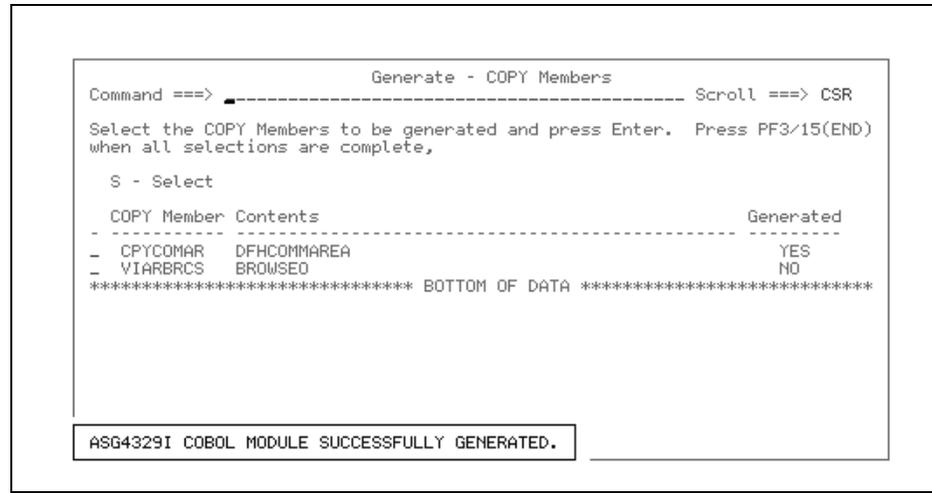
```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                Member CPYCOMAR created
Command ==>                                           Scroll ==> CSR
***** ***** Top of Data *****
000001          05 RESPONSE PIC S9(8)  COMP.
000002          05 ASG-DFHCOMM.
000003          *
000004          *
000005          10 STATS PIC X(1).
000006          10 RIDE PIC 9(6).
000007          10 RIDF PIC 9(6).
000008          05 ASG-DFHEIBVARS.
000009          10 ASG-EIBAID PIC X(1).
***** ***** Bottom of Data *****
ASG-ENCORE - DFHCOMMAREA

```

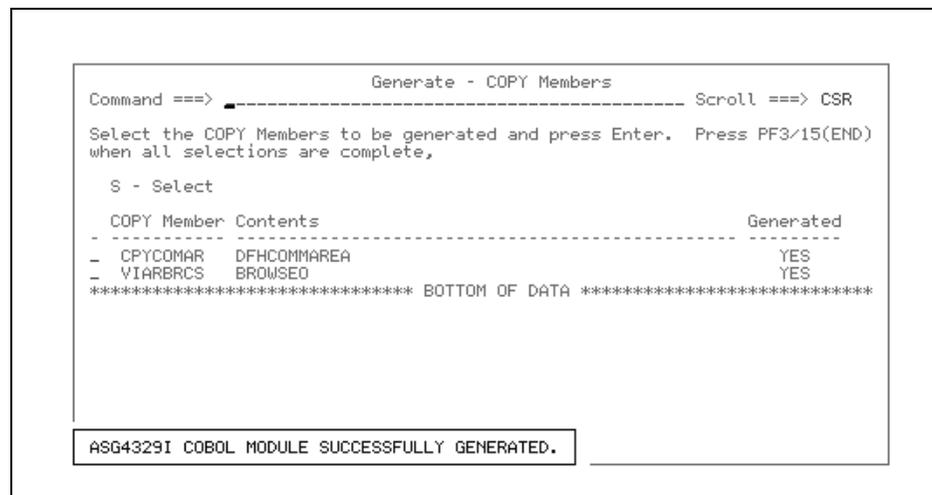
- 12 A message displays in the long message area indicating that the COBOL module has been successfully generated, as shown in [Figure 216](#). The status also changes from NO to YES under the Generated column. Select the next Copy member and repeat [step 8 on page 228](#) through [step 11 on page 229](#) to edit and save the member. Once saved, press Enter and then PF3 to return to the Generate - COPY Member pop-up (see [Figure 217 on page 230](#)).

**Figure 216 • Generate - COPY Members Pop-up - Select COPY Member**



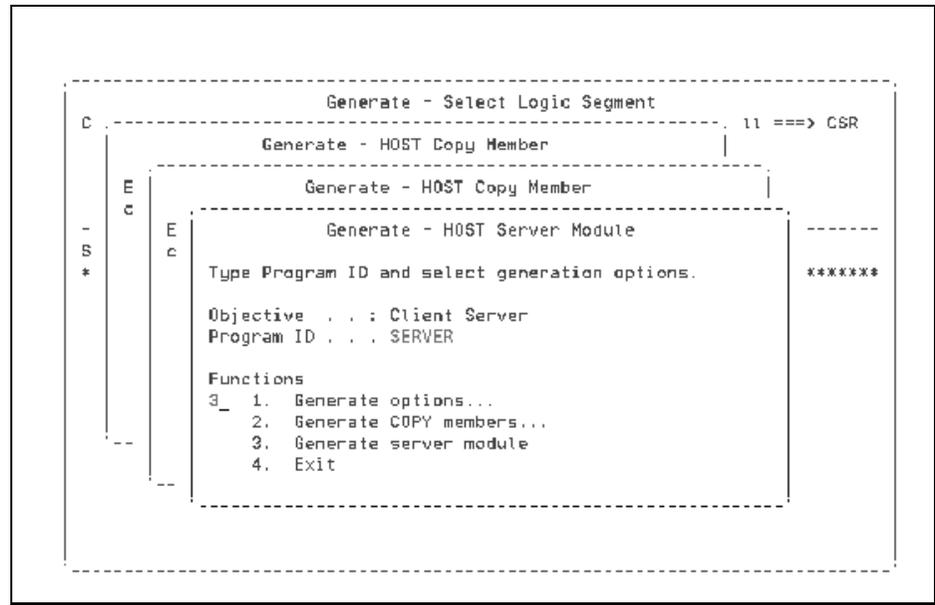
- 13 After the member has been saved, a message displays in the long message area indicating that the COBOL module has been successfully generated, as shown in [Figure 217](#). The status also changes from NO to YES under the Generated column. Press PF3 to return to the Generate - HOST Server Module pop-up (see [Figure 218](#)).

**Figure 217 • Generate - COPY Members Pop-up - Confirmation Message**



- 14 Type 3 to select Generate server module and press Enter to display the Edit screen (see [Figure 219 on page 232](#)).

Figure 218 • Generate - HOST Server Module Pop-up





- 17 The Edit screen redisplay with a message confirming that the member has been created, as shown in [Figure 221](#). Press PF3 to return to the Generate - HOST Server Module pop-up (see [Figure 222](#)).

Figure 221 • Edit Screen - Member SERVER Created

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                Member SERVER created
Command ==>                                           Scroll ==> CSR
***** ***** Top of Data *****
000001          *****
000002          *                               ASG, INC.                               *
000003          *                               *                               *
000004          * GENERATED BY ASG-ENCORE      RELEASE 6.0(000)          *
000005          *                               *                               *
000006          * PROGRAM:          SERVER                               *
000007          *                               *                               *
000008          * DATE:              DD-MMM-YYYY HH:MM:SS              *
000009          * REQUESTOR:        USERID                               *
000010          *                               *                               *
000011          * AKR INFORMATION                               *
000012          *   DS NAME:        USERID.PART.AKR                    *
000013          *   MEMBER:        VIARBRWS          ALIAS: (NONE)      *
000014          *                               *                               *
000015          * SEGMENT NAME:     CLIENT-SERVER-0001                 *
000016          *                               *                               *
000017          * OBJECTIVE TYPE:  CLIENT SERVER                       *
000018          IDENTIFICATION DIVISION.
000019          PROGRAM-ID. SERVER.
000020          ENVIRONMENT DIVISION.
000021

```

- 18 Type 4 and press Enter to exit the Generate - HOST Server Module pop-up and return to the Encore Primary screen. The CICS server module has now been generated.

Figure 222 • Generate - HOST Server Module Pop-up

```

C ----- Generate - Select Logic Segment ----- ll ==> CSR
|----- Generate - HOST Copy Member -----|
| E |----- Generate - HOST Server Module -----|
| C |
| - | Type Program ID and select generation options. |-----| |
| S | Objective . . . Client Server |-----| *****|
| * | Program ID . . . SERVER |-----|
|   |
|   | Functions
|   | 4_ 1. Generate options...
|   |    2. Generate COPY members...
|   |    3. Generate server module
|   |    4. Exit
|   |
|   |-----|
|-----|

```

## Understanding the Results of the CICS Server Module Generation

As a result of the CICS server extract, Encore created a standalone program that contains all of the necessary statements, files, and data elements to produce the CICS server.

[Figure 223](#) shows the comments generated by Encore at the beginning of the module. The comments indicate the extract objective type and the name of the file extracted (SERVER).

Figure 223 • CICS Server Module - Screen 1

```
ASG-Encore EDIT - USERID.ENCXXXXX.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:          SERVER                                           *
000007 *                               *                                       *
000008 * DATE:            DD-MMM-YYYY HH:MM:SS                             *
000009 * REQUESTOR:      VIASHG                                             *
000010 *                               *                                       *
000011 * AKR INFORMATION
000012 *   DS NAME:      USERID.PART.AKR                                   *
000013 *   MEMBER:       VIARBRWS          ALIAS: (NONE)                   *
000014 *                               *                                       *
000015 * SEGMENT NAME:   CLIENT-SERVER-0001                               *
000016 *                               *                                       *
000017 * OBJECTIVE TYPE: CLIENT SERVER
000018 * IDENTIFICATION DIVISION.
000019 * PROGRAM-ID. SERVER.
000020 * ENVIRONMENT DIVISION.
000021
```

Encore flags any code changes it has made with comments imbedded within the program, as shown in [Figure 224](#). Data definitions that are not needed by the server program are removed.

For example:

- The original DFHCOMMAREA is now renamed to CPYCOMAR and defined to be a record under the new DFHCOMMAREA (parameter list).
- Data needed to be communicated between the client and the server programs are now defined as another record (VIARBRCS) under the new parameter list.

**Figure 224 • CICS Server Module - Screen 2**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>>                                         Scroll ==> CSR
000022      DATA DIVISION.
000023
000024      WORKING-STORAGE SECTION.
000025      01 COMMAREA.
000026         02 STATS PIC X(1).
000027      *                                           BUILDS PREV BACK PAGE
000028         02 RIDB  PIC 9(6).
000029      *                                           BUILDS NEXT FWD PAGE
000030         02 RIDF  PIC 9(6).
000031         COPY DFHAID.
000032      *                                           FILEA RECORD DESCRIPT'N
000033      01 FILEA.
000034         COPY DFH$CFIL.
000035
000036      LINKAGE SECTION.
000037
000038      01 DFHCOMMAREA.
000039         COPY CPYCOMAR.
000040         COPY VIARBRCS.

```



Code referring to the original DFHCOMMAREA is changed to use the new record ASG-DFHCOMM, which is passed in by the client program. Original code not associated with the page-backward logic has been removed, as shown in [Figure 226](#). In this case, the code removed deals with the initialization of the original DFHCOMMAREA, which should be performed now by the client.

**Figure 226 • CICS Server Module - Screen 4**

```
ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==>>                                     Scroll ==>> CSR
000071      *          MOVE DFHCOMMAREA TO COMMAREA
000072      *
000073      *          GO TO PROMPT.                ASG-ENCORE- MOVE REMOVED
000074      *
000075      *          THIS COMMAND MAPS IN THE ACCOUNT NUMBER FROM THE OPERATOR
000076      *          INSTRUCTION MENU. NOTE THE EXPLICIT TESTING OF THE RESPONSE
000077      *          TO THE COMMAND.
000078      *
000079      *
000080      *          THE BACKWARD BROWSE IS SIMILAR TO THE FORWARD BROWSE.
000081      *          NOTE THAT AN EXTRA CALL TO THE "READ-PREV" ROUTINE IS NOT
000082      *          REQUIRED WHEN BROWSING BACK FROM THE HIGH END OF THE FILE.
000083      *
000084      *          PAGE-BACKWARD.
000085      *
000086      *          LOW END OF FILE
000087      *          RESET MAP 'C'
000088      *
000089      *          MOVE LOW-VALUES TO BROWSEO.
000090      *
000091      *          RIDF ---> NEXT FPAGE
000092      *
```

Error checking code has been modified since the original way of showing errors would not work in the server. You need to review, and possibly replace, all code generated from the CICS server module extraction/generation process. [Figure 227](#) shows an example of code that should be reviewed and/or replaced by other code that sets an error indicator in the parameter list so the client can display the error properly.

**Figure 227 • CICS Server Module - Screen 5a**

```
ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                           Scroll ==> CSR
000093          IF EIBCALEN = 0 THEN GO TO TEST-STATS.
000094          *
000095          *   START BROWSE WHERE WE LEFT OFF LAST TIME.
000096          *
000097          EXEC CICS STARTER FILE('FILEA') RIDFLD(RIDB IN COMMAREA)
000098             RESP(RESPONSE) END-EXEC.
000099          *
000100          *   CHECK RESPONSES
000101          *
000102          IF RESPONSE = DFHRESP(NOTFND) THEN
000103             NEXT SENTENCE.
000104          *                                     ASG-ENCORE- NEXT SENTENCE
000105          IF RESPONSE NOT = DFHRESP(NORMAL) THEN
000106             NEXT SENTENCE.
000107          *                                     ASG-ENCORE- NEXT SENTENCE
000108          *
000109          TEST-STATS.
000110          *
000111          IF STATS IN COMMAREA = 'H' THEN GO TO PREV-LINE.
000112          *
000113          *   READ AND DISCARD THE RECORD POINTED TO BY RIDB ONLY
000114          *   IF THE HI END OF THE FILE HAS NOT BEEN REACHED
```

[Figure 228](#) shows where the data generated by the page-backward logic is displayed by the original code. Encore has replaced the SEND MAP statement with statements that update the parameter list and return to the caller, which is expected to display the data in the parameter list.

**Figure 228 • CICS Server Module - Screen 6**

```

ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>                                         Scroll ==> CSR
000225      * EXEC CICS SEND MAP('BROWSE') MAPSET('VIARBRU')
000226      * ERASE END-EXEC.
000227      * MOVE EIBAID TO ASG-EIBAID.
000228      *                                     ASG-ENCORE- MOVE
000229      * MOVE COMMAREA TO ASG-DFHCOMM
000230      *                                     ASG-ENCORE- MOVE
000231      * EXEC CICS RETURN END-EXEC.
000232      *                                     ASG-ENCORE- GENERATED CICS RETURN STMT
000233      * ASG-ENCORE NOTE:
000234      * THIS CICS RETURN STATEMENT WAS REMOVED.
000235      * EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
000236      * LENGTH(13) END-EXEC.
000237      * GOBACK.
000238      *                                     ASG-ENCORE- GENERATED GOBACK STMT
000239      *
000240      * AFTER THE "RECEIVE" COMMAND EXECUTES, THE PROGRAM TESTS THE
000241      * OPERATORS RESPONSE. ONLY "CLEAR", "PF1", "PF2", "F" OR "B"
000242      * KEYS ARE TAKEN AS VALID INPUT.
000243      * ALL OTHER RESPONSES ARE IGNORED.
000244      *
000245      * PROMPT.
000246      * ASG-ENCORE NOTE:

```

Figure 229 shows that the original RECEIVE MAP statement has been removed. Data that were originally set by the RECEIVE MAP statement are expected to be passed in using the parameter list when this CICS server module is invoked by the client.

- The CLEAR key processing should have been handled by the client.
- The page-forward logic should have been extracted to a separate CICS server module. The client code should have dispatched the processing of the page-forward command to that logic. Checking for the page-forward command is therefore unnecessary. Encore replaces it with a statement that does not process it.
- The READ-NEXT paragraph has been removed because it is not needed by the page-backward logic.

Figure 229 • CICS Server Module - Screen 7

```
ASG-Encore EDIT - USERID.ENCXXXXX.COBOL                COLUMNS 00001 00072
Command ==>>                                         Scroll ==>> CSR
000247      *          RESP(RESPONSE) END-EXEC.
000248      *  MOVE DFHRESP(NORMAL) TO RESPONSE.
000249      *                                     ASG-ENCORE- MOVE RESP
000250      *
000251      *  CHECK CLEAR KEY.
000252      *
000253      *  IF EIBAID = DFHCLEAR THEN.
000254      *          CONTINUE.                                     ASG-ENCORE - CONTINUE
000255      *
000256      *  CHECK PF KEYS.
000257      *
000258      *  IF EIBAID = DFHPPF1 OR DIRI = 'F' THEN
000259      *          CONTINUE.
000260      *                                     ASG-ENCORE - CONTINUE
000261      *  IF EIBAID = DFHPPF2 OR DIRI = 'B' THEN GO TO PAGE-BACKWARD.
000262      *
000263      *  CHECK RESPONSES.
000264      *
000265      *
000266      *  THIS ROUTINE EXECUTES A "READPREV" COMMAND TO READ THE NEXT
000267      *  RECORD INTO THE FILE AREA, WITH RESPECT TO THE KEY IN "RIDB".
```

[Figure 230](#) shows that the original code to display the main menu has been replaced by a RETURN statement to transfer control back to the client.

**Figure 230 • CICS Server Module - Screen 8**

```
ASG-Encore EDIT - USERID.ENCXXXXX.COBOL          COLUMNS 00001 00072
Command ==>                                     Scroll ==> CSR
000264      *   THIS ROUTINE EXECUTES A "READPREV" COMMAND TO READ THE NEXT
000265      *   RECORD INTO THE FILE AREA, WITH RESPECT TO THE KEY IN "RIDB".
000266      *
000267      READ-PREV.
000268      EXEC CICS READPREV INTO(FILEA) FILE('FILEA')
000269      RIDFLD(RIDB IN COMMAREA) RESP(RESPONSE) END-EXEC.
000270      *
000271      *   IN SOME ERROR SITUATIONS A DUMP IS TAKEN AND THE MESSAGE
000272      *   "TRANSACTION TERMINATED" IS MOVED TO "MESSAGES" FOR DISPLAY
000273      *   ON THE OPERATOR INSTRUCTION SCREEN.
000274      *
000275      *
000276      *   DISPLAY GENERAL MENU THEN EXIT.
000277      *
000278      MENU.
000279      *
000280      *   RESET MAP 'A'
000281      *
000282      MOVE EIBAID TO ASG-EIBAID.
000283      *                                     ASG-ENCORE- MOVE
000284      EXEC CICS RETURN END-EXEC.
000285      *                                     ASG-ENCORE- GENERATED CICS RETURN STMT
```

## CICS Server Extract Compilation Issues

A module-generated from a CICS 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, as discussed in these CICS server extract conditions.

### Condition 1

Encore has generated compile errors on label names used in CICS HANDLE CONDITION statements, but not defined in the CICS server module.

Cause	Action
Encore removed error handling code from the server program that is not part of the server logic.	Review your error handling logic to see if the CICS HANDLE CONDITION statements in the server program are still viable. You may want to replace the original error handling logic with code that sets certain error indicators and returns them as output parameters. Additionally, you may also want to replace the error detection method. Current programming practices recommend using the DFHRESP option instead of HANDLE CONDITION statements to detect error conditions. See your CICS application programming guides for details about how this should be done.

---

**Condition 2**

Encore has generated incorrect NEXT SENTENCE statements that appear to lead to paths for which the preconditions have apparently failed.

Cause	Action
The preconditions that apparently failed should never happen in the first place. Encore does not automatically eliminate these conditional statements, giving the false impression that the code generated is incorrect.	Review all the conditional statements that contain NEXT SENTENCE statements generated by Encore. If these conditional statements perform checks before the proper start of the server logic, make sure they have been replaced by equivalent checks on the client side before the server is invoked. Once this is done, these conditional statements can be commented out or eliminated from the server program. For example, the original logic that causes the server to be invoked may perform a check to see if the EIBAID key is the PF1 function key. This logic may be implemented in the client as a check or to determine whether a PAGE DOWN button was clicked. The server in this case would not be invoked unless the condition is true on the client side. The corresponding check on the server side is therefore redundant and should be eliminated. They are retained in the server code generated by Encore only as a reminder that they should be reimplemented in the client code.

### Condition 3

Encore has generated some incorrect branches. They should have led to other paths but are replaced by Encore with simple NEXT SENTENCE statements that nullify the branch statements.

Cause	Action
The branches that are nullified would have lead to paths that are not in the server.	Review all the conditional statements that contain NEXT SENTENCE statements generated by Encore. If these conditional statements test for conditions that would have caused a different SEND MAP statement to be executed, replace these statements with code to set a certain indicator in the output parameter list. The client side logic should be implemented to check this indicator to decide what other actions to take after the server returns.

### Condition 4

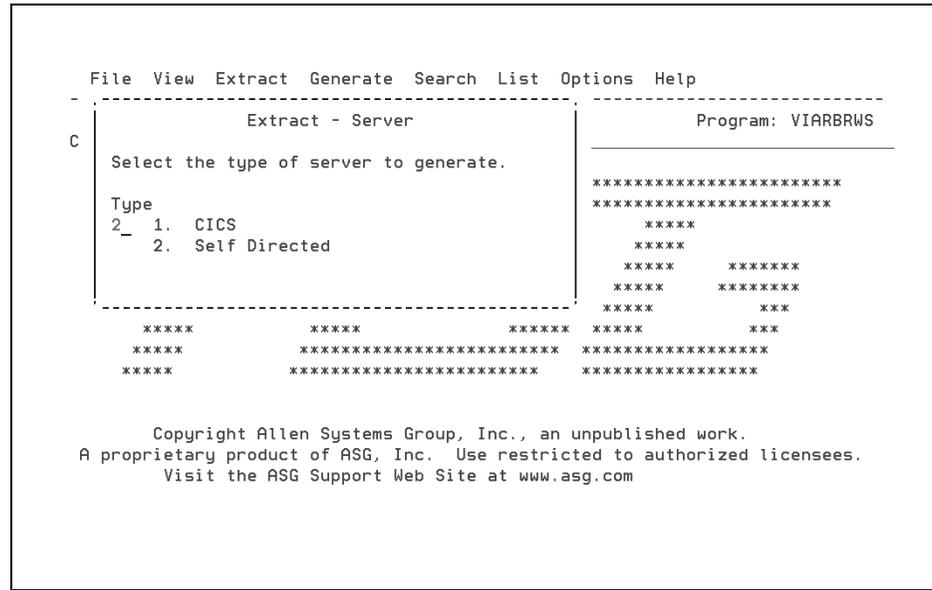
Encore has generated compile errors on COPY statements.

Cause	Action
The COPY members have not been generated or were generated but saved under incorrect names.	Go through the " <a href="#">Generating the CICS Server Module</a> " on page 223 steps until you reach the Generate - COPY Members pop-up (see <a href="#">Figure 212 on page 228</a> ). Verify that all COPY members have been generated. If the COPY members are not found, even though Encore shows them as having been generated, check your copy libraries to see if the COPY members have been generated under the correct names. Encore places you in an edit session in the generate process so you can save the result in any dataset you choose, but Encore has no way of verifying if you have named the resulting file to match the COPY statement.



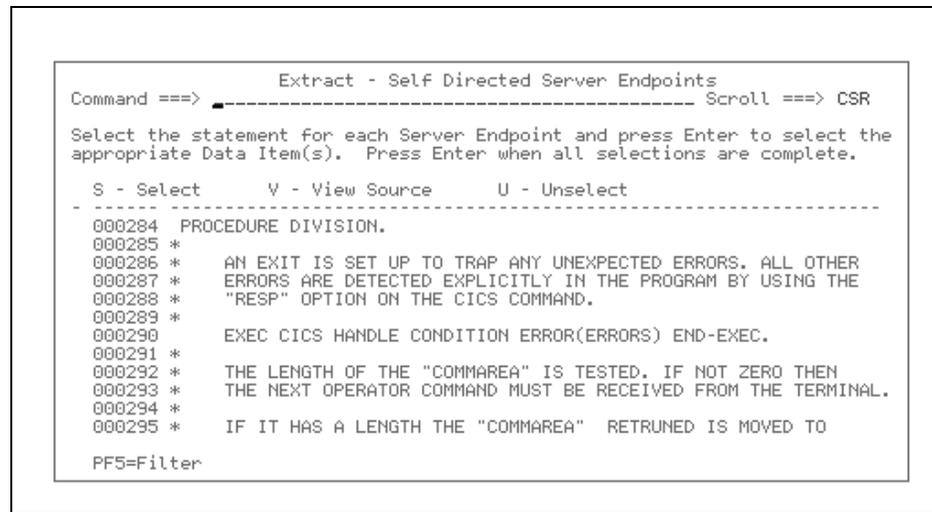
- 3 [Figure 232](#) contains a list of available server types. Type 2 to select Self Directed and press Enter to display the Extract - Self Directed Server Endpoints pop-up (see [Figure 233](#)).

**Figure 232 • Extract - Server Selection Pop-up**



- 4 You can reduce the list of selectable names by pressing PF5, or by typing Filter and pressing Enter to display the Extract - Filter - Self Directed Server Filter pop-up (see [Figure 234 on page 247](#)).

**Figure 233 • Extract - Self Directed Server Endpoints Pop-up**



- 5 This pop-up allows you to specify filtering criteria. Type the data name you want to search for (COMMAREA in this example) and enter the other options you want to use in the filter process, as shown in [Figure 234](#). Press Enter to display the Extract - CICS Server Endpoints pop-up (see [Figure 235 on page 248](#)).

**Figure 234 • Extract - Filter - Self Directed Server Filter Pop-up**

```

C                                     Extract - Self Directed Server Endpoints
S                                     Extract - Self Directed Server Filter
a Enter the desired Filtering parameters and press Enter.

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

Copy Book Name      -----
Line Range . . . . -----
Label Name . . . . -----
COBOL Subset . . . -----

000294 *
000295 * IF IT HAS A LENGTH THE "COMMAREA" RETRUNED IS MOVED TO
PF5=Filter

```

- 6 [Figure 235](#) contains all of the filtered statements associated with the CICS member. You need to extract all statements that contribute to the output data. To ensure that you are selecting the code you want to extract, type V next to the statement you want to view and press Enter to display the View Source pop-up (see [Figure 236 on page 249](#)).

**Figure 235 • 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
-----
V 000343      EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000344          RESP(RESPONSE) END-EXEC.
- 000358      IF RIDF IN COMMAREA NOT EQUAL '999999' THEN
- 000378      EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000379          RESP(RESPONSE) END-EXEC.
- 000391      IF RIDF IN COMMAREA = '000000' THEN GO TO NEXT-LINE.
- 000422      MOVE RIDF IN COMMAREA TO RIDB IN COMMAREA.
- 000490      EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
000491          LENGTH(13) END-EXEC.
- 000510      EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDB IN COMMAREA)
000511          RESP(RESPONSE) END-EXEC.
- 000522      IF STATS IN COMMAREA = 'H' THEN GO TO PREV-LINE.

PF5=Filter

```

- 7 Verify that the selected statement code represents the type of code you want to extract, as shown in [Figure 236](#). After verifying the code, press PF3 to return to the Extract - Self Directed Server Endpoints pop-up (see [Figure 237](#)). If necessary, repeat this process until you find the code you want to extract.

**Figure 236 • View - Source Pop-up**

```

View Search List Options Help
-----
View - Source          Prn  2 LINES FOUND
Command ==> _         Scroll ==> CSR

000337 *   ACCOUNT NUMBER OMITTED
000338 *
000339 *   MOVE '000000' TO RIDF IN COMMAREA.
000340 *
000341 *   THE "STARTBR" COMMAND ESTABLISHES THE BROWSE STARTING POINT.
000342 *
000343 *   EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA) LINE R
000344 *           RESP(RESPONSE) END-EXEC.                               LINE RNG
000345 *
000346 *   IF THE "NOTFND" CONDITION OCCURS AT THE START BROWSE THE
000347 *   MESSAGE "END OF FILE - PLEASE RESTART" IS MOVED TO "MESSAGES"
000348 *   FOR DISPLAY ON THE OPERATOR INSTRUCTION SCREEN.
000349 *
000350 *   IF RESPONSE = DFHRESP(NOTFND) THEN
000351 *       MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
000352 *       GO TO MENU.
000353 *   IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.

```

- 8 Type S next to the Endpoint server statement line you want to select, and press Enter to display the Extract - Self Directed Server Data Names pop-up (see [Figure 238 on page 250](#)).

**Figure 237 • 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.

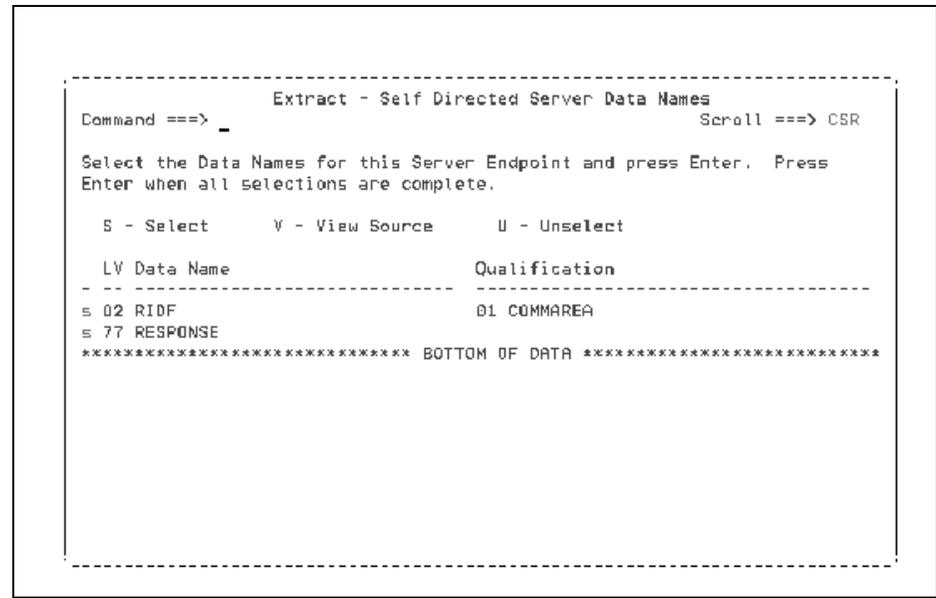
S - Select      V - View Source      U - Unselect
-----
S 000343 EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000344   RESP(RESPONSE) END-EXEC.
- 000358 IF RIDF IN COMMAREA NOT EQUAL '999999' THEN
000378 EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000379   RESP(RESPONSE) END-EXEC.
- 000391 IF RIDF IN COMMAREA = '000000' THEN GO TO NEXT-LINE.
- 000422 MOVE RIDF IN COMMAREA TO RIDB IN COMMAREA.
000490 EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
000491   LENGTH(13) END-EXEC.
- 000510 EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDB IN COMMAREA)
000511   RESP(RESPONSE) END-EXEC.
- 000522 IF STATS IN COMMAREA = 'H' THEN GO TO PREV-LINE.

PF5=Filter

```

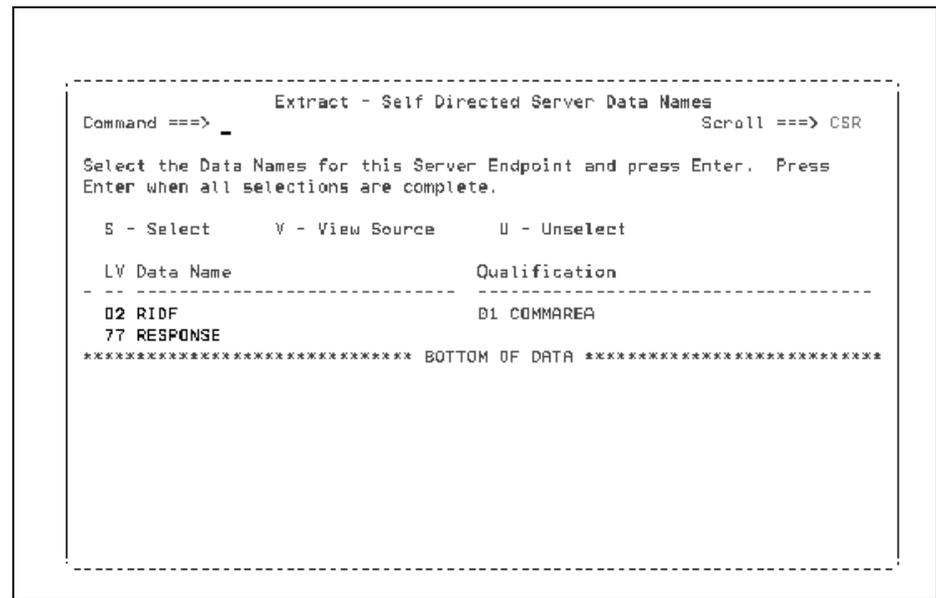
- 9 Type S to the left of the data name(s) you want, as shown in [Figure 238](#).

**Figure 238 • Extract - Self Directed Server Data Names Pop-up**



- 10 The selected data names are highlighted, as shown in [Figure 239](#). Press Enter to display the Extract - Self Directed Server Startpoints pop-up (see [Figure 240 on page 251](#)).

**Figure 239 • Extract - Self Directed Server Data Names Pop-up - Selected Data Names**



- 11 Page down until you find the startpoint and type S next to the Startpoint server statement line you want to select, as shown in [Figure 240](#). Press Enter to display the Extract - Self Directed Server Endpoints pop-up (see [Figure 241](#)).

**Figure 240 • 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
-----
000319 *
- 000320     IF KEYL NOT = ZERO THEN
000321     IF KEYI IS NUHERIC THEN
000322 *
000323 *         VALID INPUT
000324 *
S 000325         MOVE KEYI TO RIDF IN COMMAREA
000326         MOVE KEYI TO RIDB IN COMMAREA
- 000327     ELSE
000328 *
000329 *         NOT NUMERIC
000330 *

PF5=Filter
-----

```

- 12 Verify that the message in the long message area indicates that the statements for the EXEC CICS SEND code were successfully selected. Press Enter to display the Extract - Self Directed Server Paths pop-up (see [Figure 242 on page 252](#)).

**Figure 241 • 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
-----
- 000343     EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000344     RESP(RESPONSE) END-EXEC.
- 000358     IF RIDF IN COMMAREA NOT EQUAL '999999' THEN
- 000378     EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDF IN COMMAREA)
000379     RESP(RESPONSE) END-EXEC.
- 000391     IF RIDF IN COMMAREA = '000000' THEN GO TO NEXT-LINE.
000422     MOVE RIDF IN COMMAREA TO RIDB IN COMMAREA.
- 000490     EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
000491     LENGTH(13) END-EXEC.
- 000510     EXEC CICS STARTBR FILE('FILEA') RIDFLD(RIDB IN COMMAREA)
000511     RESP(RESPONSE) END-EXEC.
- 000522     IF STATS IN COMMAREA = 'H' THEN GO TO PREV-LINE.

ASG4112I ENDPOINT AND STARTPOINT STATEMENTS SUCCESSFULLY SELECTED.
-----

```

- 13 The default T (TRUE) or F (FALSE) selections have been made based on the selection criteria, as shown in [Figure 242](#). When you are done reviewing or changing the information on this pop-up press Enter to display the Extract - Self Directed Server Outputs pop-up (see [Figure 243 on page 253](#)).

Figure 242 • 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 NDT = 0 THEN
F 000311      IF RESPONSE = DFHRESP(MAPFAIL) THEN
F 000314      IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
T 000320      IF KEYL NOT = ZERO THEN
000321          IF KEYL IS NUMERIC THEN
000350      IF RESPONSE = DFHRESP(NDTFND) 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(NDTFND) THEN
_ 000386      IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
000391      IF RIDF IN COMMAREA = '000000' THEN GO TO NEXT-LINE.
000400      IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
-----

```

**Note:** \_\_\_\_\_

An X next to a generated conditional indicates that the code does not apply to this extract.

- 14 Select the Output statement you want to include in the server and press Enter to save the selection, as shown in [Figure 243](#). Press Enter again to return to the Encore Primary screen (see [Figure 244](#)).

**Figure 243 • 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 ACCDUNT NUMBER FROM THE OPERATOR
000303 *  INSTRUCTION MENU. NOTE THE EXPLICIT TESTING OF THE RESPONSE
000304 *  TO THE COMMAND.
000305 *
S 000306      EXEC CICS RECEIVE MAP('MENU') MAPSET('VIARHNU')
000307          RESP(RESPONSE) END-EXEC.

PF5=Filter
-----

```

The Encore Primary screen displays with a message indicating that criteria for the Statement extract were successfully selected.

**Figure 244 • Encore Primary screen - Confirmation Message**

```

File View Extract Generate Search List Options Help
-----
                                ASG-Encore                                Program: VIARBRWS
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

ASG4111I SELF-DIRECTED SERVER STATEMENTS SUCCESSFULLY SELECTED.
-----

```

See the online help for more information about Encore Extract features.

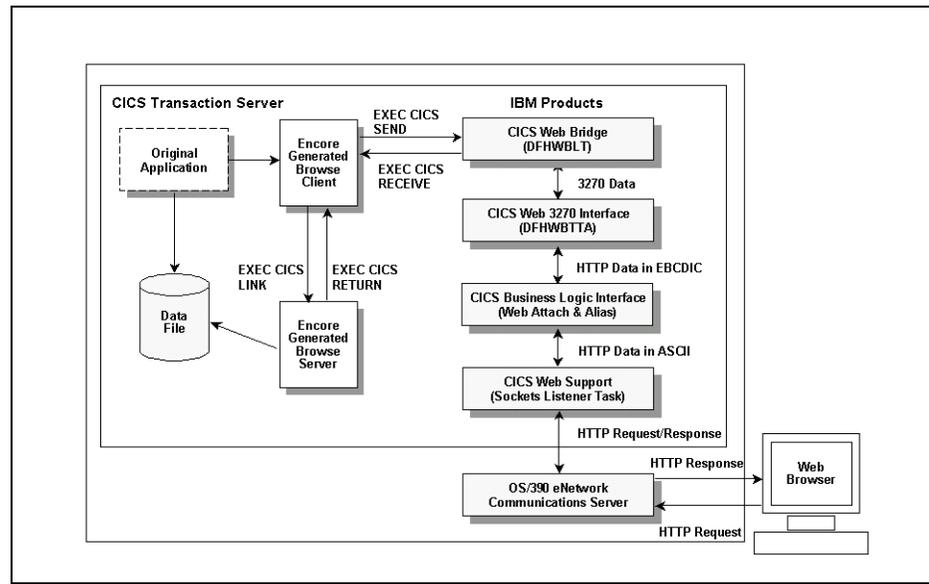
## Using IBM Solutions

After you have extracted and generated the server-side components from a CICS application, you can use a variety of existing IBM solutions to migrate to a client/server or web-based architecture. These server components may be deployed in a web-based application in a variety of ways. This section presents three alternatives for implementing a web-based application using the server-side components generated using Encore. As all three alternatives rely on IBM products to provide the communication infrastructure, the last part of this section describes the resources available from IBM to help implement these alternative solutions.

### Approach #1: Web-to-CICS Using 3270 Bridge

[Figure 245](#) shows the structure of an application that uses CICS Web Support and 3270 Bridge to translate between BMS screens and HTML pages, which provides quick access to existing 3270-based applications from the web. The boxes on the right side of the figure represent software products or components available from IBM.

Figure 245 • Web-to-CICS Using 3270 Bridge Example



Although web access using 3270 Bridge can be done without separating the business logic and the user interface, making the separation allows the logic in each component to be more readily identified and understood. It also allows the presentation logic to be isolated and tested before being migrated to other presentation technologies, such as HTML. Extracting the server components using Encore and deploying them under this approach provides a migration path to solutions based on other approaches in the future.

These are some advantages:

- Rapid implementation.
- Low risk because this solution requires minimal changes to an existing application.
- Only a single copy of presentation logic is required.

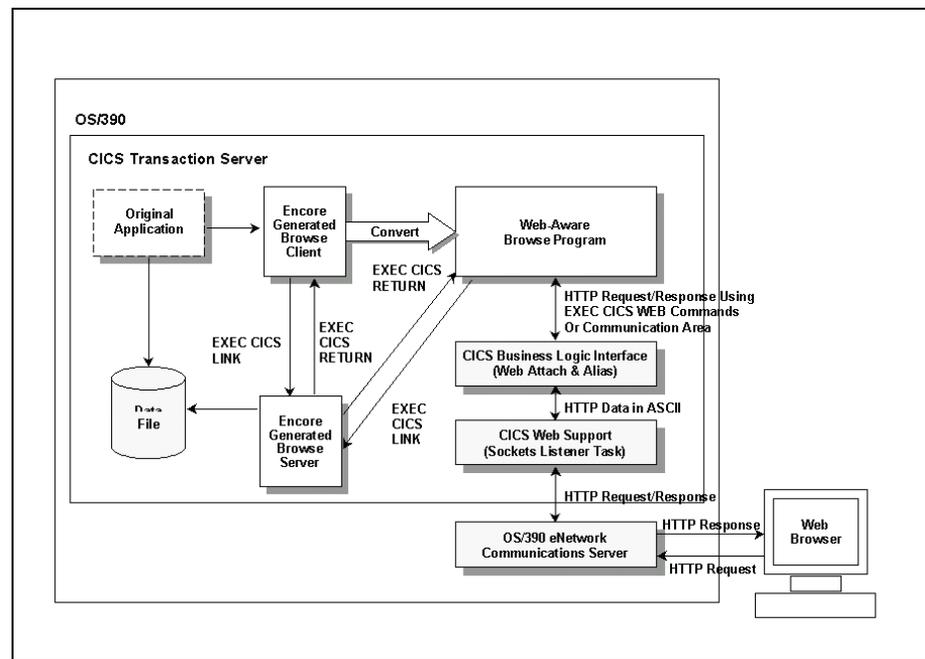
These are some disadvantages:

- Clumsy looking and inflexible user interface.
- Server resources consumed by the user interface limit scalability.
- Solutions are unlikely to be long-lived or strategic.

## Approach #2: Web-to-CICS Using Web-Aware CICS Programs

Figure 246 shows the structure of an application that uses user-written CICS programs to generate the HTML pages to be displayed in web browsers and to process requests. These programs use EXEC CICS WEB commands or the communication area to receive HTTP requests, invoke the proper CICS server programs using EXEC CICS LINK, and then format the results into HTML pages, which are returned as the HTTP response.

Figure 246 • Web-to-CICS Using Web-Aware CICS Programs Example



These are some advantages:

- Presentation logic stays on the original platform.
- Flexibility in customizing the user interface.
- Ease of integrating outputs from multiple business components.

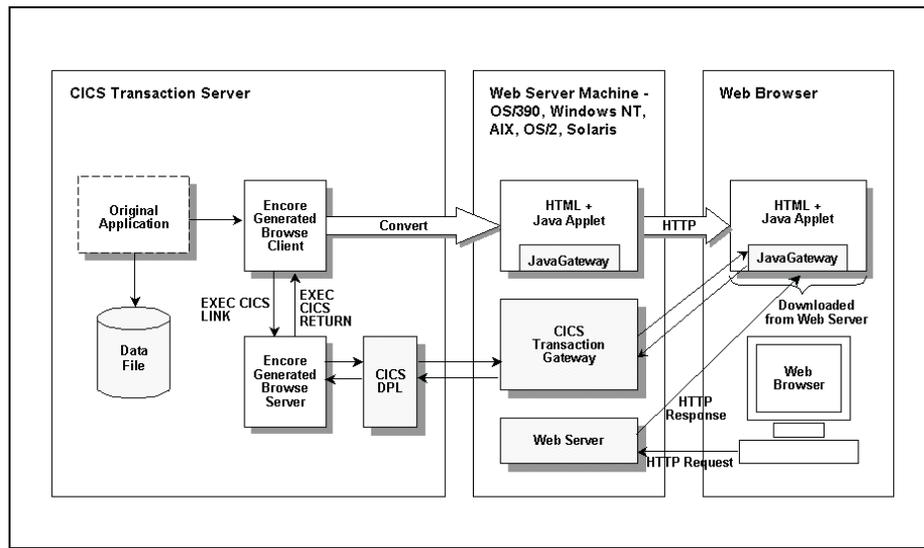
These are some disadvantages:

- Multiple versions of presentation logic; one for the web and one for the legacy users.
- Server resources consumed by the user interface may limit scalability.
- Requires combining web page design expertise with CICS development expertise.

### Approach #3: Web-to-CICS Using Java

[Figure 247](#) shows the structure of an application providing web access to CICS using CICS Transaction Gateway. CICS Transaction Gateway runs on the same platform as the web server and communicates with CICS applications running on CICS servers through ECI or EXCI. The presentation logic in this approach is located on the same platform as the web server, implemented as HTML pages and Java applets. A CICS Java-class library is provided to allow the applets to communicate between the web page displayed in the web browser and the CICS Transaction Gateway.

**Figure 247 • Web-to-CICS Using Java Example**



These are some advantages

- Presentation is controlled by web design experts.
- Multiple options are available in web server platforms.
- Diverse selection of web page and Java design tools.
- Potential of integrating outputs from multiple business components on multiple platforms.
- Scalable.

These are some disadvantages

- Multiple versions of presentation logic.
- Significant initial development time and effort.

## Available IBM Resources

Each of the approaches outlined in this section requires significant work in planning, design, implementation, and deployment. These issues must be resolved first:

- Proper selection, configuration, and implementation of the middleware (the communication infrastructures between the presentation layer and the business logic). Currently, there are a number of alternatives, each with its own strengths and weaknesses. The choice must be made with careful planning and deliberation.
- Proper implementation of the presentation layer. It should be noted that because external users of an application often have different expectations of the user interface than the internal users, a mechanical translation of the user interface rarely provides an acceptable alternative to the need for re-implementing the user interface.
- Security and authorization. Opening up an internal application for external access exposes an enterprise to new issues in security and authorization. Solving these issues is not easily automated with today's technology.
- Application development tools, processes, technologies, and expertise. The pace at which the web is evolving demands solutions that are flexible and quickly implemented, while remaining durable. Achieving a balance between flexibility and durability requires resolving issues that are far beyond what a software product can address.

Most of these issues do not have simple, product-based solutions. IBM's International Technical Support Organization (ITSO) has developed a collection of books to deal with these types of issues. Known as Redbooks, these books cover general concepts, provide background materials, and offer practical technical guidance on how to solve these problems. These are some of the documents you may want to review to gain a better understanding of the issues concerning migration of an existing OS/390 CICS application to support web access:

- *CICS Transaction Server for OS/390 Version 1 Release 3: Web Support and 3270 Bridge*, SG24-5480-00, last updated July 15, 1999.
- *Revealed! Architecting Web Access to CICS*, SG24-5466-00, published June 9, 1999.
- *Revealed! CICS Transaction Gateway with More CICS Clients Unmasked*, SG240-5277-00, published December 16, 1998.

Additional reference material may be found in IBM's product documentation libraries. One of the most useful guides is *CICS Transaction Server for OS/390 - CICS Internet Guide (SC34-5445-00)*, published March 1999.

**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 270](#).

**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 269](#).

**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.

**data name**

A standard COBOL term for fields defined in the DATA DIVISION of a COBOL program. Variable names, files, groups, array elements, and fully qualified data names.

**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 261.

**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 262](#).

**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 is displayed on line five of a screen or line three of a pop-up. Long messages are sometimes preceded by short messages that are displayed 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 265](#).

**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 data names 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 appears 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 appears 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.

**reengineering**

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 270](#).

**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 is displayed 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 261](#), ["screen subset" on page 269](#), and ["tagged line subset" on page 271](#).

**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.

## A

- action bar
  - action descriptions 7
  - description 6
  - how to select actions 6
  - overview 6
  - shortened 6
- AKR, allocating 33
- Alliance
  - accessing from ESW screen xiv
  - description xi
  - linking xiv
- Analysis/Anomaly Report parameters
  - NUMPRM(##) 38
  - RPTCEN 38
  - VIARERPT 38
  - VIARERPTAN 38
- Analyze Facility
  - analyze a program through Encore 37
  - analyze input 36
  - compile/analyze 38
  - requirements 36
- Anomaly Repair Facility parameter,  
VIAANOMF 38
- AutoChange
  - accessing from ESW screen xiv
  - description xi

## B

- Bridge
  - accessing from ESW screen xiv
  - description xi

## C

- Center, description xi
- CICS server demo, presentation logic 214
- CICS server endpoints 218
- CICS server extract
  - 3270 Bridge 254
  - Approach #1, Web-to-CICS Using  
3270 Bridge 254

- Approach #2, Web-to-CICS Using  
Web-Aware CICS  
Programs 256
- Approach #3, Web-to-CICS Using  
Java 257
- ASG-DFHCOMM 237
- available IBM resources 258
- BMS screens 254
- business logic - CICS server  
demo 214
- CICS applications 257
- CICS clients 214
- CICS EXEC 219
- CICS extract features 222
- CICS HANDLE CONDITION  
statements 242
- CICS HANDLE statements 242
- CICS Java class library 257
- CICS pseudo-conversational  
program 44, 213
- CICS server module 223
- CICS server programs 256
- CICS servers 257
- CICS statements 221
- CICS transaction gateway 257
- CICS web support 254
- COMMAREA 213
- COMMAREA-based server  
program 44
- compilation issues 242
- copy member field 224
- COPY statements 244
- CREATE command 228
- description 213
- DFHCOMMAREA 235–236
- DFHRESP 242
- ECI 257
- EIBAID 236
- EIBAID key 243
- EXCI 257
- EXEC CICS LINK 256
- EXEC CICS SEND 221

- EXEC CICS WEB commands 256
  - generation facilities 214
  - HTML pages 254, 256
  - HTTP requests 256
  - HTTP response 256
  - IBM solutions 254
  - IBM's International Technical Support Organization (ITSO) 258
  - Java applets 257
  - java servlets 214
  - MQSeries 214
  - NEXT SENTENCE statements 243
  - Next-Page server module 215
  - OS/390 CICS application 258
  - parameter list 235
  - program 44
  - programs 214
  - READ-NEXT paragraph 240
  - RECEIVE MAP statement 240
  - RETURN statement 241
  - self directed server name 218
  - SEND MAP statement 239, 244
  - server extract 214
  - starting level number 224
  - understanding generated results 234
  - understanding the results of the generated CICS server module 234
  - utilizing IBM solutions 254
  - VIARBRCS 235
  - web access 215
  - CICS server scenario, 3270 Bridge 214
  - CICS server, description 4
  - COBOL terms
    - COBOL subset 15
    - set 14
    - subsets 15
    - subsets list 15
  - COBOL, OF clause 20
  - code extract, extracting code from a logic segment 153, 207
  - command
    - END 6
    - FINDXTND 20
    - SCROLL 20
  - common code
    - eliminating within a multiple perform range extract 99
    - finding between a callable sub-module and its complement 112
    - within a multiple perform range extract 99
  - compilation issues in the server extract 242
  - complement module, contents 45
  - computation variable extract
    - business scenario 133
    - compilation issues 155
    - extracting an objective 135
    - including non-selected code in the logic segment 149
  - conventions page xvii
  - CUA features 6
- D**
- data name
    - alias 19
    - DEFINITION 20
    - description 19
    - fully qualified 20
    - group level 20
    - MODIFICATION 20
    - REFERENCE 20
    - USE 20
  - data name type
    - elementary data name 19
    - file name 19
    - group name 19
    - special name 19
    - table element name 19
  - data, element 19
  - decomposition, selecting a program 41
  - definition
    - perform range extract 42
    - report extract 44
    - statement extract 44
    - transaction extract 43
- E**
- Encore
    - accessing from ESW screen xiv
    - action bar 6
    - description xii
    - functional components 2
    - getting started 23
    - pop-ups 12
    - screens 10
  - END command 6
  - Estimate
    - accessing from ESW screen xiv
    - description xii
  - ESW
    - description x
    - invoking products xiii
    - product integration xiv
  - extract objective
    - CICS server extract 44

- computation-variable extract 43
    - perform range extract 42
    - report extract 44
    - statement extract 44
    - transaction extract 43
  - extracting multiple perform ranges
    - creating separate modules 92
    - creating the called sub-module 85
    - creating the complement module 104
    - expanding a collapsed perform range list 89
    - line commands 89
    - with multiple entry points 85
  - extracting the CICS server program 217
  - extracting the CICS server program, self directed server 245
- G**
- generate cobol module, creating the callable module 71
  - generating the CICS server module 223
  - group item 19
- I**
- Insight
    - accessing from ESW screen xiv
    - description xii
    - using analysis functions xiv
  - IO module
    - example on generating 119
    - generating 119
    - substituting calls to 118
- L**
- level number increment 224
  - logic segment
    - extracting code from 153
    - including additional lines in 149
- M**
- migration 4
- O**
- overview
    - action bar 6
    - pop-up 12
    - screen 10
    - target generation 4
- P**
- pattern string, description 21
  - perform range extract
    - business scenario 57–58
    - common code 108
    - creating the callable module 71
    - creating the complement module 75
    - definition 42
    - extracting multiple perform ranges 85
    - extracting the objective 63
    - the logic segment 67
    - using 57
    - when to use 42
  - pop-up
    - description 12
    - Extract - CICS Server Endpoint 223
    - Extract - CICS Server Startpoint 223
    - File - AKR Utilities 33
    - format example 12
    - generate - COBOL module 115
    - Generate - FD Name List 119
    - Generate - IO Module 120
    - Generate - Specify Perform Range ENTRY Names 89
    - Log/List/Punch 29
    - Log/List/Punch Definition 29
    - Options - Generate 32
    - overview 12
    - PF keys 31
    - Product Allocations 28
    - Product Parameters 27
    - Select Extract-within-Logic Segment 154, 209
    - Select FD Names 116
  - pop-up screen
    - description 6
    - how to process 6
  - product integration xiv
  - program abnormalities
    - dead code, dead data 41
    - live exits 41
    - out of PERFORM jumps 41
  - project workbook, re-engineering
    - preliminary tasks 44
  - pull-down
    - description 6
    - format example 9
    - how to select actions 6
- R**
- Recap
    - accessing from ESW screen xiv
    - description xii
  - reengineering
    - choosing a code extraction objective 42
    - program cleanup 41
  - reengineering objectives

- eliminate GOTOs 41
- eliminating obsolete code 41
- smaller modules 41
- stand-alone modules 41
- report extract
  - business scenario 182
  - complement module - understanding
    - the results 198
  - creating the complement module 194
  - creating the stand-alone report
    - module 188, 223
  - definition 44
  - when to use 44

## S

- scenario, perform range extract 57

- screen

- description 10

- overview 10

- screen subsets

- EX or X subset 15

- excluded subset 15

- HI subset 15

- highlighted subset 15

- NONE or NX subset 15

- NONExcluded subset 15

- NONH or NHI subset 15

- NONHighlighted subset 15

- target 15

- SCROLL command 20

- self directed server

- EXEC CICS SEND code 251

- extract 245

- server extract 48

- set, multiple 14

- shortened action bar 6

- SmartDoc

- accessing from ESW screen xiv

- description xii

- SmartEdit

- accessing from ESW screen xiv

- description xiii

- SmartTest

- accessing from ESW screen xiv

- description xiii

- statement extract 48

## T

- table entry 19

- tagged line subsets

- tags 15

- target tags 15

- target

- data name description 19

- description 18

- label name description 18

- line range description 19

- paragraph name description 19

- pattern 21

- pattern string description 21

- perform range name description 18

- program name description 19

- subset name description 19

- target generation, overview 4

- transaction extract

- definition 43

- when to use 43

- transaction variable extract

- business scenario 157

- compilation issues 178

- creating the replacement module 165

- selecting the variable name 161

- start/end usage notes 176

## U

- user options

- how to set online operation

- parameters 27

- how to setting log, list, punch,

- customization and work file 28

- setting log, list, punch, customization

- processing options 29

- setting PF key values 31

## V

- variable, redefined 19

- VIARBRWS, CICS demo program 49

- VIARDEMO, Encore demonstration

- program 49

## W

- wildcard patterns

- member list 14

- selection list 14



ASG Worldwide Headquarters Naples Florida USA | [asg.com](http://asg.com)