

ASG-Insight™ User's Guide

Version: 6.0

Publication Number: INX0200-60

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.

© 1987-2002 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

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
Product:		
Publication:		
Tape VOLSER:		

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

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

Non-Business Hours - Emergency Support

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

Preface	v
About this Publication	v
Related Publications	vi
ASG-Existing Systems Workbench (ASG-ESW)	vii
Invoking ESW Products	x
ESW Product Integration	xi
Examples	xii
Publication Conventions	xiv
1 Introduction	1
Introduction	1
Insight Overview	1
Benefits	2
2 Product Overview	5
Major Components of Insight	5
CUA Interface	6
The Action Bar	6
Pull-downs.....	8
Pop-ups	9
Screens	10
Concepts	11
Screen Messages	11
Analyze Job (Batch)	12
Application Knowledge Repository (AKR)	12
Sets	12
Subsets.....	12
Targets.....	16

3 Getting Started with Insight	21
Introduction	21
Initiating an Insight Session	22
Defining User Options	23
Setting Online Operation Parameters	24
Allocating the Log, List, Punch, and Work Files	25
Specifying Log/List/Punch Processing Options	26
Mapping PF Key Values	29
Using an AKR	30
Allocating an AKR	30
Verifying AKR Allocation Results	32
Analyze Facility	33
Program Analyze Requirements	33
Program Analyze Input	33
Analyze Submit Facility	33
Verifying Analyze Results	35
4 Program Understanding and Maintenance Techniques	37
Introduction	38
Insight Program Analysis Methodology	38
Using Automated Tasks	38
Generating a Program Overview	38
Generating a Paragraph Overview	39
Identifying Data Item Usage	39
Analyzing the Program Logic	39
Applying Quality Assurance Standards	39
Starting an Insight Session	40
Using Automated Tasks	41
Summary: Using Automated Tasks	54
Using the Insight Program Understanding Checklist	54
Getting a Program Overview	55
Discovering Program Structure and Coding Techniques	57
Getting a Paragraph Overview	73
Summary: Getting a Program Overview	79
Identifying Data Item Usage Cross-References	80
Summary: Identifying Data Item Usage Cross-References	96

Conducting an Analysis of Program Logic 96

Summary: Conducting an Analysis of Program Logic 116

Applying Quality Assurance and Standards 116

Summary: Applying Quality Assurance Standards..... 132

Index 133

Preface

This *ASG-Insight User's Guide* provides instruction for using ASG-Insight (herein called Insight). Insight is a powerful interactive system that helps programmers by automating the process of analyzing COBOL programs.

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

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

About this Publication

This publication consists of these chapters:

- [Chapter 1, "Introduction,"](#) provides an overview of the Existing Systems Workbench (ESW) and Insight benefits.
- [Chapter 2, "Product Overview,"](#) describes the Insight interface and concepts.
- [Chapter 3, "Getting Started with Insight,"](#) describes for a new user how to invoke Insight, customize user options, allocate an Application Knowledge Repository (AKR), and analyze a program.
- [Chapter 4, "Program Understanding and Maintenance Techniques,"](#) describes how to use Insight to gain program understanding and perform maintenance.

Related Publications

The documentation library for ASG-Insight 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 the ASG-Existing Systems Workbench (ESW) suite of products.
- *ASG-Insight Installation Guide* (INX0300-*nn*) provides instruction about the installation and maintenance of ASG-Insight.
- *ASG-Insight Quick Reference Card* (INX0900-*nn*) provides a summary of the commands and functions within ASG-Insight.
- *ASG-Insight Quick Reference Guide* (INX0600-*nn*) summarizes the syntax and use of ASG-Insight commands.
- *ASG-Insight Reference Guide* (INX0400-*nn*) provides detailed information about the pop-ups, screens, and commands used in ASG-Insight, including field descriptions, usage notes, and command syntax.
- *ASG-Insight User's Guide* (INX0200-*nn*) provides user information for ASG-Insight.

Note: _____

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

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 vii](#) 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 Or 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:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, panel (or screen), option names, and names of keys. A plus sign (+) is inserted for key combinations (e.g., Alt+Tab).
<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 introduces Insight and contains these sections:

Topic	Page
Introduction	1
Insight Overview	1
Benefits	2

Introduction

Insight is the Program Understanding component of ESW 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 you with debugging program abends, changing a computation, and resolving incorrect program output values.

Insight Overview

Insight is the first online technology that analyzes COBOL programs using IBM's TSO/ISPF interface. It is an electronic window through which you gain insight into the workings of a COBOL program.

Insight features Common User Access (CUA) screens, pull-downs, and pop-ups that are designed to provide easy access to all of the product features.

An action bar is the line of keywords displayed at the top of a screen. Each keyword represents a category of actions that you can perform on that screen. To view the list of actions on a pull-down, move the cursor to the desired keyword and press Enter. To make a selection, type the number that corresponds to the desired action and press Enter. The result may be a generated command or a pop-up.

A pop-up is a window that displays on your screen and allows you to enter information for the requested action. Type the desired data and options and press Enter to execute the desired action.

See ["CUA Interface" on page 6](#) for more information.

Powerful Insight facilities provide a precise, consistent, and thorough analysis of COBOL source code. You no longer need to summarize information from source listings, cross-references, and primitive online browse facilities. Each task is automated, yet blends with your current maintenance practices. You can determine what information you want and when you want it. You can also direct whether an overview or the details of a program are displayed. Insight reduces the time spent understanding programs by half.

Benefits

Insight offers COBOL programmers unprecedented speed and accuracy while analyzing source code.

You can get an overview of either structured or unstructured programs with Insight's flexible options. Insight also offers you online access to cross-referenced data, either by selecting an option on an action bar or by using a single command. Its references include automatic checks for group level and redefined datanames. In addition, Insight retains the results for further evaluation with other commands.

Insight reduces effort, eliminates oversights, and saves time.

Efficiency. You save time because you do not manually collect information from limited listings and browse facilities. Use an action bar selection or command to evaluate your program and Insight presents you with concise results.

Accuracy. The Insight analysis is accurate. It consistently and precisely evaluates COBOL source code.

Usability. The easy to use CUA screens, pull-downs, and pop-ups featured in Insight eliminate your need to search and summarize program listings. Insight displays summarized results in screen formats that prepare you for your next query.

Accessibility. Insight is at your fingertips. It is integrated into your daily TSO/ISPF work environment and supplies useful help facilities.

Focus. Insight manages technical details so you can focus on understanding programs and assessing your changes.

Understandability. Insight pull-downs, pop-ups, and commands help to automate each task you need to perform in order to learn a program. COBOL program characteristics are easily revealed through pop-ups or Insight commands. Insight presents the resulting analysis clearly, when and where you need it.

Insight offers these five features to help you through the understanding process:

- Determining program structure
- Identifying data usage and relationships
- Tracing data usage in logic
- Tracing logic flow
- Saving, reusing, and printing information

2

Product Overview

This chapter discusses components of the Insight product, and contains these sections:

Topic	Page
Major Components of Insight	5
CUA Interface	6
Concepts	11

Major Components of Insight

The Insight system consists of these major components:

- Program analyzer (Batch)
- Application Knowledge Repository (AKR)
- Source view, Tree view, and Structure view
- Search and logic facilities
- Automated program maintenance task facility

CUA Interface

Insight features CUA (Common User Access) screens, pull-downs, and pop-ups that give you easy access to all of the product features.

An action bar is the line of keywords that displays at the top of a screen. Each keyword represents a category of actions that you can perform on that screen. Process the action bar by selecting an action. Select an action by moving the cursor to the desired keyword and pressing Enter.

A pull-down is the list that displays when you select an action on the action bar. On a pull-down, actions followed by... displays a pop-up when selected. Actions not followed by ... immediately activate internal commands. Select pull-down items one of these ways:

- Move the cursor to the desired keyword and press Enter
- Type the number of the desired action in the input field and press Enter.

A pop-up is a window that displays when you select an item on a pull-down, a pop-up, or you enter certain commands. It displays on your screen to allow you to enter information for the requested action. Enter the desired data and/or option, and then follow the instructions on the pop-up to process the information.

Note: _____

Use the END command (default PF03/15) to exit a pull-down or pop-up without processing any actions. If the cursor is on the action bar when you press PF03/15, the cursor will move to the command input area of the screen.

The Action Bar

The Action bar displays on all Insight screens. On a few pop-ups, a shortened action bar displays. The shortened action bar contains fewer actions than the regular action bar, and some of the pull-downs on the shortened action bar contain fewer actions.

Select an action by moving the cursor with the tab or arrow keys to the desired keyword and pressing Enter. If the cursor is on the action bar when you press PF03/15, the cursor will move to the command input area of the screen. If the cursor is on a full screen, pressing HOME moves the cursor to the first action on the action bar.

Action	Description
Options	Displays the Options pull-down used to customize your Insight environment by setting certain parameters and options, defining PF keys, and so forth.
Help	Displays the Help pull-down used to access the online help facility.

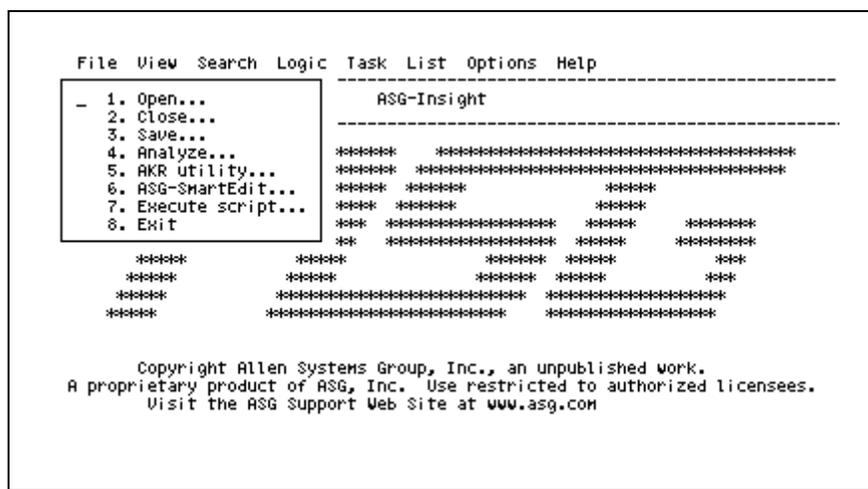
Pull-downs

A pull-down displays in a window when you select an action on the action bar (see [Figure 2](#)). Two types of actions are available on pull-downs: an action followed by ellipsis (...) displays a pop-up when selected; an action not followed by ... immediately activates the requested action and usually redisplay the previous screen.

There are two ways to select an item on a pull-down:

- Move the cursor to the desired keyword and press Enter
- Type the number of the desired action in the input field and press Enter.

Figure 2 • Pull-down Format



On the File pull-down, all of the actions except Exit display a pop-up when selected.

Note:

Use the END command or PF03/15 to exit a pull-down without processing any actions.

Pop-ups

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

Follow the instructions on the pop-up to process the information. Use the END command (default PF03/15) to exit from a pop-up without processing it.

Pop-ups have two types of fields: entry fields and selection fields. Entry fields are used to type textual information and selection fields are used to select one or more choices.

[Figure 3](#) shows the Search - Data Name pop-up.

Figure 3 • Pop-up Example (Search - Data Name Pop-up)

```

File View Search Logic Task List Options Help
-
C                               Search - Data Name                               U
*                               Type a data name and select search options. Then press Enter. For  R
*                               a selection list, enter a pattern (e.g. ABC*) in the name area.      **
0
0 Data name _____
0
0 References                    Indirect impact                    Size change
0 1 1. All                      1 1. None                    levels . . . ---
0                               2. Of size change
0                               3. Of value change
0                               4. Mods
0
0 Direction                    Options                    Action
0 1 1. All                      - No data aliasing          1 1. Find
0                               - IN-clause...              2. Highlight
0                               3. Previous                  3. Scroll
0                               4. First                     4. Print
0                               5. Last                      5. Punch
0                               6. Exclude
0
000018 *                               IN ITS SKELETAL FORM, THIS EXIT COPIES THE INPUT

```

Entry Fields. Text entry fields are used to enter textual information, such as a dataname, line range, number of levels, etc. The length of each text entry field is designated by an underline. In the [Figure 3](#), Data Name is an example of a text entry field.

Selection Fields. Numbered selection fields allow you to select one item from a list. Type the number of the desired selection in the field input area to the left of the first numbered item.

Select unnumbered selection fields by entering a slash (/) in the field input area. To unselect a selection field, enter a blank space to remove the slash.

The field input area is designated by an underline. When a pop-up first displays, each field contains its default value. On some pop-ups, if you change the default Insight retains the new value for subsequent sessions.

In [Figure 3 on page 9](#), Direction is an example of a numbered selection field. No Data Aliasing is an example of an unnumbered selection field.

Note:

Use the END command or PF03/15 to exit from a pop-up without processing it.

Screens

A screen is a full-width display of information, containing a full action bar as the first line. Insight screens are modeled after the TSO/ISPF/PDF edit screen, to which CUA features have been added.

[Figure 4](#) shows the Source View screen. Each field or area is described below.

Figure 4 • Insight Screen Sample

```
File View Search Logic Task List Options Help
-----
Source View                               Program: UIAIDEMO
Command ==>                               Scroll ==> PAGE
ASG1679I 3 TARGET STATEMENTS FOUND
-----
000283  CALL 'UIAIDEM1' USING MASTER-IN      282 LINES NOT DISPLAYED
000284                MASTER-END-OF-FILE    TARGET
000285                MASTER-REPORT-DATE.
-----
000353  MOVE LOAN-START-DATE TO DET-START-DATE.  67 LINES NOT DISPLAYED
000354  WRITE MAST-RPT FROM DETAIL-LINE1.      I/TGT-U:
-----
000405  READ MASTERIN                        50 LINES NOT DISPLAYED
-----
***** BOTTOM OF DATA *****
```

(A) The action bar displays at the top of every screen. See ["The Action Bar" on page 6](#) for more information.

(B) The name of the screen.

(C) The name of the active program. If there is no open program, this field is not displayed. This area of the screen is also used to temporarily display error and informational messages.

(D) The command input area. Insight primary commands are entered here.

(E) SCROLL specifies the number of screen lines or columns to scroll. This field is omitted from screens that can not be scrolled, such as the primary screen. You can enter these values:

Value	Description
1 through 9999	Specifies a number from 1 through 9999 to enter the number of lines or columns to scroll.
CSR	Specifies a scroll value of one page from the current cursor location.
HALF	Specifies a scroll value of a half page.
MAX	Specifies that the top, bottom, right, or left margin is the scroll value.
PAGE	Specifies a scroll value of one page.
DATA	Specifies a scroll value of one line less than a page.

(F) The long message field displays more descriptive error and informational messages. A long message displays if there is no corresponding short message, or if you enter HELP while a short message is shown.

Concepts

Insight is a powerful, easy-to-use system that helps programmers analyze and understand COBOL programs faster and more accurately. This section introduces some concepts and terms that are key to the operation of Insight.

Screen Messages

Insight messages are displayed in the long message area on the screen as described above. Short messages are displayed when available. Long messages are displayed if a short message does not exist or when you request help immediately following a displayed short message. You can enter the HELP command with the message number to display an explanation and any actions for the message.

Note: _____

See the Help Facility chapter in the *ASG-Insight Reference Guide* for additional information.

Analyze Job (Batch)

Insight analyzes COBOL II, and COBOL/370 programs using the Analyze job. The Analyze job produces a detailed program analysis that includes this information:

- Program organization and structure
- Data relationships
- Paths of modification and usage for all data fields
- Logic and control flow between modules, paragraphs, and statements

The analysis is saved in the AKR.

The Analyze job technology is similar to an optimizing compiler. The major difference is that you can use Insight to access the information saved in the AKR.

Note: _____

For more information, see the Analyze chapter in the *ASG-Insight Reference Guide*.

Application Knowledge Repository (AKR)

The AKR holds all the information the Analyze job produced about an application program. Access this information by using one of the View methods.

The AKR is a BDAM or VSAM file organization. A single AKR can store many programs, and you can define more than one AKR. To define new AKRs and to expand existing AKRs, use the online or Batch AKR utilities provided by Insight

Note: _____

For more information, see the AKR Utilities chapter in the *ASG-Insight Reference Guide*.

Sets

A set is a grouping of source lines in a COBOL source program. Sets consist of subsets, mark name sets, and line range sets. For more information on mark name sets and line range set, see ["Targets" on page 16](#). Many Insight commands use one or more of these sets or subsets. To concatenate sets, place a plus sign (+) between them.

Subsets

There are three types of subsets: COBOL language subsets, screen subsets, and tagged lines subsets. To show each subset, either select the Subsets action on the List pull-down or type LIST SUBSETS to display the List - COBOL Subset Names pop-up.

COBOL Subsets

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

For a complete list of the verbs included in each of the COBOL subset names, select the Subsets action on the List pull-down or type `LIST SUBSETS` to display the Subsets List screen. This table describes each of the COBOL subsets and the entities to which they correspond.

COBOL Subset	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 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	COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, -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 datanames and their aliases that are not referenced in the PROCEDURE DIVISION.

COBOL Subset	Description
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 as well as the entire DATA DIVISION.
DIRective	Statements that direct the compiler to take specific actions during compilation such as BASIS, EJECT, and TITLE.
DL/I DL/1	EXEC DL/I commands, such as ENTRY 'DLITCBL' or CALL 'CBLTDLI'.
DML	SQL Data Manipulation Language statements, such as SELECT, UPDATE, INSERT.
ENtry	The PROCEDURE DIVISION statement and all ENTRY statements.
EXIt_PGMExit	Statements containing a STOP RUN, GOBACK, or EXIT PROGRAM verb as well as 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.
INClude	COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, -INC 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.

COBOL Subset	Description
MATH	Statements containing ADD, SUBTRACT, MULTIPLY, DIVIDE, or COMPUTE verbs.
PERform	Statements containing the PERFORM, SORT, or MERGE verbs.
RETurn	Statements of a PERFORMed paragraph ranges that return control.
SORTMerge	Statements containing SORT, MERGE, or RELEASE verbs; paragraph or section names referred to in INPUT/OUTPUT PROCEDURES.
TESted	Identifies the lines of code that have been tested, based on information created and updated using TCA Reports.
UNTested	Identifies the lines of code that have not been tested, based on information created and updated using TCA Reports.

Screen Subsets

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

Highlighted | HI
 NONHighlighted | NHI
 Excluded | X
 NONExcluded | NX

Tag Subsets

Tag subsets display in the right-most columns of the screen to provide you with immediate information about the source code. Insight uses tags to mark the results of commands and actions from pull-downs or pop-ups, and to mark data items that are never referenced, tag statements that contain dead code, tag program exits, and tell you if PERFORMed paragraphs fallthrough or return. Specific tags are provided for TRACE option information. These tags help you to trace through the code quickly and easily.

These tags are used as subsets in commands that accept subsets as targets:

Tag	Description
TAGged TAGS	Refers to all lines that have information tags on them.
DECision	Refers to a line where TRACE is waiting for a decision.
OPTions	Refers to lines which are the optional choices for the decision point of the trace.
STArt	Refers to a line where the flow or trace started.
TARGet TGT	Refers to a line containing the target for FLOW or TRACE commands.

Targets

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

- Mark name
- Line range
- Data name
- Perfrange name
- Label name
- Pattern string
- Subset name
- Program name
- Paragraph name

Mark Name

A mark name is any user-generated path or set of lines named with the COPY, MARK, MERGE, or RENAME command, one of the Utility - Scratchpad pop-ups, a system-generated path created by the FLOW or TRACE command, or one of the Logic Order Search pop-ups. These paths are NETWORK, SUBNET_n, and TRACK.

Marks can be created, listed, renamed, copied, deleted, and merged with other Mark name sets. You may view all of the existing marks on the List - User Marks pop-up, or by typing LIST MARKS on the Source View screen, which displays the Marks List pop-up.

Perfrange Name

A perfrange name consists of all executable statements within a perform range, including all code that is executed by following that PERFORM. For example, the COBOL statement:

```
PERFORM PARAGRAPH-ABC THRU PARAGRAPH-XYZ
```

indicates that the range is comprised of the paragraphs beginning with PARAGRAPH-ABC and continuing through the end of paragraph PARAGRAPH-XYZ, including any paragraphs that are executed in between.

Note: _____

If a single paragraph is PERFORMed, see the paragraph name itself.

Program Name

A program name consists of all the code in all the divisions of the given program. For COBOL II Release 3 and COBOL/370, the program name may refer to a nested subprogram.

Subset Name

A subset name is one of the previously described COBOL language subsets, screen subsets, or tag subsets.

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.

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.

Dataname

A dataname can be any of these types:

- Elementary dataname
- File name
- Group name

- Table name
- Table element name
- Special name

Specify any legal COBOL reference for a data element as a dataname. If a variable is redefined to another name, Insight 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. Insight 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 datanames are specified by following them with a standard COBOL OF clause, followed by the group level dataname. For example:

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

Locate multiple datanames at the same time by concatenating the datanames with a plus sign (+) between them. For example:

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

Pattern String

A pattern string is a sequence of characters, surrounded by single or double quotes if it contains blanks. Specify strings of non-alphanumeric characters by the *X' string'*, *T' string'*, and *P' string'* operands. Further qualify the string using the WORD, PREFIX, or SUFFIX subordinate operands.

- *X' string'*
A hexadecimal string, enclosed in single or double quotes.
- *T' string'*
A text string, which disregards upper and lowercase, enclosed in single or double quotes.
- *P' string'*
A picture string, enclosed in single or double quotes. These are the valid picture strings:

P'=' Any character

P'-' Any nonblank character

P'.'	Any nondisplay character
P' #'	Any numeric character
P'-'	Any non-numeric character
P'@'	Any alphabetic character (upper or lowercase)
P'<'	Any lowercase alphabetic character
P'>'	Any uppercase alphabetic character
P'\$'	Any special character (not alphabetic or numeric)

- **WORD**
A string preceded and followed by any non-alphanumeric character (except hyphen).
- **PREFIX**
A word that begins with the specified string.
- **SUFFIX**
A word that ends with the specified string.

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.

3

Getting Started with Insight

This chapter discusses how to use Insight and contains these sections:

Topic	Page
Introduction	21
Initiating an Insight Session	22
Defining User Options	23
Using an AKR	30
Analyze Facility	33

Introduction

The steps necessary to invoke Insight vary by site. Check with the system administrator at your site for these details. Before following along with the sample Insight session presented, you may find it helpful to review Insight terminology, in the glossary of the *ASG-Insight Reference Guide*.

The first time you activate Insight, you must define some parameters. This chapter describes how to perform these functions:

- Invoke an Insight session
- Customize user options
- Allocate an AKR
- Analyze a program

To set online operation parameters

- 1 Select Options ► Product Parameters and press Enter. The Options - Product Parameters, shown in [Figure 8](#), displays.

Figure 8 • Options - Product Parameters Pop-up

```

- File View Search Logic Task List Options Help
-
C Command ==> Options - Product Parameters
-----
Type the desired parameters below to customize the operation of
the product. Then press Enter.
**
*
ALARM . . . . . YES (Yes or No)
Save Equates . . . . YES (Yes or No)
Save Marks . . . . . YES (Yes or No)
Cursor character . . % (token substitution character)

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

```

- 2 Type the appropriate information in the fields and press Enter.
- 3 Press PF03/PF15 to return to the previous screen.

Allocating the Log, List, Punch, and Work Files

Use the Product Allocations pop-up to set and/or verify Direct Access Storage Device (DASD) information for the Log, List, Punch, and Work files. Some options may be initially set to default values established during product installation by the systems programmer.

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

- 1 Select Options ▶ Product allocations and press Enter. The Options - Product Allocations pop-up, shown in [Figure 9](#), displays.

Figure 9 • Options - Product Allocations Pop-up with SMS

```

Command ==>> Options - Product Allocations
-----
Log file:
Management Class  MGMT1      Generic unit . . . SYSDA
Storage Class . . STOR1      Volume serial . . _____
List file:
Management Class  MGMT1      Generic unit . . . SYSDA
Storage Class . . STOR1      Volume serial . . _____
Punch file:
Management Class  MGMT1      Generic unit . . . SYSDA
Storage Class . . STOR1      Volume serial . . _____
Work file:
Management Class  MGMT1      Generic unit . . . SYSDA
Storage Class . . STOR1      Volume serial . . _____
Data Class . . . . DATA1    Space units . . . CYLS
Primary space . . 1
Secondary space   1
    
```

- 2 Type the appropriate information in the fields and press Enter.
- 3 Press PF03/PF15 to return to the previous screen.

Specifying Log/List/Punch Processing Options

Use the Options - Log/List/Punch Definition pop-up to define and set and/or verify processing values for the Insight Log, List, and Punch files. These files are used for system message logging, error handling, and holding the results of several Insight commands. Some options may be initially set to default values established by the systems programmer during product installation.

To specify the Log/List/Punch Definition options

- 1 Select Options ► Log/list/punch and press Enter. The Options - Log/List/Punch Definition pop-up, shown in [Figure 10](#), displays.

Figure 10 • 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      . . . . . K              PK              PK
Primary tracks      . . . . . 1              1              1
Secondary tracks    . . . . . 2              5              5
Lines per page      . . . . . 56             56             56
Sysout class        . . . . . *              *              *

Process options:  PK (print/keep), PD (print/delete), K, or D.

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

- 2 Type the appropriate information in the fields and press Enter.
- 3 Press PF03/PF15 to return to the previous screen.

Note:

When the PK or PD option is specified, a valid job card must be specified in the JOB STATEMENT INFORMATION field prior to processing any Log, List, or Punch files.

Fields

Field	Description
File Naming	Specifies whether the Log, List, and Punch files are named by system default (S) or user-defined (U) dataset names.
Prompt later for DSN	Enables you to define a custom name during processing. If you specify N, you must enter a dataset name in the corresponding Data set name field, and specify Overwrite or Append in the File Mode field. If you specify Y in the Prompt later for DSN field, Insight prompts you for the dataset name during file processing.

Field	Description
Log Data set name List Data set name Punch Data set name	Indicates the name of the dataset in which the Log, List, or Punch files will be saved during processing.
File Mode	Specifies whether Log, List, and Punch files will overwrite or be appended to the existing files during processing.

To customize the Log, List, or Punch dataset name

If you specify K or PK process option, you can customize the dataset where the log, list, or punch file is allocated. By default, Insight allocates this Log, List, and Punch file name:

USERID.yyyynnnn.VIAxxxxx

where:

yyy is the product ID.

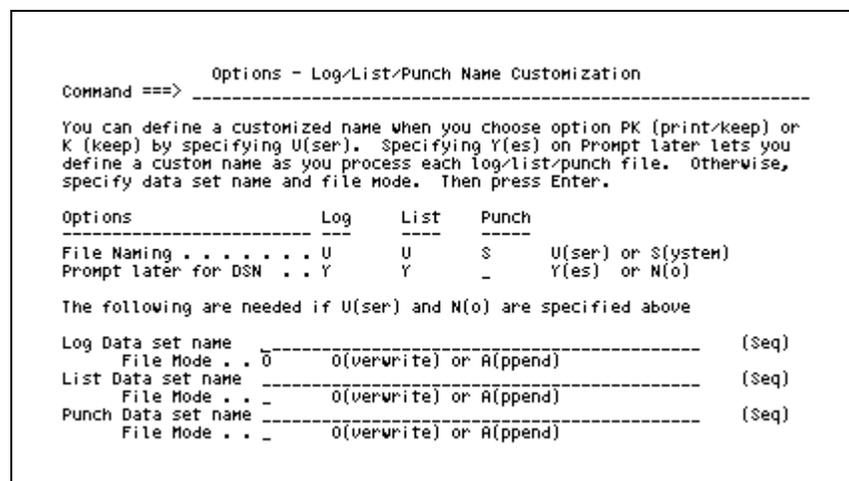
nnnn is a sequential number from 00001 to 99999.

xxxxx is LOG for Log, LIST for List, and PUNCH for Punch files.

If you have specified a TSO Prefix, the prefix will be appended to the beginning of the file name allocated for the Log, List, and Punch files.

- 1 From the Options - Log/List/Punch Definition pop-up, type 4 and press Enter. The Options - Log/List/Punch Name Customization pop-up, shown in [Figure 11](#), displays.

Figure 11 • Options - Log/List/Punch Name Customization Pop-up



- 2 Type U in the File Naming field for Log, List, and/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 Overwrite or Append in the File Mode field.

If you specify Y in the Prompt later for DSN field, Insight prompts you for the dataset name during file processing. For example, if you specify Yes in the Prompt later for DSN field for the Log file, the Log Name Customization pop-up shown in [Figure 12](#) will be displayed.

Figure 12 • Log Name Customization Pop-up

```

Options - Log Name Customization
Command ==> _____
-----
You can define a customized name when you choose option PK (print/keep) or
K (keep) by specifying U(ser). Specifying Y(es) on Prompt later lets you
define a custom name as you process each log file. Otherwise, specify
data set name and file mode. Then press Enter.

Options          Log
-----          ---
File Naming . . . . . S      U(ser) or S(ystem)
Prompt later for DSN . . . _  Y(es) or N(o)

The following are needed if U(ser) and N(o) are specified above

Log Data set name _____ (Seq)
File Mode . . . _      O(verwrite) or A(ppend)
    
```

Mapping PF Key Values

To set your PF keys

- 1 Select Options ► PF keys and press Enter. The Options - PF Key Definition pop-up, shown in [Figure 13](#), displays.

Figure 13 • PF Key Definition Pop-up

```

Options - PF Key (01-12) Definition
Command ==> _____
-----
Press Enter to process changes and/or to display alternate keys.
Press PF3/15 (END) to exit.

Number of PF keys: 24      Terminal type: 3276

PF01 HELP
PF02 SPLIT
PF03 END
PF04 RPREF
PF05 RFIND
PF06 FX %
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 BRANCH
PF11 BRANCH BACKUP
PF12 RECALL
    
```

- 2 Type the appropriate PF key values in the fields. PF keys 1 through 12 are displayed initially. To display PF keys 13 through 24, press Enter.
- 3 Press PF03/PF15 to return to the previous screen.

Using an AKR

Analyzed programs are stored in a BDAM or VSAM file organization in the AKR for Insight to use when testing. Define a AKR to be shared by all users, or define multiple AKRs for use by departments, groups, or individuals.

Online and batch utilities help you manage the AKR. The utilities provide these capabilities:

- Allocating or expanding the AKR
- Renaming and deleting programs
- Printing or punching AKR directory and program information

Note: _____

For more information about the Analyze options and how to analyze programs, see the Analyze chapter in the *ASG-Insight Reference Guide*.

Allocating an AKR

To allocate an AKR

- 1 Select File ► AKR utility bar and press Enter. The File - AKR Utility pop-up, shown in [Figure 14](#), displays.

Figure 14 • File - AKR Utility Pop-up

```
File - AKR Utility
Command ==> _____
Blank - Display member list      D - Delete member
A   - Allocate/expand AKR      R - Rename member

Application Knowledge Repository (AKR):
Data set name . . 'VIATEST_APPL.AKR'
Member . . . . . (if "R" or "D" selected)
New name . . . . . (if "R" selected)
Volume serial . . ----- (if not cataloged)
Password . . . . . (if password protected)
```

- 2 Specify the name of the AKR in the Data set name field.

- 3 Type A in the primary command input area and press Enter. The File - AKR Allocate/Expand pop-up, shown in [Figure 15](#), displays.

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 if SMS is turned on for your site.

Figure 15 • File - AKR Allocate/Expand Pop-up with SMS

```

File - AKR Allocate/Expand
Command ==> _____
S - Submit JCL      E - Edit JCL      C - Specify Catalog
Expand existing AKR . . . NO          (Yes or No)
AKR data set name . . . . 'USER.TEST.AKR'
Management Class . . . . _____ (Blank for default MGMTCLAS)
Storage Class . . . . _____ (Blank for default STORCLAS)
Volume . . . . . _____
Data Class . . . . . _____ (Blank for default DATACLAS)
Space units . . . . . RECORDS (Records, Tracks or Cylinders)
Space amount . . . . . 4000 (Total AKR size in above units)
Unique . . . . . YES (Enter NO for a VSAM data space)

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

```

- 4 Type the appropriate information in the required fields:
- Type NO in the Expand existing AKR field.
 - Verify the AKR name in AKR data set name field.
 - Specify a valid Job card in the Job statement information field.
 - Press Enter.
- 5 Type S in the primary command input area and press Enter to submit the JCL.

Or

Type E in the primary command input area and press Enter to edit the JCL. While in the editor, review the generated JCL, type the standard ISPF SUBMIT command in the primary command input area and press Enter. To return to the File - AKR Allocate/Expand pop-up, type END in the primary command input area or press PF03/PF15.

- 6 Verify the AKR allocation results by examining the job output. Sample job outputs are presented in ["Verifying AKR Allocation Results" on page 32](#).

Verifying AKR Allocation Results

After the AKR allocation Batch job has completed, review the job output to verify successful allocation and initialization. [Figure 16](#), [Figure 17](#), and [Figure 18](#) show output excerpts with messages that indicate successful AKR allocation and initialization.

Figure 16 • IDCAMS Utility Condition Code Message Output

```

IDCAMS  SYSTEM SERVICES                TIME: HH:MM:SS      MM/DD/YY      PAGE  1

  DEFINE CLUSTER -
    (NAME (USER.TEST.AKR) -
     CYLINDERS (5) -
     VOLUME (DISK01) -
     CONTROLINTERVALSIZE (4096) -
     NUMBERED -
     RECORDSIZE (4089 4089) -
     RECOVERY -
     UNIQUE -
     SHAREOPTIONS (3 3)) -
  DATA -
    (NAME (USER.TEST.AKR.DATA))
IDC0508I DATA ALLOCATION STATUS FOR VOLUME DISK01 IS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
    
```

Figure 17 • AKR Initialization Message Output

```

ASG-CENTER-OS (ESA)      AKR UTILITY LOG      DD-MMM-YY  HH:MM:SS  PAGE 1

  INIT DSNAME (USER.TEST.AKR)

  ASG1316I AKR "USER.TEST.AKR" INITIALIZED.

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

Figure 18 • AKR Utility Log Summary Message Output

```

ASG-CENTER-OS (ESA)      AKR UTILITY LOG - SUMMARY  DD-MMM-YY  HH:MM:SS  PAGE 2

  ASG1300I      1 AKR(S) INITIALIZED      0 FAILED.

  ASG1315I *** END OF SUMMARY REPORT ***
    
```

Analyze Facility

The Program Analyze facility extracts knowledge about the program and populates the AKR with this information.

Program Analyze Requirements

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

- The COBOL language as specified in the IBM OS/VS COBOL, COBOL II, and COBOL/370 Language Reference Guides.
- A program which can be compiled without errors by the IBM OS/VS COBOL, COBOL II, or COBOL/370 compiler.
 - OS/VS COBOL programs that receive conditional (C), error (E), or disaster (D) messages from the IBM compiler cannot be successfully analyzed.
 - COBOL II and COBOL/370 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 includes these items:

- JCL to compile the program. The JCL should be the complete JCL used to compile the program and should contain the appropriate steps to complete these tasks:
 - Fetch the source from the source manager (such as Librarian or Panvalet)
 - Execute any preprocessors
 - Invoke the compiler
- Program Analyze features. These features indicate the type of analysis to be performed.
- An initialized AKR to receive Program Analyze output.
- Program Analyze options.

Analyze Submit Facility

Use the Analyze Submit pop-up to specify the necessary information and to submit a program to be analyzed through Insight.

Note: _____

If the compile JCL resides in a source manager such as Librarian or Panvalet, you cannot use this method.

To display the Analyze Submit pop-up

- 1 Select File ► Analyze and press Enter. The File - Analyze Submit pop-up, shown in [Figure 19](#), displays.

Figure 19 • Analyze Submit Pop-up

```

                                     File - Analyze Submit
Command ==> _____
                E - Edit JCL                      S - Submit JCL

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

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

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 only)

```

Note: _____
If you started your Insight session using the ESW Primary screen, the product names listed under the Analyze features (Y/N) field may differ from the names shown in this example.

- 2 Type the appropriate information in the required fields, then press Enter:
 - a Specify the PDS member or sequential dataset containing the compile/link JCL in the Data set name field.
 - b Specify the analyze feature by typing Y in the desired Analyze features (Y/N) fields.

Note: _____
Other analyze features are shown on this screen when additional ESW products are installed at your site. For more information on analyze options, see the Analyze chapter in the *ASG-Insight Reference Guide*.

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

- 3 Type `S` in the primary command input area and press Enter to submit the JCL.

Or

Type `E` in the primary command input area and press Enter to edit the JCL. While in the editor, review the generated JCL, type the standard TSO `SUBMIT` command in the primary command input area and press Enter. To return to the File - Analyze Submit pop-up, either type `END` or press PF03/PF15.

- 4 Verify the analyze results by examining the job output. Sample job outputs are presented in ["Verifying Analyze Results" on page 35](#).

Verifying Analyze Results

After the Compile - Analyze Batch job completes, review the job output to verify the results. Check the output for these items:

- Acceptable compiler results.
- Messages indicating the program has been successfully analyzed and stored on the AKR.
- Acceptable linkage editor results, if applicable.

Storage on the AKR does not occur if the program does not compile and analyze successfully.

Note: _____

Insight does not require a compile or link for you to use the Source View facility.

[Figure 20](#) shows an output excerpt that indicates a successful analyze.

Figure 20 • Output Excerpt Indicating Successful Analyze

```
ASG-CENTER-OS (ESA)          PROGRAM: VIAIDEMO          DD-MMM-YY  HH:MM:SS PAGE 1

LINE  ERROR MESSAGE

ASG0237I  166 SYMBOLS PROCESSED.
ASG0238I  165 SYMBOLS MATCHED.
ASG0240I  207 VERBS PROCESSED.
ASG0057W SOURCE PROGRAM CONTAINS DEAD CODE. A COMPLETE LIST MAY BE OBTAINED ONLINE.
(A)© ASG0248I PROGRAM 'VIAIDEMO' WAS STORED IN AKR 'USER.TEST.AKR'.

(B)
_
DIAGNOSTICS LINES:  1 TOTAL - 1 WARNINGS, 0 CONDITIONALS, 0 ERRORS, 0 DISASTERS
SOURCE LINES: 468 TOTAL - 122 DATA DIVISION STATEMENTS, 228 PROCEDURE DIVISION STMTS
PARAMETERS PASSED:  NOCOBOLII,LANGLVL(2),FEATURES=(S,X)
OPTIONS IN EFFECT:  BUFMAXK=2000K, FEATURES=(ASG-Insight), FLAG(W), LANGLVL(2),
LINECNT=60, NORECUR, NOSEQ, NOSOURCE, SPACE1
ENTRY POINTS:      VIAIDEMO
EXTERNAL CALLS:    VIAIDEM1,  ILBOABN0
                   NORETURN:  ILBOABN0
END OF PROCESSING: DD-MMM-YY  HH:MM:SS
```

(A) Message indicated program was stored on the AKR.

(B) Any Program Diagnostics is shown in this area.

4

Program Understanding and Maintenance Techniques

This chapter discusses techniques for using Insight to gain program understanding and perform maintenance, and contains these sections:

Topic	Page
Introduction	38
Insight Program Analysis Methodology	38
Starting an Insight Session	40
Using Automated Tasks	41
Summary: Using Automated Tasks	54
Using the Insight Program Understanding Checklist	54
Getting a Program Overview	55
Getting a Paragraph Overview	73
Summary: Getting a Program Overview	79
Identifying Data Item Usage Cross-References	80
Summary: Identifying Data Item Usage Cross-References	96
Conducting an Analysis of Program Logic	96
Summary: Conducting an Analysis of Program Logic	116
Applying Quality Assurance and Standards	116
Summary: Applying Quality Assurance Standards	132

Introduction

The Source View facility of Insight provides you with powerful program analysis tools within an interactive environment. You can quickly and easily navigate through source code and focus on the information you need in order to understand your program and save time when performing maintenance or enhancement tasks. In addition, Insight automates some common maintenance and enhancement tasks that guide you through each step of the process from analysis to source modification.

This chapter provides various programming tasks and problems, along with step-by-step instructions to help you manage them. The results of the steps, when performed on the ESW demonstration programs, are also included. These steps apply for any program that you need to understand, maintain, or enhance.

The examples in this chapter use the demonstration programs VIAIDEMO and VIAIDEM1. Use VIAIDEM3 for COBOL II Release 3 examples.

Insight Program Analysis Methodology

Program analysis is an important first step in the ESW Existing Systems Cycle (ES/Cycle). This section describes how Insight program analysis techniques are applied using a basic program analysis methodology.

Using Automated Tasks

Insight has an automated Task function you can use as a quick and easy way to solve problems involving program understanding and maintenance. From here, you can modify incorrect report and file output, enhance a data computation, debug a data-related abend, and then make changes in your editor without ever leaving your Insight session. Using automated tasks helps you save time and increase confidence in your results when performing maintenance and enhancement tasks.

Generating a Program Overview

Program Analysis generally falls into the categories shown in [Figure 20 on page 36](#). It is important to gain an overview of program structure and coding techniques to examine logical program units and all CALLing and CALLEd relationships between the units. Logical units consist of PERFORM ranges, including GOTO statements, or CALLEd programs.

Another important aspect of a program overview includes the external dimension, often determined by investigating the source code listings and program documentation. This information is crucial to your understanding of how data is passed between CALLEd programs and how the values are read in and output.

Generating a Paragraph Overview

Program paragraph cross-referencing gives you the ability to grasp how paragraphs transfer control. The interactive capability of Insight permits you to automate your process of determining how paragraphs are referenced in a program, thus eliminating the time-consuming task of manually browsing listings.

Identifying Data Item Usage

Identifying data item cross-referencing is important to understanding how and where fields are used, modified, moved to and from, and where definitions or picture clauses are located in a program. This process eliminates the need to manually examine a program for alias names, tag, and highlight them. Insight also permits you to specify the indirect references to data items by tracking their use across intermediate fields. This feature permits you to electronically highlight how specific data is manipulated in the program and the location of its final destination, a process commonly referred to as the ripple effect.

Analyzing the Program Logic

Another important part of any analysis is understanding program logical execution. The extended logic analysis component of Insight consists of these three components:

- The Program Analyzer - a Batch process that reads your COBOL source code and copybooks and gathers information on execution paths, data relationships, and program organization.
- The AKR - a BDAM or VSAM file organization that contains a copy of the COBOL source code and all of the information gathered by the Program Analyzer.
- Source View - an Insight display of a COBOL program that resides in the AKR.

Logic analysis allows you to follow program logic interactively without data or executing the program. This allows you to investigate all of the statements in all of the execution paths to all of the targets, or to a specific target and automatically record these results on an electronic scratchpad for later reference.

Applying Quality Assurance Standards

Quality assurance standards are often measured during code inspections. The intelligent search capability of Insight allows you to quickly check a COBOL program. For example, you can use Insight to quickly and easily check for these:

- Proper indentation and alignment of IF and ELSE statements
- ELSE statements on a line by themselves
- Unique sequential naming and numbering of sections and paragraphs

- PERFORM statements coded with the THRU clause
- Dead code and dead data
- Acceptable programming expressions

Starting an Insight Session

To start an Insight analysis session

- 1 Log on to a terminal and open Insight.
- 2 Select File ► Open and press Enter. The File - Open Program pop-up, shown in [Figure 21](#), displays.

Figure 21 • File - Open Program Pop-up

```
File - Open Program Request
Type AKR information and program name. Then press Enter.
Application Knowledge Repository (AKR):
  Data set name . . 'VIAIEST_AAPPL.AKR'
  Program name . . _____ (blank for selection list)
  Volume serial . . _____ (if not cataloged)
  Password . . . . _____ (if password protected)
```

- 3 Type the AKR name in the Data set name field.
- 4 Type VIAIDEMO in the Program name field.
- 5 Type the required volume serial and password, if applicable, and press Enter.

Note: _____

The logon procedure and AKR information is unique to your programming environment. Contact your systems programmer to verify the logon and AKR information for your location.

Using Automated Tasks

Insight automates some common program maintenance and enhancement tasks. These automated tools guide you through selection pop-ups that isolate statements and datanames that pertain to a particular maintenance or enhancement project. When completed, Insight displays the source lines that modify the selected dataname in every execution path that leads to the selected statement. These statements help you understand how datanames are built prior to the given source statement.

You have the option of either viewing the results in Source View or in an edit session. These automated tasks provide you with the ability to perform these tasks quickly:

- Modify incorrect report or file output
- Enhance data computations
- Debug data-related program abends

In this section, you use Insight to determine these items:

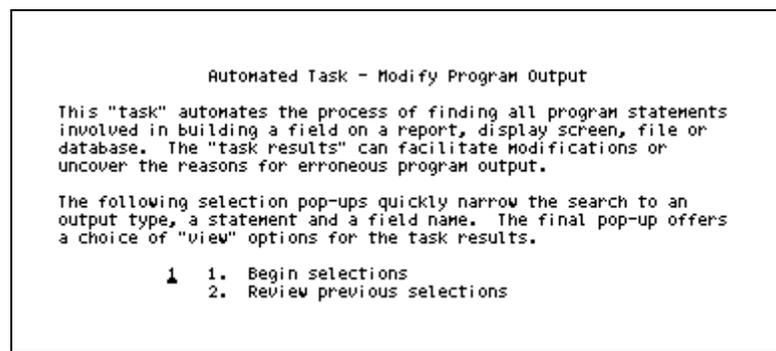
- The source of incorrect output in this program
- How to isolate the statements that are involved in a computation to make modifications
- How to debug a data-related abend in a program using Insight
- What to do to repair or enhance this program, based upon the results of the analysis
- The source of incorrect output in this program

Assume that your program is producing bad output on a report. The report output contains incorrect information in the LOAN-AMT field. The objective for this example is to determine which statements are involved in the initialization of the LOAN-AMT column of the report.

To find these statements using the automated Task function

- 1 Select Task ► Modify output and press Enter. The Automated Task - Modify Program Output pop-up, shown in [Figure 22](#), displays.

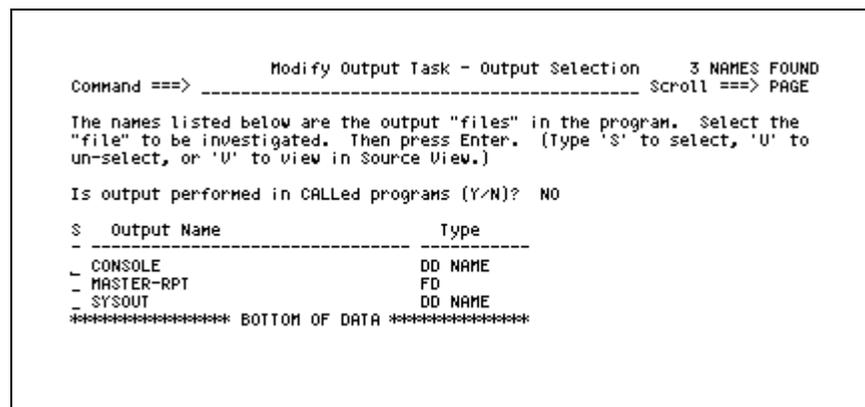
Figure 22 • Automated Task - Modify Program Output Pop-up



- 2 Specify the Begin selections option and press Enter.

The Modify Output Task - Output Selection pop-up, shown in [Figure 23](#), provides a list of all files (i.e., FDs, DDs, SDs, and RDs) to which VIAIDEMO provides output. The screen also lists CALLED programs which may perform output.

Figure 23 • Modify Output Task - Output Selection Pop-up



Assume that you have determined that the File Definition (FD), MASTER-RPT, contains incorrect output.

- 3 To examine the FD for modification of the output in a detail line, type S to the left of MASTER-RPT on the pop-up and press Enter. The Modify Output Task - Statement Selection pop-up, shown in [Figure 24](#), displays.

Figure 24 • Modify Output Task - Statement Selection Pop-up

```

                                Modify Output Task - Statement Selection
Command ==> ----- Scroll ==> PAGE

The statements listed below "write" to the selected "file". Select the
statement to be investigated. Then press Enter. (Type 'S' to select, 'U'
to un-select, or 'V' to view in Source View.)

Selected output: MASTER-RPT

$ Line   Output statement
-----
- 000354 WRITE MAST-RPT FROM DETAIL-LINE1.
- 000359 WRITE MAST-RPT FROM DETAIL-LINE2.
- 000367 WRITE MAST-RPT FROM DETAIL-LINE3.
- 000374 WRITE MAST-RPT FROM DETAIL-LINE4.
- 000426 WRITE MAST-RPT FROM SUB-PRINT.
- 000436 WRITE MAST-RPT FROM RPT-HDG-LINE1.
- 000437 WRITE MAST-RPT FROM RPT-HDG-LINE2.
- 000456 WRITE MAST-RPT FROM TOTAL-PRT
***** BOTTOM OF DATA *****

```

- 4 Determine the WRITE statement that is producing the incorrect output. If it is not clear which statement is producing the error, you may choose one of these options:
 - a Type V on the line of the suspected WRITE statement and press Enter to view the source code around that statement.
 - b Type S on the line of a suspected WRITE statement and press Enter to view the data (definitions) that are written by this statement.

Assume that statement 000354, WRITE MAST-RPT FROM DETAIL-LINE1, is producing the incorrect output.

- 5 Type S next to this output statement and press Enter. A list of all of the individual fields contained in DETAIL-LINE1 displays, as shown in [Figure 25](#).

Figure 25 • Modify Output Task - Data Name Selection Pop-up

```

                                Modify Output Task - Data Name Selection   5 NAMES FOUND
Command ==> -----> Scroll ==> PAGE

The names listed below are data items that are output by the "write"
statement selected. Select the item to be investigated. Then press Enter.
(Type 'S' to select, 'U' to un-select, or 'V' to view in Source View.)

Selected statement: 000354 WRITE MAST-RPT FROM DETAIL-LINE1.

$  Data name
-----
  01 DETAIL-LINE1
   05 DET-CC
  -   05 DET-NUMBER
  -   05 DET-LOAN-AMT
  -   05 DET-START-DATE
***** BOTTOM OF DATA *****

```

Assume that the field, DET-START-DATE (of DETAIL-LINE1) is being truncated on MASTER-REPORT.

- 6 Type S next to the DET-START-DATE dataname and press Enter. The Automated Task - Result Options pop-up, shown in [Figure 26](#), displays providing a choice of ways to view the results identified by Insight.

Figure 26 • Automated Task - Result Options Pop-up

```

                                Automated Task - Result Options

The selections are complete for this "task". All program statements that
relate to these selections are available for viewing. Select the desired
option. Then press Enter.

  1. 1. Source View - See the task results in Source View**
     2. Editor      - Modify the results in your editor
     3. Review      - Return to first selection pop-up

** Source View tags:
   START - Selected starting statement
   IT    - Intermediate target
   TARGET - Statement that initializes a component variable
   TARGET-U - Target with uninitialized variable
   TARGET-U? - Target with possible uninitialized variable

```

- 7 Select the Source View option and press Enter to display the Results in Source View pop-up, shown in [Figure 27](#).

Note:

Notice the description of Source View tags that appear at the bottom of the Automated Task - Result Options pop-up. These tags accompany the task results in Source View.

Figure 27 • Results in Source View

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> PAGE
ASG1679I 3 TARGET STATEMENTS FOUND
-----
000283  CALL 'VIAIDEM1' USING MASTER-IN
000284  MASTER-END-OF-FILE
000285  MASTER-REPORT-DATE.
-----
67 LINES NOT DISPLAYED
000353  MOVE LOAN-START-DATE TO DET-START-DATE.
000354  WRITE MAST-RPT FROM DETAIL-LINE1.
-----
50 LINES NOT DISPLAYED
000405  READ MASTERIN
-----
93 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

This result permits you to see the source statements that are involved in initializing the data field, DET-START-DATE, in Source View. The statement, MOVE LOAN-START-DATE TO DET-START-DATE, is tagged I/TGT-U?. The I represents an intermediate target used in the building of DET-START-DATE. The tag, TGT-U?, identifies a possibly uninitialized variable.

After reviewing the statements displayed by the automated task, it is often necessary to make changes to one or more of the displayed source statements in an edit session.

To make changes to displayed source statements

Note:

This example is demonstrated with the assumption that your installation's standard Editor is not SmartEdit.

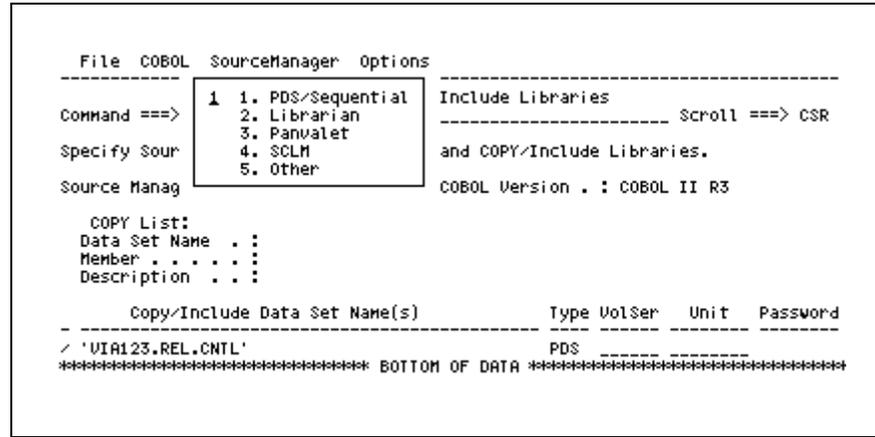
- 1 Select Task ▶ See task results in editor and press Enter. The Options - COPY/Include Libraries screen displays.

Note:

If SmartEdit is installed as your editor, proceed to [step 4](#) to display the Edit - Entry Panel shown on [Figure 29 on page 46](#).

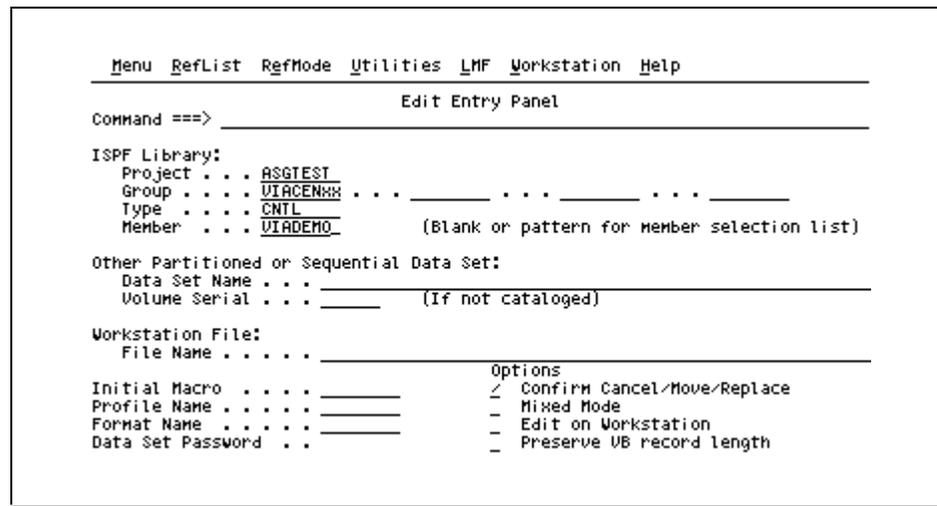
- 2 Select Source Manager and press Enter. The Edit Options pull-down, shown in [Figure 28](#), displays.

Figure 28 • Edit Options Pull-down



- 3 Select a source manager option (i.e., Librarian, Panvalet) for your environment and press Enter.
- 4 Press Enter again. The Edit - Entry Panel, shown in [Figure 29](#), displays.

Figure 29 • Edit - Entry Panel



- 5 On the Edit - Entry Panel, verify the ISPF Library Project, Group, and Type.

- 6 Type VIAIDEMO in the Member field and press Enter to display the source code in your editor. [Figure 30](#) displays VIAIDEMO using SmartEdit.

Note:

The source displayed in the editor is the source that resides in your installation's source manager, and not the source that resides in the AKR. You cannot use the Editor to edit the source code that is in the AKR.

Figure 30 • VIAIDEMO Program Display in SmartEdit

```

ASG-Insight Rx.x -- ASG.VIACENxx.CNTL(VIAIDEMO) - 40.00 -- COLUMNS 007 072
COMMAND ==> SCROLL ==> CSR
ASG1673I ENTER THE DISPLAY COMMAND TO VIEW THE LINES FOUND.
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. VIAIDEMO.
000300 AUTHOR. WRITTEN BY ASG AT LANGLVL 2.
000400 ENVIRONMENT DIVISION.
000500 CONFIGURATION SECTION.
000600 SOURCE-COMPUTER. IBM-370.
000700 OBJECT-COMPUTER. IBM-370.
000800 INPUT-OUTPUT SECTION.
000900 FILE-CONTROL.
001000 SELECT MASTERIN ASSIGN TO S-MASTERIN.
001100 SELECT MASTER-RPT ASSIGN TO S-MREPORT.
001200 DATA DIVISION.
001300 FILE SECTION.
001400 FD MASTERIN
001500 RECORDING MODE IS F
001600 BLOCK CONTAINS 0 RECORDS
001700 LABEL RECORDS ARE STANDARD.
001800
001900 COPY VIAIMAST.
002000
002100 FD MASTER-RPT
    
```

- 7 Type DISPLAY in the primary command input area and press Enter to display the highlighted results of your automated Task analysis in Insight, as shown in [Figure 31](#).

Figure 31 • Highlighted Results of Insight Analysis in SmartEdit

```

ASG-Insight Rx.x -- ASG.VIACENxx.CNTL(VIAIDEMO) - 01.00 -- COLUMNS 001 072
COMMAND ==> SCROLL ==> CSR
ASG1675I 3 LINES FOUND.
----- 326 LINE(S) NOT DISPLAYED
HI==> 032700 MOVE LOAN-START-DATE TO DET-START-DATE.
HI==> 032800 WRITE MAST-RPT FROM DETAIL-LINE1.
----- 50 LINE(S) NOT DISPLAYED
HI==> 037900 READ MASTERIN
----- 93 LINE(S) NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
    
```

You can now make any corrections or modifications to output statements in the VIAIDEMO MASTER-RPT.

- 8 To return to your Source View session, press PF03/PF15 until you reach your Source View screen.

Note: _____

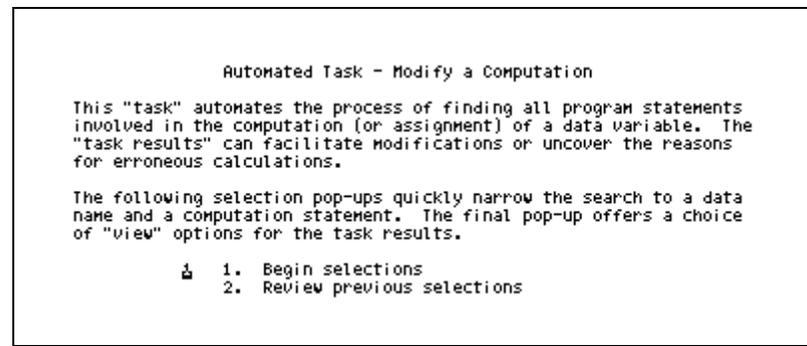
If you make changes in the editor and want to see those changes in Source View, you must reanalyze the program.

To isolate the statements that are involved in a computation

Assume that the computation of the value, ZIP-LOAN-AMT, in the program, VIAIDEMO, needs to be modified. Using Insight, you can quickly determine which statements are involved in the calculation.

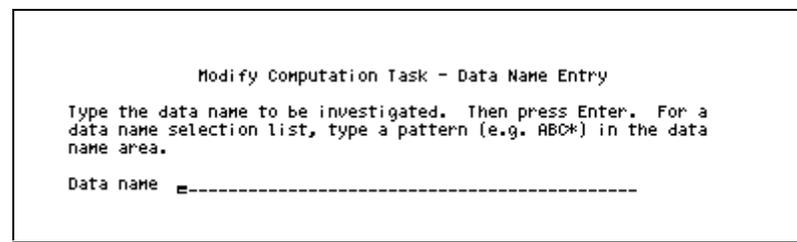
- 1 Select Task ► Modify computation and press Enter. The Automated Task - Modify a Computation pop-up, shown in [Figure 32](#), displays.

Figure 32 • Automated Task - Modify a Computation Pop-up



- 2 Select the Begin selections and press Enter. The Modify Computation Task - Data Name Entry pop-up, shown in [Figure 33](#), displays.

Figure 33 • Automated Task - Modify Computation Task Pop-up



If you decide you want to change the way the data value, ZIP-LOAN-AMT, is computed in VIAIDEMO, you can locate all of the modifications to the data item in the entire program. This gives you information on all of the possible places that the data is modified in logical execution order in the source code.

To review how ZIP-LOAN-AMT is used in the source code

Note:

You can use the wild card symbol, an asterisk (*), to designate a partial dataname. For this example, assume that you are not quite certain of the exact spelling of the dataname ZIP-LOAN-AMT.

- 1 Specify the Data name, ZIP*, and press Enter. The Modify Computation Task - Data Name Selection pop-up, shown in [Figure 34](#), displays.

Figure 34 • Modify Computation Task - Data Name Selection Pop-up

```

                                Modify Computation Task - Data Name Selection
Command ==> ----- $Scroll ==> PAGE

The data names listed below are assigned value in this program.  Select the
name to be investigated.  Then press Enter.  (Type 'S' to select, 'U' to
un-select, or 'V' to view in Source View.)

Pattern ZIP**

$  Data name                Qualification
-----
= ZIP                       01 MASTER-IN
- ZIP-CODE                   05 ZIP, 01 MASTER-IN
- ZIP-LOAN-AMT               01 ZIP-TOTALS
- ZIP-LOAN-CNT               01 ZIP-TOTALS
- ZIP-TOTALS
- ZIP-YTD-INT                01 ZIP-TOTALS
***** BOTTOM OF DATA *****

```

- 2 Type S to the left of the dataname, ZIP-LOAN-AMT, and press Enter. The Modify Computation Task - Statement Selection pop-up, shown in [Figure 35](#), displays.

Note:

The Data Name Selection pop-up does not display if you enter the dataname ZIP-LOAN-AMT on the Automated Task - Modify Computation Task pop-up.

Figure 35 • Modify Computation Task - Statement Selection Pop-up

```

                                Modify Computation Task - Statement Selection
Command ==> ----- Scroll ==> PAGE

The statements listed below assign value to the selected data name.
Select the statement to be investigated. Then press Enter. (Type 's'
to select, 'u' to un-select, or 'v' to view in Source View.)

Selected name: ZIP-LOAN-AMT

$ Line      Computation/assignment statement
-----
_ 000386 COMPUTE ZIP-LOAN-AMT = { ZIP-LOAN-AMT + DET-LOAN-AMT }.
_ 000392 COMPUTE ZIP-LOAN-AMT = { ZIP-LOAN-AMT + DET-LOAN-AMT }.
***** BOTTOM OF DATA *****
```

The Modify Computation Task - Statement Selection pop-up displays every statement in the program that computes ZIP-LOAN-AMT. In most cases, several other statements initialize the data fields that are used in the computation.

- 3 Select a statement and press Enter. The Automated Task - Result Options pop-up, shown in [Figure 36](#), displays.

Figure 36 • Automated Task - Result Options Pop-up

```

                                Automated Task - Result Options

The selections are complete for this "task". All program statements that
relate to these selections are available for viewing. Select the desired
option. Then press Enter.

  1. 1. Source View - See the task results in Source View**
     2. Editor      - Modify the results in your editor
     3. Review      - Return to first selection pop-up

** Source View tags:
   START - Selected starting statement
   IT    - Intermediate target
   TARGET - Statement that initializes a component variable
   TARGET-U - Target with uninitialized variable
   TARGET-U? - Target with possible uninitialized variable
```

- 4 Select Source View and press Enter to display a screen similar to the Source View screen shown in [Figure 37](#).

Figure 37 • Statement in VIAIDEMO that Compute ZIP-LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> PAGE
ASG1679I 4 TARGET STATEMENTS FOUND
-----
000283  CALL 'VIAIDEM1' USING MASTER-IN          282 LINES NOT DISPLAYED
000284          MASTER-END-OF-FILE          TARGET
000285          MASTER-REPORT-DATE.
-----
000352  MOVE LOAN-AMT TO DET-LOAN-AMT.          66 LINES NOT DISPLAYED
-----
000386  COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ).  33 LINES NOT DISPLAYED
-----
000392  COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ).  5 LINES NOT DISPLAYED
-----
000405  READ MASTERIN                          12 LINES NOT DISPLAYED
-----
***** ***** BOTTOM OF DATA *****

```

As described in the previous example using automated tasks to modify output, you can choose now to view the results in an edit session.

Debugging a Data-related Abend

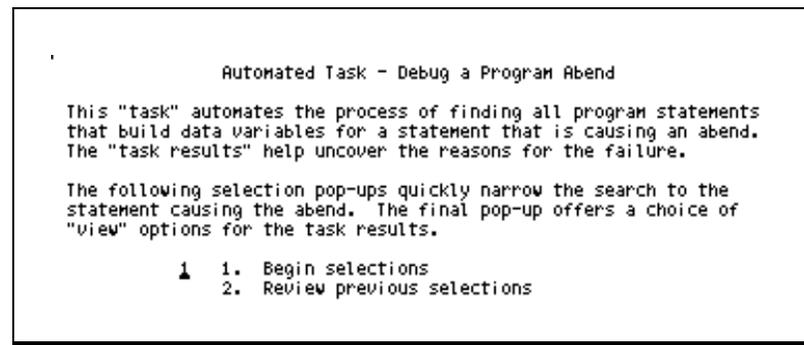
Many program abends that occur are related to data assignment and computation errors. Insight automates your task of finding every source statement that builds datanames found on an abending program statement. For example, assume that you have examined a program dump to determine the range of lines where the suspected abend occurred in your program. You would perform these steps:

- Examine the range in the source code.
- Find the source of the bad data value in a CALLED program, READ statement, or uninitialized value.

To determine the source of the bad value causing the program to abend

- 1 Select Task ▶ Debug abend and press Enter. The Automated Task - Debug a Program Abend pop-up, shown in [Figure 38](#) displays.

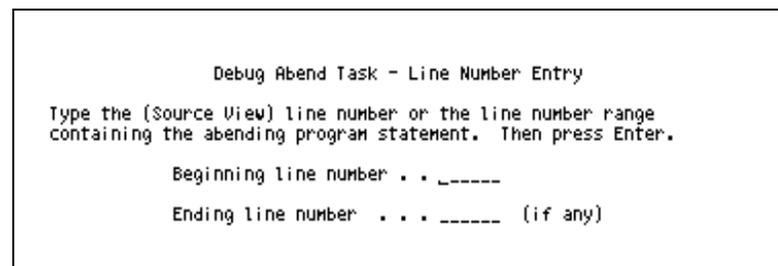
Figure 38 • Automated Task - Debug a Program Abend Pop-up



- 2 Select ▶ 1. Begin selections and press Enter. The Debug Abend Task - Line Number Entry pop-up, shown in [Figure 39](#), displays.

To use this technique, you must first determine either the line number or the line number range of the abending statement.

Figure 39 • Debug Abend Task - Line Number Entry Pop-up



- Specify the Beginning line number (450) and the Ending line number (460) on the pop-up and press Enter. The Debug Abend Task - Statement Selection pop-up, shown in [Figure 40](#), displays.

Figure 40 • Debug Abend Task - Statement Selection Pop-up

```

                                Debug Abend Task - Statement Selection
Command ==> _____ Scroll ==> PAGE

The statements listed below are in the selected range and reference one or
more data variables. Select the statement to be investigated. Then press
Enter. (Type 'S' to select, 'U' to un-select, or 'V' to View in Source
View.)

Selected lines: 450 - 460

$ Line   Statements containing variables
-----
- 000452 MOVE ' ' TO FIRST-TIME
- 000453 MOVE TOTAL-AMT TO PRT-TOTAL-AMT
- 000454 MOVE TOTAL-CNT TO PRT-TOTAL-CNT
- 000455 MOVE TOTAL-YTD-INT TO PRT-TOTAL-YTD-INT
- 000456 WRITE MAST-RPT FROM TOTAL-PRT
- 000457 MOVE 'Y' TO MASTER-END-OF-FILE.
***** BOTTOM OF DATA *****

```

- Type S next to a statement (such as line 000453, MOVE TOTAL-YTD-INT TO PRT-TOTAL-YTD-INT), where the suspected program abend occurred on the pop-up and press Enter. The Automated Task - Result Options pop-up, shown in [Figure 41](#), displays.

Figure 41 • Automated Task - Result Options Pop-up

```

                                Automated Task - Result Options

The selections are complete for this "task". All program statements that
relate to these selections are available for viewing. Select the desired
option. Then press Enter.

  1. Source View - See the task results in Source View**
  2. Editor      - Modify the results in your editor
  3. Review     - Return to first selection pop-up

** Source View tags:
START - Selected starting statement
IT    - Intermediate target
TARGET - Statement that initializes a component variable
TARGET-U - Target with uninitialized variable
TARGET-U? - Target with possible uninitialized variable

```

- 5 Select the Source View option and press Enter. The Source View screen, shown in [Figure 42](#), displays.

Figure 42 • Selected Statements in Source View

```

File View Search Logic Task List Options Help
-----
Source View                                Program: VIAIDEMO
Command ==>                               Scroll ==> CSR
ASG1679I 3 TARGET STATEMENTS FOUND
-----
000388      COMPUTE TOTAL-YTD-INT = ( TOTAL-YTD-INT          I/TGT-U
000389      + YEAR-TO-DATE-INTEREST ).
-----
000403      READ MASTERIN                                TARGET
-----
000453      MOVE TOTAL-YTD-INT TO PRT-TOTAL-YTD-INT        S/TGT-U?
-----
***** ***** BOTTOM OF DATA *****

```

Notice that these results reveal that Insight found three targets. On line 388, note the tag, I/TGT-U, which indicates that the COMPUTE statement is an intermediate target with an uninitialized variable. Also, the MOVE statement on line 453 shows a target with a possible uninitialized variable, designated by the tag, TGT-U?.

- 6 When you have completed your modifications to the program, press PF03/PF15 to return to your Source View screen.

Summary: Using Automated Tasks

- Using Insight automated Tasks, you can quickly examine and modify incorrect output, modify a data computation, or debug a data-related abend.
- Insight permits you to view the results of your investigation in the context of the program.
- Using automated Task features, you can also display the results of your analysis and make changes in your editor, without having to leave your Insight session.

Using the Insight Program Understanding Checklist

To develop a complete understanding of a COBOL program and to use this information to make enhancements, use the Insight Program Understanding Checklist. The examples in this section describe how to apply the methodology to perform program analysis and maintenance.

Getting a Program Overview

VIAIDEMO is a program provided with Insight that demonstrates the features and capabilities of the product. This program is designed to resemble a typical loan processing application. Since you are unfamiliar with this source code, begin by examining program structure and processing requirements. Assume that you choose to view only the results of your inquiry and exclude all other lines from the display.

To exclude all unwanted lines from the display

- 1 Select Options ► Modes and press Enter. The Options - Processing Modes pop-up, shown in [Figure 43](#), displays.

Figure 43 • Options - Processing Modes Pop-up

```

                                Options - Processing Modes
Command ==> _____ Scroll ==> PAGE
Type option settings. Then press PF3/15 (END) to save options and exit.
-----
Option      Set      Description
-----
LEARN       OFF     Display internally generated Primary Commands
SCRIPT      OFF     Script facility is disabled
XMODE       ON      Exclude all lines before executing Primary Commands

```

- 2 Type ON in the Set field for XMODE and press Enter.

Excluded lines are portions of the program which do not contain a result from your request. They are represented on your display by dashed lines. Excluding lines from the display allows you to see a more concise view of the results of your request.

Note: _____

To exit a pull-down or pop-up, press PF03/PF15 in succession until you reach the desired screen.

To see how LEARN mode works

- 1 On the Options - Processing Modes pop-up, type ON in the Set field for the LEARN option and press Enter.

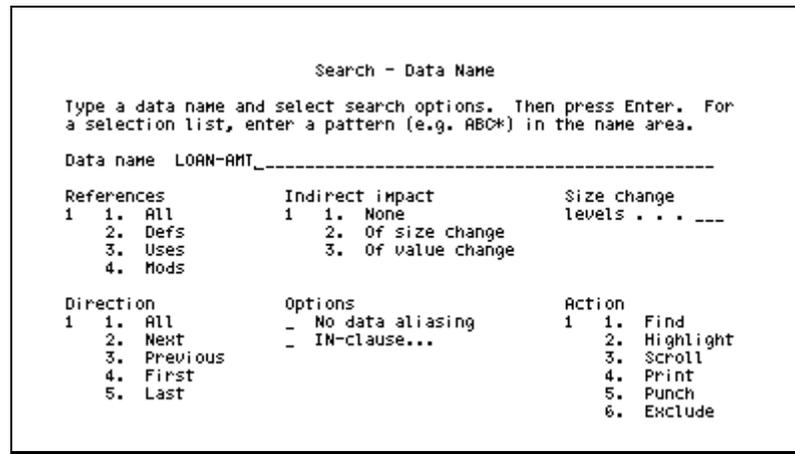
Note: _____

With LEARN mode ON, the complete command syntax displays on a pop-up.

- 2 Press PF03/PF15.

- 3 With the LEARN option set to ON, select Search ▶ Data and press Enter.
- 4 On the Search - Data pop-up, type LOAN-AMT in the Data name field and select the Indirect impact option, Of size change.
- 5 Type a slash mark (/) to the left of the IN-clause Option and press Enter. The Search - Data Name pop-up, shown in [Figure 44](#), displays.

Figure 44 • Search - Data Name Pop-up

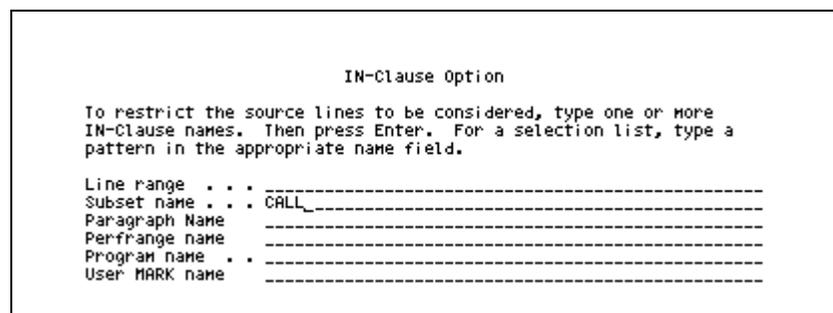


Note:

Activate selections on the pull-down by typing the appropriate number(s) in the pseudo blanks. Choices on a pop-up are not activated when you position the cursor next to an entry and press Enter. Accept the default conditions for fields on a pop-up unless you are directed otherwise.

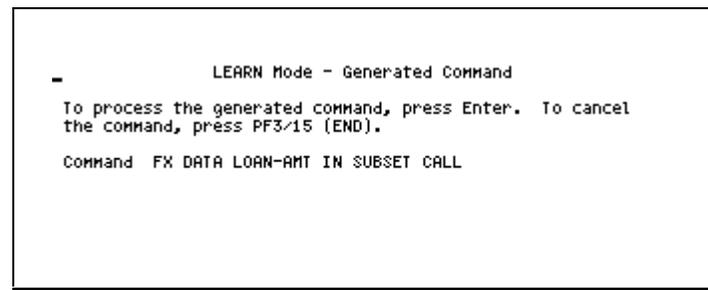
- 6 On the IN Clause Option pop-up, shown in [Figure 45](#), designate CALL as the subset name and press Enter.

Figure 45 • IN-CLAUSE Option Pop-up



The Generated Command pop-up, shown in [Figure 46](#), displays. This pop-up identifies the entire Insight command syntax. The shortest form of the command displays in upper case letters. The Command field may display results that wrap to the next line.

Figure 46 • LEARN Mode - Generated Command Pop-up



When you use the Search action, the options you select are retained until you execute another search in Insight. This permits you to conduct additional searches in your program by making only those changes to the Search action options you choose.

Note: _____

The examples presented are shown with LEARN mode set to OFF.

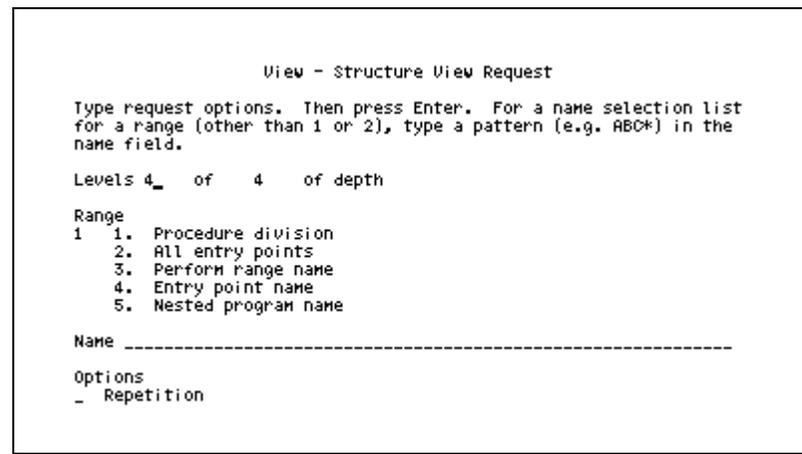
Discovering Program Structure and Coding Techniques

Insight gives you an overview of either structured or unstructured programs. With Source View, you can gain valuable understanding about various aspects of the program. To understand a particular program, how it works, and how it was written, you must have some basic information.

To investigate the structure of the program, VIAIDEMO

- 1 Select View ▶ Structure and press Enter. The View - Structure View Request pop-up, shown in [Figure 47](#), displays.

Figure 47 • View - Structure View Request Pop-up



- 2 Specify 4 of 4 Levels of depth and press Enter. If you leave the Range field blank, the default is the Procedure Division.

If you selected the Repetition option, you obtain a display of all duplicate PERFORM ranges. If you do not choose this option, duplicate PERFORM ranges are only flagged as REPEATED and are not expanded unless you use the View action and choose the Zoom in option on those ranges.

You can view different structural levels in the program by inserting a value in the Levels of depth field on the Structure View pop-up. [Figure 48](#) displays all four of the structural levels in VIAIDEMO. Insight automatically inserts the maximum number of program levels in the field, of depth.

Figure 48 • Perform Structure View Pop-up

```

File View Search Logic Task List Options Help
-----
                Perform Structure View                Program: VIAIDEMO
Command ==>----- Scroll ==> PAGE
ASG1582I 4 OF 4 LEVELS DISPLAYED IN THE VIEW.
                                         More: + >

+--- 001+
|VIAIDEMO|
+-----+

|-----|
+--- 002+
|PROGRAM-|
|INIT|
+-----+

|-----|-----|-----|-----|
+--- 004+   +--- 005+   +--- 006+   +--- 007+
|P005-VAL|   |P010-OPE|   |P155-CL-|   |P120-REA|
|-PARM TH|   |N THRU P|   |SUBTOT T|   |D THRU P|
|RU P005*|   |019-EXIT|   |HRU P15*|   |129-EXIT|
+-----+   +-----+   +-----+   +-----+

|-----|-----|-----|-----|

```

The Structure View screen is a graphical representation of the PERFORM structure of a COBOL program. It displays the logical program units, which include CALLED programs and PERFORM ranges.

Note:

The plus (+) and right (>) signs indicate that you can view additional information by moving your display down or to the right.

The content and shape of the Structure View are dependent upon the COBOL program you are viewing. The long highlighted message indicates the number of levels displayed and the total number of structural levels in the program. You can use the View action to select the Source Code display or Structure View display at any time during your analysis.

If you press PF03/PF15, you can jump from a Structure View display to Source View. To return to your previous location in Structure View, you can type RSTV in the primary command input area and press Enter.

To change the scope or reformat the display, select View ► Structure View action and enter the new values.

Use these methods to scroll the display (where *n* represents the number of columns or rows):

Direction	Method
LEFT	Type LEFT <i>n</i> in the primary command input area and press Enter.
RIGHT	Type RIGHT <i>n</i> in the primary command input area and press Enter.
UP	Type UP <i>n</i> in the primary command input area and press Enter.
DOWN	Type DOWN <i>n</i> in the primary command input area and press Enter.

Logical program units are represented in a Structure View by boxes. Relationships among these units are shown with lines between the boxes.

To examine paragraphs in PERFORM range

- 1 Select View ▶ Zoom and press Enter.
- 2 Position your cursor inside the box containing the PERFORM range (for example, P120-READ THRU P129-EXIT) and press Enter. The Perform Range Source Code pop-up, shown in [Figure 49](#), displays.

Figure 49 • Perform Range P120-READ THRU P129-EXIT

```

Command ==> _____ Perform Range Source Code _____ Scroll ==> CSR
000399 P120-READ.
000400     IF END-INPUT
000401         NEXT SENTENCE
000402     ELSE
000403         READ MASTERIN
000404             AT END
000405                 GO TO P170-FINAL.
000406     IF END-INPUT
000407         GO TO P129-EXIT
000408     ELSE
000409         IF DISTRICT-ID EQUAL ZEROES
000410             GO TO P120-READ.
000412 P129-EXIT.
000413     EXIT.
000446 P170-FINAL.
000447     IF FIRST-TIME = 'Y'
000448         PERFORM P999-ABEND-PROGRAM
000449     ELSE
000450         MOVE ' ' TO FIRST-TIME
    
```

- 3 Press PF03/PF15 to exit this expanded display of the PERFORM range.
- 4 When you have completed your examination of the program Structure View, press PF03/PF15 to return to the Source View screen.

The results of Zoom in include all of the code executed by the PERFORM range. You can scroll through the zoom in window to view the entire PERFORM range which would execute. Using this technique, you are able to gain a complete understanding of the source code.

To view a list of the CALLED programs

Assume that you think programs CALLED by VIAIDEMO may contain code that impacts this loan processing application.

- 1 Select List ▶ Calls on the pull-down and press Enter. The Calls List screen, shown in [Figure 50](#), displays.

Figure 50 • List - CALL Statements Pop-up

```

List - CALL Statements          9 CALLS
Command ==> _____ scroll ==> CSR
Select the program CALL(s) to be viewed. Then press Enter.

$  Called program                                     Mode
-  -----
-  'CD106'                                           STATIC
-  'IOMOD'                                           DYNAMIC
-  'I032'                                           DYNAMIC
-  'MESSAGE'                                         DYNAMIC
-  '$$MBMDT2'                                        DYNAMIC
-  '$$M68SEP'                                        DYNAMIC
-  'USERDUMP'                                        DYNAMIC
-  'UT45'                                           DYNAMIC
-  'UNOUE'                                           DYNAMIC
***** BOTTOM OF DATA *****

```

- 2 Press PF08 to scroll down and view additional CALLED Subprograms, if necessary. Internal subprograms are indented in relation to their hierarchy.

The Calls List screen displays all of the static, dynamic, and internally CALLED programs. If you want to examine individual CALL statements in a program, use the Search action and specify the CALL COBOL language subset on the Search - COBOL Subset Name pop-up. The difference between these results is subtle but important.

For example, ABENDPGM displays once on the Calls List screen, but it may be CALLED more than once in the program. This is revealed when you Search the CALL statements in VIAIDEMO.

The CALL mode is static, dynamic, or internal. In a static CALL, the object of the CALL is a literal and is link-edited into the same load module as the calling program. When the object of the CALL is an identifier that is loaded at run time, it is referred to as a dynamic CALL. An internal CALL is one that calls a nested source subprogram and is unique to COBOL II.

To view an internally CALLED program on the list

- 1 Type S in the line command area to the left of one of the desired Internal Called Subprograms on the List - CALL Statements pop-up and press Enter. You can then use all of the Insight functions to examine the CALLED program.

Note:

CALLED subprograms must be tagged as INTERNAL to be viewed from the Calls List screen. Static and dynamic programs cannot be viewed from this display.

- 2 Press PF03/PF15.

CALL is one of the COBOL language subsets. Insight classifies COBOL statements into subsets by grouping together COBOL verbs of a similar nature. This feature permits you to more quickly isolate specific information, for example, that relates to a program abend.

To examine the list of COBOL language subsets, follow this step:

- ▶ Select List ▶ Subsets and press Enter. The List - COBOL Subset Names pop-up, shown in [Figure 51](#), displays.

Figure 51 • LIST COBOL Subset Names Pop-up

```

List - Subset Names
Command ==> _____ $scroll ==> PAGE
Select the subset to be viewed. Then press Enter.
$ Subset Description
-----
- Assignment - COBOL: ACCEPT, ADD, SUBTRACT, MULTIPLY, DIVIDE,
                BY, COMPUTE, MOVE, FROM, INSPECT, EXAMINE,
                STRING, UNSTRING, SET, SEARCH, TRANSFORM;
                IDMS: ACCEPT, CHECK, ERASE, GET, INQUIRE, LOAD,
                PUT, RETURN;
                SQL: FETCH, SELECT INTO, SET.
- CALL        - COBOL: CALL, CANCEL, ENTRY,
                intrinsic function calls;
                CICS: LINK, XCTL;
                IDMS: ATTACH, LINK, XCTL.
- CICS        - EXEC CICS, EXEC DLI commands.
- COBOLII     - CALL (COBOLII), PERFORM (COBOLII), CONTINUE, SET,
                SERVICE LABEL, EVALUATE, INITIALIZE, end verbs.
- COBOL/370   - Statements and clauses unique to COBOL/370, such as
                intrinsic function calls, procedure pointers and
    
```

To view a list of any subset

- 1 From the List - COBOL Subset Name pop-up, position your cursor to the left of the subset name.
- 2 Type S and press Enter. You can view any COBOL subset using the same method.

To identify the INPUT/OUTPUT processing

Assume that you are interested in gaining an understanding of where the I/O processing occurs in this program in order to recommend changes to the output report, MAST-RPT.

- 1 On the List - COBOL Subset Names pop-up, press PF08 and move your cursor position down to IO.
- 2 Type S to the left of the IO subset name and press Enter.

In [Figure 52](#), the highlighted lines in VIAIDEMO include all of the COBOL I/O statements (I/O, Input, or Output, respectively), plus the CALL statements that contain I/O, Input, or Output. This information identifies all of the possible places that values are read in and/or output. It is useful in isolating statements that may produce incorrect output.

Figure 52 • Source View Showing I/O Processing Statements

```

File View Search Logic Task List Options Help
-----
Source View
Command ==>> 14 LINES FOUND
                SCROLL ==>> CSR
000247  DISPLAY 'TOTAL INPUT RECORDS - ' REC-CNT. IO
000248  DISPLAY 'END VIAIDEMO PROCESSING' UPON CONSOLE. IO
-----
000312  OPEN INPUT MASTERIN. IO
000313  OPEN OUTPUT MASTER-RPT. IO
-----
000352  WRITE MAST-RPT FROM DETAIL-LINE1. IO
-----
000357  WRITE MAST-RPT FROM DETAIL-LINE2. IO
-----
000365  WRITE MAST-RPT FROM DETAIL-LINE3. IO
-----
000372  WRITE MAST-RPT FROM DETAIL-LINE4. IO
-----
000403  READ MASTERIN IO
-----
000424  WRITE MAST-RPT FROM SUB-PRINT. IO
-----
000434  WRITE MAST-RPT FROM RPT-HDG-LINE1. IO
-----

```

You can also investigate the OPEN statements in a program using the same method.

To search for specific COBOL verbs within a program

- 1 Select Search ▶ String and press Enter. The Search - Pattern String pop-up displays.
- 2 Type OPEN in the String field, select the IN-clause on, and press Enter.

- 3 Specify the IO Subset name and press Enter. The Source View pop-up displays the Open I/O Statements, as shown in [Figure 53](#).

Figure 53 • Open I/O Statements

```
File View Search Logic Task List Options Help
-----
Source View                                2 PATTERNS FOUND
Command ==> _____ Scroll ==> CSR
000312      OPEN INPUT MASTERIN.                PATTERN
000313      OPEN OUTPUT MASTER-RPT.             PATTERN
-----
***** ***** BOTTOM OF DATA *****
```

To save the results of the analysis for later reference

Assume that you want to save this information for later reference during analysis. To accomplish this, create a Mark on the Scratchpad.

- 1 Select Options ▶ Scratchpad and press Enter. The Options - Scratchpad pop-up, shown in [Figure 54](#), displays.

Figure 54 • Options - Scratchpad Pop-up

```
Options - Scratchpad
Select the desired option to create or modify
Equates and Marks. Then press Enter.

Options
1  1. Equate...
   2. Mark...
   3. Copy...
   4. Delete...
   5. Merge...
   6. Rename...
```

- 2 Select the Mark option and press Enter. The Options - Scratchpad Mark pop-up, shown in [Figure 55](#), displays.

Figure 55 • Options - Scratchpad Mark Pop-up

```

Options - Scratchpad Mark

To create a mark, type a name and a target. Then press Enter. For a
name selection list for a target type, type a pattern (e.g. ABC*) in
the target name area.

Mark name . . _-----
Target name  -----
Target type
1  1. Mark name
   2. Perfrange name
   3. Program name
   4. Subset name
   5. Line range
   6. Label name

Comments . . _-----
    
```

- 3 To specify a name for your Scratchpad entry, type in your unique mark name, target name and type, and comments.
- 4 Press Enter to display the Utility - Scratchpad Mark pop-up.

To recall a Mark at any time during your analysis, follow this step:

- ▶ Select List ▶ Marks and press Enter. The List - User Marks pop-up, shown in [Figure 56](#), displays.

Figure 56 • List - User Marks Pop-up

```

List - User Marks

Command ==> _____ Scroll ==> CSR

C : Copy set/path 1 to set/path 2.   D : Delete existing set/path 1.
M : Merge set/path 1 with set 2.    R : Rename set/path 1 to set/path 2.
S : Select the mark to be viewed.

Set/path 1      Set/path 2      Comments
-----
_ MARK1        SET _____ ALL OPEN STATEMENTS
***** BOTTOM OF DATA *****
    
```

To print or punch these results for later reference during your analysis, select either the Print or the Punch option on a pop-up. Select the Print option to copy lines containing this result to a List file. Select the Punch option to copy the lines to a Punch file.

The List or Punch file is processed according to the default you set up in your Log/List/Punch definition.

To set up the default print and punch process on your Job Card

- 1 From the User Marks pop-up, press PF03/PF15.
- 2 Select Options ▶ Log/list/punch and press Enter. The Options - Log/List/Punch Definition pop-up, shown in [Figure 57](#), displays.

Figure 57 • 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      . . . . . K              PD              PK
Primary tracks      . . . . . 1              1              1
Secondary tracks    . . . . . 2              5              5
Lines per page      . . . . . 56             56             56
Sysout class        . . . . . *              *              *

Process options:  PK (print/keep), PD (print/delete), K, or D.
Log file is allocated

Job statement information:
//VIA123 JOB (DEUJXS,283200),
//          MSGCLASS=A
//*  INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
```

- 3 Designate one of these Process Options:
 - PK (Print/Keep)
 - PD (Print/Delete)
 - K (Keep)
 - D (Delete)
- 4 To print the file from Log/List/Punch Definition screen immediately, select the option, 2 - Process LIST File and press Enter. If you do not elect to print immediately, the List file is processed when you end your Insight session.

To customize the Log, List, or Punch dataset name

If you specify K or PK process option, you can customize the dataset where the Log, List, or Punch file is allocated. By default, Insight allocates the Log, List, and Punch files in this format:

USERID.yyyynnnnn.VIAxxxxxx

where:

yyy is the product ID.

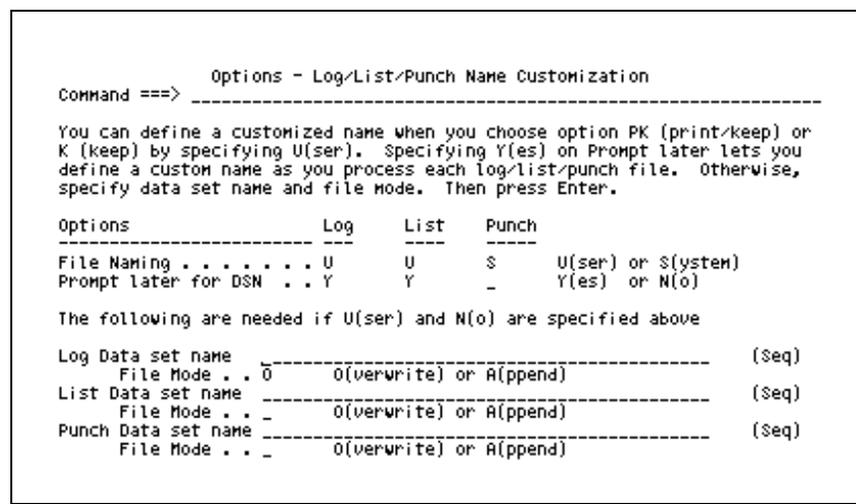
nnnnn is a sequential number from 00001 to 99999.

xxxxxx is LOG for Log, LIST for List, and PUNCH for Punch files.

If you have specified a TSO Prefix, the prefix will be appended to the beginning of the file name allocated for the Log, List, and Punch files.

- 1 From the Options - Log/List/Punch Definition pop-up, type 4 and press Enter to display the Options - Log/List/Punch Name Customization pop-up shown in [Figure 58](#).

Figure 58 • Options - Log/List/Punch Name Customization Pop-up



- 2 Type U in the File Naming field for Log, List, and/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 Overwrite or Append in the File Mode field.

If you specify Y in the Prompt later for DSN field, Insight prompts you for the dataset name during file processing. For example, if you specify Yes in the Prompt later for DSN field for the Log file, the Log Name Customization pop-up, shown in [Figure 59](#), displays.

Figure 59 • Log Name Customization Pop-up

```

                                Log Name Customization
Command ==> -----
The current log file's data set name is shown below. To have a custom
name, specify a new sequential data set name. If it already exists, it
has to have LRECL=137 and RECFM=U; and specify the file mode. Then press
Enter.
Current Data set name : ''
Custom Data set name  _-----
File Mode . . . _      0(overwrite) or A(append)
    
```

To find internal SORTs used in this program

A SORT is used primarily to process data from a master file that is not in an appropriate sequence. SORTs occur in SORT sections using input and output procedures. Assume, for this example, that you want to locate any SORTs in VIAIDEMO, remove them from the program and use an external SORT utility that is prescribed by your department standards.

In this example, Insight looks for one specific verb represented by the PERFORM COBOL subset. The PERFORM subset also looks for MERGE and PERFORM statements, in addition to locating SORT.

- 1 Select Search ▶ String and press Enter. The Search - Pattern String pop-up, shown in [Figure 60](#), displays.

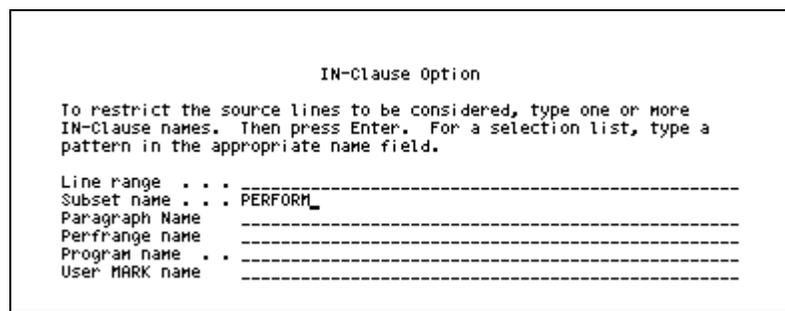
Figure 60 • Search - Pattern String Pop-up

```

                                Search - Pattern String
Type a string and select search options. Then press Enter.
String _-----
String type                Location
1  1. Simple                1  1. Any
   2. Hexadecimal           2. Word
   3. Text                  3. Prefix
   4. Picture                4. Suffix
Direction                  Options                Action
1  1. All                    _ IN-clause...  1  1. Find
   2. Next                   2. Highlight
   3. Previous                3. Scroll
   4. First                   4. Print
   5. Last                    5. Punch
Begin column ___           End column ___
                                6. Exclude
    
```

- 2 Type SORT in the String field, select the IN-clause Option on the pop-up, and press Enter.
- 3 Type PERFORM as the subset name, as shown in [Figure 61](#), and press Enter.

Figure 61 • IN-Clause Option Pop-up



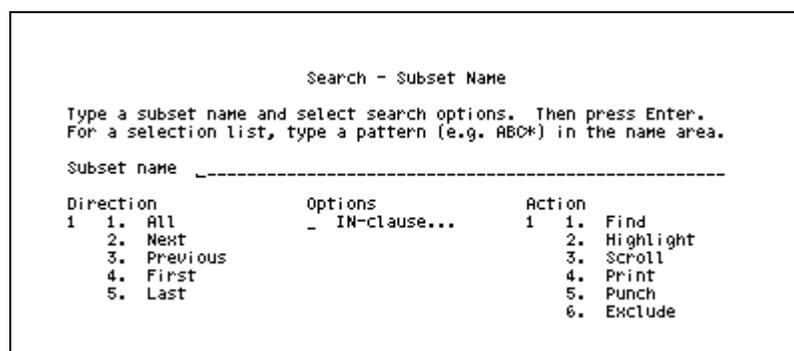
In this example, no SORTs were found in the program, VIAIDEMO, indicated by the short message at the top right of the screen, NO PATTERNS FOUND.

To find COBOL debugging facilities used in this program

Assume that you want to examine the source to determine the expected output before you debug your program. You can use Insight to identify all of the statements containing a DEBUG, EXHIBIT, ON, READY, or RESET verb, or a D in column 7 in the program. This information may be of greatest value to you if you are unfamiliar with the source code.

- 1 Select Search ▶ Subset and press Enter. The Search - COBOL Subset Name pop-up, shown in [Figure 62](#), displays.

Figure 62 • Search - COBOL Subset Name Pop-up



- 2 Type DEBUG as the subset name, turn off the IN-clause Option on the pop-up, and press Enter.

In many cases, programs set these options to turn on based upon an internal switch setting or by a parameter passed to the program. Locating these statements allows you to investigate how these internal DEBUGs are activated.

The results, shown in [Figure 63](#), indicate that there is one debug statement found in this program. If enabled, the READY TRACE statement, when executed, sends paragraph names that are referenced to the system output file. This file provides a listing of all paragraphs executed.

Figure 63 • Source View - COBOL DEBUGging Search

```
-----
File View Search Logic Task List Options Help
-----
Source View                                1 STATEMENT FOUND
Command ==> _____ SCROLL ==> CSR
-----
- - - - - 255 LINES NOT DISPLAYED
00256      READY TRACE.                      DEBUG
- - - - - 240 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
```

To make this result more meaningful in your analysis, you can now examine what paragraph contains this READY statement. You may also want to investigate how the READY statement gets invoked.

- 3 Select View ► Zoom and press Enter.
- 4 Position the cursor on the READY statement and press Enter.

Note: _____

To activate this debugging facility, you must set the compiler option.

- 5 Press PF07 to move the display up or PF08 to move down.

In [Figure 64](#), the READY TRACE is activated by the parameter, DEBUG-PARM, with a value of TEST. A Search for the definitions of the data item reveals that it is part of 01 INPUT-PARM.

Figure 64 • Source View - Expanded Statements

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> CSR

***** ***** TOP OF DATA *****
----- 252 LINES NOT DISPLAYED
000253 PROGRAM-INIT.
000254
000255     IF DEBUG-PARM = 'TEST'
000256         READY TRACE.                                DEBUG
000257
000258     PERFORM P005-VAL-PARM
000259         THRU P005-EXIT.
000260
000261     PERFORM P010-OPEN
000262         THRU P019-EXIT.
000263
000264     PERFORM P155-CL-SUBTOT
000265         THRU P159-EXIT.
000266
000267     PERFORM P120-READ
000268         THRU P129-EXIT.
000269

```

To locate GOTO clauses in a program

Locating GOTO clauses is useful in determining the degree to which a program is structured.

- 1 Select Search ▶ Subset and press Enter.
- 2 Type GOTO as the subset name, as shown in [Figure 65](#), and press Enter.

Figure 65 • Search - COBOL Subset Name

```

Search - Subset Name

Type a subset name and select search options. Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

Subset name GOTO

Direction          Options          Action
1  1. All           - IN-clause...  1  1. Find
   2. Next
   3. Previous
   4. First
   5. Last

```

These results, shown in [Figure 66](#), reveal that three of the five GOTOs are to EXIT paragraphs. If there were many GOTOs to paragraph names, for example, the program might be considered unstructured.

Figure 66 • Source View - GO TO Statements

```
File View Search Logic Task List Options Help
-----
Source View                               5 STATEMENTS FOUND
Command ==>                               Scroll ==> PAGE
-----
000407          GO TO P170-FINAL.          GOTI
----- 1 LINE NOT DISPLAYED
000409          GO TO P129-EXIT           GOTI
----- 2 LINES NOT DISPLAYED
000412          GO TO P120-READ.          GOTI
----- 16 LINES NOT DISPLAYED
000429          GO TO P159-EXIT.          GOTI
----- 41 LINES NOT DISPLAYED
000471          GO TO P129-EXIT.          GOTI
----- 27 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
```

To examine the contents of P120-READ

You can investigate what happens in paragraphs P170-FINAL and P120-READ.

- 1 Select Search ▶ Subset and press Enter. The Search - COBOL Subset Name pop-up displays.
- 2 Specify the subset and PARAGraph, select Previous as the Direction, and press Enter.
- 3 Position the cursor on the GO TO P120-READ statement and press Enter.
- 4 Select View ▶ Zoom and press Enter.

- 5 Position the cursor on the P120-READ statement on line 399 and press Enter. The Source View screen, shown in [Figure 67](#), displays.

Figure 67 • Source View - Paragraph P120-READ

```
File View Search Logic Task List Options Help
-----
Source View                               12 DETAIL LINES
Command ==>                               Scroll ==> CSR
000399 P120-READ.                           PARAGRAP
000400     IF END-INPUT
000401     NEXT SENTENCE
000402     ELSE
000403         READ MASTERIN
000404         AT END
000405             GO TO P170-FINAL.
000406     IF END-INPUT
000407         GO TO P129-EXIT
000408     ELSE
000409         IF DISTRICT-ID EQUAL ZEROES       FALLTHRU
000410             GO TO P120-READ.
000411
-----
***** 85 LINES NOT DISPLAYED *****
***** BOTTOM OF DATA *****
```

Getting a Paragraph Overview

Now that you have gained an understanding of how this program is constructed, you can examine the details at the paragraph level.

Many organizations create programming standards that dictate that PERFORM statements must also contain a THRU clause indicating the intended ending point for that PERFORM.

To examine a list of program PERFORM ranges, follow this step:

- ▶ Select List ▶ Performs and press Enter. The List - Perform Range Names pop-up, shown in [Figure 68](#), displays.

Figure 68 • List - Perform Range Names Pop-up

```

                                     List - Perform Range Names      10 PERFORM RANGES
Command ==> _____ Scroll ==> PAGE
Select the perform range to be viewed. Then press Enter.

  Perform range name
-----
- PROGRAM-INIT
- P000-NEXT THRU P000-EXIT
- P005-UAL-PARM THRU P005-EXIT
- P010-OPEN THRU P019-EXIT
- P100-PRINT THRU P119-EXIT
- P120-READ THRU P129-EXIT
- P150-SUBTOT THRU P169-EXIT
- P155-CL-SUBTOT THRU P159-EXIT
- P160-HDG THRU P169-EXIT
- P999-ABEND-PROGRAM
***** BOTTOM OF DATA *****
```

These results indicate that the THRU clause has not been coded with the PERFORM LABEL, PROGRAM-INIT, or the Section PERFORM, P999-ABEND-PROGRAM OF ABEND-PROGRAM.

To examine the source code in a PERFORM range on this list, type S to the left of the desired range and press Enter. You may want to make a note of these results when you examine the program during the next analysis phase involving Quality Assurance and Standards.

Using the List action, you can also isolate possible recursion of PERFORMed routines in a program. [Figure 69](#) shows a Perform Ranges List screen with one recursion found. VIAIDEMO does not contain recursion. This screen sample has been created for illustration purposes only and cannot be found in VIAIDEMO. If there is recursion in your program, it is marked with an asterisk (*) on the Perform Ranges List screen.

Figure 69 • Program Recursion - Perform Ranges List

```

====>                                Perform Ranges List                28 PERFORM RANGES
ASG1365I 28 PERFORM RANGES, 1 RECURSIVE.                                SCROLL ====> CSR

      Enter S before Perform Label to select for viewing.

      PERFORM LABEL (* INDICATES ROUTINE IS RECURSIVELY PERFORMED)
-----
000-INITIALIZE-STORAGE
050-HANDLE-CONDITION
060-SUBSCRIPT
070-INDEX
100-CHOICE-CLEAR
1000-CAUSE-CWA-ERROR THRU 1000-CWA-ERROR-EXIT
1050-CAUSE-SEVEN-ERROR THRU 1050-SEVEN-ERROR-EXIT
1100-CAUSE-FILE-ERROR THRU 1100-FILE-ERROR-EXIT
1150-RETURN-TO-CICS
850-CAUSE-STORAGE-VIOLATION THRU 850-STORAGE-VIOLATION-EXIT
900-CAUSE-ALOT-OF-STORAGE THRU 900-ALOT-OF-STORAGE-EXIT
* 9000-CLEAR-THE-SCREEN
9100-SEND-MSG
950-CAUSE-FREEMAIN-ERROR THRU 950-FREEMAIN-ERROR-EXIT
  
```

Note:

For more information regarding the List screens, see the online help or the *ASG-Insight Reference Guide*.

To find the statements executed by the PERFORM range, P100-PRINT THRU P119-EXIT

Assume that you suspect a problem in the PRINT routine. You decide to examine all of the code associated with a PERFORM statement. To quickly locate and display the statements in the PERFORM range:

- 1 Type S to the left of the PERFORM range, P100-PRINT THRU P119-EXIT, on the List Perform Range Names pop-up and press Enter.

Or

Select Search ▶ Performs and press Enter.

- 2 Type the perform range in the Perfrange Name field and press Enter (or, press Enter on the Search - Perform Range Name pop-up to obtain a selection list, type S next to the perform range, and press Enter).

These results reveal all of the statements that could potentially be executed as a result of this PERFORM. You can scroll down through the code to investigate the location of the suspected problem in the PRINT routine.

Notice that no statements are displayed with paragraph P100-PRINT. This occurred because XMODE is turned on, excluding the comments that follow this statement.

- 3 Type S4 in the line command input area to show the two excluded lines and press Enter.

In [Figure 70](#), note the fall-through condition identified in this result on line 341, indicating that this PERFORMed paragraph does not return to the PERFORM statement on line 340. It falls through to the next paragraph.

Figure 70 • Perform Range P100-PRINT THRU P119-EXIT

```
File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ===>                               Scroll ===> CSR
000324 P100-PRINT.                           PERF RNG
000325
000326 * IF THE FIRST DIGITS OF THE ZIP CODE CHANGE, PRINT THE SUBTOTALS
000327 * AND SKIP TO THE TOP OF THE PAGE.
000328
000329 P105-NOT-FIRST-TIME.                   PERF RNG
000330
000331     MOVE ZIP-CODE TO HLD-ZIP.             PERF RNG
000332     IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX    PERF RNG
000333         NEXT SENTENCE                     PERF RNG
000334     ELSE                                  PERF RNG
000335         PERFORM P150-SUBTOT                 PERF RNG
000336             THRU P169-EXIT
000337         MOVE HLD-ZIP-PREFIX TO CUR-PREFIX.  PERF RNG
000338
000339     IF LINE-CNT GREATER THAN 53              PERF RNG
000340         PERFORM P160-HDG                   PERF RNG
000341             THRU P169-EXIT.                FALLTHRU
000342
```

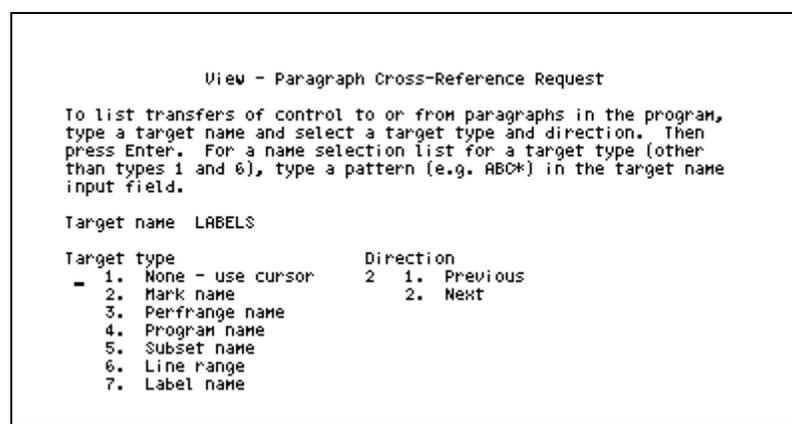
Also, notice that this program CALLs DBAREAD2 on line 345. You can use the List action with the Calls option to see if this subprogram is internally called by VIAIDEMO. If it is, you can use the File action to Open DBAREAD2 to determine how logic paths are followed across CALLED programs. This technique could be used to determine how values are passed between programs without ending your current Insight analysis session.

To perform paragraphs transfer control

The Paragraph Cross-Reference function permits you to see how control is transferred to or from the target paragraph(s). If no operand is entered after you select a Target name on the pop-up, the target paragraph is determined by the cursor location when the function is executed. Select a Target type if you do not specify a Target name. For example, if you are uncertain of the Target name, leave that field blank and choose a Target type.

- 1 Select View ► Paragraph X-ref on the pull-down and press Enter. The View - Paragraph Cross-Reference Request pop-up, shown in [Figure 71](#), displays.

Figure 71 • View - Paragraph Cross-Reference Request Pop-up



- 2 Specify the Target name as LABELS, the Direction as NEXT, and press Enter.

Note: _____

The default setting for the Paragraph Cross-Reference function is PREVIOUS.

The Target(s) field requests the paragraph name(s) of interest, in this case the Subset, Labels, as shown in [Figure 72](#). The Direction was set to determine the paragraphs to which control is transferred. If you choose, you may use the same function to determine paragraphs from which control is transferred. To accomplish this, select Previous as the Direction on the View - Paragraph Cross Reference Request pop-up and press Enter. To Add to the target paragraph(s), type A in the column to the left of the selected paragraph and press Enter.

Figure 72 • View - Paragraph Cross Reference Pop-up

```

View - Paragraph Cross Reference
Command ==> _____ Scroll ==> PAGE
Target(s): LABELS
Direction: NEXT

  A : Add to target      P : Execute PREF      S : Select for viewing
-----
S Target paragraph(s)   How                Goes to
-----
- PROCEDURE DIVISION   PERFORM            PROGRAM-INIT
- " "                  PERFORM            P000-NEXT
- " "                  PROGRAM EXIT
- PROGRAM-INIT         RETURN             PROCEDURE DIVISION
- " "                  PERFORM            P005-UAL-PARM
- " "                  PERFORM            P010-OPEN
- " "                  PERFORM            P120-READ
- " "                  PERFORM            P155-CL-SUBTOT
- P000-NEXT             FALLTHRU           P000-EXIT
- " "                  PERFORM            P100-PRINT
- " "                  PERFORM            P120-READ
- P000-EXIT             RETURN             PROCEDURE DIVISION
  
```

If you accept the default direction, PREVIOUS, the target paragraph is added to the top of the list. If you specify NEXT, the target paragraph is added to the end of the list. Type P to execute another paragraph cross-reference or S to return to the source code with the first line of the paragraph displayed at the top of the screen. Ditto marks (") are shown when multiple paragraphs transfer control to or from the same paragraph.

You can now print or punch this information for later reference during your analysis or to document your findings as described earlier in this section.

Hints

COBOL Language Subsets. Several examples use COBOL Language Subsets to examine program and paragraph structure. Insight classifies COBOL statements into subsets by grouping together COBOL verbs of a similar nature. You can isolate statements in a program that contain any of the approximately 30 verbs recognized with the added language intelligence of DB2, SQL, and IDMS.

The IN-Clause Option. You can use the IN-Clause option to restrict a search to specific areas of the source code. The IN-Clause option accepts any set or path. You can use concatenation to add sets together, permitting you to use just one action to check more than one set for the same target. For example, you can use the Search action to locate a dataname using the IN-Clause on the Search - Data pop-up to specify all I/O and CALLED programs in concatenated form in the field marked COBOL Subset as IO + CALL.

Summary: Getting a Program Overview

- This program is reasonably well-structured, based upon the findings from Structure View.
- Save the analysis results anytime using the scratchpad, print, and punch features.
- You can examine multiple internally CALLED programs in the same Insight session.
- You saved significant time during testing by using Insight to identify program structure, paragraph cross-references, and locate statements executed in a PERFORM range of interest.
- COBOL language subsets group together verbs of a similar nature and make it quick and easy to pinpoint targets for analysis.

Identifying Data Item Usage Cross-References

Unlike the standard compiler cross-reference, Insight offers extended cross-referencing of data items. Insight can differentiate between a use, modification, or definition of a data item. This phase of the analysis permits you to identify how loan processing data items are used and cross-referenced in this application.

Use Insight to determine as much information as you can about LOAN-AMT, including all data references, as well as the impact of a change to either size or value of the data item. Use these questions to guide you through your analysis of VIAIDEMO:

Data Item Names

Specify any COBOL reference to a data element as a dataname. If a variable is redefined to another name, Insight searches for the specified variable name and, optionally, the redefined name.

When data items overlap (for example, when a name refers to parts of multiple data items), searches are performed on each part and all references are reported. Any reference to an entry in a table is regarded as a reference to the entire table. These references are called aliases. Insight locates the direct and alias references (definitions, uses, or modifications) as opposed to simple pattern matching of the characters in the variable name.

You can specify fully qualified datanames followed by a standard COBOL OF clause and the group level dataname, for example:

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

Assume, for example, that the dataname, CUST-ID-NUM, is part of two records, MASTER1-IN and MASTER2-IN. If you use the Search feature to locate CUST-ID-NUM and do not qualify either MASTER1-IN or MASTER2-IN, Insight identifies references to the data item in both records. To isolate the data item in one record or the other, you must specify the record you want to search.

You can search for multiple data items at the same time by concatenating the datanames with a plus sign (+) between them. For example:

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

An ALIAS for a dataname refers to the Parent (higher level group item), Child (lower level group item) or Rename/Redefinition (renamed, redefined, and 88 level items). This example shows an ALIAS (all Insight references to LOAN-INFORMATION in record MASTER-IN include PAYMENT-AMT, LOAN-AMT, and INTEREST-RATE).

```
01 MASTER-IN.  
  05 LOAN-INFORMATION.  
    10 PAYMENT-AMT      PIC 9(7)V99.  
    10 LOAN-AMT        PIC 9(13)V99.  
    10 INTEREST-RATE   PIC V99999.
```

To reveal all references to LOAN-AMT in VIAIDEMO

- 1 Select Search ► Data and press Enter.

You can designate a dataname on the Search - Data pop-up or select one from the Selection List - Data Names pop-up. In the Data name field on the Search - Data Selection List pop-up, you can enter a pattern (for example, ABC* for all datanames starting with ABC) or blanks in the Data name field.

- 2 Leave the Data name field blank, leave the Indirect impact Option blank and press Enter on the Search - Data Name pop-up.
- 3 Press PF08 to move the display down until the dataname, LOAN-AMT displays, as shown in [Figure 73](#).

Figure 73 • Selection List - Data Names Pop-up

```
Selection List - Data Names          LINE 1 OF 101  
Command ==> _____ Scroll ==> PAGE  
$  Data name  
-----  
- ABEND-CODE  
- ADDRESS1  
- AREA-CODE  
- AVG-AMT (DEAD)  
- BILLING-DAYS  
- CHECK-CODE (DEAD)  
- CITY  
- CLIENT-ID  
- CONSOLE  
- CUR-PREFIX  
- CURRENT-ZIP-DATA  
- CUSTOMER-ID  
- DBA-DEPT-CODE  
- DEBUG-PARM  
- DET-ADDRESS  
- DET-AREA-CODE  
- DET-BILLING-DAYS
```

- 4 Type S to the left of the LOAN-AMT dataname and press Enter. The Search - Data Name pop-up, shown in [Figure 74](#), displays.

Figure 74 • Search - Data name Pop-up

```

Search - Data Name

Type a data name and select search options. Then press Enter. For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data name  LLOAN-AMT

References          Indirect impact          Size change
1  1. All           1  1. None                levels . . . ---
   2. Defs         2  2. Of size change
   3. Uses         3  3. Of value change
   4. Mods

Direction          Options                      Action
1  1. All          - No data aliasing         1  1. Find
   2. Next        - IN-clause...            2. Highlight
   3. Previous
   4. First
   5. Last
  
```

- 5 Press Enter to display the screen shown in [Figure 75](#).

Figure 75 • All Occurrences of LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> CSR
ASG0443I 7 DATA REFS: 4 DEFS, 1 USE, 2 MODS, FOUND FOR LOAN-AMT.
***** TOP OF DATA *****
----- 13 LINES NOT DISPLAYED
000014  FD  MASTERIN                                DATA DEF
000015          RECORDING MODE IS F
000016          BLOCK CONTAINS 0 RECORDS
000017          LABEL RECORDS ARE STANDARD.
----- 2 LINES NOT DISPLAYED
000020  01  MASTER-IN.                                DATA DEF
----- 14 LINES NOT DISPLAYED
000035  05  LOAN-INFORMATION.                          DATA DEF
----- 1 LINE NOT DISPLAYED
000037  10  LOAN-AMT                                  PIC 9(13)V99.  DATA DEF
----- 243 LINES NOT DISPLAYED
000281  CALL 'VIAIDEM1' USING MASTER-IN                DATA MOD
000282          MASTER-END-OF-FILE
000283          MASTER-REPORT-DATE.
----- 66 LINES NOT DISPLAYED
000350  MOVE LOAN-AMT TO DET-LOAN-AMT.                  DATA USE
----- 52 LINES NOT DISPLAYED
  
```

LOAN-AMT or its aliases are referenced in seven locations in this program. Line 281 identifies a modification of LOAN-AMT in the CALLED module, VIAIDEM1. If VIAIDEM1 resides in the AKR, LOAN-AMT is actually modified in VIAIDEM1. If VIAIDEM1 is not in the AKR, then Insight identifies the occurrence of LOAN-AMT as a possible modification.

Locate Modifications to the Data Item

To isolate the modifications to LOAN-AMT

- 1 Select Search ► Data and press Enter.
- 2 Specify the dataname as LOAN-AMT with the Mods option in the References field, as shown in [Figure 76](#), and press Enter.

Figure 76 • Search - Data Name Pop-up

```

Search - Data Name
Type a data name and select search options. Then press Enter. For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data name LOAN-AMT

References          Indirect impact          Size change
4_ 1. All           1 1. None                levels . . . ---
   2. Defs          2 2. Of size change
   3. Uses          3 3. Of value change
   4. Mods

Direction          Options                    Action
1 1. All           - No data aliasing        1 1. Find
   2. Next         - IN-clause...           2. Highlight
   3. Previous
   4. First
   5. Last

```

LOAN-AMT is modified in a CALLED module and again in the READ statement. [Figure 77](#) shows the modification of LOAN-AMT found in the CALLED program, VIAIDEM1. When you investigate datanames in CALLED programs, you need to identify the corresponding field name in the parameters of the CALLED program.

Figure 77 • Modifications of LOAN-AMT

```

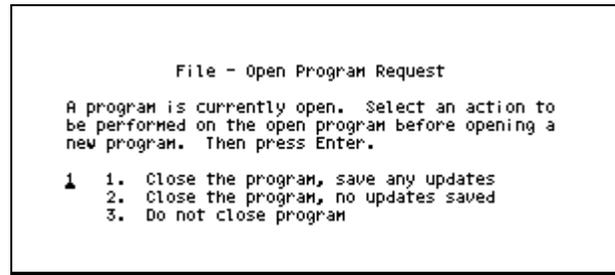
File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEM1
Command ==>                               Scroll ==> PAGE
ASG0443I 2 DATA MODS: 2 PARENTS, FOUND FOR LOAN-AMT.
***** ***** TOP OF DATA *****
----- 282 LINES NOT DISPLAYED -----
000283      CALL 'VIAIDEM1' USING MASTER-IN      DATA MOC
000284      MASTER-END-OF-FILE
000285      MASTER-REPORT-DATE.
----- 119 LINES NOT DISPLAYED -----
000405      READ MASTERIN                        DATA MOC
----- 93 LINES NOT DISPLAYED -----
***** ***** BOTTOM OF DATA *****

```

To examine the CALLED program, VIAIDEM1

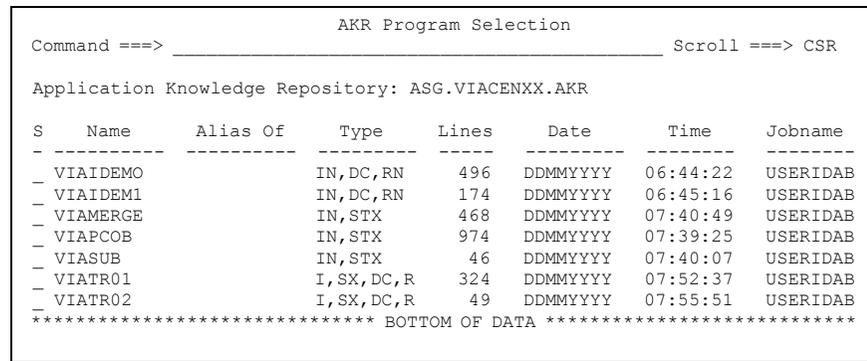
- 1 Select File ▶ Open and press Enter. The File - Open Program Request pop-up, shown in [Figure 78](#), displays.

Figure 78 • File - Open Program Request Pop-up



- 2 Select 3. Do not close program and press Enter. The AKR Program Selection pop-up, shown in [Figure 79](#), displays.
- 3 Type S to the left of VIAIDEM1 to select the program from the list of programs in the AKR. Press Enter.

Figure 79 • AKR Program Selection Pop-up



You can now use all of the Insight features to investigate VIAIDEM1, shown in [Figure 80](#).

Figure 80 • VIAIDEM1

```

File View Search Logic Task List Options Help
-----
Command ==> Source View Program: VIAIDEM1
Scroll ==> CSR

***** ***** TOP OF DATA *****
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. VIAIDEM1.
000003 AUTHOR. WRITTEN BY ASG AT LANGLVL 2.
000004 ENVIRONMENT DIVISION.
000005 CONFIGURATION SECTION.
000006 SOURCE-COMPUTER. IBM-370.
000007 OBJECT-COMPUTER. IBM-370.
000008 INPUT-OUTPUT SECTION.
000009 FILE-CONTROL.
000010     SELECT DAILY-TOTALS ASSIGN TO S-DTOTALS.
000011 DATA DIVISION.
000012 FILE SECTION.
000013 FD DAILY-TOTALS
000014     RECORDING MODE IS F
000015     BLOCK CONTAINS 0 RECORDS
000016     LABEL RECORDS ARE STANDARD.
000017
000018 01 TOTAL-REPORT.

```

To locate the modification of LOAN-AMT in VIAIDEM1

- 1 Select Search ▶ Data and press Enter. The Search - Data Name pop-up, shown in [Figure 81](#), displays.

Figure 81 • Search - Data Name Pop-up

```

Search - Data Name

Type a data name and select search options. Then press Enter. For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data name LOAN-AMT

References          Indirect impact          Size change
4_ 1. All           1 1. None                levels . . . ___
   2. Defs         2 2. Of size change
   3. Uses         3 3. Of value change
   4. Mods

Direction          Options                    Action
1 1. All           - No data aliasing        1 1. Find
   2. Next         - IN-clause...           2. Highlight
   3. Previous
   4. First
   5. Last

```

- 2 Identify the dataname (LOAN-AMT), specify Mods in the References field, and press Enter.

The result identifies the modification of LOAN-AMT in VIAIDEM1, as shown in [Figure 82](#).

Figure 82 • Modifications of LOAN-AMT in VIAIDEM1

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEM1
Command ==>                               Scroll ==> CSR
ASG0443I 2 DATA MODS: 1 PARENT, FOUND FOR LOAN-AMT.
***** ***** TOP OF DATA *****
----- 107 LINES NOT DISPLAYED
000108 PROCEDURE DIVISION USING MASTER-IN,          DATA MOD
000109          MASTER-END-OF-FILE,
000110          MASTER-REPORT-DATE.
----- 3 LINES NOT DISPLAYED
000114          MOVE INIT-AMTS TO LOAN-AMT          DATA MOD
----- 60 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

To expand the MOVE statement to look at the detail associated with the passing of the value INIT-AMTS to LOAN-AMT

- 1 Select View ► Zoom in and press Enter.
- 2 Position the cursor on the statement MOVE INIT-AMTS TO LOAN-AMT in VIAIDEM1 and press Enter. The result is shown in [Figure 83](#).

Figure 83 • Expanded Statements in VIAIDEM1

```

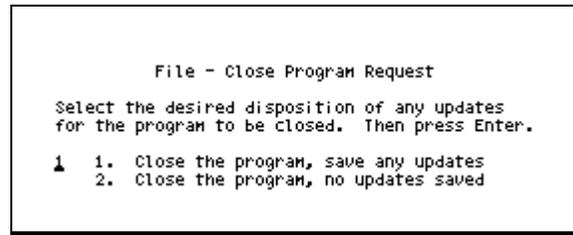
File View Search Logic Task List Options Help
-----
Source View                               13 DETAIL LINES
Command ==>                               Scroll ==> CSR
***** ***** TOP OF DATA *****
----- 107 LINES NOT DISPLAYED
000108 PROCEDURE DIVISION USING MASTER-IN,          DATA MOD
000109          MASTER-END-OF-FILE,
000110          MASTER-REPORT-DATE.
000111
000112          IF FIRST-TIME = 'Y'
000113          OPEN OUTPUT DAILY-TOTALS
000114          MOVE INIT-AMTS TO LOAN-AMT          DATA MOD
000115          MOVE ' ' TO FIRST-TIME          ELSE          FALLTHRU
000116          IF END-INPUT
000117          PERFORM P200-PRINT-TOTALS
000118          THRU P299-EXIT
000119          CLOSE DAILY-TOTALS
000120          GO TO P009-END-PROGRAM
000121          ELSE
000122          PERFORM P100-COMPUTE-TOTALS
000123          THRU P199-EXIT.          FALLTHRU
000124

```

- 3 Select File ► Close and press Enter.

- 4 On the File - Close Program Request pop-up, shown in [Figure 84](#), select 2. Close the program, no updates saved and press Enter.

Figure 84 • File - Close Program Request



Determining the Impact of Changes to the Size or Value of LOAN-AMT

When you change the way a data item is used, you impact related program logic, which may create a ripple effect. You can use the Search function with the Indirect impact option, Of value change, to follow the data flow from one field to the next (Field A to B, B to C, C to D, etc.).

For example, data item A has only two values. You change the code to permit three values. If the value in A is transferred to other data items and an unexpected data item value is found, the program may fail. Search, used in conjunction with the Of value change option, locates those indirectly related data items and logic that may depend on the values contained in A.

First, determine the impact to this program of a change to the size of LOAN-AMT. Then determine the impact to this program of a change to the value of LOAN-AMT in the same manner.

Specify the number of Levels to designate the depth of the indirect references to the data item. If you designate the Indirect impact Of size change without specifying levels, the default value is ALL. All datanames indirectly affected by a change in the size of a dataname are located.

If you specify the number of levels you want to search, only the results of the desired levels are highlighted. You can see the ripple effect more clearly if you start your investigation by setting Levels to 1, then increasing the number on each subsequent Search action.

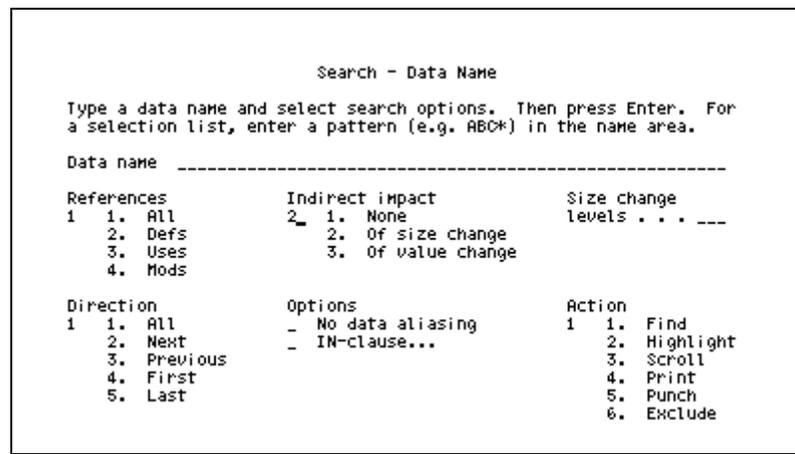
To search levels for indirect impact

- 1 Select Search ▶ Data and press Enter.
- 2 Specify the LOAN-AMT dataname, select the Of size change Indirect impact option, as shown in [Figure 85](#), and press Enter.

Note:

For the purpose of this example, leave the Levels field blank on the Search - Data Name pop-up. This reveals all indirect references at the maximum levels of depth for LOAN-AMT.

Figure 85 • Search - Data Name Pop-up



This information reveals all of the references to LOAN-AMT that would be impacted if you modified the size of LOAN-AMT. To view the results that were not displayed in [Figure 86](#), press PF08 to scroll down. To scroll back up, press PF07.

Figure 86 • Indirect Impact of Size Change to LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> CSR

000281 CALL 'VIAIDEM1' USING MASTER-IN          DATA MOD
000282             MASTER-END-OF-FILE
000283             MASTER-REPORT-DATE.
-----
66 LINES NOT DISPLAYED
000350 MOVE LOAN-AMT TO DET-LOAN-AMT.          DATA REF
-----
1 LINE NOT DISPLAYED
000352 WRITE MAST-RPT FROM DETAIL-LINE1.      DATA REF
-----
2 LINES NOT DISPLAYED
000355 MOVE YEAR-TO-DATE-INTEREST TO DET-YTD-INT. DATA REF
000356 MOVE LAST-BILL-DATE TO DET-LAST-BILL-DATE. DATA USE
000357 WRITE MAST-RPT FROM DETAIL-LINE2.      DATA REF
-----
5 LINES NOT DISPLAYED
000363 MOVE PAYMENT-AMT TO DET-PAYMENT-AMT.   DATA USE
-----
1 LINE NOT DISPLAYED
+-----+ TA US
| ASG0443I 79 DATA REFS: 49 DEFS, 14 USES, 16 MODS, 26 LEVELS FOUND FOR | LAYED
| LOAN-AMT.                                     | TA US
+-----+ LAYED
000384 COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ). DATA MOD

```

If you change the size of LOAN-AMT, you impact the program in 79 different places.

Now, repeat the process again. This time, determine the Indirect impact, Of value change, to LOAN-AMT to determine the ripple effect of a change in value to the same data item.

Figure 87 • Indirect Impact of Value Change to LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                               Program: VIAIDEMO
Command ==>                               Scroll ==> CSR
ASG0443I 23 DATA REFS: 13 DEFS, 8 USES, 2 MODS FOUND FOR LOAN-AMT.
000281 CALL 'VIAIDEM1' USING MASTER-IN          DATA MOD
000282             MASTER-END-OF-FILE
000283             MASTER-REPORT-DATE.
-----
66 LINES NOT DISPLAYED
000350 MOVE LOAN-AMT TO DET-LOAN-AMT.          DATA USE
-----
1 LINE NOT DISPLAYED
000352 WRITE MAST-RPT FROM DETAIL-LINE1.      DATA USE
-----
31 LINES NOT DISPLAYED
000384 COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ). DATA USE
-----
5 LINES NOT DISPLAYED
000390 COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ). DATA USE
-----
12 LINES NOT DISPLAYED
000403 READ MASTERIN                          DATA MOD
-----
18 LINES NOT DISPLAYED
000422 MOVE ZIP-LOAN-AMT TO SUB-LOAN-AMT.      DATA USE
-----
1 LINE NOT DISPLAYED
000424 WRITE MAST-RPT FROM SUB-PRINT.          DATA USE
-----
72 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

As shown on [Figure 87](#), if you change the value of LOAN-AMT, you would impact this program in 23 different locations in the program, VIAIDEMO.

You can limit your impact search to specific types of COBOL statements by using the IN clause. For example, you can search the indirect impact of a size change to a data item for all references in CONditiONal statements. You can also search for the indirect impact of a change in a data value for all modifications in I/O and CALL statements.

This method provides a way to isolate specific or explicit references to the data item quickly and easily. For example, you can specify MODs, USEs, DEFs individually or collectively during your investigation of a data item.

Identifying Data Items Referred to in INPUT/OUTPUT Operations or CALLs

To determine where LOAN-AMT occurs in I/O and CALL statements

- 1 Select Search ► Data and press Enter.
- 2 Type LOAN-AMT in the Data name field, specify None for the Indirect impact field, and select the IN-clause option, as shown in [Figure 88](#).

Figure 88 • Search - Data Pop-up

```

Search - Data Name
Type a data name and select search options. Then press Enter. For
a selection list, enter a pattern (e.g. ABC*) in the name area.

Data name  L_OAN-AMT_-----

References          Indirect impact          Size change
1  1. All           1  1. None           levels . . . ---
   2. Defs         2. Of size change
   3. Uses         3. Of value change
   4. Mods

Direction          Options          Action
1  1. All          - No data aliasing  1  1. Find
   2. Next         / IN-clause...     2. Highlight
   3. Previous
   4. First
   5. Last
  
```

- 3 Press Enter to display the IN-Clause Option pop-up, shown in [Figure 89](#).

Figure 89 • IN-Clause Option Pop-up

```

IN-Clause Option

To restrict the source lines to be considered, type one or more
IN-Clause names. Then press Enter. For a selection list, type a
pattern in the appropriate name field.

Line range . . . : -----
Subset name . . . : IO + CALL_-----
Paragraph Name   : -----
Perfrange name   : -----
Program name . . : -----
User MARK name   : -----
  
```

- 4 Type IO + CALL and press Enter.

Figure 90 shows that one of the references to LOAN-AMT occurs in the CALLED program, VIAIDEM1, and the other in a READ statement. You can open VIAIDEM1 to see how this data item is being modified without terminating your current examination of VIAIDEMO, as described earlier.

Figure 90 • References to LOAN-AMT in Modifications and CALLS

```

File View Search Logic Task List Options Help
-----
Source View                                PROGRAM: VIAIDEMO
Command ==>                                SCROLL ==> CSR
ASG0443I 2 DATA REFS: 2 MODS, FOUND FOR LOAN-AMT IN IO + CALL.
***** ***** TOP OF DATA *****
- - - - - 280 LINES NOT DISPLAYED
000281      CALL 'VIAIDEM1' USING MASTER-IN          DATA MOD
000282                                MASTER-END-OF-FILE
000283                                MASTER-REPORT-DATE.
- - - - - 119 LINES NOT DISPLAYED
000403      READ MASTERIN                            DATA MOD
- - - - - 93 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

This information is significant since it may reveal how data is being passed between programs or systems. you could also use it to identify the impact of changes when you move data from one file structure to another, or to a system.

Determining if the Data Item LOAN-AMT is Modified in the PERFORM Range

Earlier, you determined that LOAN-AMT is modified in the READ statement on line 403.

To verify how and where this statement is used in the PERFORM range

- 1 Select Search ▶ Data and press Enter.
- 2 Specify the LOAN-AMT dataname, select the Mods option and IN-Clause, as shown in Figure 91.

Figure 91 • Search - Data Name Pop-up

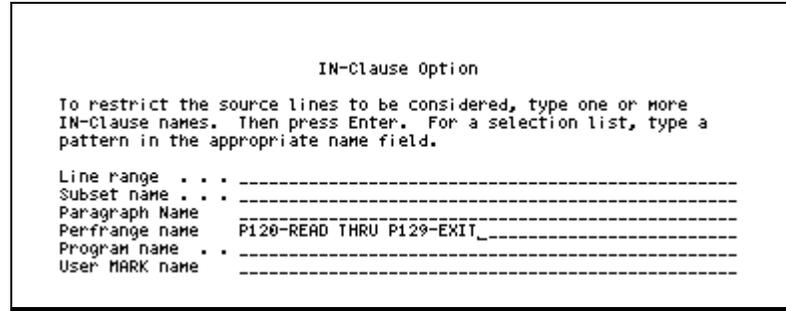
```

Search - Data Name
Type a data name and select search options. Then press Enter. For
a selection list, enter a pattern (e.g. ABC*) in the name area.
Data name  LOAN-AMT
References      Indirect impact      Size change
4  1. All        1  1. None          levels . . . ____
   2. Defs      2. Of size change
   3. Uses      3. Of value change
   4. Mods
Direction      Options      Action
1  1. All       7  No data aliasing  1  1. Find
   2. Next      7  IN-clause...     2. Highlight
   3. Previous  3. Of value change  3. Scroll
   4. First     4. Print
   5. Last      5. Punch
                6. Exclude

```

- 3 Press Enter to display the IN-Clause Option pop-up, shown in [Figure 92](#).

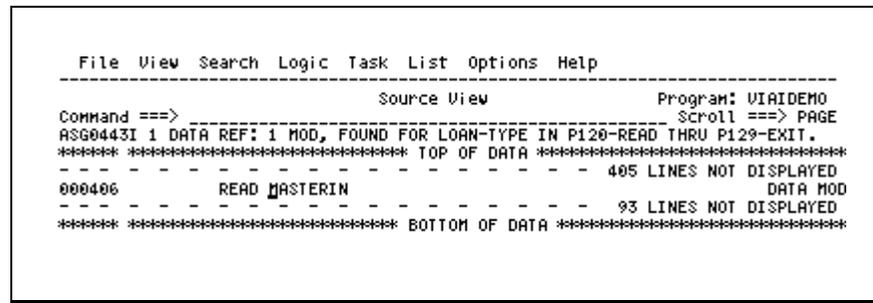
Figure 92 • Search - IN Clause Pop-up



- 4 Specify Perfrange name, P120-READ THRU P129-EXIT and press Enter.

The Source View, shown in [Figure 93](#), displays.

Figure 93 • Modifications of Loan-AMT in P120-READ THRU P129-EXIT



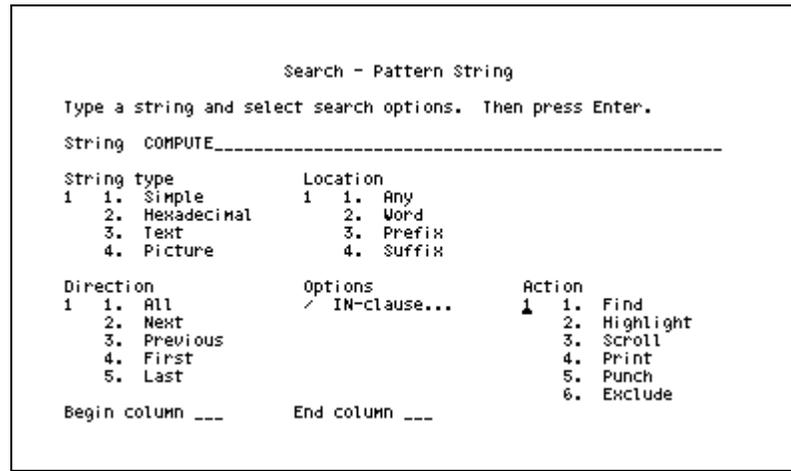
Although the dataname, LOAN-AMT, is not directly referenced in the PERFORM range, P120-READ THRU P129-EXIT, Insight is able to determine that the field is modified by the READ MASTERIN statement. This statement is found as a modification because LOAN-AMT is in the record associated with MASTERIN. To review the data relationship, use the Search - Data feature to locate the definition of LOAN-AMT.

Locating COMPUTEd Values

To locate the COMPUTE statements located in a program

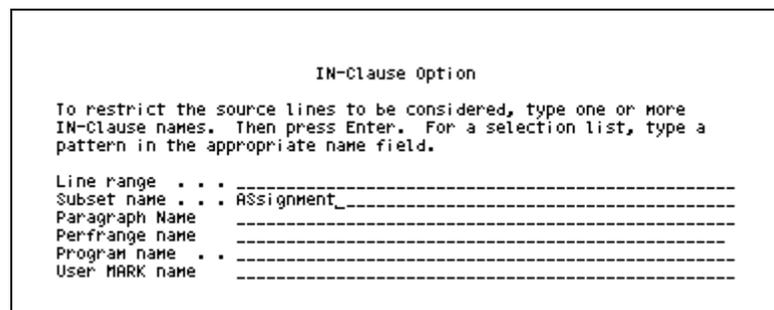
- 1 Select Search ▶ String and press Enter.
- 2 Designate COMPUTE as the string and select the IN-clause Option, as shown in [Figure 94](#).

Figure 94 • Search - Pattern Pop-up



- 3 Specify the Subset name as ASSignment on the IN-Clause Option pop-up, as shown in [Figure 95](#), and press Enter.

Figure 95 • Search - Pattern String Pop-up



This limits the result to only those COMPUTE statements that assign values in VIAIDEMO. [Figure 96](#) indicates that seven patterns containing COMPUTE were found.

Figure 96 • COMPUTE Statements in VIAIDEMO

```

File View Search Logic Task List Options Help
-----
Source View
Command ==> _____ 7 PATTERNS FOUND
                          Scroll ==> CSR
000376 COMPUTE LINE-CNT = ( LINE-CNT + 5 ). PATTERN
----- 7 LINES NOT DISPLAYED
000384 COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ). PATTERN
000385
000386 COMPUTE ZIP-YTD-INT = ( ZIP-YTD-INT PATTERN
----- 1 LINE NOT DISPLAYED
000388 COMPUTE TOTAL-YTD-INT = ( TOTAL-YTD-INT PATTERN
----- 1 LINE NOT DISPLAYED
000390 COMPUTE ZIP-LOAN-AMT = ( ZIP-LOAN-AMT + DET-LOAN-AMT ). PATTERN
----- 45 LINES NOT DISPLAYED
000436 COMPUTE PAGE-CNT = ( PAGE-CNT + 1 ). PATTERN
----- 42 LINES NOT DISPLAYED
000479 COMPUTE AVG-AMT = ( ZIP-LOAN-AMT / ZIP-LOAN-CNT ). PATTERN
----- 17 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

To determine what PERFORM statements are conditionally executed

- 1 Select Search ▶ Subset and press Enter.
- 2 Specify the CONDITIONAL subset, set the Direction to All, and select the IN-clause Option, as shown in [Figure 97](#), and press Enter.

Figure 97 • Search - COBOL Subset Name Pop-up

```

Search - Subset Name

Type a subset name and select search options. Then press Enter.
For a selection list, type a pattern (e.g. ABC*) in the name area.

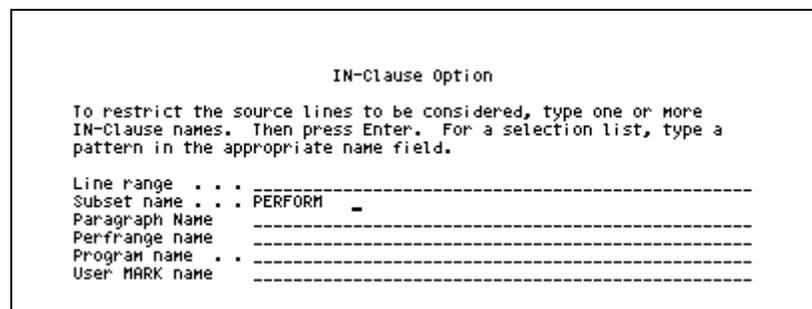
Subset name  CONDitional_____

Direction      Options      Action
1  1. All        / IN-clause...  1  1. Find
   2. Next                               2. Highlight
   3. Previous                               3. Scroll
   4. First                                           4. Print
   5. Last                                           5. Punch
                                           6. Exclude

```

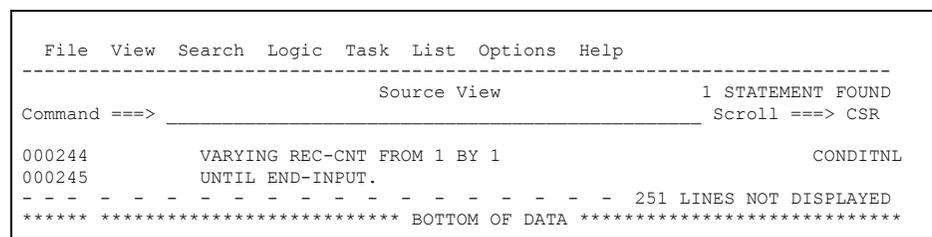
- Specify the PERFORM subset, as shown in [Figure 98](#), and press Enter.

Figure 98 • IN-Clause Option Pop-up



This search results, shown in [Figure 99](#), reveal that REC-CNT is incremented from 1 by a value of 1 in the VARYING statement located at line 244 each time this statement is performed in the program.

Figure 99 • PERFORMs of VARYING Statement



Hint

Use the extended search capability with a Pattern option in Insight to locate an exact string of characters. Many of the dialog pop-ups that use the Search pull-down allow you to enter a Pattern to obtain a list for the Target Name.

A pattern string is a sequence of characters, surrounded by single or double quotes if it contains blanks. You can specify strings of non-alphanumeric characters using the X'string' (hexadecimal), T'string' (test), and P'string' (picture) operands. You may also qualify by using the WORD, PREFIX, or SUFFIX subordinate operands.

Note: _____

For more detailed information on using Pattern Strings in Insight, see the *ASG-Insight Reference Guide*.

The extended search capability of Insight also permits you to specify a data usage. You may locate a MODification (value set or altered), USE (used), DEF (defined), or REF (all of the preceding). You can specify any legal COBOL reference for a data element as a dataname.

You can specify the Data action on the Search pull-down and select any of these types of datanames:

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

Summary: Identifying Data Item Usage Cross-References

- Using Insight, you can quickly determine all of the references to LOAN-AMT, including DEFs, MODs, and USEs. You can perform limited searching in a program to locate the impact of a change to the size and value of the data item would be before you make recommendations for modifications to the program.
- You were able to quickly see how LOAN-AMT was used in INPUT/OUTPUT statements and in CALLED programs. Once the references were located, you could open the CALLED program and examine the logic execution without ending the current Insight analysis session.
- Insight permitted you to quickly determine how the data item was used in PERFORM ranges and in specific COMPUTE statements.
- You can use the Search feature in Insight with the Pattern option. This allows you to quickly locate an exact string of characters. You can also qualify your search using the WORD, PREFIX, or SUFFIX subordinate operands.

Conducting an Analysis of Program Logic

Investigating program logic is potentially the most time consuming program analysis task. Insight provides you with the capability to analyze program logic and identify critical execution paths for further examination:

Before you investigate specific program execution from a start point to a target, you may want to look at the structural representation of the source code in logical order. To do this using Insight, use the View action with the Tree View option.

Insight tags and highlights portions of the source code based on the result of the function requested. There are also options that allow you to exclude search results from the display. The Insight function Reset allows you to remove tags and highlights from the display and to restore all lines of your source code that may have been excluded from the display.

To reset the display

- 1 Select View ► Reset and press Enter. The View - Reset Request pop-up, shown in [Figure 100](#), displays.

Figure 100 • Reset Display to Show all Source Code



- 2 Type a slash (/) in the field marked All on the pop-up. Press Enter to execute the Reset function and return to the editor screen.

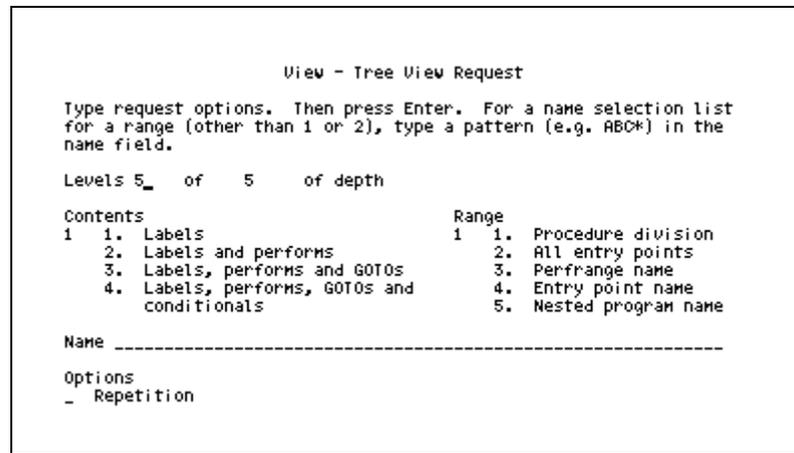
Note: _____

You can use the reset function at any time to remove tags and highlights that are no longer needed and to restore excluded lines to the display.

To view the structural representation of the source code in logical order

- 1 Select View ▶ Tree and press Enter.
- 2 Designate Levels 5 of 5 and accept the defaults for Contents and Range, as shown in [Figure 101](#), and press Enter.

Figure 101 • View - Tree View Request Pop-up



When you select the Contents option 1. Labels, you generate a Tree View display of the program structure using the paragraphs and section labels of control flow verbs. The Labels and performs option provides you with a display of the paragraphs and section labels of control flow verbs and perform range hierarchy for the program. The Labels, performs, and GOTOs option generates a structural display using control flow verbs (for example, PERFORM, GOTO, ALTER) and program labels. When you choose the Labels, performs, GOTOs, and conditionals option, you obtain a Tree View display of the conditional statements that surround control flow statements plus the control flow statements and labels. Select the Repetition Option if you choose to display all duplicate PERFORM ranges.

If you do not select the Repetition option, duplicate PERFORM ranges are flagged as REPEATED and are not expanded unless you use the Zoom in feature from the View pull-down on a PERFORM range.

The default for Name is the program you are currently viewing in Insight. The maximum number of structural levels in the program is automatically displayed to the left of the notation of depth. You must insert the number of those levels you want to view. If you choose to examine fewer than the maximum number of levels in the program, Insight displays the plus sign (+) to indicate that lower levels of program structure are not displayed. To view the lower level structure, you can use the Zoom in and Zoom out options on the View action pull-down.

In [Figure 102](#), Tree View shows all five program hierarchical levels. Tree View tags are displayed on the right of the screen.

Figure 102 • Tree View Screen

```

File View Search Logic Task List Options Help
-----
                                Tree View                                LINE 1 OF 35
Command ==> _____ Scroll ==> PAGE
ASG1582I 5 OF 5 LEVELS DISPLAYED IN THE VIEW.
***** ***** TOP OF DATA *****
000000 PROCEDURE DIVISION.
000002     PROGRAM-INIT.
000003         P005-UAL-PARM.
000004             P999-ABEND-PROGRAM.
000004             P999-ABEND-PROGRAM. REPEATED
000003         P005-EXIT.
000003         P010-OPEN.
000003         P019-EXIT.
000003         P155-CL-SUBTOT.
000004             P159-EXIT.
000003         P120-READ.
000004             P170-FINAL.
000005             P999-ABEND-PROGRAM. REPEATED
000004             P179-EXIT.
000004             P200-CLOSE.
000005             P129-EXIT.
000004             P129-EXIT. REPEATED
000004 +             P120-READ. REPEATED

```

Note:

See the online help or the *ASG-Insight Reference Guide* for more information on Tree View tags.

To view the logical execution paths that lead to the statement

This procedure examines all execution paths from the beginning of the PROCEDURE DIVISION to the COMPUTE statement on line 384.

- 1 Press PF03/PF15 to return to Source View from Tree View.
- 2 Select Logic ► Label and press Enter.

- 3 Specify PROCEDURE as Label name, 384 as Start line/label, Backward (all) as Direction, Show execution path(s) from starting point to reference(s) as Action as shown on [Figure 103](#), and press Enter. Program execution displays in a backward direction from the source of the problem on line 384 to the beginning of the Procedure Division.

Figure 103 • Logic Order Search - Label Name Pop-up

```
Logic Order Search - Label Name

To find label references in execution order, type the label and
starting point, and select search options. Then press Enter. For a
label name selection list, type a pattern (e.g. ABC*) in the
appropriate input area.

Label name  PROCEDURE_____
Start line/label (if any) 384_____

Direction          Options
4  1. Next          - Trace IN-clause...
   2. Previous
   3. Forward (all)
   4. Backward (all)

Action
_2  1. Show label name reference(s)
    2. Show execution path(s) from starting point to reference(s)
    3. Trace (single-step) from starting point to reference(s)
```

If you choose Data on the pull-down, you can view data Definitions, Uses, Modifications, or Analysis Type. When you select the Label action on the pull-down, you can follow program logic to a specific label name. Use the Subset action on the Logic pull-down to view program logic to a COBOL language subset. By selecting the Perform action, you can examine logic within a PERFORM range. The Mark action provides you with the ability to look at program execution in a specified set of marked lines. The Program action on the Logic pull-down traces the logic of the program and include pertinent DATA DIVISION/WORKING-STORAGE data elements. If you choose the Line action, you can follow the execution of the program within a given range of lines.

Examine the statements in the execution path from the Procedure Division to line 384.

The NETWORK tags on the right of the screen indicate that each statement is contained in the designated execution path. In all, there are 126 statements in this NETWORK, as shown in [Figure 104](#). You can scroll down through the statements in the NETWORK to view them online or you can rename them for later examination using the Mark selection on the Options - Scratchpad pull-down.

Figure 104 • Source View - Procedure Division to Line 384

```

File View Search Logic Task List Options Help
-----
Source View                               125 STATEMENTS FOUND
Command ==> _____ Scroll ==> CSR

***** ***** TOP OF DATA *****
- - - - - 230 LINES NOT DISPLAYED
000231 PROCEDURE DIVISION. NETWORK
- - - - - 5 LINES NOT DISPLAYED
000237 PERFORM PROGRAM-INIT. NETWORK
- - - - - 5 LINES NOT DISPLAYED
000243 PERFORM P000-NEXT THRU P000-EXIT NETWORK
000244 VARYING REC-CNT FROM 1 BY 1 NETWORK
000245 UNTIL END-INPUT. NETWORK
- - - - - 7 LINES NOT DISPLAYED
000253 PROGRAM-INIT. NETWORK
- - - - - 1 LINE NOT DISPLAYED
000255 IF DEBUG-PARM = 'TEST' NETWORK
000256 READY TRACE. NETWORK
- - - - - 1 LINE NOT DISPLAYED
000258 PERFORM P005-VAL-PARM NETWORK
000259 THRU P005-EXIT. NETWORK
- - - - - 1 LINE NOT DISPLAYED
000261 PERFORM P010-OPEN NETWORK

```

Note:

If you want to include the definitions of the data items in the COMPUTE statement, select Program from the Logic pull-down, and enter the program name in the Program name field. The selections in the Direction and Action fields would be the same.

Hint

Creating NETWORKs and SUBNETs. When you examine program execution using the Logic action, you automatically create a NETWORK and SUBNETs.

A NETWORK contains all of the statements in all of the execution paths to all reachable targets.

A SUBNET contains all of the statements in all of the execution paths to a specified target.

Each time you examine source code execution in this way, you create a new NETWORK and SUBNETs replacing the previous ones. You can save the results of your analysis at anytime using these Insight features:

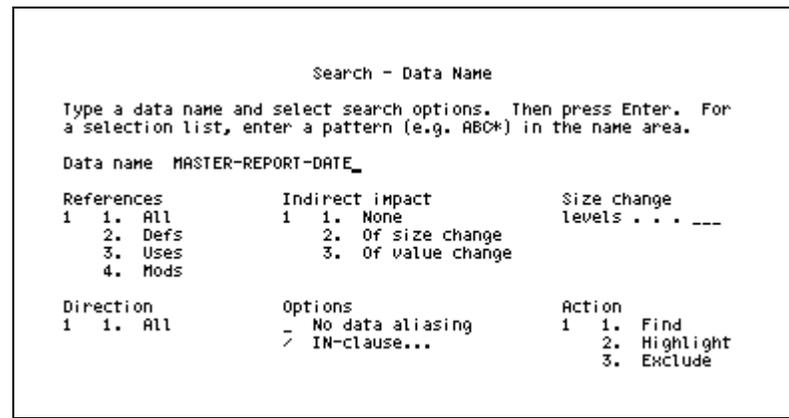
- OPTIONS (SCRATCHPAD pop-up)
- COPY (Action)
- MERGE (Action)
- PRINT (SEARCH pop-ups)
- PUNCH (SEARCH pop-ups)

To determine where a data item is referenced in the CALLED program

Assume that you need to know where MASTER-REPORT-DATE received the data it is passing to the CALLED program, VIAIDEM1. How is the field, MASTER-REPORT-DATE, set prior to the CALL statement?

- 1 Select Search ▶ Data and press Enter. The Search - Data pop-up, shown in [Figure 105](#), displays.
- 2 Specify the MASTER-REPORT-DATE dataname, specify References as All, select the IN-clause option, and press Enter.

Figure 105 • Search - Data Pop-up



- 3 Specify the Subset name CALL, as shown in [Figure 106](#), and press Enter.

Figure 106 • In-Clause Option Pop-up

```

                                IN-Clause Option

To restrict the source lines to be considered, type one or more
IN-Clause names. Then press Enter. For a selection list, type a
pattern in the appropriate name field.

Line range . . . : _____
Subset name . . . : CALL_____
Paragraph Name   : _____
Perfrange name  : _____
Program name . . : _____
User MARK name  : _____
    
```

[Figure 107](#) shows that MASTER-REPORT-DATE displays in the CALL statement on line 283.

Figure 107 • Source View - CALLED Programs in VIAIDEMO

```

File View Search Logic Task List Options Help
-----
                                Source View                                Program: VIAIDEMO
Command ==>> _____ Scroll ==>> CSR
ASG0443I 1 DATA REF: 1 USE, FOUND FOR MASTER-REPORT-DATE IN CALL.
***** ***** TOP OF DATA *****
----- 280 LINES NOT DISPLAYED
000281     CALL 'VIAIDEM1' USING MASTER-IN
000282                                     MASTER-END-OF-FILE
000283                                     MASTER-REPORT-DATE.                                DATA USE
----- 213 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
    
```

To examine the logic from line 283 to the previous modification of the data item

- 1 Select Logic ► Data and press Enter. The Logic Order Search - Data Name pop-up, shown in [Figure 108](#) displays.

- 2 Type MASTER-REPORT-DATE as the dataname and 283 as the Start line/label, select the Mods only References option, set the Direction to Backward (all), select the Show dataname reference(s) action, as shown in [Figure 108](#). Press Enter.

Figure 108 • Logic Order Search - Data Name Pop-up

```

                                Logic Order Search - Data Name

To find data name references in execution order, type the name and
starting point, and select search options. Then press Enter. For a
data or label name selection list, type a pattern (e.g. ABC*) in the
appropriate input area.

Data name MASTER-REPORT-DATE
Start line/label (if any) -----

References          Direction          Options
3  1. All           4_ 1. Next           - Trace
   2. Uses only    2. Previous         - IN-clause...
   3. Mods only    3. Forward (all)
                   4. Backward (all)

Action
1  1. Show data name reference(s)
   2. Show execution path(s) from starting point to reference(s)
   3. Trace (single-step) from starting point to reference(s)
    
```

The results shown in [Figure 109](#) indicate the target data reference as the only modification of MASTER-REPORT-DATE, looking backward from the first CALL statement.

Figure 109 • Source View - Modification of MASTER-REPORT-DATE

```

File View Search Logic Task List Options Help
-----
                                Source View          1 MOD(S) REACHABLE
Command ==>> _____ Scroll ==>> CSR

- - - - - 430 LINES NOT DISPLAYED
000431      MOVE SPACES TO RPT-HDG-LINE1.          DATA MOD
- - - - - 65 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
    
```

To modify the field after the CALL statement

This time, follow the program logic in a forward direction to determine where MASTER-REPORT-DATE is modified following the CALL statement.

- 1 Select Logic ▶ Data and press Enter.
- 2 Type MASTER-REPORT-DATE as the dataname and line 283 as the Start line/label, set References to Mods only, Direction to Forward, and Action to Show dataname reference(s), as shown in [Figure 110](#), then press Enter.

Figure 110 • Logic - Data Pop-up

```

Logic Order Search - Data Name

To find data name references in execution order, type the name and
starting point, and select search options. Then press Enter. For a
data or label name selection list, type a pattern (e.g. ABC*) in the
appropriate input area.

Data name MASTER-REPORT-DATE
Start line/label (if any) -----
References          Direction          Options
3  1. All           4  1. Next              - Trace
   2. Uses only    2. Previous           - IN-clause...
   3. Mods only    3. Forward (all)
                   4. Backward (all)

Action
1  1. Show data name reference(s)
   2. Show execution path(s) from starting point to reference(s)
   3. Trace (single-step) from starting point to reference(s)

```

The next modification of MASTER-REPORT-DATE occurs on line 431, where spaces are added to the report heading, as [Figure 111](#) reveals.

Figure 111 • Execution Path to Next Mod of MASTER-REPORT-DATE

```

File View Search Logic Task List Options Help
-----
Source View          1 MOD(S) REACHABLE
Command ==> _____ Scroll ==> CSR

***** ***** TOP OF DATA *****
-- -- -- -- -- -- -- -- -- -- -- 430 LINES NOT DISPLAYED
000431      MOVE SPACES TO RPT-HDG-LINE1.          DATA MOD
-- -- -- -- -- -- -- -- -- -- -- 65 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

To review the code executed in the path to this modification

- 1 Repeat [step 1](#).
- 2 Perform [step 2](#), but set the Action to Show execution path(s) from starting point to reference(s), and press Enter.

The statements in this execution path make up the NETWORK, as shown in [Figure 112](#)

Figure 112 • Execution Path to Next Modification of MASTER-REPORT-DATE

```

File View Search Logic Task List Options Help
-----
Source View                                100 STATEMENTS FOUND
Command ==>> _____ Scroll ==>> CSR

***** ***** TOP OF DATA *****
- - - - - 242 LINES NOT DISPLAYED
000243     PERFORM P000-NEXT THRU P000-EXIT                NETWORK
000244         VARYING REC-CNT FROM 1 BY 1                NETWORK
000245         UNTIL END-INPUT.
- - - - - 33 LINES NOT DISPLAYED
000279 P000-NEXT.                                        NETWORK
000280
000281     CALL 'VIAIDEM1' USING MASTER-IN                NETWORK
000282         MASTER-END-OF-FILE
000283         MASTER-REPORT-DATE.
000284
000285     PERFORM P100-PRINT                                NETWORK
000286         THRU P119-EXIT.
000287
000288     PERFORM P120-READ                                  NETWORK
000289         THRU P129-EXIT.
000290
000291 P000-EXIT.                                        NETWORK

```

To examine program execution step-by-step and determine the cause of a known problem

Assume that you are examining how a COMPUTE statement, which uses the data value DET-LOAN-AMT, receives its value. This might be a resolution to an Abend or just an incorrect total on a report. For the purpose of this example, assume that you suspect that the problem occurs at line 384.

To understand how the value was assigned to DET-LOAN-AMT prior to line 384, you decide to identify the last place that this field was modified.

- 1 Select Logic ▶ Data and press Enter. The Source View pop-up, shown in [Figure 113](#), displays.

Figure 113 • Previous Modification of DET-LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                                1 MOD(S) REACHABLE
Command ==>> _____ Scroll ==>> CSR

***** ***** TOP OF DATA *****
- - - - - 349 LINES NOT DISPLAYED
000350     MOVE LOAN-AMT TO DET-LOAN-AMT.                DATA MOD
- - - - - 146 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

- 2 Complete these fields:
 - a Type DET-LOAN-AMT as the dataname.
 - b Specify the Start line/label as 384.
 - c Select the Mods only References option.
 - d Set the Direction to Previous.
 - e Select the Show dataname reference(s) action.
- 3 Press Enter.

To examine all statements in the execution path

Now that you have identified the possible targets from line 384, you can examine all of the statements in the execution path to the previous modification of the data item, DET-LOAN-AMT.

- 1 Select Logic ► Data and press Enter. The Source View pop-up, shown in [Figure 114](#), displays.

Figure 114 • Execution Path from Line 384 to Pervious Modification

```

File View Search Logic Task List Options Help
-----
Source View                               24 STATEMENTS FOUND
Command ===> _____ Scroll ===> CSR
000350 MOVE LOAN-AMT TO DET-LOAN-AMT. NETWORK
000351 MOVE LOAN-START-DATE TO DET-START-DATE. NETWORK
000352 WRITE MAST-RPT FROM DETAIL-LINE1. NETWORK
000353
000354 MOVE NAME TO DET-NAME. NETWORK
000355 MOVE YEAR-TO-DATE-INTEREST TO DET-YTD-INT. NETWORK
000356 MOVE LAST-BILL-DATE TO DET-LAST-BILL-DATE. NETWORK
000357 WRITE MAST-RPT FROM DETAIL-LINE2. NETWORK
000358
000359 MOVE ADDRESS1 TO DET-ADDRESS. NETWORK
000360 MOVE AREA-CODE TO DET-AREA-CODE. NETWORK
000361 MOVE EXCHANGE TO DET-EXCHANGE. NETWORK
000362 MOVE PHONE-NUMBER TO DET-PHONE-NUMBER. NETWORK
000363 MOVE PAYMENT-AMT TO DET-PAYMENT-AMT. NETWORK
000364 MOVE LOAN-TYPE TO DET-LOAN-TYPE. NETWORK
000365 WRITE MAST-RPT FROM DETAIL-LINE3. NETWORK
000366
000367 MOVE CITY TO DET-CITY. NETWORK
000368 MOVE STATE TO DET-ST. NETWORK

```

- 2 Complete these fields:
 - a Type DET-LOAN-AMT as the dataname.
 - b Specify the Start line/label as 384.

- c Select the Mods only References option.
 - d Set the Direction to Previous.
 - e Select the Show execution path(s) from starting point to reference(s) action.
- 3 Press Enter. Insight automatically created a Network of the statements executed from the previous modification of DET-LOAN-AMT. Since DET-LOAN-AMT contains the values set by LOAN-AMT, you now need to locate where those values are set in LOAN-AMT.

To locate the data item values

- 1 Select Logic ▶ Data and press Enter. The Logic Order Search pop-up displays.
- 2 Type LOAN-AMT as the dataname, line 350 as the Starting point, Direction as Previous, Action as Show dataname reference(s), then press Enter.

These results, shown in [Figure 115](#), indicate that one of the previous data modifications occurs in the CALLED program, VIAIDEM1, and the other in the reading of file, MASTERIN. You could now use the procedure described earlier in this guide to open VIAIDEM1 and view how the data item is modified in VIAIDEM1.

Figure 115 • Execution Path from DET-LOAN-AMT Modification to LOAN-AMT

```

File View Search Logic Task List Options Help
-----
Source View                               2 MOD(S) REACHABLE
Command ==> _____ Scroll ==> CSR

***** ***** TOP OF DATA *****
000281 CALL 'VIAIDEM1' USING MASTER-IN          DATA MOD
000282          MASTER-END-OF-FILE
000283          MASTER-REPORT-DATE.
- - - - - 119 LINES NOT DISPLAYED
000403 READ MASTERIN                          DATA MOD
- - - - - 93 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

When you use the Logic function in Insight to examine the execution paths of your program, the process may be suspended at either Decision Points or Stop Points.

An Execution Decision is one in which you choose an execution path in the program. These Decision Points occur on conditional statements (IF statements, for example).

A Viewing Decision is one where you choose to look at a particular section of code that is part of an execution sequence. These Decision Points occur on PERFORM statements.

A Stop Point occurs at any CALL statement. This lets you know that during actual execution, another program will be invoked. You can use the Logic function to examine program execution in the CALLED subprogram without leaving your current Source View session, providing the CALLED subprogram is on the AKR.

When your display must be scrolled because it is full or a branch takes the trace to another part of your program, a Stop Point is also encountered.

To trace the execution path from the paragraph line xxx forward to all program exits

- 1 Select Options ▶ Modes and press Enter.
- 2 Turn XMODE OFF on the pop-up and press Enter. This enables you to view all of the source code when tracing execution paths.
- 3 Press PF03/PF15.
- 4 Select View ▶ Reset and press Enter.
- 5 Select All on the pop-up and press Enter.
- 6 Select Logic ▶ Subset and press Enter. The Logic Order Search - Line Number Range pop-up displays.
- 7 Type EXIT as the subset, designate Start line/label as 329, set the Direction to Forward, and set the Action to Trace (single-step) from starting point to reference(s) action, as shown in [Figure 116](#). Press Enter.

Figure 116 • Logic Order Search - Line Number Range Pop-up

```
Logic Order Search - Line Number Range

To find line range references in execution order, type the range and
starting point, and select search options. Then press Enter. For a
label name selection list, type a pattern (e.g. ABC*) in the "start"
field.

Line range EXIT_____
start line/label (if any) 329_____

Direction          Options
3 1. Next           - Trace IN-clause...
  2. Previous
  3. Forward (all)
  4. Backward (all)

Action
3_ 1. Show line number range reference(s)
   2. Show execution path(s) from starting point to reference(s)
   3. Trace (single-step) from starting point to reference(s)
```

The third line of the long message area, shown in [Figure 117](#), indicates that a Decision Point was reached at line 332 and that two options are available. This is an Execution Decision based on a conditional.

Figure 117 • Execution Path to Target(s)

```

File View Search Logic Task List Options Help
-----
Command ==> Source View Program: VIAIDEMO
ASG0456I DECISION: 332; 2 OPTIONS: 333, 334. Scroll ==> CSR
000327 * AND SKIP TO THE TOP OF THE PAGE.
000328
000329 P105-NOT-FIRST-TIME. START
000330
000331 MOVE ZIP-CODE TO HLD-ZIP.
000332 IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX DECISION
000333 NEXT SENTENCE OPTION 1
000334 ELSE OPTION 2
000335 PERFORM P150-SUBTOT
000336 THRU P169-EXIT
000337 MOVE HLD-ZIP-PREFIX TO CUR-PREFIX.
000338
000339 IF LINE-CNT GREATER THAN 53 FALLTHRU
000340 PERFORM P160-HDG
000341 THRU P169-EXIT. FALLTHRU
000342
000343 P110-CONTINUE-PRINT.
000344
000345 CALL 'DBAREAD2' USING DBA-DEPT-CODE.

```

Note: _____
 The conditional and first line of each option are tagged.

To make a selection on the Trace Decision Options pop-up

- 1 Press Enter to view to the Trace Decision Options pop-up.

The Trace Decision Options pop-up presents you with options for proceeding from the current Decision Point. The Trace Decision Options pop-up, shown in [Figure 118](#), displays the source for the decision, type of branch, line number, and first statement of each destination. An asterisk (*) next to an option indicates that you have already traced that option.

Figure 118 • Trace Decision Options Pop-up

```

Trace Decision Options
Command ==> Scroll ==> CSR

Trace of: EXITS
Direction: FORWARD
Paragraph: 000329 P105-NOT-FIRST-TIME.
Decision: 000332 IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX

Option Info indicates TARGET's and '*' for code already TRACE'd

Select one of the following decision options: Option
S Option Info
-----
- 1. TRUE (000333 NEXT SENTENCE)
- 2. FALSE (000334 ELSE)
***** BOTTOM OF DATA *****

```

- 2 Type S on the Trace Decision Options pop-up next to your choice and press Enter.
- 3 Type the number of your choice in the primary command input area and press Enter.

Note:

Insight returns you to the Source View screen and proceeds to the next Decision Point. If you elect to stop the single-step trace at the Trace Decision Menu and return to Source View, press PF03/PF15.

Assume now that you elect Option 1, the TRUE condition in this execution path.

- 4 Select Option 1 and press Enter. The Source View pop-up displaying the Trace Decision Point ([Figure 119](#)) displays.

Figure 119 • Trace Decision Point

```

File View Search Logic Task List Options Help
-----
Source View                                Program: VIAIDEMO
Command ==>                                Scroll ==> CSR
ASG0456I DECISION: 339; 2 OPTIONS: 340, 343.
000331 MOVE ZIP-CODE TO HLD-ZIP.
000332 IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333     NEXT SENTENCE
000334 ELSE
000335     PERFORM P150-SUBTOT
000336         THRU P169-EXIT
000337     MOVE HLD-ZIP-PREFIX TO CUR-PREFIX.
000338
000339 IF LINE-CNT GREATER THAN 53              DECISION
000340     PERFORM P160-HDG                      OPTION 1
000341         THRU P169-EXIT.                  FALLTHRU
000342
000343 P110-CONTINUE-PRINT.                    OPTION 2
000344
000345     CALL 'DBAREAD2' USING DBA-DEPT-CODE.
000346
000347 * PRINT THE MASTER FILE DETAIL REPORT BODY FOR THIS MASTER
000348
000349     MOVE CLIENT-ID TO DET-NUMBER.

```

The trace continues until you reach the next Decision Point, another Execution Decision.

- 5 Select Option 1 and press Enter.

You can elect to suspend your trace and perform a Search for LINE-CNT. Once you have located the references to LINE-CNT, you can return to the point at which you suspended your trace by typing RTRACE in the primary command input area and pressing Enter.

[Figure 120](#) shows that a Viewing Decision is encountered at the Decision Point at line 340. The PERFORMed code is part of the execution sequence. If you elect Option 1, you choose not to look at the PERFORMed code (for example, this part of your program may contain a standard housekeeping routine which you do not need to review). If you elect Option 2, you display the PERFORMed source code.

Figure 120 • Trace Viewing Decision

```

File View Search Logic Task List Options Help
-----
Source View                                     Program: VIAIDEMO
Command ==>                                     Scroll ==> CSR
ASG0456I DECISION: 340; 2 OPTIONS: 343, 430.
000331 MOVE ZIP-CODE TO HLD-ZIP.
000332 IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX
000333 NEXT SENTENCE
000334 ELSE
000335 PERFORM P150-SUBTOT
000336 THRU P169-EXIT
000337 MOVE HLD-ZIP-PREFIX TO CUR-PREFIX.
000338
000339 IF LINE-CNT GREATER THAN 53                                FALLTHRU
000340 PERFORM P160-HDG                                         DECISION
000341 THRU P169-EXIT.                                         FALLTHRU
000342
000343 P110-CONTINUE-PRINT.                                     OPTION 1
000344
000345 CALL 'DBAREAD2' USING DBA-DEPT-CODE.
000346
000347 * PRINT THE MASTER FILE DETAIL REPORT BODY FOR THIS MASTER
000348
000349 MOVE CLIENT-ID TO DET-NUMBER.

```

To display PERFORMed source code

- 1 Select Option 2 and press Enter. The Source View screen, shown in [Figure 121](#), displays.

Figure 121 • PERFORM Range Source Code

```

File View Search Logic Task List Options Help
-----
Source View                                     Program: VIAIDEMO
Command ==>                                     Scroll ==> CSR
ASG0318I RETURNING TO 343 FROM P160-HDG THRU P169-EXIT.
000428 P159-EXIT.
000429 EXIT.                                                    RET/FALL
000430 P160-HDG.
000431 MOVE SPACES TO RPT-HDG-LINE1.
000432 MOVE CUR-PREFIX TO HDG-ZIP-PREFIX.
000433 MOVE PAGE-CNT TO MASTER-REPORT-PAGE-CNT.
000434 WRITE MAST-RPT FROM RPT-HDG-LINE1.
000435 WRITE MAST-RPT FROM RPT-HDG-LINE2.
000436 COMPUTE PAGE-CNT = (PAGE-CNT + 1).
000437 MOVE 4 TO LINE-CNT.                                         FALLTHRU
000438
000439 P169-EXIT.
000440 EXIT.                                                    STOP PNT
000441 SKIP3
000442 *****
000443 * PRINT THE GRAND TOTALS OF THE LOAN PORTFOLIO *
000444 *****
000445
000446 P170-FINAL.

```

- 2 Press PF08 to view the Stop Point encountered at line 440. At this point, you elect to stop and examine all of the statements that are part of your trace.

The Track is a series of scratchpad entries generated when you trace the execution path from the start point to the target(s). The Track contains all of the statements in the execution path chosen during a trace.

Each time you use single-step through program logic using Insight, a Track is generated automatically and replaces any already existing on the scratchpad. To save a Track, you must use the Options function and Copy feature to save it on the Scratchpad.

To view all of the statements on the current Track

- 1 Select Search ► Mark and press Enter.
- 2 On the Source View pop-up, shown in [Figure 122](#), type TRACK as the mark name and press Enter.

Figure 122 • Track of Execution Path Statements

```
File View Search Logic Task List Options Help
-----
Source View                               18 STATEMENTS FOUND
Command ==>                               Scroll ==> CSR
000329 P105-NOT-FIRST-TIME.                TRACK
000330                                     TRACK
000331     MOVE ZIP-CODE TO HLD-ZIP.        TRACK
000332     IF HLD-ZIP-PREFIX EQUAL CUR-PREFIX TRACK
000333     NEXT SENTENCE                    TRACK
000334     ELSE
000335     PERFORM P150-SUBTOT
000336     THRU P169-EXIT
000337     MOVE HLD-ZIP-PREFIX TO CUR-PREFIX.
000338
000339     IF LINE-CNT GREATER THAN 53        TRACK
000340     PERFORM P160-HDG                    TRACK
000341     THRU P169-EXIT.                  FALLTHRU
000342
000343 P110-CONTINUE-PRINT.                   TRACK
000344
000345     CALL 'DBAREAD2' USING DBA-DEPT-CODE. TRACK
000346
000347 * PRINT THE MASTER FILE DETAIL REPORT BODY FOR THIS MASTER
```

To invoke the contents of a paragraph

Assume that you are interested in the contents of paragraph P155-CL-SUBTOT.

- 1 Select View ► Paragraph X-ref and press Enter.
- 2 Type P155-CL-SUBTOT as the target name, specify the target type, and label name, as shown in [Figure 123](#). Set the Direction option to previous to see how P155-CL-SUBTOT is invoked, and press Enter.

Figure 123 • View - Paragraph Cross-Reference

```

View - Paragraph Cross-Reference Request

To list transfers of control to or from paragraphs in the program,
type a target name and select a target type and direction. Then
press Enter. For a name selection list for a target type (other
than types 1 and 6), type a pattern (e.g. ABC*) in the target name
input field.

Target name P155-CL-SUBTOT-----
Target type          Direction
7_ 1. None - use cursor 1 1. Previous
   2. Mark name          2. Next
   3. Perfrange name
   4. Program name
   5. Subset name
   6. Line range
   7. Label name
    
```

The results, shown in [Figure 124](#), reveal paragraphs that transfer control to P155-CL-SUBTOT as well as the type of transfers that are used in those paragraphs.

Figure 124 • Paragraph Cross-Reference Screen

```

View - Paragraph Cross Reference
Command ==> _____ Scroll ==> PAGE
Target(s): LABEL P155-CL-SUBTOT
Direction: PREVIOUS

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

$ Comes from          How          Target paragraph(s)
-----
_ PROGRAM-INIT        PERFORM        P155-CL-SUBTOT
_ P150-SUBTOT         FALLTHRU       " "
***** BOTTOM OF DATA *****
    
```

Now, you can examine paragraph contents of P150-SUBTOT, which falls through to P155-CL-SUBTOT.

- 3 Type S to the left of P150-SUBTOT and press Enter. The source view shown in [Figure 125](#) displays.

Figure 125 • Paragraph P150-SUBTOT

```

File View Search Logic Task List Options Help
-----
Source View                                Program: VIAIDEMO
Command ===>                               Scroll ===> CSR

000420 P150-SUBTOT.
000421     MOVE ZIP-LOAN-CNT TO SUB-LOAN-CNT.
000422     MOVE ZIP-LOAN-AMT TO SUB-LOAN-AMT.
000423     MOVE ZIP-YTD-INT TO SUB-YTD-INT.
000424     WRITE MAST-RPT FROM SUB-PRINT.                                FALLTHRU
000425 P155-CL-SUBTOT.
000426     MOVE 0 TO SUB-LOAN-CNT, SUB-LOAN-AMT, SUB-YTD-INT.
000427     GO TO P159-EXIT.
000428 P159-EXIT.
000429     EXIT.                                                        RET/FALL
000430 P160-HDG.
000431     MOVE SPACES TO RPT-HDG-LINE1.
000432     MOVE CUR-PREFIX TO HDG-ZIP-PREFIX.
000433     MOVE PAGE-CNT TO MASTER-REPORT-PAGE-CNT.
000434     WRITE MAST-RPT FROM RPT-HDG-LINE1.
000435     WRITE MAST-RPT FROM RPT-HDG-LINE2.
000436     COMPUTE PAGE-CNT = (PAGE-CNT + 1).
000437     MOVE 4 TO LINE-CNT.                                          FALLTHRU
000438

```

The previous example can also be applied to quickly and accurately cross-reference any COBOL language subset. For example, you could perform a paragraph cross-reference on the INPUT subset name to reveal all paragraphs containing input statements and how they are reached. Using the technique demonstrated earlier in this chapter, in the section on Saving the Results of Analysis, you could print out the list of these paragraphs for future program walk-throughs.

Hint

You can examine program logic without designating a target. When you specify a direction with these functions on a pop-up, you get these results:

- FORWARD/NEXT follows program execution to locate the target(s) from the Start Point FORWARD to the program EXITs.
- BACKWARD/PREVIOUS follows program execution to locate the target(s) from the Start Point BACKWARD to the Procedure Division or Entry statement.

Summary: Conducting an Analysis of Program Logic

- You used Insight to look at the hierarchical structure of this program and to view the execution sequence from a compute statement backward to a target to determine how the value was set.
- You identified how MASTER-REPORT-DATE was populated and how it was modified after a CALL statement.
- You identified the execution path from the Procedure Division to the first READ statement, where a known bug existed in the initialization routine.
- You determined how LOAN-AMT is used in execution sequence after the READ statement.
- You determined how the SUBTOTAL parameter gets invoked.

Applying Quality Assurance and Standards

Code inspections are the primary means for ensuring adherence to programming quality standards. Assume that you are the moderator for the quality assurance walk-through before an actual inspection is conducted. Your department has new standards. Before you begin your quality assurance code inspection, you decide to record your analysis as a SCRIPT to run on several other programs in the Loan Processing Department. You recognize that using a SCRIPT against other programs online saves you time and increases your confidence in the results over manual inspection.

Saving an Analysis

If you want to exclude unwanted lines from the display, turn XMODE ON, as shown in ["Getting a Program Overview" on page 55](#).

To save the steps you perform during a Quality Assurance code inspection using the SCRIPT feature

- 1 Select Options ► Modes and press Enter to display the Options - Processing Modes pop-up shown in [Figure 126](#).

Figure 126 • Options - Processing Modes Pop-up

```

                                Options - Processing Modes
Command ==> _____ Scroll ==> PAGE
Type option settings. Then press PF3/15 (END) to save options and exit.
-----
Option      Set      Description
-----
LEARN       OFF     Display internally generated Primary Commands
SCRIPT      OFF     Script facility is disabled
XMODE       ON      Exclude all lines before executing Primary Commands

```

- 2 Set SCRIPT mode ON on the Options - Set Processing Modes pop-up to save the steps in your quality assurance analysis. Press PF03/PF15.

You can use the SCRIPT feature on any analyzed program in the AKR.

- 3 When you have completed your analysis, repeat [step 1](#) and [step 2](#) and set SCRIPT mode OFF on the Options - Set Processing Modes pop-up. Press PF03/PF15.

To execute each command in order in a SCRIPT

- 1 Select File ► Execute script and press Enter. The File - Execute Script pop-up, shown in [Figure 127](#), displays.

Figure 127 • File - Execute Script File Pop-up

```

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

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

Data set name _____

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

```

- 2 Type `USERID.INSXXX01.VIASCRIPT` in the Data set name field, select the Single step option, and press Enter.

During execution of the script you can change a command, if desired, when it displays in the primary command input area.

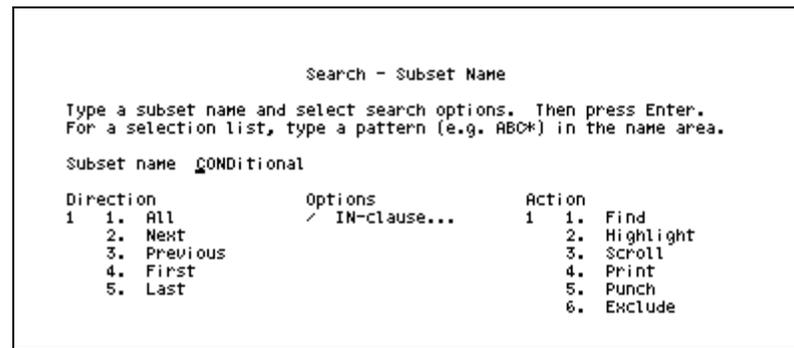
- 3 To bypass a command, delete the contents of the primary command input area and press Enter.

IF and ELSE Statements

To determine if all IF statements are indented and aligned with their respective ELSE statements

- 1 Select Search ► Subset and press Enter.
- 2 Designate the Subset name, CONditionals, select no Options, as shown in [Figure 128](#), and press Enter.

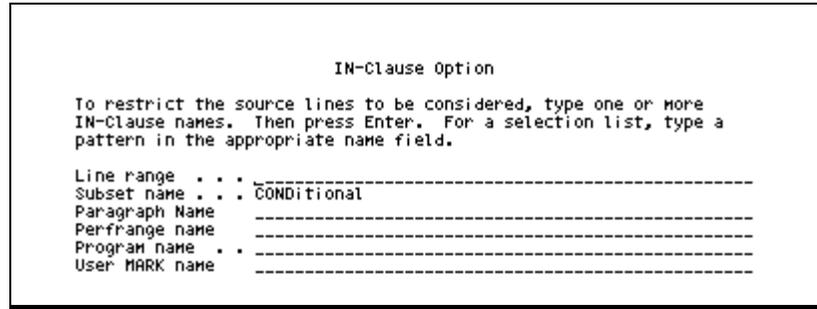
Figure 128 • Search - COBOL Subset Pop-up



As a result of this subset option, all statements or parts of statements that conditionally change the flow of control in a program, like IF, ELSE, and WHEN, for example, are highlighted.

On the IN-Clause Option pop-up, shown in [Figure 131](#), type CONDitionals as the subset name, and press Enter.

Figure 131 • IN-Clause Option Pop-up



In this example, no negative IF statements were used in the program, as indicated by the short message, NO PATTERNS FOUND.

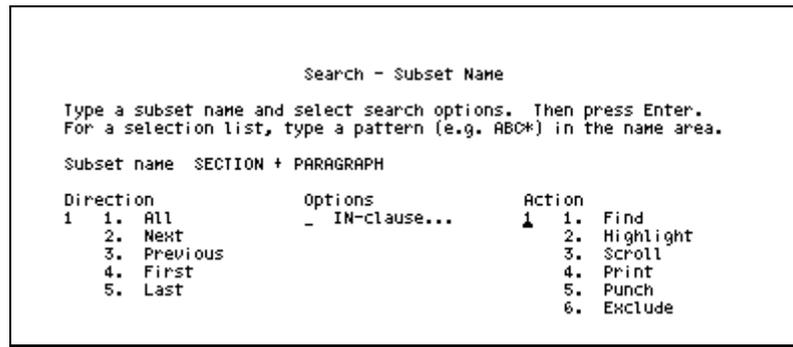
Paragraph Names and Numbering Schemes

To view the section and paragraph names and numbering scheme used in this program

Repeat these same steps to view paragraph names and numbering.

- 1 Select Search ► Subset and press Enter.
- 2 Specify SECTION + PARAGRAPH in the Subset name field, as shown in [Figure 132](#), and press Enter.

Figure 132 • Search - COBOL Subset Name Pop-up



The results, shown in [Figure 133](#), indicate that a unique, sequential numbering scheme was used and that all of the section names are meaningful to this program. To view the additional statements located in this Search, press PF08 to scroll down.

Figure 133 • Source View - Program Section Names

```

File View Search Logic Task List Options Help
-----
Source View                               31 STATEMENTS FOUND
Command ==>                               Scroll ==> CSR

***** ***** TOP OF DATA *****
- - - - - 4 LINES NOT DISPLAYED
000005 CONFIGURATION SECTION. SECTION
- - - - - 2 LINES NOT DISPLAYED
000008 INPUT-OUTPUT SECTION. SECTION
- - - - - 4 LINES NOT DISPLAYED
000013 FILE SECTION. SECTION
- - - - - 38 LINES NOT DISPLAYED
000052 WORKING-STORAGE SECTION. SECTION
- - - - - 170 LINES NOT DISPLAYED
000223 LINKAGE SECTION. SECTION
- - - - - 29 LINES NOT DISPLAYED
000253 PROGRAM-INIT. PARAGRAPH
- - - - - 25 LINES NOT DISPLAYED
000279 P000-NEXT. PARAGRAPH
- - - - - 11 LINES NOT DISPLAYED
000291 P000-EXIT. PARAGRAPH
- - - - - 6 LINES NOT DISPLAYED
000298 P005-VAL-PARM. PARAGRAPH

```

Verifying PERFORM Statement Standards

Assume that a quality assurance standard requires all PERFORM statements using the THRU clause to go to a program EXIT. In addition, coding standards require that PERFORM statements contain only one paragraph.

To determine if this program meets these criteria

- 1 Select List ▶ Perform and press Enter. The List - Perform Range Names pop-up, shown in [Figure 134](#), displays. There are two PERFORM labels that do not use the THRU clause, PROGRAM-INIT and P999-ABEND-PROGRAM.

Figure 134 • List - Perform Range Names Pop-up

```
Command ==> _____ List - Perform Range Names      10 PERFORM RANGES
                                      Scroll ==> PAGE

Select the perform range to be viewed. Then press Enter.

  Perform range name
-----
- PROGRAM-INIT
- P000-NEXT THRU P000-EXIT
- P005-VAL-PARM THRU P005-EXIT
- P010-OPEN THRU P010-EXIT
- P100-PRINT THRU P119-EXIT
- P120-READ THRU P129-EXIT
- P150-SUBTOT THRU P169-EXIT
- P155-CL-SUBTOT THRU P159-EXIT
- P160-HDG THRU P169-EXIT
- P999-ABEND-PROGRAM
***** BOTTOM OF DATA *****
```

- 2 Press PF03/PF15 to exit.

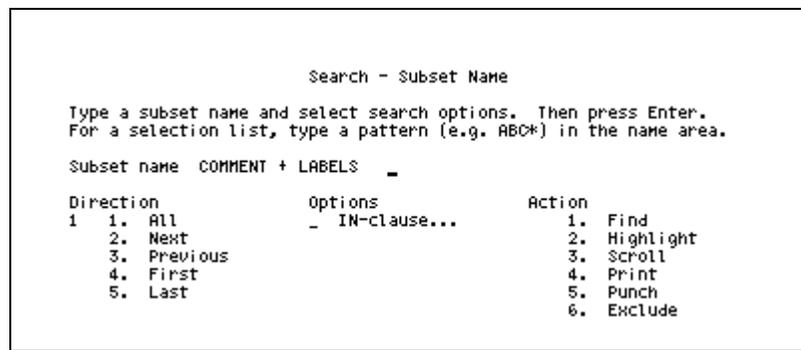
Useful Program Comments

Although the term useful is subjective, you may want to examine the program for comments that you believe are appropriate for long-term analysis and maintenance by those who may not be familiar with the source code.

To quickly isolate the comments in the program

- 1 Select Search ▶ Subset and press Enter.
- 2 Type COMMENTS + LABELS in the Subset name field, as shown in [Figure 135](#), and press Enter.

Figure 135 • Search - COBOL Subset Name Pop-up

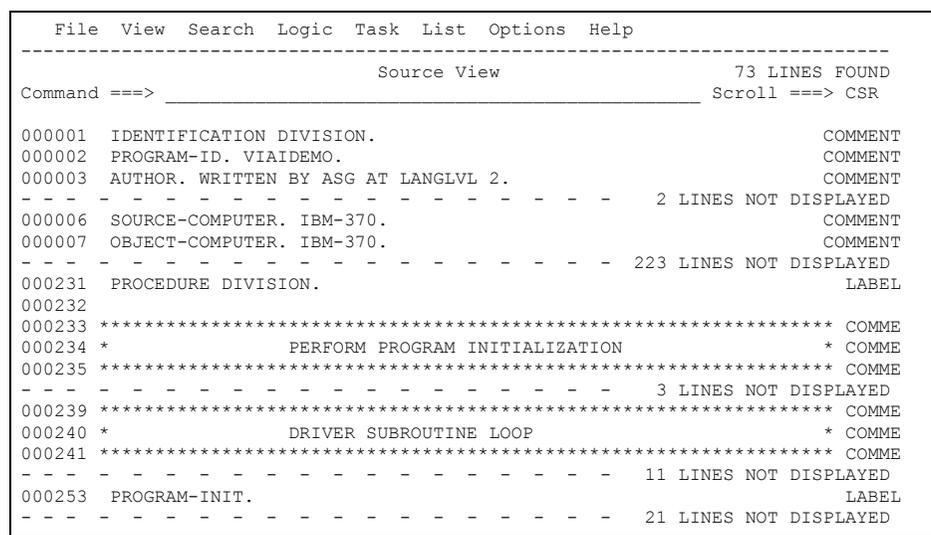


Note:

You may want to also select the Print Action on the pop-up to provide you with a printed copy of these results for later reference during your analysis. See online help or ["Specifying Log/List/Punch Processing Options" on page 26](#).

This program contains 73 lines of comments and labels. Scroll through the results shown in [Figure 136](#). You can add and remove any of these comments.

Figure 136 • Source View - Program Comments and Labels



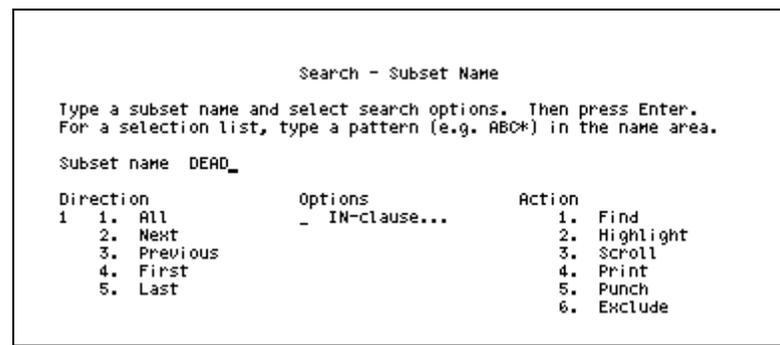
Finding Dead Code or Dead Data in a Program

Using Insight, you can quickly locate all of the statements in a program that cannot be executed under any conditions. You can also identify all of the DATA DIVISION statements containing datanames and their aliases that are not referenced in the PROCEDURE DIVISION.

To locate all occurrences of dead code and dead data in the program and determine why they are dead

- 1 Select Search ► Subset and press Enter.
- 2 Type DEAD as the subset name, as shown in [Figure 137](#), and press Enter.

Figure 137 • Search - Subset Pop-up



To isolate just the dead code in a program, Search the DEADCODE subset name. Use the DEADDATA subset name to identify just those DATA DIVISION statements containing datanames and their aliases that are not referenced in the PROCEDURE DIVISION.

The results, shown in [Figure 138 on page 125](#), reveal that 14 lines of dead code and dead data were located in the program. Before you make a recommendation to delete any of the statements that represent dead code or dead data, you should determine if they are used in any COPY statements. In general, use these simple guidelines to determine which dead code or data statements can safely be deleted:

- If the dead code or dead data are used in COPY statements, then do not delete it.
- If the dead code or dead data are not used in COPY statements, then the dead code and/or dead data can be safely deleted.

Upon examination of these results, you discover that there are no COPY statements that contain dead code or dead data.

Figure 138 • Source View - Dead Code and Dead Data in VIAIDEMO

```

File View Search Logic Task List Options Help
-----
Source View
14 LINES FOUND
Command ==> SCROLL ==> CSR
000060 77 CHECK-CODE PIC X VALUE '0'. DEADDATA
000061 01 AVG-AMT PIC 9(8)V99 COMP. DEADDATA
000062 01 NEW-DATA-FLAGS. DEADDATA
000063 05 NEW-CUSTOMER-DATA PIC 9(3). DEADDATA
000064 05 NEW-ACCOUNT-DATA PIC 9(3). DEADDATA
000065 05 NEW-ACCOUNT-TYPE PIC 9(3). DEADDATA
- - - - - 160 LINES NOT DISPLAYED
000226 05 PARM-LENGTH PIC 9(4) COMP. DEADDATA
- - - - - 244 LINES NOT DISPLAYED
000471 P200-EXIT. DEADCODE
000472 EXIT. DEADCODE
- - - - - 5 LINES NOT DISPLAYED
000478 P250-AVG-AMT. DEADCODE
000479 COMPUTE AVG-AMT = ( ZIP-LOAN-AMT / ZIP-LOAN-CNT ). DEADCODE
- - - - - 1 LINE NOT DISPLAYED
000481 P259-AVG-EXIT. DEADCODE
000482 EXIT. DEADCODE
- - - - - 13 LINES NOT DISPLAYED
000496 GOBACK. DEADCODE

```

MOVE CORRESPONDING Clauses

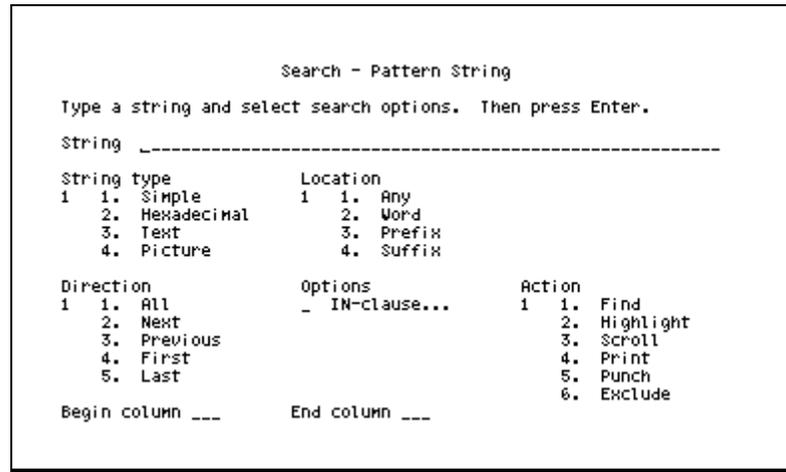
Assume that your quality assurance standards do not permit the use of the MOVE CORRESPONDING clause. The use of MOVE CORRESPONDING clauses can result in these actions:

- Generation of additional machine code that may require significant core memory.
- Masking of the identity of actual source and target fields being moved.
- Creation of duplicate datanames, since both the source and target names must be identical.

To identify MOVE CORRESPONDING clauses in this program

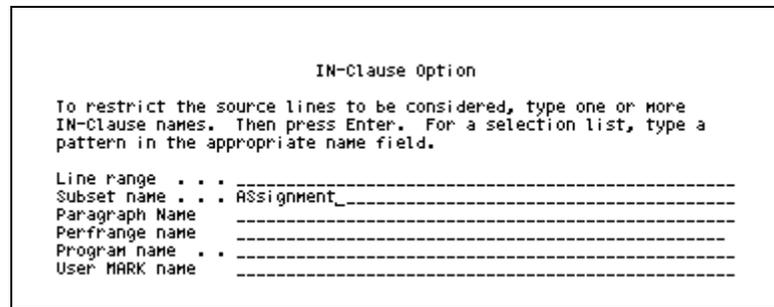
- 1 Select Search ▶ String and press Enter. The Search - Pattern String pop-up, shown in [Figure 139](#), displays.

Figure 139 • Search - Pattern String Pop-up



- 2 Type CORRESPONDING as the string name, choose the IN-clause Option on the pop-up, and press Enter.
- 3 Type ASSignment as the subset name on the IN-Clause Option pop-up, as shown in [Figure 140](#).

Figure 140 • IN-Clause Option Pop-up



There are no MOVE CORRESPONDING clauses in this program, as shown in the short message area.

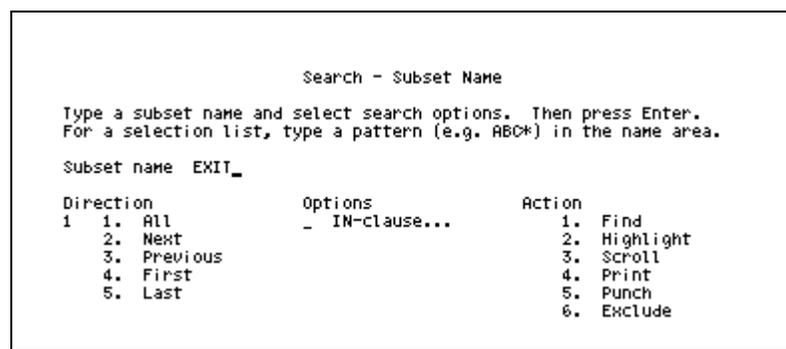
Determining the Number of EXITs Used in a Program

Assume that standards require that only one program exit is permitted.

To determine if this program meets your established standards criteria

- 1 Select Search ► Subset and press Enter. The Search - COBOL Subset Name pop-up, [Figure 141](#), displays.

Figure 141 • Search - COBOL Subset Name Pop-up



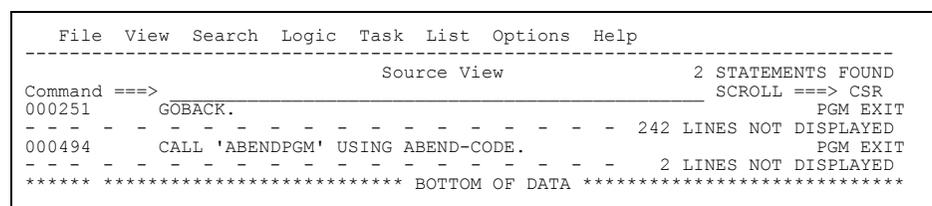
- 2 Type EXIT as the subset name and press Enter.

Note:

EXIT or PGMEXIT are COBOL subsets. They are statements containing a STOP, RUN, GOBACK, or EXIT PROGRAM verb as well as CALL statements that are indicated as NORET (non-returning).

The results, shown in [Figure 142](#) indicate two program EXITs, one to a GOBACK and the other to a CALLED program.

Figure 142 • Source View - Program EXITs



Note:

The CALLED program was designated during the Program Analyze process as a program that does not return.

You can use the Scratchpad feature demonstrated in the section, Saving the Results of Analysis, earlier in this section, to save the results of this Search for later reference.

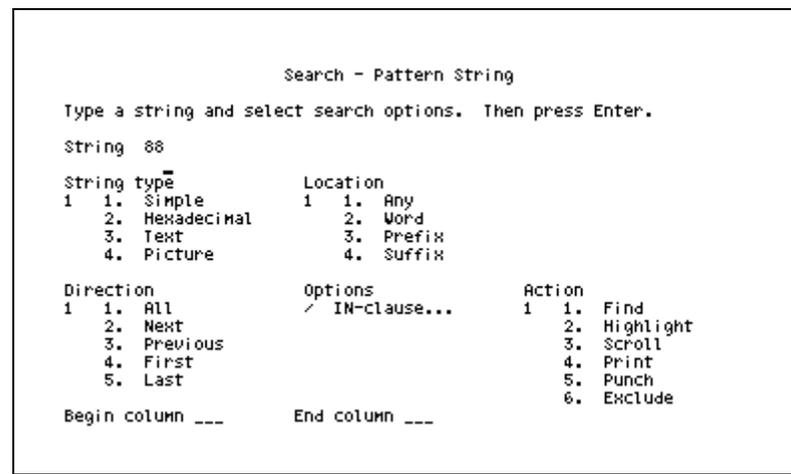
Identifying 88 Levels Used in a Program

Assume that you permit 88 levels in your programming standards to create a higher level value definition.

To determine how many and where the potential 88 levels exist

- 1 Select Search ► String and press Enter.
- 2 Type 88 in the String field, select the IN-clause Option, as shown in [Figure 143](#), and press Enter.

Figure 143 • Search - String Pop-up



- 3 Designate `DEFinitions` as the subset name and press Enter to see how and where the 88 levels are located in this program, as shown in [Figure 144](#).

Figure 144 • IN-Clause Option Pop-up

```

                                IN-Clause Option

To restrict the source lines to be considered, type one or more
IN-Clause names. Then press Enter. For a selection list, type a
pattern in the appropriate name field.

Line range . . . : _____
Subset name . . . : DEFinitions_
Paragraph Name   : _____
Perfrange name   : _____
Program name . . : _____
User MARK name   : _____
    
```

[Figure 145](#) reveals that only one 88 level is found. Using this 88 level permits you to define the value for END-INPUT for the entire program, VIAIDEMO.

Figure 145 • Source View - 88 Level Statement

```

File View Search Logic Task List Options Help
-----
                                Source View                                1 PATTERN FOUND
Command ==> _____ SCROLL ==> CSR
000055      88  END-INPUT _____ VALUE 'X'. _____ PATTERN
----- 441 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****
    
```

Identifying =, <, or > Expressions Used Instead of the Full Spelling in IF Statements

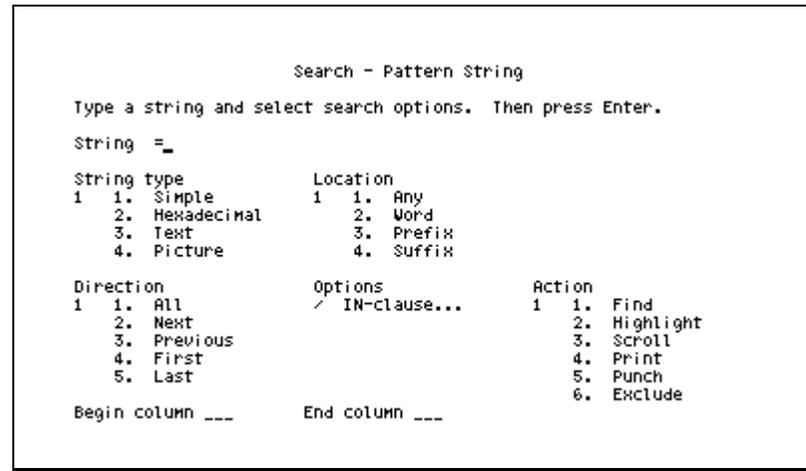
Your department prescribes that these arithmetic symbols should not be used in program code.

To determine if any arithmetic symbols exist

- 1 Select Search ▶ String and press Enter.
- 2 Specify the expression, =, <, or > as the String, select the IN-clause Option on the pop-up, and press Enter.

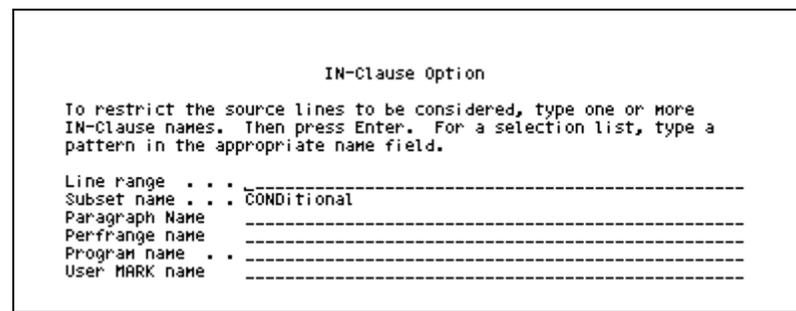
Repeat the same sequence shown in [Figure 146](#) for each expression, designated as the Target on the pop-up.

Figure 146 • Search - Pattern String Pop-up



- 3 Specify CONditionals as the subset name, as shown in [Figure 147](#), and press Enter. This provides you with a list of all occurrences of the designated expression(s) used in IF statements in VIAIDEMO.

Figure 147 • Search - IN-Clause Option Pop-up



Summary: Applying Quality Assurance Standards

- In this module, you were able to determine that all IF statements were indented and aligned with their respective ELSE statements and that ELSE statements were coded on lines by themselves.
- Negative expressions were not used in IF statements.
- Paragraph numbering, section numbering, and naming conventions were sequential and meaningful.
- Not all PERFORM statements used the THRU clause.
- The program may or may not need additional comments, based upon your own personal preference and standards.
- Dead data and dead code do not appear to be used in COPY statements in this program and may be deleted.
- 88 levels, the use of symbolic expressions (such as =, <, and >) and multiple WRITE statements to the same file are consistent with your new programming standards.

You can now execute this same analysis SCRIPT across other Loan Processing programs to validate your quality assurance standards.

A

abends, debugging data-related 51
action bar
 definition 1, 6
 example 7
AKR 12, 30
AKR Utilities pop-up 30
Alliance
 accessing from ESW screen xi
 description viii
 linking xi
allocating
 AKRs 30
 files 25
analysis session, starting 40
analyze job 12
AutoChange
 accessing from ESW screen xi
 description viii

B

Bridge
 accessing from ESW screen xi
 description viii

C

CALLED programs 61
Center, description viii
COBOL
 debugging facilities 69
 OF clause 18
 subset 13
 verbs 63
coding techniques 57
command input area 10
comments in a program, isolating 122
COMPUTED values, locating 93
conventions page xiv
CUA
 overview 1
 terms 6

D

data element 18
data item
 COBOL references 80
 impact of change to size 87
 impact of change to value 87
 locating references 90
 modification 83, 91
 name 80
dataname 17
 fully qualified 18
 group level 18
dataname types 17
display
 excluding unwanted program lines 55
 investigative request 55

E

elementary dataname 17
Encore
 accessing from ESW screen xi
 description ix
Estimate
 accessing from ESW screen xi
 description ix
ESW
 description vii
 invoking products x
 product integration xi
excluded subset 15
Existing Systems Workbench screen 23

F

features, overview 3
File - Analyze Submit pop-up 34
frequently asked questions 40

G

getting started 21
GOTO clause, locating 71
group item 18
group level dataname 18

H

HI subset 15
highlighted subset 15

I

initiating an Insight session 22
INPUT/OUTPUT processing, locating 63
Insight
 accessing from ESW screen xi
 customizing 23
 description ix
 facilities 2
 getting started 21
 initiating a session 22
 using analysis functions xi
introduction for new users 21

L

label, name set 17
lines (number, range, set) 17
List - COBOL Subset Names pop-up 12
List file allocation 25
Log file allocation 25
Log/List/Punch Definition pop-up 26
long messages 11

M

mark name 16
messages 11
modifying computation task 48

N

New user introduction 21
NHI subset 15
nonexcluded subset 15
nonhighlighted subset 15
NX subset 15

O

OPEN statements, locating 63
Options - Product Allocations pop-up 25
Options Menu 23

P

Parameter Definition screen 24
pattern string 18
PERFORM range, locating executed statements 75
perfrange name 17
PF Key Definition pop-up 29

pop-up

definition 2, 6
description 9
example 9
List - COBOL Subset Names 12

product integration xi

program

CALLED list 61
different structural levels 59
structure 57
program abend, debugging 51
program analysis
 88 levels 128
 analyzing program logic 39
 applying quality assurance standards 39
 COBOL debugging facilities 69
 comments 122
 COMPUTED values 93
 examining known problems 106
 EXIT 127
 field settings 102, 105
 GOTO clauses 71
 identifying data item usage 39
 IF statements 119, 129
 internal sorts 68
 logical execution paths 99
 methodology 38
 MOVE CORRESPONDING clause 125
 negative expressions 119
 overview 38
 paragraph control transfer 77
 paragraph names 120
 paragraph overview 39
 PERFORM statements 121
 program overview 38
 recursion 75
 results 116
 saving analysis steps 116
 saving results 64
 section names 120
 THRU clause 121

program analyzer 39

program output 41

pull-down 8

Punch file allocation 25

R

Recap

accessing from ESW screen xi
description ix

recursion 75

S

Sample SmartDoc session 21
screen
 AKR Utilities 30
 characteristics 10
 description 10
 Existing Systems Workbench 23
 format 10
 Log/List/Punch Definition 26
 messages 11
 Options - Product Allocations 25
 Options Menu 23
 Parameter Definition 24
 PF Key Definition 29
 subsets 15
session, initiating 22
set
 concatenate 12
 description 12
 label name 17
 multiple 12
setup, customizing Insight 23
short messages 11
SmartDoc
 accessing from ESW screen xi
 description ix
SmartEdit
 accessing from ESW screen xi
 description x
SmartTest
 accessing from ESW screen xi
 description x
SORT, locating 68
Source View facility
 demonstration programs 38
 introduction 38
subset name 17
subsets 12

T

table entry 18
tag subsets 15
targets 16

U

user options 23

V

variable, redefined 18

W

Work file allocation 25

X

X subset 15

ASG Worldwide Headquarters Naples Florida USA | asg.com