

ASG-SmartTest™ for COBOL and Assembler User's Guide

Version: 6.0

Publication Number: STA0200-60

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.

© 1989-2002 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 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	ix
About this Publication	x
Related Publications	x
ASG-Existing Systems Workbench (ASG-ESW)	xii
Invoking ESW Products	xv
ESW Product Integration	xvi
Examples	xvii
Publication Conventions	xix
1 Introduction	1
SmartTest Overview	2
Testing and Debugging	3
Testing Support	4
Supported Execution Environments	4
Supported Databases.....	5
Screen Format	5
The Common User Access Interface	7
SmartTest CUA Screens	7
CUA Action Bar	7
CUA Pull-downs	8
CUA Pop-ups	10
Insight Analysis	11
Reviewing Program Structure	12
Reviewing Data Usage	13
Paragraph Cross-reference	16
Reusing Command Results	16
Optimization Considerations	18

Command Targets	19
Mark Name Target	19
Perfrange Name Target.....	20
Subset Name Target	20
Line Range Target.....	20
Label Name Target	20
Program Name Target.....	20
Dataname Target.....	21
Pattern String Target.....	23
Paragraph Name Target.....	23
2 Test Session	25
Test Session Overview	26
Beginning a Test Session	26
User Options	28
Online Operation Parameters	29
Log/List/Punch/Work File Allocations.....	29
Log/List/Punch Processing Options	30
Specifying Script File Allocations	32
Setting PF Key Values	33
Setting Mode Options	34
Application Knowledge Repository (AKR)	35
Verifying AKR Allocation Results.....	37
Analyze Facility	38
Program Analyze Requirements	38
Program Analyze Input.....	38
Analyzing a Program.....	39
Method 1 - Analyzing a Program Using SmartTest	39
Method 2 - Analyzing a Program from an Edit Session	41
Method 3 - Analyzing a Program from a User Screen	43
Verifying Analyze Results	44
Steps in Setting up the Test Session	44
File Pull-down.....	45
Selecting the Testing Environment.....	47
Language Environment Testing	52
MVS Programs in TSO Foreground	53
Specifying TSO Setup Information.....	53
Converting Batch Execution JCL to a TSO CLIST	55
Initiating a Test Session	61
Setup Considerations.....	61

Saving the SmartTest Testing Setup	62
Sharing Test Setups	63
Sharing an Alternate Profile Dataset	63
Restoring the Testing Environment	64
Terminating a Test Session	65
3 Test Session - Additional Environments	67
ISPF Dialog Manager	68
Specifying ISPF Dialog Manager Information	68
Specifying Programs to be Tested	70
Specifying ISPF File Allocation Information	70
Initiating an ISPF Test Session	78
IMS/DB Programs in TSO Foreground	79
Specifying IMS/DB Setup Information	79
Specifying IMS File Allocation Information	82
BTS in TSO Foreground	98
Specifying BTS Setup Information	98
Selecting BTS Transactions to Monitor	100
Specifying BTS File Allocation Information	102
Specifying IMS File Allocation Information	120
DB2 Programs in TSO Foreground	121
Specifying DB2 Setup Information	121
DB2 Stored Procedure Testing Option	123
Requirements	123
Setting Up the DB2 Stored Procedure Test	124
Reviewing DB2 Stored Procedure Parameters	126
Initiating the DB2 Stored Procedures Test	129
Testing Programs in a Batch Region	130
Batch Connect Facility	130
Specifying Batch Connect Setup Information	130
Submitting and Connecting to a Batch Job	134
Batch Test Initiation	136
Testing DL/I in the Batch Environment	138
Testing BTS in the Batch Environment	139
Testing DB2 in the Batch Environment	140
Testing DFHDRP in the Batch Environment	141

4 Testing Techniques	143
Learning the SmartTest Commands	144
Controlling Program Execution Using SmartTest	144
Testing with SmartTest	145
Testing Using the MONITOR Method	145
Testing Using the NOMONITOR Method	146
Guidelines for Using the MONITOR/NOMONITOR Methods	146
Testing LINKed, ATTACHed, and CALLED Load Modules	147
Controlling Program Execution	148
Selecting Test Environment Using the Setup Wizards	148
Executing the Program Continuously Using RUN	148
Executing a Specified Number of Statements Using STEP	149
Changing Program Execution Sequence Using GO	149
Interrupting Test Execution Using Keystrokes	150
Intercepting Program Abends	151
Setting Program Address Stops	152
Inserting Breakpoints to Interrupt Execution	153
Automatically Inserting Breakpoints in the Program	153
Inserting Breakpoints with Impact Datasets	155
Displaying Breakpoints	160
Generating a Dump	161
Canceling a Test Session	161
Exiting a SmartTest Test Session	162
Viewing and Changing Test Session Data	162
Viewing Test Session Data Values	162
Viewing Test Session Data Items Inline	162
Removing Zoom Data Windows	165
Viewing Data Values at the Top of the Screen	165
Changing Test Session Data Values	167
Execution History (Backtrack)	167
Recording Program Execution History	167
Reviewing Backtrack History	168
Backtrack Recording/Review Session	168
Using the BackTrack Variable History Function	172
Using Pseudo Code	173
Pseudo Code Concepts	173
Pseudo Code Statements Available	174
Entering and Editing Pseudo Code	176
Executing Pseudo Code in a Test Session	177
Viewing Pseudo Code in a Test Session	177
Removing Pseudo Code from a Program	178
Using Multiple Programs	178
Tailoring a Test Session by Program	179

Setting Test Session Options	180
Displaying Program and Test Information	181
Printing Displayed Information (LPRINT Command)	182
Linking to Alliance	183
5 Program Analysis Features	187
COBOL Intelligent Search Function	187
Search for COBOL Subsets	188
Search for Dataname References	191
Search for Indirect Dataname References	191
Limit the Search Scope	192
Excluding Lines from the Display	192
Redisplaying Excluded Lines	194
Displaying All Excluded Lines	194
Finding Program Information Using the Search Function	195
Finding All Input and Output Statements	195
Determining References to a Data Field	198
Determining Where a Data Field is Modified	201
Determining if a Data Field is Used in Conditional Logic	203
Determining the Impact of a Data Field Size Change	205
Determining the Impact of a Data Field Value Change	207
Highlighting Search Results	209
Printing Program Information	209
Repositioning the Display	210
Following Branching Logic	211
Using the Branch Function	212
Using the BRANCH Command	213
Searching the Program in Execution Sequence	215
6 Additional Testing Features	217
Capturing and Replaying Command Sequences	217
Replaying a Script File	218
Locating the Next Executable Statement (LOCATE * Command)	220
Simplifying Commands	221
Using the Cursor Substitution Character	221
Creating Short Names for Character Strings (EQUATE Command)	221
Keep Commands in the Command Input Area (& Command)	222
Recall Primary Commands and Messages (RECALL Command)	222
Displaying Product Release and Level Numbers (PRODLVL Command)	223

Other Primary Commands	223
Other Line Commands	224
Commands Available Only with Insight	224
7 Analyze	225
Analyzing a COBOL Program	225
Analyze Input Descriptions	226
The Analyze Process	227
Using the File - Analyze Submit Pop-up	228
Options	229
Fields	230
Using ISPF	231
Using ISPF/PDF Edit	232
Analyze Submit Parameters Screen	234
Options	235
Fields	235
Automatic JCL Modifications	237
Analyze Summary Report	244
Adding Analyze Facilities to a Standard Compile Mechanism	246
CLIST Compile Mechanism	246
ISPF Compile Mechanism	247
Assembler Analyzer	248
Assembler Analyzer Input	248
Automatic JCL Modifications	249
Assembler Analyze JCL	250
Analyze Options	252
Buffers	252
COBOL Level	253
DB2 Load Library	253
DB2 Application Plan	253
Dynamic CALLS	254
Flag Messages	255
Input	255
IO	256
Language Level	256
Line Count	257
Main	257
Maximum Number of Errors	257
Output	257
Program	258

Recursion.....	258
Return.....	258
Sequence.....	259
Source.....	259
Spacing.....	259
SQL Authorization ID.....	260
DB2 Subsystem.....	260
Live Exit.....	260
Memory.....	261
8 Additional Language Support	263
SmartTest and Assembler Language	263
Analyzing an Assembler Program.....	264
Starting a Test Session.....	264
Assembler Source Testing and Debugging.....	264
Display Expanded Assembler Macros.....	265
The Assembler Specific Commands.....	267
Commands with Limited Use in Assembler.....	267
Commands Not Available with Assembler.....	268
SmartTest and INTERSOLV APS	269
Analyzing an INTERSOLV APS Program.....	269
Starting an INTERSOLV APS Test Session.....	269
APS Testing and Debugging.....	270
APS Program.....	270
Considerations with APS Program Painter Code.....	273
Considerations with APS Generated COBOL Code.....	273
SmartTest and PL/I.....	273
9 Help Facility.....	275
Introduction.....	275
Help Navigational Commands.....	277
Screen Help.....	278
Command Help.....	279
General Information.....	281
Specific Information.....	282
Help Abends.....	283
Help Messages.....	284
Severity Levels.....	284
Printing Messages.....	285

10 COBOL Compiler Options	289
Introduction	289
Compiler Limitations	290
COPYLIBs With Debug Limitations	290
TEST Option Limitations	290
Compiler Optimization Limitations	291
Glossary	293
Index	303

Preface

This *ASG-SmartTest for COBOL and Assembler User's Guide* provides user information about ASG-SmartTest (herein called SmartTest). SmartTest brings a new approach to the testing and debugging application programs. Alone or as part of the ASG-Existing Systems Workbench (ASG-ESW), SmartTest has language-related features that support COBOL and Assembler programmers in the development and testing phases of software systems. The uniqueness of the SmartTest automated solution is due to the ability of the product to gather and store knowledge, about the application being tested, in the Application Knowledge Repository (AKR). This knowledge base provides SmartTest with a foundation of information in these areas:

- Program relationships, logic and data, execution paths, etc.
- COBOL intelligence
- Testing/debugging session experience

This stored knowledge enhances the productivity and the quality of testing because it provides programmers with an online programming expert to assist with testing. SmartTest runs under MVS and uses ISPF as its standard user interface. Full-screen source support is provided for COBOL and Assembler, as well as disassembled object support for other languages.

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 introduction to SmartTest.
- [Chapter 2, "Test Session."](#) describes how to set up a test session for the TSO execution environment.
- [Chapter 3, "Test Session - Additional Environments."](#) describes setup information for other test environments.
- [Chapter 4, "Testing Techniques."](#) presents common SmartTest testing and debugging functions available with SmartTest.
- [Chapter 5, "Program Analysis Features."](#) presents the basic problem investigation features available with SmartTest.
- [Chapter 6, "Additional Testing Features."](#) describes additional testing features that are available with SmartTest.
- [Chapter 7, "Analyze."](#) describes the analyze process that is required before a program can be tested using SmartTest.
- [Chapter 8, "Additional Language Support."](#) provides information on Assembler, INTERSOLV APS, and PL/I programming languages support in SmartTest.
- [Chapter 9, "Help Facility."](#) describes how to use the comprehensive and context sensitive Help facility, including an online Help Tutorial.
- [Chapter 10, "COBOL Compiler Options."](#) list the COBOL compiler options used by SmartTest.

Related Publications

The documentation library for ASG-SmartTest 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-ESW suite of products.
- *ASG-SmartTest CICS User's Guide* (STC0200-*nn*) contains specific commands and test session setup information for the CICS environments.
- *ASG-SmartTest for COBOL and Assembler User's Guide* (STA0200-*nn*) contains introductory and usage information for COBOL and Assembler. It also contains test session setup information for the TSO, ISPF, IMS/DB, DB/2, BTS, and Batch environments.

- *ASG-SmartTest IMS User's Guide (STM0200-nn)* contains specific commands and test session setup information for the IMS/DC environments.
- *ASG-SmartTest Installation Guide (STX0300-nn)* contains information for installing and maintaining ASG-SmartTest.
- *ASG-SmartTest PLI User's Guide (STL0200-nn)* contains introductory and usage information about how to use ASG-SmartTest with the PL/I language. It also contains test session setup information for the TSO, ISPF, IMS/DB, DB/2, BTS, and Batch environments.
- *ASG-SmartTest Quick Start for COBOL/ASM (STA0900-nn)* summarizes how to use ASG-SmartTest with the COBOL or Assembler language.
- *ASG-SmartTest Quick Start for PL/I (STL0900-nn)* summarizes how to use ASG-SmartTest with the PL/I language.
- *ASG-SmartTest Reference Guide (STX0400-nn)* contains detailed reference information about CUA pull-downs and pop-ups, ASG-SmartTest command syntax, and pseudo code.
- *ASG-SmartTest Reference Summary (STX0600-nn)* summarizes the syntax and usage of ASG-SmartTest commands.
- *ASG-SmartTest TCA User's Guide (STT0200-nn)* contains procedures for using the ASG-ASG-SmartTest-TCA (Test Coverage Analysis) option.

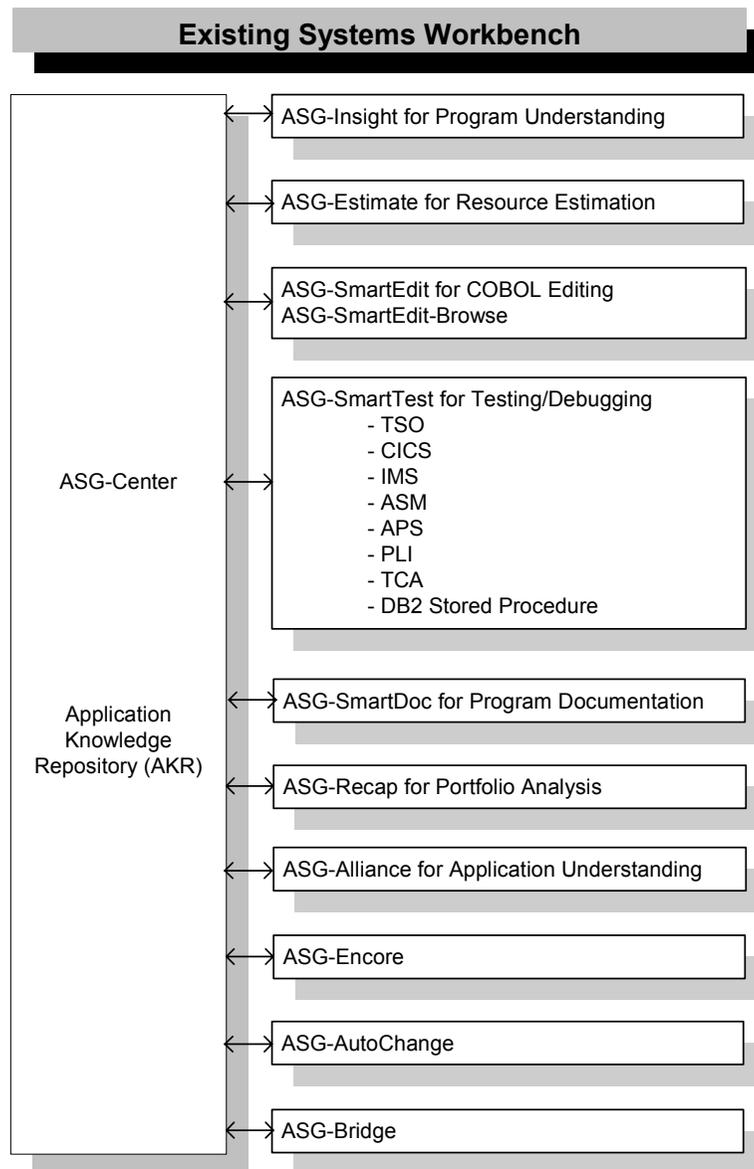
Note: _____

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

ASG-Existing Systems Workbench (ASG-ESW)

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

Figure 1 • ASG Existing Systems Workbench



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

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

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

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

Invoking ESW Products

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

The product names can also vary depending on whether you access a product directly or through ESW. See ["ESW Product Integration" on page xvi](#) 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 SmartTest and contains these sections:

Topic	Page
SmartTest Overview	2
Testing and Debugging	3
Screen Format	5
The Common User Access Interface	7
Insight Analysis	11
Reviewing Program Structure	12
Reviewing Data Usage	13
Reusing Command Results	16
Optimization Considerations	18
Command Targets	19

SmartTest is the Testing/Debugging component of ESW 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.

SmartTest Overview

Testing and debugging application programs can be a difficult and time-consuming process. SmartTest brings a new approach to the testing and debugging process. Alone or as part of ESW, SmartTest has language-related features that support COBOL and Assembler programmers in the development and testing phases of software systems.

SmartTest provides an electronic window through which COBOL, PL/I, and Assembler programs are viewed, analyzed, tested, and debugged. SmartTest runs under MVS and uses ISPF as its standard user interface. Full screen source support is provided for COBOL and Assembler, as well as disassembled object support for other languages. It executes in the TSO/ISPF environment and provides full ISPF compatibility. This compatibility facilitates learning and usage through standard IBM interfaces.

SmartTest features optional Common User Access (CUA) screens, pull-downs, and pop-ups designed to provide easy access to all of the product features. Powerful SmartTest commands can be used as an alternative to the selections offered on the CUA action bar.

The uniqueness of the SmartTest automated solution is based on its ability to gather and store knowledge, about the application being tested, in the Application Knowledge Repository (AKR). This knowledge base provides SmartTest with a foundation of information in these areas:

- Program relationships, logic and data, execution paths, etc.
- COBOL intelligence
- Testing and debugging session experience

This stored knowledge enhances the productivity and quality of testing because it provides programmers with an online programming expert to assist with testing.

Testing and Debugging

SmartTest provides comprehensive testing and debugging capabilities, which includes these features and functions:

Feature	Description
Comprehensive Program Knowledge	The testing environment contains complete knowledge about the program including its syntax, logic relationships, and execution flow. This knowledge is stored in the AKR.
COBOL Intelligence	COBOL-compatible pseudo code can be temporarily added to a program to test proposed changes, saved permanently in the AKR, or automatically added to the existing program. Dynamic breakpoints can be set based on COBOL syntax statements or conditions.
Script Automation	A predefined command sequence can be included in any test session. These script files can be created automatically during a test session, then used for regression testing of a program.
Scrollable Window Displays	In-context windows for data display and logic provide easy access to related information.
Hot Key Branching	Hot key branching between paragraphs, sections, and called routines provides easy access to related logic.
Assembler Integration	Full screen Assembler integration of programs, called modules, or memory display of data elements is provided.
ESW Integration	Integrates the ESW family of products that support the automation of the software maintenance cycle. If Insight is installed and specified during the analyze, the Insight program analysis functions are available for viewing logic/data relationships and execution paths. If Alliance is installed, an interface to Alliance is provided. SmartTest provides canned queries you can customize and run in Alliance to extract information about a program.
Execution Review (Backtrack)	Execution history can be viewed using the Backtrack Facility. This feature allows you to step backward or forward through the executed code and view data values as they existed when specific statements were executed.
Automatic Setting of Breakpoints	Breakpoints can be set with a dataname list produced in Alliance, AutoChange, or Estimate; or produced by users.

The dramatic advantage SmartTest has over conventional program testing and debugging tools is that it is COBOL intelligent. It provides comprehensive analysis information about a program.

SmartTest effectively presents information in an interactive environment for online querying and/or testing on a display screen. The results of new code or enhancements can be quickly and accurately tested to determine their impact on the rest of the program logic. Program errors can be easily recognized and quickly corrected.

Testing Support

SmartTest runs under MVS ESA, TSO/ISPF (Release 3.5 through 4.8) and supports these languages, execution environments, and databases:

- COBOL II
- COBOL/370
- COBOL for MVS and VM
- COBOL for OS/390
- CASE-generated COBOL (APS)
- High Level Assembler
- CA-Optimizer II (COBOL II)
- INTERSOLV APS
- OS PL/I Versions 2.3
- PL/I MVS & VM

Supported Execution Environments

- TSO
- BTS
- CICS Version 4.1
- CICS/TS 1.1, 1.2, 1.3, and 2.1
- CICS command level programs
- IMS/DB
- IMS/DC Version 3.1 through 7.1
- ISPF Dialog Manager

- HOGAN
- LANGUAGE ENVIRONMENT Version 1.3 through 2.9

Supported Databases

- VSAM
- IMS Version 3.1 through 7.1
- DB2 Version 2.2 through 6.1
- CA-IDMS/DB Release 14.0
- SYSTEM 2000
- DATACOM/DB
- TOTAL/TIS

Screen Format

SmartTest screens are modeled after the TSO/ISPF edit screen. [Figure 6](#) is an example of the SmartTest Program View screen.

Figure 6 • SmartTest Screen Example

```

File View Test Search Logic List Options Help
-----
                                (A)                                (B)
                                Program View                       VIAPCOB.VIAPCOB -A
Command ==> __ (C) _____ (D) __ Scroll ==> CSR

000360      MOVE ALL '/?X!' TO SOC7-DATA.
>>>>>>      ADD +1 TO DATA-PACKED-DEC.                                FALLTHRU
|          +-----+-----+-----+-----+
|          | 10 DATA-PACKED-DEC          PIC S9(5)V99 C3      ADDR 000D06A8 |
|          |          VALUE > /?X!          < * INVALID NUMERIC * |
|          +-----+-----+-----+-----+
0 (F) 62 *-----*
000363 *
000364 * THIS IS AN EXAMPLE OF A SOC7 ABEND (DATA EXCEPTION). THE *
000365 * VALUES INVOLVED IN DATA RELATED ABENDS ARE AUTOMATICALLY *
000366 * DISPLAYED BY SMARTTEST (BY SIMULATING A 'ZD' LINE COMMAND). *
000367 *
000368 * THERE ARE SEVERAL WAYS TO FIX THIS SOC7: *
000369 *      1) OVERTYPE THE VALUE OF 'DATA-PACKED-DEC' IN THE ZOOMDATA*
(E)
(G)
+-----+-----+-----+-----+
|STATUS: DATA EXCEPTION (OC7)          PROGRAM: VIAPCOB  DATE: DDMMYYYY |
| STMT: 000361  OFF: 000D64  AMODE: 24  MODULE: VIAPCOB  TIME: 10:33:00 |
|SOURCE: ADD +1 TO DATA-PACKED-DEC. |
+-----+-----+-----+-----+

```

This table describes the sections highlighted on the Program View screen:

Screen Section	Description
(A) Program View	Specifies the name of the screen.
(B) VIAPCOB.VIAPCOB	Identifies the <i>module.program</i> being viewed. This area of the screen is called the short message area and is also used to temporarily display short informational or error messages. A value of -A (Active program) following the <i>module.program</i> name indicates that this program is currently being tested. A value of -Q (Qualified) indicates this program has been displayed for viewing on the Program View screen while another program is being tested.
(C) Command ==>	Specifies the is the primary field for entering SmartTest commands.
(D) Scroll ==>	<p>Specifies the number of lines or columns to scroll the display screen. The Scroll ==> field is omitted from screens that cannot be scrolled. These are the valid values:</p> <p>1 through 9999. Specifies the number of lines or columns to scroll.</p> <p>CSR. Specifies the screen is to be scrolled until the cursor is at the top, bottom, left side, or right side of the screen.</p> <p>MAX. Specifies that the top, bottom, right, or left margin is the scroll value.</p> <p>HALF. Specifies a scroll value of a half screen.</p> <p>PAGE. Specifies a scroll value of one screen.</p> <p>DATA. Specifies a scroll value of one line less than a screen. This field is omitted from screens that cannot be scrolled.</p>
(E) Long message area	Displays descriptive, informational, or error messages in this area. Long messages are displayed when there are no corresponding short messages, or when you type <code>HELP</code> in the command input area while a short message displays.
(F) Line command input area	Indicates the line command area, in which you can enter over the displayed line numbers in columns 1 through 6 as with other TSO/ISPF editor screens.
(G) Status box at the bottom of the SmartTest screen	Provides current status information about the program being tested including the reason for an interruption in execution, the current statement number and offset in the program, and the text of the next source statement to be executed.

The Common User Access Interface

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

SmartTest CUA Screens

For CUA purposes, screen denotes the whole screen, as opposed to a pop-up. The action bar is usually displayed at the top of SmartTest screens.

CUA Action Bar

The Program View screen is shown in [Figure 7](#). Notice the action bar, which designates the primary functional organization of SmartTest. Each action is briefly described in these sections.

You can remove the CUA action bar from the screen by typing `SET CUA OFF`. SmartTest functions will be available using the command interface. To reinstate the CUA action bar, type `SET CUA ON`.

Figure 7 • SmartTest Screen Action Bar Example

```

File View Test Search Logic List Options Help
-----
                                Program View                                VIAPCOB.VIAPCOB -A
Command ==> _____ Scroll ==> CSR

001120 * VALIDATE EXECUTION PARAMETER.
001121   PARM-EDIT.
>>>>>>   IF CURRENT-PARM-NUMBER = 0 THEN
001123     PERFORM BUILD-PADDED-PARM
001124     IF PADDED-PARM = 'ALL' THEN
001125       MOVE PADDED-PARM TO FIRST-PARM-TEXT
001126     ELSE
001127       PERFORM FIND-FIRST-PARAMETER
001128         VARYING SUB1 FROM 1 BY 1
001129         UNTIL SUB1 > VALID-PARMS-COUNT.
001130
001131     IF FIRST-PARM-TEXT = 'ALL' THEN                                RETURN
001132       MOVE CURRENT-PARM-NUMBER TO SUB1
001133       MOVE +0 TO CURRENT-PARM-NUMBER
+-----+
|STATUS: STOPPED BY STEP REQUEST      PROGRAM: VIAPCOB   DATE: DDMMYYYY |
| STMT: 001122  OFF: 001D0C  AMODE: 24  MODULE: VIAPCOB  TIME: 10:46:05 |
|SOURCE: IF CURRENT-PARM-NUMBER = 0 THEN                                     |
+-----+

```

File Pull-down

Use the File pull-down to set up the test environment, select Test Coverage Analysis (if the option is installed at your site), manage the AKR, open, view, and/or test a program, update a source file, and to exit SmartTest.

View Pull-down

Use the View pull-down to access the various methods used to view a program and related testing history in SmartTest.

Test Pull-down

Use the Test pull-down to access the test setup wizards and the various methods used to run or step through a program, manipulate data values in SmartTest, initiate a link to Alliance, generate dumps, or begin the process of automatically setting breakpoints with an impact dataset.

Search Pull-down

Use the Search pull-down to find, highlight, scroll, print, punch, or exclude a specified target, which may be a dataname, label name, paragraph name, pattern, COBOL subset, perform range, program, user mark, or line number.

Logic Pull-down

Use the Logic pull-down to follow the logical execution paths of a program.

Note: _____

This action displays on the action bar only if Insight is installed. See the *ASG-Insight User's Guide* for more information on the Logic pull-down.

List Pull-down

Use the List pull-down to access pop-ups that list information about the program and the testing environment.

Options Pull-down

Use the Options pull-down to customize your SmartTest environment by setting certain parameters and options.

Help Pull-down

Use the Help pull-down to access the online Help features.

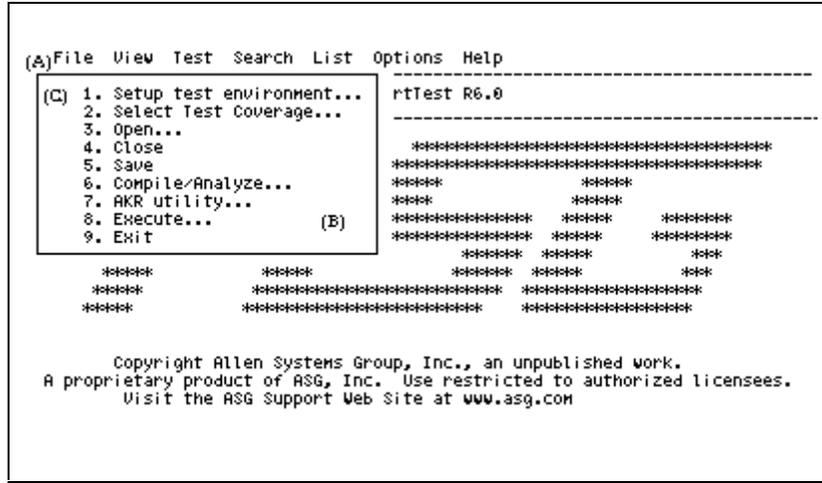
CUA Pull-downs

Pull-downs are displayed by selecting items from the action bar. Action bar choices are selected by positioning the cursor on the item and pressing Enter to display the corresponding pull-down.

Actions on pull-downs are selected by typing the number of the action in the selection field and pressing Enter, or by positioning the cursor on the line with the desired action and pressing Enter.

Figure 8 shows the SmartTest File pull-down.

Figure 8 • CUA Screen Features



Screen Section	Description
(A) Action Bar	Presents selectable items that represent functions in SmartTest.
(B) Pull-down	Displays the selectable actions available for each action bar choice.
(C) Action	Describes the SmartTest feature. When selected, an action followed by an ellipsis (...) displays additional information; actions without the ellipsis immediately execute the action.

Note:

Use the END PF key (usually PF3 or PF15) to exit a pull-down screen.

CUA Pop-ups

Pop-ups contain entry fields and selection fields. In [Figure 9](#), Data name is an entry field, while References and Indirect impact are selection fields.

Figure 9 • Pop-up Example

```

File View Test Search Logic List Options Help
-
      Search - Data Name
C
Type a data name and select search options. Then press Enter. For
0 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. Defs            2. Of size change
0 3. Uses            3. Of value change
0 4. Mods
0
0 Direction          Options          Action
0 1 1. All            - No data aliasing    1 1. Find
0 2. Next            - IN-clause...        2. Highlight
0 3. Previous
0 4. First
0 5. Last
0
000019          BLOCK CONTAINS 0 RECORDS.
  
```

Screen Section	Description
(A) Entry Fields	These are fields in which you type textual information, such as a dataname, line range, or number of levels. Underscores indicate the maximum length of the field.
(B) Selection Fields	<p>These are fields in which you may select one or multiple choices.</p> <p>If the selections are numbered, one choice is allowed. To make a selection, type the number of the choice in the selection field.</p> <p>If the selections are not numbered, more than one choice is allowed. To make a selection, type a slash (/), or a non-blank character, in the field you wish to select.</p> <p>If the choices are contained on a scrollable list, use the S line command to make a selection.</p>

Note: _____
 Use the END PF key (usually PF3 or PF15) to exit a pop-up.

Insight Analysis

If Insight is installed with SmartTest, Insight analysis extends SmartTest to include comprehensive analysis functions. To take advantage of the Insight functionality, select Extended Analysis on the File - Analyze Submit screen.

Note: _____

If you do not plan on using this advanced analysis feature, ASG recommends that you specify N in the Extended Analysis field because the advanced analysis requires additional AKR space and analysis time.

Technology that captures and stores a complete summary of how a COBOL program works (the underlying logic and data relationships, control flow, organization and structure, etc.) and an interface of analysis functions are integrated directly into the test session. This technology is the Analytical Engine, which automates the important and otherwise time-consuming task of capturing and storing complete information about a program.

Insight Analysis Benefits

Integrated analysis and testing provides these productivity and quality benefits:

- All logic paths are analyzed and tested, with or without data.
- Abends can be traced backward through the logic to their source.
- Data modification paths (including modifications, uses, and references in called modules) are displayed.
- Program and COBOL sensitivity are included in each testing command.

SmartTest provides many valuable facilities for a test session, significantly reducing the time required to perform testing and debugging tasks. SmartTest with Insight ensures the full impact of new code or a change to existing code is analyzed and available to the programmer when testing. It also provides a means of determining underlying causes for problems while debugging a program.

SmartTest is an integrated analysis and testing system that brings intelligence to software testing. Testing and debugging are based on a foundation of program knowledge. Automation of the program understanding process improves the quality of the testing/debugging session, resulting in additional productivity gains.

Reviewing Program Structure

SmartTest provides commands to reveal the structure of a COBOL program: FINDXTND, ZOOMIN, ZOOMOUT, and BRANCH.

The highest level of structure can be displayed by typing `FX` referring to the subset `STRUCTURE`, for example:

```
X;FINDXTND STRUCTURE
```

The Zoom functions show the subprograms' (COBOL II and above) divisions, sections, paragraphs, and paragraph code exactly as they occur in the program. `ZOOMIN` and `ZOOMOUT` functions can be used to view major structure statements one level at a time, or to focus on specific areas of code within a paragraph.

If you begin with all source lines excluded from the screen and enter successive `ZOOMIN` commands, source lines are displayed in hierarchical order. The `PROCEDURE DIVISION` lines would be displayed in this order:

- Subprograms
- Division headings
- Sections
- Paragraphs
- Paragraph code

This feature provides a means of viewing various structure levels with or without the corresponding code. Type `ZOOMIN` to see detailed information from one of these levels, then type `ZOOMOUT` to return to the previous level.

Use the COBOL Intelligent Search function or the `FINDXTND` command to locate specific subsets within the structure levels. For example, this command reveals all IO and assignment statements:

```
FX IO + AS
```

Use the `BRANCH` function to scroll from any `GO TO` or `PERFORM` statement to the label to which it transfers control. It can also be used to return to the location from where the `BRANCH` primary command was issued. For example, this command positions the display to the paragraph labeled `P120-READ`:

```
BRANCH P120-READ
```

This BRANCH command causes the cursor to be repositioned to the P120-READ paragraph. You can view the code within the paragraph, then type BRANCH BACKUP to return. BRANCH and BRANCH BACKUP have also been assigned to PF keys. Instead of entering the command in the command input area, you can place the cursor on a GO TO P120-READ statement and press the PF key for BRANCH (default PF10/PF22). Then press the BRANCH BACKUP key (default PF11/PF23) to return.

Reviewing Data Usage

The FINDXTND command searches the program in sequential order from the current location for a target. A target is the object of a search and can be a set of lines, a path, a dataname, or a pattern. Several target types are available.

The entire program can be searched for a target, or the scope of the search may be limited by using the IN clause. The results of a FINDXTND primary command are all lines containing the specified target. These lines are highlighted and the cursor is positioned below the first target. Targets that were highlighted as a result of a previous command are reset so only the results of the current command are highlighted.

Complex items can be located that would take many separate searches if done manually.

Every statement containing the IO or ASsignment COBOL subset is searched for FX-DATA or DEMO-STUFF with the single command. For example:

```
FX FX-DATA + DEMO-STUFF IN IO + AS
```

The INDIRECT operand of the FX primary command can be used to locate the references to a dataname and its aliases. It can also be used to locate other datanames that are indirectly affected by the change in size, or the change in value, of a dataname. [Figure 10](#) shows all datanames affected by the change in size of STOP-PAY-AMOUNT.

Figure 10 • FX INDIRECT Results

```

File View Test Search Logic List Options Help
-----
Program View                                VIAPCOB.VIAPCOB -A
Command ==> _____ Scroll ==>
CSR

000505          UNTIL STOP-PAYEE > STOP-MAX-INIT.          DATA USE
-----
3 LINES NOT DISPLAYED
000509          VARYING STOP-PAYEE FROM 1 BY 1
000510          UNTIL STOP-PAYEE > STOP-MAX-PAYEES.        DATA USE
-----
34 LINES NOT DISPLAYED
000545          VARYING STOP-PAYEE FROM 1 BY 1
000546          UNTIL STOP-PAYEE > STOP-MAX-PAYEES.        DATA USE
-----
2 LINES NOT DISPLAYED
000549          VARYING STOP-PAYEE FROM 1 BY 1
000550          UNTIL STOP-PAYEE > STOP-MAX-PAYEES.        DATA USE
-----
2 LINES NOT DISPLAYED
000553          VARYING STOP-PAYEE FROM 1 BY 1
000554          UNTIL STOP-PAYEE > STOP-MAX-INIT.          DATA USE
-----
10 LINES NOT DISPLAYED

| ASG0443I 29 DATA REFS: 16 DEFS, 11 USES, 2 MODS, 2 LEVELS FOUND FOR |
| STOP-PAY-AMOUNT. |
|
000570          ADD STOP-PAY-AMOUNT (STOP-PAYEE) TO STOP-PAY-TOTAL.  DATA MOD

```

This screen is the result of an X ALL;FX STOP-PAY-AMOUNT INDIRECT SIZE command. All levels is the default for this command. All direct and indirect references of the STOP-PAY-AMOUNT size change are displayed.

The information is shown one level at a time to see the ripple effect. [Figure 11](#) shows all datanames that are affected in the first indirect level for STOP-PAY-AMOUNT.

Figure 11 • FX INDIRECT Levels

```

File View Test Search Logic List Options Help
-----
Command ==> Program View VIAPCOB.VIAPCOB -A
Scroll ==> CSR

- - - - - 350 LINES NOT DISPLAYED
000565 COMPUTE STOP-PAY-AMOUNT (STOP-PAYEE) = DATA USE
000566 STOP-PAY-HOURS (STOP-PAYEE) * DATA USE
000567 STOP-PAY-RATE (STOP-PAYEE). DATA USE
- - - - - 2 LINES NOT DISPLAYED
000570 ADD STOP-PAY-AMOUNT (STOP-PAYEE) TO STOP-PAY-TOTAL. DATA MOD
- - - - - 17 LINES NOT DISPLAYED
000588 MOVE 0 TO STOP-10-NUMBERS (STOP-PAYEE). DATA USE
- - - - - 386 LINES NOT DISPLAYED
***** BOTTOM OF DATA *****

U
| ASG0443I 20 DATA REFS: 13 DEFS, 6 USES, 1 MOD, 1 LEVEL FOUND FOR |
| STOP-PAY-AMOUNT. |
|

```

This screen shows the references to STOP-PAY-AMOUNT and the first level of indirect references created with this command:

```
X ALL;FX STOP-PAY-AMOUNT INDIRECT LEVELS 1
```

A powerful feature of SmartTest is that command results can be used in a more detailed examination. The results of a FINDXTND command can be used to perform these tasks:

- Scroll through highlighted lines using the SCROLL command.
- Group highlighted lines together without intervening source code using the EXCLUDE command X NHI (exclude non-highlighted lines).
- Search for another target within the highlighted lines.
- Save the highlighted lines for later use by naming them with the MARK command.
- Save the results for later reference using the LPRINT and LPUNCH commands.

Paragraph Cross-reference

The PREF command is used to view the View - Paragraph Cross Reference pop-up, which shows program logic at the paragraph level and the control flow between paragraphs. The paragraph cross-reference feature identifies this information:

- The paragraphs that transfer control to this paragraph and what mechanism is used to transfer control.
- The paragraphs to which this paragraph can transfer control and what mechanism is used to transfer control.

This command displays the labels of paragraphs to which the paragraph PARM-EDIT can transfer control as shown in [Figure 12](#):

```
PREF LABEL PARM-EDIT NEXT
```

Figure 12 • Paragraph Cross Reference with Direction NEXT

```

View - Paragraph Cross Reference
Command ==> _____ Scroll ==> CSR
Target(s): LABEL PARM-EDIT
Direction: NEXT

  A : Add to target      P : Execute PREF      S : Select for viewing
S  Target paragraph(s)  How                    Goes to
-----
-  PARM-EDIT           RETURN                PROGRAM-INIT
-  " "                RETURN                MAIN-ROUTINE-CHECKALL
-  " "                PERFORM               BUILD-PADDED-PARM
-  " "                PERFORM               FIND-FIRST-PARAMETER
-  " "                PERFORM               FIND-NEXT-PARAMETER
***** BOTTOM OF DATA *****

```

Reusing Command Results

A particularly useful feature SmartTest provides is the ability to reuse command results. For example:

- Mark names are used to save sets and paths for later use when doing in-depth searches.
- The IN clause of the FINDXTND and TRACE commands can be used to limit the scope of a search to only the desired sets and paths.

This FINDXTND command results in a search for the string MAST-RPT occurring within an IO statement:

```
FX 'MAST-RPT' ALL IN IO
```

All input/output statements containing 'MAST-RPT' are highlighted. Since the results of the FINDXTND command are highlighted, the HI screen subset can be specified in another command to include them.

This FINDXTND command results in a search of the highlighted statements for the string LINE2:

```
FX 'LINE2' ALL IN HI
```

Since the highlighted statements resulted from the search for the string 'MAST-RPT' occurring within IO statements, the result of the above FINDXTND command is all IO statements that contain the string 'MAST-RPT' and the string 'LINE2'.

There is no limit to the number of times command results can be reused in this manner. This table lists the different ways the IN clause is used with search commands and the results:

Command	Results
FINDXTND 'WRITE' WORD IN IO	Finds all IO statements containing WRITE as a distinct word (WRITE statements).
FLOW FROM 291 TO PROD-COST	Highlights statements containing references to PROD-COST and creates a NETWORK of all paths from Line 291 to those statements.
TRACE ZIPCODE IN NETWORK	Follows the execution paths in NETWORK to references of ZIPCODE, stopping at each branch for specification of a statement to follow. A path called TRACK is created which includes all statements that have been followed.

Optimization Considerations

SmartTest processes COBOL II and later programs compiled with the OPTIMIZE compiler option, with these considerations:

- Repeated Code Segments. When testing with the ASM option set ON, the disassembled code that displays for repeated code segments (i.e., embedded PERFORMs or subprograms) are shown only for the first occurrence of such code (i.e., the lowest address).
- Pseudo Code User Labels. When inserting pseudo code into repeated code segments (i.e., embedded PERFORMs or subprograms), you may not insert user labels. This is because all pseudo code variables and labels are global, and insertion of multiple labels (one for each occurrence of the code) causes duplicate labels.
- Pseudo Code in Repeated Segments. When inserting pseudo code into repeated code segments (i.e., embedded PERFORMs or subprograms), avoid the use of the &COUNT internal variable. &COUNT maintains a separate instruction count for each separate occurrence of the repeated code. Instead, create a pseudo code variable, increment it on each pass through the pseudo code, and test the pseudo code variable rather than the &COUNT variable.
- Source Code Appears to be Unexecutable. On the Program View screen, it may appear that certain source lines are not executable. Using commands such as STEP, BREAK, and ZA (Zoom Assembler) on these lines may cause unexpected results.
- GO Command. Use the SmartTest GO command with caution. Paragraphs and lines of source code can be severely altered or eliminated in the optimization process. As a result, unpredictable or erroneous results can occur with the GO command. As a navigational tool for testing optimized programs, see the optimized Assembler output generated by the compiler.

Note: _____

For specific SmartTest command syntax, see the online help or the *ASG-SmartTest Reference Guide*. For additional information on the effects of optimization, see the appropriate Application Programming Guide for your version of COBOL.

Command Targets

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

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

Mark Name Target

Note: _____

Mark name targets may be used only if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

A mark name target is any user-generated path or set of lines named with the COPY, MARK, MERGE, or RENAME primary command; or an Insight system-generated path created by the FLOW or TRACE command. System-generated paths are NETWORK, SUBNETn, and TRACK.

Marks can be listed, renamed, copied, deleted and merged with other Mark name targets. All existing Mark names can be listed by using the LIST MARKS command, which displays the List - User Marks pop-up.

Perfrange Name Target

Note: _____

Perfrange name targets may be used only if Insight is installed and an Insight analysis has been run on the COBOL program being tested.

A perfrange name target consists of all executable statements within a PERFORM range, including all code that is executed by following that PERFORM. In this COBOL statement, the PERFORM range includes the paragraphs beginning with PARAGRAPH-ABC and continuing through the end of PARAGRAPH-XYZ, including any paragraphs that are executed or performed from within these paragraphs:

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

A single paragraph that is PERFORMed should be referenced by its name. Perfrange can also be a section contained in the Declaratives.

Subset Name Target

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

Line Range Target

A line range target 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 on the Program View screen. If the specified line number is greater than the last line in the program, the last line is assumed.

Label Name Target

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

Program Name Target

A program name target consists of all the code in all the divisions of the given program. For example, a nested program name in COBOL II or later.

Dataname Target

A dataname target can be any of these items:

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

Any legal COBOL reference for a data element can be specified as a dataname. If a variable is redefined to another name, SmartTest 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.

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. SmartTest locates valid references to the variable item as opposed to simple pattern matching of the characters in the variable name.

Fully-qualified datanames can be specified using the standard COBOL syntax, for example:

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

For COBOL II or later programs, a data item, label name, or program that may be ambiguous or used multiple times can be qualified with OF. If the COBOL label P120-READ exists in VIAPDEM2 and also in the program VIAPDEM1, it can be qualified with OF as shown in this example:

```
P120-READ OF VIAPDEM1
```

Multiple datanames can be located at the same time by concatenating the datanames with a + (plus sign) between them as shown in this example:

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

Datanames can be specified with one of these subordinate operands:

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

The BREAK, EXCLUDE, FINDXTND, HIGH, LPRINT, LPUNCH, and SCROLL primary commands also offer ALIAS/NOALIAS and DIRECT/INDIRECT operands. These are the valid operands:

Operand	Description
ALIAS/NOALIAS	ALIAS includes all aliases for the specified dataname and is the default.
DIRECT/INDIRECT	DIRECT includes only the specified dataname. INDIRECT locates any dataname indirectly affected by the specified dataname. INDIRECT data items can be further qualified using SIZE, VALUE, and LEVELS subordinate operands.
	SIZE Occurrences of a dataname indirectly affected by a change in the size of the dataname.
	VALUE Occurrences of a dataname directly or indirectly affected by a change in the value of the dataname.
	LEVELS Identifies the depth of the indirect references.

Pattern String Target

A pattern string target is a sequence of characters, surrounded by single or double quotes if it contains blanks. Strings of non-alphanumeric characters can be specified by the *X'string'*, *T'string'*, and *P'string'* operands. The string can be further qualified using the WORD, PREFIX, or SUFFIX subordinate operands.

Operand	Description
<i>X'string'</i>	A hexadecimal string, enclosed in single or double quotes.
<i>T'string'</i>	A text string, which disregards upper and lowercase, enclosed in single or double quotes.
<i>P'string'</i>	A picture string, enclosed in single or double quotes. These are the valid picture strings: <ul style="list-style-type: none"> 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 Target

A paragraph name target 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.

2

Test Session

This chapter describes how to set up a SmartTest test session for the TSO execution environment and contains these sections:

Topic	Page
Test Session Overview	26
Beginning a Test Session	26
User Options	28
Application Knowledge Repository (AKR)	35
Analyze Facility	38
Steps in Setting up the Test Session	44
Language Environment Testing	52
MVS Programs in TSO Foreground	53
Initiating a Test Session	61
Saving the SmartTest Testing Setup	62
Restoring the Testing Environment	64
Terminating a Test Session	65

Test Session Overview

The steps necessary to invoke SmartTest vary by site. Check with the systems administrator at your site for these details. The sample SmartTest session presented assumes you are familiar with the terminology used by SmartTest. The sample session assumes that your program has already been compiled, link-edited, and analyzed.

This section describes how to set up a test session for the TSO execution environment. For information on how to set up a CICS test session, see the *ASG-SmartTest CICS User's Guide*. For information on how to set up a IMS/DC test session, see the *ASG-SmartTest IMS User's Guide*.

After setting up a program for testing the first time, subsequent test sessions may not require all of these steps.

Note: _____

The steps involving a TSO file allocation CLIST relate only to execution in TSO foreground.

Beginning a Test Session

The method you use to invoke SmartTest depends on your system setup. If you need assistance to activate SmartTest, see your systems administrator.

If your site starts SmartTest directly, use the ISPF selection or CLIST as indicated by your systems administrator. After you activate the session, the SmartTest Primary Screen displays, as shown in [Figure 13 on page 27](#).

User Options

The first time you use SmartTest, you may need to customize options to reflect the appropriate settings for your environment. Some options are initially set to default values during product installation.

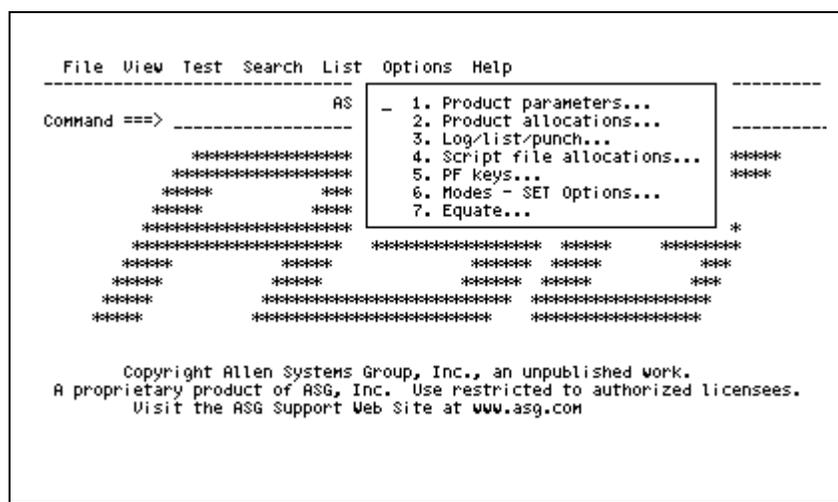
Use the Options pull-down to customize the SmartTest environment. Customization may include these functions:

- Setting and verifying online operations parameters on the Options - Product Parameters pop-up.
- Setting and verifying the Log, List, Punch, and Work file allocations in the Options - Product Allocations pop-up.
- Setting and verifying Log, List, and Punch file processing options and provide a Job card on the Options - Log/List/Punch Definition pop-up.
- Specifying Script file allocations on the Options - Script File Allocations pop-up.
- Setting and verifying values on the Options - PF Key Definition pop-up.
- Setting mode options on the Options - Modes screen.

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

- ▶ Select Options on the action bar. The Options pull-down, shown in [Figure 15](#), displays.

Figure 15 • Options Pull-down



Online Operation Parameters

Use the Options - Product Parameters pop-up to set parameters affecting the online operation of SmartTest. This includes specifying whether pseudo code and equates are to be saved.

To set online operation parameters

- 1 Select Options ► Product Parameters to display the Options - Product Parameters pop-up.
- 2 Enter the appropriate information in the fields. For a description of the fields, see the online help.

Log/List/Punch/Work File Allocations

Use the Options - Product Allocations pop-up to set and verify the DASD allocations for the Log, List, Punch, and Work files. Some options are 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 to display the Options - Product Allocations pop-up shown in [Figure 16](#).

Figure 16 • Options - Product Allocations Pop-up

```

                                Options - Product Allocations
Command ==> -----
Log file:
Generic unit . . . SRTDA      (generic group name or unit address)
Volume serial . . SRT801    (blank for authorized default volume)
List file:
Generic unit . . . SYSDA      (generic group name or unit address)
Volume serial . . SRT801    (blank for authorized default volume)
Punch file:
Generic unit . . . SRTDA      (generic group name or unit address)
Volume serial . . SRT801    (blank for authorized default volume)
Work file:
Generic unit . . . SYSDA      (generic group name or unit address)
Volume serial . . -----    (blank for authorized default volume)
Space units . . . CYLS      (BLKS, TRKS or CYLS)
Primary space . . 5          (space units)
Secondary space . 5          (space units)

```

- 2 Enter the appropriate information in the fields. For a description of the fields, see the online help.

Log/List/Punch Processing Options

Use the Options - Log/List/Punch Definition pop-up to set or change options and to process the SmartTest Log, List, and Punch files. These files are used for system message logging, error handling, and holding the results of several SmartTest commands. It is not necessary to exit SmartTest to process these files.

To specify the Log/List/Punch processing options

- 1 Display the Options - Log/List/Punch Definition pop-up using one of these methods:

From the action bar, select Options ► Log/list/punch.

Or

Type PRINTLOG (or PRINTLST) on any SmartTest screen that has a command prompt and press Enter.

The Options - Log/List/Punch Definition pop-up, shown in [Figure 17](#), displays.

Figure 17 • 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

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:
//USER1 JOB (ACCT),'USER1',PRTY=6
//          MSGCLASS=X
//*JOBPARM SYSAFF=CPUC
//*
    
```

- 2 Enter the appropriate information. For a description of the fields, see the online help.

Note: _____

If you specify the PK or PD option, a valid job card must be specified in the Job Statement Information field prior to processing any Log, List, or Punch files.

Customizing 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, SmartTest allocates the Log, List, and Punch files as:

```
userid.STTnnnnn.VIAxxxxx
```

where:

nnnnn 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.

To customize dataset names

- 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 18](#).

Figure 18 • Options - Log/List/Punch Name Customization Pop-up

```

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

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

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

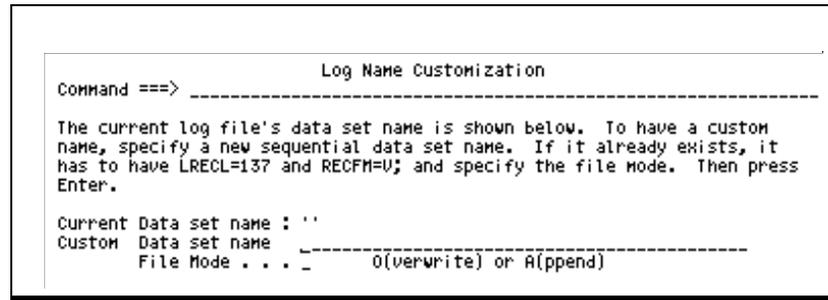
Log Data set name ----- (Seq)
  File Mode . . -      0(overwrite) or A(ppend)
List Data set name ----- (Seq)
  File Mode . . -      0(overwrite) or A(ppend)
Punch Data set name ----- (Seq)
  File Mode . . -      0(overwrite) or A(ppend)

```

- 2 Type U in the File Naming field for Log, List, and/or Punch to indicate a user-defined dataset name. If you type 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 type Y in the Prompt later for DSN field, SmartTest prompts you for the dataset name during file processing. If you specify Yes in the Prompt later for DSN field for the Log file, the Log Name Customization pop-up shown in [Figure 19](#) displays.

Figure 19 • Log Name Customization Pop-up



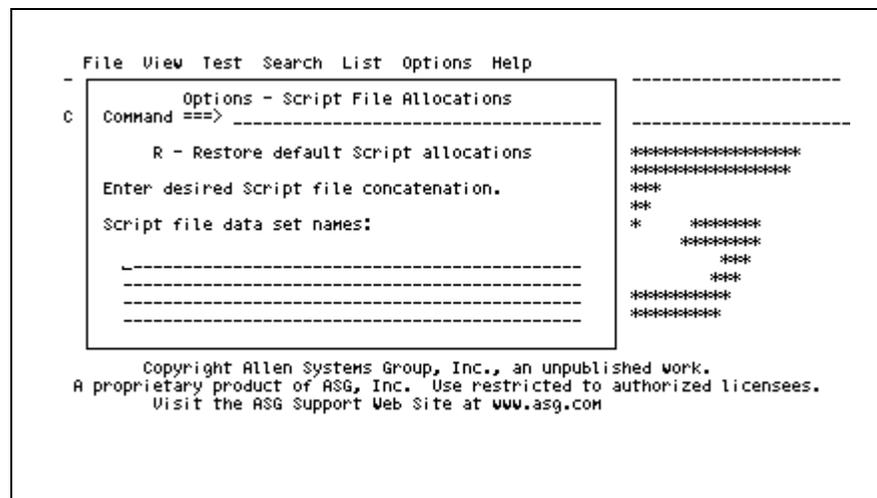
Specifying Script File Allocations

Use the Options - Script File Allocations pop-up to display and/or modify the default script file concatenation sequence for your session.

To specify your default script files

- 1 Select Options ► Script file allocations to display the Options - Script File Allocations pop-up as shown in [Figure 20](#).

Figure 20 • Options - Script File Allocations Pop-up



- 2 Enter or modify the dataset names to be used for the default script files for your session. To restore the default values for your installation, type R.

Note:

When initializing for defaults, the lines are filled beginning from the bottom to allow concatenation.

Setting PF Key Values

Use the Options - PF Key Definition pop-up to display and/or redefine PF key values used with SmartTest.

To set your PF Keys

- 1 Select Options ► PF Keys or type KEYS on any SmartTest screen that has a command prompt and press Enter. The Options - PF Keys Definition pop-up, shown in [Figure 21](#), displays.

Figure 21 • Options - PF Keys 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: 3278

PF01 HELP
PF02 SPLIT
PF03 END
PF04 RUN
PF05 RFIND
PF06 STEP
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 BRANCH
PF11 BRANCH BACKUP
PF12 RECALL

```

- 2 Enter 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.

Setting Mode Options

Use the Options - Modes screen to enable, disable, or enter a value for the mode of each listed option.

To set your modes

- 1 Select Options ► Modes to display the Options - Modes screen shown in [Figure 22](#).

Figure 22 • Options - Modes Screen

Option	Set	Description
ASM	OFF	Display Assembler code in status box, and instruction STEP
ASMVIEW	OFF	Display Assembler code for programs not on the AKR
AUTOQUAL	ON	QUALIFY to the Active Program after a RUN or STEP
BACKTRACK	OFF	BACKtrack Recording Mode is disabled with a buffer of 1M
BREAKS	ON	The BREAKpoint facility is enabled
COLUMNS	OFF	The COLUMNS option is disabled
CUA	ON	The CUA Menu facility is enabled
DATA	AUTO	Number of lines for Zoom Data windows, or AUTO
DELAY	1	Number of seconds to delay between steps during STEP AUTO
FLOATING	OFF	Display floating point registers in the status box
GENERATED	ON	Display generated code with the original source code
HEX	OFF	Display data in hexadecimal format
KEEP	AUTO	Number of lines for KEEP window, or AUTO
LANGUAGE	COB	The current LANGUAGE for the test session is COBOL
LE	OFF	Normal Language Environment error processing
LEARN	OFF	Display internally generated Primary Commands
LINK	OFF	Monitor LINKed-to programs for the test session
MAIN	ON	The program being tested is a MAINLINE
MONITOR	ON	The default for the RUN command is MONITOR

- 2 Type this command:

```
SET operand mode
```

where:

operand is the corresponding option.

mode is the desired mode. For example, SET ASM ON.

You can also modify an option's mode using the Set field on the Options - Modes screen.

Note:

The current setting for each option is saved between sessions, with the exception of BREAKS, PSEUDO, SCRIPT, and WHENS. The defaults for these options are restored when you initiate a new session.

Application Knowledge Repository (AKR)

Before programs can be tested, your program must be analyzed. Analyzed programs are stored in the AKR for use by SmartTest when testing. An AKR can be allocated in a BDAM or VSAM file organization. An AKR can be defined to be shared by all users, or multiple AKRs can be defined for use by departments, groups, or individuals.

Online and batch utilities are provided for managing the AKR. These utilities furnish these capabilities:

- Renaming and deleting programs
- Working with the AKR directory
- Allocating or expanding the AKR

To allocate an AKR

- 1 Select File ► AKR utility or type UTILITY. The File - AKR Utility pop-up, shown in [Figure 23](#), displays.

Figure 23 • 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 . . 'USER12,GENERAL,AKR'
Member . . . . . ----- (if "R" or "D" selected)
New name . . . . . ----- (if "R" selected)

Volume serial . . ----- (if not cataloged)
Password . . . . . ----- (if password protected)

```

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

- 3 Display the File - AKR Allocate/Expand pop-up by typing A in the command input area and pressing Enter.

Figure 24 • File - AKR Allocate/Expand Pop-up

```
Command ==> ----- File - AKR Allocate/Expand -----
S - Submit JCL      E - Edit JCL      C - Specify Catalog
Expand existing AKR . . . NO          (Yes or No)
AKR data set name . . . 'USER12.GENERAL.AKR'
Volume . . . . . -----
Unit . . . . . 12          (Generic unit name)
Space units . . . . . RECORDS (Records, Tracks or Cylinders)
Primary space . . . . . 4000  (Primary amount in above units)
Secondary space . . . . . 0   (Secondary amount in above units)

Job statement information:
//USER12 JOB (          ),
//          MSGCLASS=A
//*          INSERT '*ROUTE PRINT MODE.USER' HERE IF NEEDED.
/*
```

- 4 Enter the appropriate information in these required fields:
 - a Type NO in the Expand existing AKR field.
 - b Enter the AKR dataset name in AKR data set name field.
 - c Enter the appropriate SMS classes or the volume and space information as required for your site.
 - d Enter a valid Job card in the Job statement information: field.

For more information on the File - AKR Allocate/Expand pop-up, see the FILE command in the *ASG-SmartTest Reference Guide*.

- 5 Submit the JCL by typing S in the command input area and pressing Enter.
Or
Edit the JCL by typing E in the command input area and pressing Enter.
 - a While in the editor, make the appropriate modifications and issue the standard ISPF SUBMIT command.
 - b Return to the File - AKR Allocate/Expand by issuing the END primary command or pressing PF3.
- 6 Verify the AKR allocation results by examining the job output. Sample job outputs are presented in ["Verifying AKR Allocation Results" on page 37](#))

Verifying AKR Allocation Results

After the AKR allocation batch job has completed, review the job output to verify successful allocation and initialization. These examples show output excerpts with messages that indicate successful allocation and initialization of a VSAM AKR.

[Figure 25](#) shows the output of the IDCAMS step to allocate the AKR.

Figure 25 • IDCAMS Utility Condition Code Message Output

```
IDCAMS  SYSTEM SERVICES

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

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

[Figure 26](#) shows the output of a successful initialization of the AKR.

Figure 26 • AKR Initialization Message Output

```
ASG-CENTER-OS (390)          AKR UTILITY LOG

INIT DSNAME (USER.TEST.AKR)

ASG1316I AKR "USER.TEST.AKR" INITIALIZED.

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

[Figure 27](#) shows the output of the AKR utility after successful allocation of an AKR.

Figure 27 • AKR Utility Log Summary Message Output

```
ASG-CENTER-OS (390)          AKR UTILITY LOG - SUMMARY

ASG1300I      1 AKR(S) INITIALIZED      0 FAILED.

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

Analyze Facility

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

Program Analyze Requirements

The Program Analyze is similar to a COBOL compile or Assembler assembly. These basic program standards are required:

- COBOL II and later programs that receive error (E), severe (S), or unconditional (U) messages from the IBM compiler cannot be successfully analyzed.

Note: _____

For information on the COBOL compiler options needed by SmartTest, see ["COBOL Compiler Options" on page 289](#).

- Assembler programs that receive condition codes of 8 or greater from the IBM Assembler cannot be successfully analyzed.

Program Analyze Input

These items are input to the Program Analyze:

- JCL to compile and link the program. The JCL should be the complete JCL used to compile and link the program, and should contain these appropriate steps:
 - Retrieve the source from the source manager (such as Librarian or Panvalet)
 - Execute any preprocessors
 - Invoke the compiler
 - Invoke the linkage editor
- Program Analyze features. These features indicate the type of analysis to be performed.
- An allocated Application Knowledge Repository to receive Program Analyze output.
- Program Analyze options.

Analyzing a Program

The Program Analyze job can be submitted using these methods:

- From the File - Analyze Submit pop-up (see ["Method 1 - Analyzing a Program Using SmartTest" on page 39](#)).
- From a TSO/ISPF edit screen by executing the VIASUB edit macro (see ["Method 2 - Analyzing a Program from an Edit Session" on page 41](#)).
- From any TSO/ISPF screen or user screen by executing the VIASUBDS CLIST (see ["Method 3 - Analyzing a Program from a User Screen" on page 43](#)).

This table shows the conditions for using each method of analysis:

When the Compile JCL Resides in:	Analyze the Program...
A PDS or Sequential Dataset	From within SmartTest. From within a TSO/ISPF edit session using the VIASUB edit macro. Using the VIASUBDS CLIST.
Librarian, Panvalet, or other user source manager when editing the JCL	From the edit session using the VIASUB edit macro.
Screen-driven submit facility that generates the JCL	From the edit session using the VIASUB edit macro. By executing the VIASUBDS CLIST.

Method 1 - Analyzing a Program Using SmartTest

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

Note:

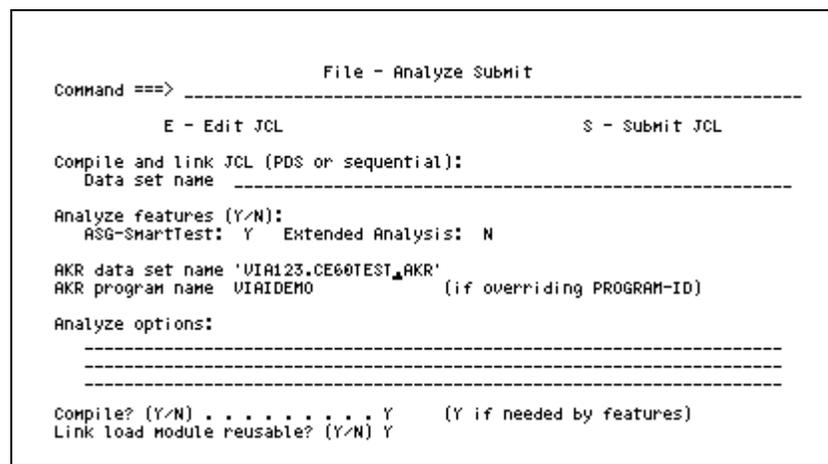
If the compile/link JCL resides in a source manager such as Librarian or Panvalet, use ["Method 2 - Analyzing a Program from an Edit Session" on page 41](#).

For information on the COBOL compiler options needed by SmartTest, see ["COBOL Compiler Options" on page 289](#).

To analyze a program while in SmartTest

- 1 Select File ► Compile/Analyze. The File - Analyze Submit pop-up, shown in [Figure 28](#), displays.

Figure 28 • File - Analyze Submit Pop-up



- 2 Enter the appropriate information in the required fields:
 - 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 SmartTest field. If you have Insight installed and want to take advantage of the extended analyze feature, type Y in the Extended Analysis field.

Other analyze features are available on this screen when additional ESW products are installed at your site.
 - c Specify the AKR dataset name in the AKR Data set name field.

For more information on the File - Analyze Submit pop-up, see ["The Analyze Process" on page 227](#).

- 3 Submit the compile/link JCL by typing S in the command input area and pressing Enter.

Or

Edit a temporary copy of the compile/link JCL by typing E in the command input area and pressing Enter.

- a While in the editor, make the appropriate modifications and issue the standard ISPF SUBMIT command.
- b To return to the File - Analyze Submit pop-up, issue the END primary command or press PF3/PF15.

Edits to the JCL are not saved.

- 4 Verify the analyze results by examining the output reports. (See ["Verifying Analyze Results" on page 44](#) for more information.)

Method 2 - Analyzing a Program from an Edit Session

You may submit an analyze job outside of SmartTest through your regular edit session (ISPF, source manager) or user compile/linkedit dialog. Use the Analyze Submit Parameters screen to specify SmartTest-specific information.

For information on the COBOL compiler options needed by SmartTest, see ["COBOL Compiler Options" on page 289](#) for more information.

To analyze a program while in an edit session

- 1 Initiate an edit session on your standard compile/link JCL (see [Figure 29](#)).

Figure 29 • Sample JCL

```

EDIT ----- ASG.VIACENXX.CNTL(VIAMECII) - 01.03 ----- COLUMNS 001 072
COMMAND ==>                                     SCROLL ==> CSR

//ASG JOB (ASG), 'ASG-SMARTTEST DEMO'
//*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//**
//*****
//**          ASG-SMARTTEST  SAMPLE COBOL DEMO PROGRAM
//*****
//** SUPPLY A VALID JOBCARD AND A 'USERLIB' OVERRIDE (IF NECESSARY).
//**
//** IF ASG-SMARTTEST'S COBOL COMPILE AND LINK PROC ('VIAPCII') WAS
//** INSTALLED INTO A USER PROCLIB DURING INSTALLATION, JUST SUBMIT
//** THIS JOB. OTHERWISE, COPY 'VIAPCII' FROM THE ASG 'CNTL'
//** LIBRARY AFTER THESE COMMENTS, UNCOMMENT THE 'PEND', AND SUBMIT.
//**
//*****
//**
//VIASUB   EXEC VIAPCII,
//**       USERLIB='USER.TEST.LOADLIB',
//**       MEMBER=VIASUB
//VIAMERGE EXEC VIAPCII,
//**       USERLIB='USER.TEST.LOADLIB',
//**       MEMBER=VIAMERGE
//**

```

- 2 Make any changes necessary, such as job card information, program and/or load module names, copybook and/or load library names, and so forth.
- 3 Type VIASUB in the command input area on your Edit screen and press Enter. The Analyze Submit Parameters screen, shown in [Figure 30](#), displays.

Figure 30 • Analyze Submit Parameters

```

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

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

AKR data set name 'UIA123,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

Display this panel by default in the future? (Y/N) Y

```

- 4 Enter the appropriate information in these required fields:
 - a Verify that the SmartTest field under Analyze features contains a Y. If you have Insight installed and you want to use the extended analysis features, type Y in the Extended Analysis field.
 - b Enter the AKR dataset name in the AKR Data set name field.

Other analyze features are available on this screen when additional ESW products are installed at your site.

- 5 Submit the compile/link JCL by typing S in the command input area and pressing Enter.

Or

Edit a temporary copy of the compile/link JCL by typing E in the command input area and pressing Enter.

While in the editor, make the appropriate modifications and issue the standard ISPF SUBMIT command.

- 6 Type END and press Enter to return to the editor screen.

Note: _____

Edits to the JCL are not saved.

- 7 Verify the analyze results by examining the output reports. (See ["Verifying Analyze Results" on page 44](#) for more information.)

Method 3 - Analyzing a Program from a User Screen

If you use compile/link dialogs to enter information and invoke job submission, an option may be added to the user screens to specify that a SmartTest Analysis is to be performed. Check with the systems administrator at your site for these details.

If no SmartTest options have been added to your compile/link dialog screens, and there is an opportunity to edit the JCL prior to job submission, see ["Method 2 - Analyzing a Program from an Edit Session" on page 41](#).

For information on the COBOL compiler options needed by SmartTest, see ["COBOL Compiler Options" on page 289](#).

Verifying Analyze Results

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

- Acceptable compiler results.
- Acceptable linkage editor results.
- Messages indicating the program has been successfully analyzed.
- Program is stored in the AKR.

Storage in the AKR does not occur if the program does not compile and analyze successfully. The message `ASG0248I PROGRAM 'XXXXXXXX' WAS STORED...` indicates whether the program was stored successfully on the AKR.

DIAGNOSTICS LINES indicates the number of warnings, conditionals, errors, and disasters that occurred.

Steps in Setting up the Test Session

These are the steps generally involved in setting up a test for the first time in SmartTest:

- Compile/link and analyze programs to be tested.
- Prepare execution JCL for conversion to a TSO file allocation CLIST.
- Prepare input data for testing.
- Activate SmartTest.
- Provide program, module, and testing environment information.
- Convert execution JCL to TSO file allocation CLIST.
- Initiate test session.

After you set up a program for testing, subsequent test sessions may not require all of these steps.

The steps involving a TSO file allocation CLIST relate only to execution in TSO foreground.

For these sections, assume the program has already been compiled, linked, and analyzed.

Actions

Action	Description
Setup test environment...	Displays the Environment Selection pop-up to set up the test environment.
Select Test Coverage...	Displays the File TCA Test Plan Selection pop-up if the SmartTest-TCA (Test Coverage Analysis) option is installed at your site. If it is not, this action displays a message indicating this option is not installed at your site.
Open...	Opens an analyzed program for viewing, inserting pseudo code, inserting breakpoints, and performing non-executable functions.
Close	Closes an open program.
Save	Saves pseudo code, marks, and/or equates in the AKR.
Compile/Analyze...	Displays the Analyze Submit pop-up that is used to submit a compile/analyze job.
AKR utility...	Allocates and expands an AKR, and deletes and renames members of an AKR.
Edit pseudo...	Inserts pseudo code into source code.
Execute...	Displays pop-ups to Read and execute a SmartTest script file.
Exit	Exits SmartTest.

Selecting the Testing Environment

To display the Environment Selection pop-up

- 1 Select File ► Setup test environment action.
- 2 Type 2 in the Setup Options field or type ENV in the primary command area on any screen and press Enter. The Environment Selection pop-up, shown in [Figure 32](#), displays.

Figure 32 • Environment Selection Pop-up

```

Command ==> Environment Selection
-----
A - Specify additional AKRs      L - Specify additional LOADLIBS
P - Specify PROCLIBS           D - Display AKR Directory

Environment selection:  Current environment is TSO
Online: 1 - TSO          5 - IMS/DB   Batch Connect: 9 - MVS Batch
        2 - CICS         6 - BTS      10 - IMS Batch
        3 - ISPF Dialog  7 - DB2    11 - BTS Batch
        4 - IMS/DC      8 - DB2 Procedure 12 - DB2 Batch

Application Knowledge Repositories (AKR):  1 Specified
'USER12.GENERAL.AKR'
-----

Application Load Libraries:  2 Specified
'USER12.GENERAL.LOAD'
'COB2.U400.COB2LIB'
-----

```

Options

Option	Description
A - Specify additional AKRs	Displays the System AKR Libraries pop-up to specify additional AKRs.
L - Specify additional LOADLIBS	Displays the System Load Libraries pop-up.
P - Specify PROCLIBS	Displays the Procedure Libraries pop-up specifying the PROCLIBS required for the test session.

Note:

If you do not enter any procedure libraries on the Procedures Libraries pop-up, the libraries listed in the PROCLIBS entry in VIA\$PRMS are used to search for procedures.

Option	Description
D - Display AKR Directory	Displays the Environment - AKR Directory pop-up showing members in the concatenated AKRs.
1 - TSO	Displays the TSO Session Setup screen to test MVS programs in TSO foreground (see "Specifying TSO Setup Information" on page 53 for more information).
2 - CICS	Displays the CICS Session Setup screen to test programs in the CICS environment. For more information, see the <i>ASG-SmartTest CICS User's Guide</i> .
3 - ISPF Dialog	Displays the ISPF Session Setup screen to test programs through ISPF Dialog Manager (see "Specifying ISPF Dialog Manager Information" on page 68).
4 - IMS/DC	Displays the IMS/DC Session Setup screen to test programs in the IMS/DC environment. For more information, see the <i>ASG-SmartTest IMS User's Guide</i> .
5 - IMS/DB	Displays the IMS/DB Session Setup screen to test IMS/DB programs in TSO foreground (see "Specifying IMS/DB Setup Information" on page 79).
6 - BTS	Displays the BTS Session Setup screen to test IMS/DC programs with BTS in TSO foreground (see "Specifying BTS Setup Information" on page 98).
7 - DB2	Displays the DB2 Session Setup screen to test DB2 programs in TSO foreground (see "Specifying DB2 Setup Information" on page 121).
8 - DB2 Procedure	If the DB2 Stored Procedure test option is installed at your site, displays the DB2 Stored Procedure Setup screen (see "DB2 Stored Procedure Testing Option" on page 123). If not, displays a message indicating that this option is not installed at your site.
9 - MVS Batch	Displays the Batch Session Setup screen to test MVS programs with batch connect (see "Specifying Batch Connect Setup Information" on page 130).
10 - IMS Batch	Displays the IMS Batch Session Setup screen to test IMS/DB programs with batch connect (see "Testing DL/I in the Batch Environment" on page 138).

Option	Description
11 - BTS Batch	Displays the BTS Batch Session Setup screen to test IMS/DC programs with BTS with batch connect (see "Testing BTS in the Batch Environment" on page 139).
12 - DB2 Batch	Displays the DB2 Batch Session Setup screen to test DB2 programs with batch connect (see "Testing DB2 in the Batch Environment" on page 140).
Application Knowledge Repositories (AKRs)	Specifies the AKR dataset name(s). The AKRs are concatenated in the order they are entered. Additional AKRs can be specified on the System AKR Libraries pop-up by typing A in the command area.
Application Load Libraries	Specifies the application load libraries to be used for the test session. Application load libraries are searched in the order entered. Note: _____ The Application Load Libraries field does not apply to the CICS or batch connect environments.

Fields

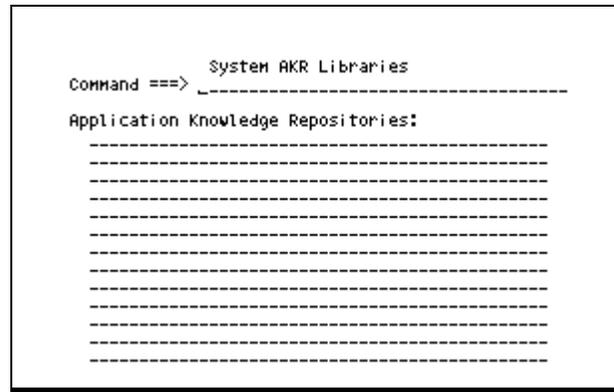
Field	Description
Specified	Indicates the total number of AKRs or Load libraries specified, including those specified on the System AKR Libraries pop-up or the System Load Libraries pop-up, respectively.

System AKR Libraries

To specify additional load libraries for the test session

- 1 Type A in the primary command area on the Environment Selection pop-up and press Enter. The System AKR Libraries pop-up, shown in [Figure 33](#), displays. These libraries are concatenated to those specified on the Environment Selection pop-up.

Figure 33 • System AKR Libraries Pop-up



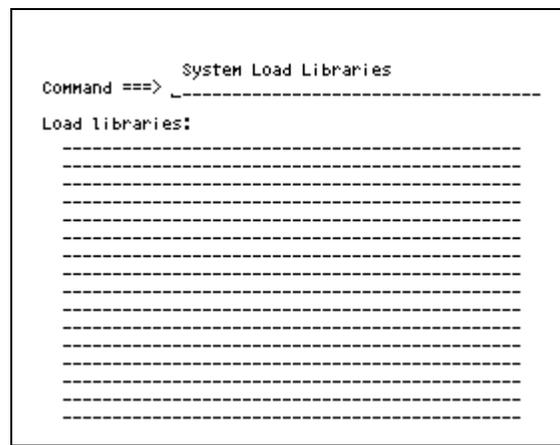
- 2 Press PF3/PF15 to return to the Environment Selection pop-up.

System Load Libraries

To specify additional load libraries for the test session

- 1 Type L in the primary command area on the Environment Selection pop-up and press Enter. The System Load Libraries pop-up, shown in [Figure 34](#), displays. These libraries are concatenated to those specified on the Environment Selection pop-up.

Figure 34 • System Load Libraries Pop-up



- 2 Press PF3/PF15 to return to the Environment Selection pop-up.

Language Environment Testing

This section details the requirements for testing Language Environment enabled programs in TSO.

- On the Options - Modes screen, set the LE and LECOND Set options to ON (see ["Setting Test Session Options" on page 180](#) for more information). These options affect testing of Language Environment enabled programs.

Option	Description
LE	Allows the Language Environment to process all errors without interpretation by SmartTest when set to ON. If set to OFF, SmartTest processes program checks before the Language Environment does. Setting LE to ON causes SmartTest to bypass monitoring of many CEE modules and gives the Language Environment control.
LECOND	Specifies whether SmartTest passes all errors to an error handler (registered through the CEEHDLR callable service) upon notification of the error from LE. When this option is set to OFF, SmartTest stops on the error condition. This option is used in conjunction with LE. It does not appear on the Options - Modes screen unless LE is enabled.

- On the Test Session Tailoring screen, set the Break on Entry and Break on Return options to YES. In addition, the Break on Entry option should be set to YES for the COBOL routine that issues the CEEHDLR call. (See ["Tailoring a Test Session by Program" on page 179](#) for more information.)
- On the Load Module Intercept List pop-up, request an intercept on the error handler routine. (See ["Specifying Programs to be Tested" on page 70](#) for more information.)
- Execute the TSO test session using either the MONITOR or NOMONITOR method (see ["Testing with SmartTest" on page 145](#), ["Testing Using the MONITOR Method" on page 145](#), and ["Testing Using the NOMONITOR Method" on page 146](#)) for more information. When an error occurs, you receive control in your error exit. You may step or run in your error handler code. When a break on return is encountered in the error handler, you must use the RUN NOMON command to complete the test. This gives control back to SmartTest on any break that is encountered in code that is executed after returning from the error handler. If there are no breaks, the RUN NOMON command takes you to TEST ENDED.

Note:

It is very important to use the RUN NOMON command at a break on return from your error handler. Results are unpredictable if you use a RUN MONITOR command at that point, especially on program checks (as distinguished from LE conditions). If you use a RUN MONITOR command, the next RUN command entered results in a ASG2153 message.

MVS Programs in TSO Foreground

Specifying TSO Setup Information

Use the TSO Session Setup screen to specify the TSO test session parameters for MVS programs and to initiate a test session.

To select the TSO environment

- 1 Select TSO on the Environment Selection pop-up to display the TSO Session Setup screen shown in [Figure 36](#).

Figure 36 • TSO Session Setup Screen

```

Command ==> _____ TSO Session Setup
                        -----
                        R - Begin TSO test session (RUN)
                        C - Convert batch JCL to TSO CLIST

Execution:              Options:
Load module  TESTCOBA   Break on entry (Y/N) NO
                        Break CSECT/pgm id  _____

Execution parameters:  (quotes are optional)
'B'
-----
File allocation CLIST:
Data set name 'USER.CLIST'
Member . . . VIEWEJCL           Deallocate after test NO

```

- 2 Enter the Load module in the Load module field. Required.
- 3 Enter any desired options. This step is optional.
- 4 Enter the CLIST dataset name and member in the Data set name and Member fields, respectively. Optional.

If a CLIST has already been created for the program to be tested, see ["Initiating a Test Session" on page 61](#) for more information. If a CLIST does not exist for the program to be tested, see ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#).

Options

Option	Description
R	Initiates the IMS/DB test session. This is the equivalent of the RUN command
C	Displays the Convert Batch JCL screen. (See "Converting Batch Execution JCL to a TSO CLIST" on page 55 for more information.)

Fields

Field	Description
Execution	
Load module	The initial load module to be tested. This should be the name of the program that is specified on the EXEC statement in the execution JCL for the program.
Options	
Break on entry (Y/N)	YES causes the test session to stop at the start of the test session. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
Break CSECT/pgm id	Entering a program name causes the test session to stop on entry to the specified CSECT in a statically linked module.
File allocation CLIST	
Dataset name	The dataset containing the allocation CLIST for allocating files to be used during testing. If the Convert Batch JCL facility is used, the dataset specified is shown here.
Member	The name of the allocation CLIST. If the Convert Batch JCL facility is used, the member name specified is shown here.

Field	Description
Execution parameters	Any required application parameters can be entered in this field.
Deallocate after test	<p>YES causes the allocation CLIST to be automatically executed to deallocate the test files when:</p> <ul style="list-style-type: none"> • At the end of the test session • After a CANCEL command is entered <p>Issuing another RUN command causes the datasets to be reallocated automatically.</p> <p>The default is NO, which causes the CLIST to be automatically executed to deallocate the test files before any of these situations:</p> <ul style="list-style-type: none"> • Exiting SmartTest • Switching test programs • Switching test environments

Converting Batch Execution JCL to a TSO CLIST

Execution in TSO foreground requires the data files used during testing to be allocated to the TSO session. This is accomplished by a TSO file allocation CLIST. The CLIST allocates the datasets to the TSO session, making them available for testing.

A JCL to CLIST conversion facility is provided in SmartTest to automate the process of CLIST creation. All DD statements in the JCL stream are converted regardless of the number of steps, unless otherwise indicated. A comment statement is included for each step.

Converting Batch JCL to CLIST

To specify the dataset containing the batch execution JCL to be converted

Note: _____

If the TSO Session Setup screen is already displayed, skip [step 1](#) and [step 2](#) and proceed to [step 3 on page 56](#).

- 1 Select File ► Setup test environment and press Enter. The File - Setup Test Environment pop-up displays.
- 2 Select Setup Options ► Select current environment and press Enter. The Session Setup screen displays.

- 3 Type C in the primary command input area and press Enter to display the Convert Batch JCL pop-up as shown in [Figure 37](#).

Figure 37 • Convert Batch JCL Pop-up

```

                                Convert Batch JCL
Command ==> -----
C - Convert batch JCL into CLIST      S - Extract setup libraries
E - Edit file allocation CLIST       F - Edit CLIST using panels
P - Specify procedure libraries      J - Edit batch JCL
A - Execute allocation CLIST        D - Execute deallocation CLIST

Batch Execution JCL:
Data set name 'VIASOFT.VIACEN50.CNTL'
Member . . . VIAMEJCL

File allocation CLIST:
Data set name 'USER.CLIST'
Member . . . ----- (Blank defaults to JCL member)

Options:
Delete [Y/N] NO      (Delete before create 'DISP=NEW' datasets)
Step   [Y/N] NO      (Only convert step with program to be tested)
Subsystem . . ---- (Source Library Manager subsystem name)
Step Name . . ----- (Unique step name in JOB to convert)
    
```

- 4 Enter the Batch Execution JCL dataset name and member in the Data set name and Member fields, respectively. This is the JCL to be converted.
- 5 Enter the File allocation CLIST dataset name and member in the Data set name and Member fields, respectively. This is where the converted CLIST is stored.

Options

Option	Description
C	Converts the specified batch execution JCL member into a file allocation CLIST. The CLIST dataset name and member name must be entered in the File Allocation CLIST fields.
E	Invokes the ISPF editor for the specified allocation CLIST member.
P	Displays the Procedure Libraries pop-up used to specify procedure libraries. If you do not enter any procedure libraries on this screen, the libraries listed in the PROCLIBS entry in VIA\$PRMS are used to search for procedures.
A	Invokes the CLIST processor with the generated CLIST as input and uses the ALLOC (allocate) parameter. This allocates the datasets to the TSO session and provides the files for testing.
S	Converts the specified batch execution JCL member and extracts Library information, such as STEPLIB, DFSRESLB, and so forth.
F	Displays the File Allocation screen, which enables you to edit the SmartTest allocation CLIST using screens rather than directly editing the CLIST.

Option	Description
J	Displays the batch JCL for editing.
D	Invokes the CLIST processor with the generated CLIST as input and uses the DEALC (deallocate) parameter. This frees the datasets from the TSO session.

Fields

Field	Description
Data set name	Specifies the partitioned dataset containing the JCL to be converted to a TSO CLIST. If the Subsystem option is to be used, specify the subsystem library dataset name.
Member	Specifies the member to be converted to a TSO CLIST.
Data set name	Specifies the partitioned dataset where the generated CLIST will be placed. Note that this dataset need not be defined by the TSO SYSPROC DD statement.
Member	Specifies the partitioned dataset member containing the generated allocation CLIST.
Delete (Y/N)	For datasets that are created 'NEW,CATLG', YES causes a DELETE to be issued each time the CLIST is executed, before the dataset is allocated. The default value is NO.
Step (Y/N)	YES causes only the step with the load module specified on the Session Setup screen to be converted. NO causes all DD statements in the JCL stream to be converted regardless of the number of steps in the procedure. The default value is NO.
Subsystem	If the JCL to be converted is stored in a source library that is installed with a subsystem option, enter the name of that subsystem. Also, the dataset and member name used by the source library management system must be specified in the Batch Execution JCL field.
Step Name	Specifies the unique step name within the JCL to be converted. This is applicable if Step is specified as YES.

Note:

The Edit Facility is not available for JCL stored in a library subsystem.

Editing Allocation CLIST

Method 1. Selecting E on the Convert Batch JCL screen displays the generated allocation CLIST, as shown in [Figure 38](#), which allows you to edit the CLIST manually.

Figure 38 • Generated Allocation CLIST Example

```

EDIT ---- USER.CLIST(VIAMEJCL) - 01.00 ----- COLUMNS 001 072
COMMAND ==>                                SCROLL ==> CSR
***** ***** TOP OF DATA *****
000001      PROC 0 VIAPARM(ALLOC)
000002      /* ASG-SMARTTEST CLIST ALLOCATE PROCESSING *****
000003      IF &VIAPARM = ALLOC THEN DO
000004          CONTROL NOMSG
000005          /* PROGRAM=VIAMERGE
000006          FREE FILE(INFILE1,INFILE2,INFILE3,OUTFILE,OUTRPT)
000007          FREE ATTRLIST(VATTR1,VATTR2)
000008          CONTROL MSG
000009          ATTRIB VATTR1 RECFM(F B) LRECL(230) BLKSIZE(460)
000010          ATTRIB VATTR2 RECFM(F B) LRECL(132) BLKSIZE(13200)
000011          ALLOC FI(INFILE1) DA('ASG.VIACENXX.CNTL(VIAMIN01)')      -
000012              SHR KEEP
000013          ALLOC FI(INFILE2) DA('ASG.VIACENXX.CNTL(VIAMIN02)')      -
000014              SHR KEEP
000015          ALLOC FI(INFILE3) DA('ASG.VIACENXX.CNTL(VIAMIN03)')      -
000016              SHR KEEP
000017          ALLOC FI(OUTFILE) DUMMY USING(VATTR1)
000018          ALLOC FI(OUTRPT) UNIT(SYSDA) SPACE(1,1) CYLINDERS -
000019              USING(VATTR2) NEW DELETE
000020          /* PROGRAM=VIAMERGE
000021      END
    
```

Special considerations apply for Generation Data Group (GDG) datasets. The TSO ALLOCATE statement created for Generation Data Groups specifies the absolute generation number represented by the catalog at the time of CLIST generation. Automatic incrementing of GDG entries does not occur. Each time the CLIST is executed the same generation is used unless the CLIST is manually edited or regenerated prior to execution. For example:

```
//FILE1 DD DSN=ASG.VIACENxx.IN1(0),DISP=SHR
```

converts to:

```
ALLOC FI(FILE) DA('ASG.VIACENxx.IN1.G0006V01') SHR KEEP
```

Method 2. Selecting **F** on the Convert Batch JCL screen displays the Allocations from JCL screen, as shown in [Figure 39](#), which provides a user interface for editing the CLIST. The basic editor provides a list of DDNAME and dataset information that corresponds to the CLIST allocations.

Figure 39 • Allocations from JCL Screen

```

----- Allocations from JCL ----- Row 1 to 5 of 5
===>                               SCROLL ==> PAGE
Long,Short,Attrib,CANCEL,END
Dataset:'VIAUSR.TEST.DATA(VIAMEJCL)'
Delete (Y/N) NO      (Delete before create 'DISP=NEW' datasets)

Options: Insert,Replace,Delete,Edit,View,Browse,Select
-----
DD Name  Data Set Names in Order of Concatenation      Disp
-----
INFILE1  'VIINST.CE50T001.CNTL(VIAMIN01)'              SHR   KEEP
INFILE2  'VIINST.CE50T001.CNTL(VIAMIN02)'              SHR   KEEP
INFILE3  'VIINST.CE50T001.CNTL(VIAMIN03)'              SHR   KEEP
OUTFILE  DUMMY
OUTRPT   NEW   DELETE
***** Bottom of data *****

```

Fields

Field	Description
Delete (Y N)	For datasets that are created NEW,CATLG, type YES to cause a DELETE to be issued each time the CLIST is executed, before the dataset is allocated. The default value is NO.
DD Name	The file name for the allocation entry. If this field is left blank, the dataset is concatenated to the previous entry.
Data Set Names in Order of Concatenation	The name of the dataset to be allocated. If DISP=DATA, this field is ignored.
Disp	The disposition with which to allocate the dataset. The valid values are SHR, OLD, NEW, MOD, DATA, and SYSOUT.

Commands

Command Type	Command	Description
Primary		
	END	Writes a CLIST from the current information and end the edit session.
	CANcel	Ends the edit session without saving changes.
	Attrib	Displays an additional line of information for each entry showing the dataset attributes, if present.
	Long	Displays all additional information for each entry in the list.
	Short	Resets the display to one line per entry.
Line		
	I	Inserts a new blank line after the selected line.
	R	Adds a new line after, and identical to, the selected line.
	D	Deletes the selected line.
	S	Invokes the attribute editor for the selected line.
	E	Invokes the ISPF editor for the dataset on the current line.
	V	Invokes the ISPF view for the dataset on the current line.
	B	Invokes the ISPF browse for the dataset on the current line.

Initiating a Test Session

After all information for testing a program in TSO foreground is specified, you must have completed these tasks:

- Selected the environment. (See ["Selecting the Testing Environment" on page 47](#) for more information.)
- Specified the Load module, AKR, load libraries, and procedure libraries (if needed). (See ["System Load Libraries" on page 50](#) for more information.)
- Generated a CLIST. (See ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#) for more information.)

Test initiation involves executing the TSO file allocation CLIST and displaying Program View. When test initiation is complete, the test session is active and waiting for a command. Program View displays program source or disassembled object code.

To initiate a test session, follow this step:

- ▶ Type R in the primary command input area on the TSO Session Setup screen and press Enter.

Or

Type RUN or STEP on any SmartTest screen or pop-up with a primary command input area. You may also use the PF4 (RUN) or PF6 (STEP) key.

Setup Considerations

TSO Foreground

During setup for testing in TSO foreground, keep these input/output dataset and execution JCL considerations in mind.

If a TSO file allocation CLIST does not exist at the time a test session is to be initiated, one must be created. A JCL to CLIST conversion facility is provided to automate the process.

These points may require you to modify the batch execution JCL prior to CLIST conversion:

- The JCL to be converted must be valid and executable outside of SmartTest.
- JCL SYSOUT designations of * (for example, //SYSPRINT DD SYSOUT=*) are routed to the terminal by the CLIST. Output routed to the terminal displays as it is written. It can only be scrolled forward, and is not available once the test session is terminated. You may want to assign the SYSOUT destinations to sequential files or the held output queue.
- Make temporary datasets permanent to insure their availability outside the TSO session.

When testing a multiple step job, complete these actions:

- Specify YES in the Step field of the Convert Batch JCL pop-up or edit the multiple step JCL so it contains only the step to be tested.
- Execute programs/utilities invoked in other steps separately (for example, SORT, IDCAMS). Those programs serving to allocate or delete files (for example, IEFBR14) may be skipped and the appropriate statements manually added to the CLIST after conversion.

Saving the SmartTest Testing Setup

For testing programs running in multiple environments, SmartTest provides the LIST PROFILE primary command. LIST PROFILE is used to save a test session setup and restore it when you need to test the same program again or a different program using the same setup.

To save the setup information in a profile dataset

- 1 Type LIST PROFILE in the primary command area of any SmartTest screen to display the Profile Data Set Member List screen. The command may be abbreviated LI PR.

- 2 Type W in the line command area beside any line that says AVAILABLE. In the User profile description field, enter any text that reminds you of the purpose of the setup being saved. You can include the program being tested, the application or project ID, and the date, as shown in [Figure 40](#).

Figure 40 • Profile Data Set Member List Screen

```

Command ==> ----- Profile Data Set Member List -----
The current environment is: TSO
Profile dataset name : 'USER12.ISPF.PROFILE'
COPY TO dataset name :
$ - Select member to restore      W - Write current environment to member
C - Copy selected member          R - Replace member (Pending status)
D - Delete member                 * denotes TCA Profile
Profile   Environ   User profile description (optional)
-----
- VIAPST01 AVAILABLE
- VIAPST02 AVAILABLE
- VIAPST03 AVAILABLE
- VIAPST04 TSO      test1
- VIAPST05 TSO      test for autotester
- VIAPST06 TSO      test viapcob and keep window
- VIAPST07 TSO      tca test tso profile
- VIAPST08 TSO      this is new stuff
- VIAPST09 TSO      Q/A transfer for TESTCOB RELEASE 3.3
- VIAPST10 TSO      test count for tom
- VIAPST11 TSO      analz test w/br
- VIAPST12 CICS     cics test
- VIAPST13 ISPF     ISPF setup 09/21/00_

```

Sharing Test Setups

To copy setup information from other profile datasets

- 1 In the Profile dataset name field on the Profile Data Set Member List screen, enter the dataset name of the profile dataset containing the desired setup. The saved profile list is refreshed to display the profiles from the new profile dataset.
- 2 In the line command area of the Profile Data Set Member List screen, select the desired profile to copy to your profile dataset by typing C on the line containing the profile.

The profile is now copied to your profile dataset and may be used and modified as desired.

Sharing an Alternate Profile Dataset

SmartTest allows you to create an alternate profile dataset, to enable team members to share test profile setups. The alternate dataset must be a non-ISPF dataset. Test profiles may be copied from the individual users' profile datasets to the alternate dataset, and from the alternate dataset to the ISPF datasets. The common dataset may replace the individual profile datasets as the default.

To copy test profiles to a common, alternate dataset

- 1 In the Profile dataset name field on the Profile Data Set Member List screen, type the name of the profile dataset containing the desired test setup profiles, for example:

'USER.ISPF.PROFILE'

- 2 In the COPY TO dataset name field, type the name of the common, alternate dataset to receive the profiles, for example:

'USER.TSO.CNTL'

Note:

The alternate must be a non-ISPF dataset.

The profiles are now copied to the common, alternate dataset.

To copy test profiles from a common, alternate dataset to an individual, ISPF dataset

- 1 In the Profile dataset name field on the Profile Data Set Member List screen, type the name of the common, alternate dataset containing the desired test setup profiles, for example:

'USER.TSO.CNTL'

- 2 In the COPY TO dataset name field, type the name of the individual ISPF dataset to receive the profiles, for example:

'USER.ISPF.PROFILE'

The profiles are now copied to the individual, ISPF dataset.

Restoring the Testing Environment

To restore a saved test session environment

- 1 Type LIST PROFILE in the primary command area of any SmartTest screen to display the Profile Data Set Member List screen.
- 2 Select a profile by typing S in the line command area of the line containing the profile and pressing Enter.

The saved setup is now restored. You can start the test session using the RUN or STEP command.

Terminating a Test Session

To end the active test session, follow this step:

- ▶ Select Test ▶ Cancel, or type CANCEL on any command line.

To exit SmartTest, follow this step:

- ▶ Select File ▶ Exit to end SmartTest. Any pseudo code, marks, or breaks are saved in the AKR if the online operation parameters have been saved as the default. See "[Online Operation Parameters](#)" on page 29 for more information on setting these parameters.

You can also exit SmartTest by pressing PF3 until you exit the product.

3

Test Session - Additional Environments

This chapter describes how to set up a SmartTest test session for the other execution environments and contains these sections:

Topic	Page
ISPF Dialog Manager	68
IMS/DB Programs in TSO Foreground	79
BTS in TSO Foreground	98
DB2 Programs in TSO Foreground	121
DB2 Stored Procedure Testing Option	123
Testing Programs in a Batch Region	130
Testing DL/I in the Batch Environment	138
Testing BTS in the Batch Environment	139
Testing DB2 in the Batch Environment	140
Testing DFHDRP in the Batch Environment	141

ISPF Dialog Manager

Specifying ISPF Dialog Manager Information

Use the ISPF Session Setup screen to specify the ISPF Dialog Manager test session parameters and to initiate a test session.

To select the ISPF Dialog Manager environment

- 1 Select ISPF Dialog Manager on the Environment Selection pop-up and press Enter. The ISPF Session Setup screen, shown in [Figure 41](#), displays.

Figure 41 • ISPF Session Setup Screen

```

                                ISPF Session Setup
Command ==> _____

R - Begin ISPF test session (RUN)    U - Verify ISPF allocations
S - Select programs to be tested

Execution:                          Options:
Load Module . . . _____        Break on entry (Y/N) YES
                                      Break CSECT/pgm id _____

ISPF Options:
Test Profile DSN _____
NEVAPPL . . . _____
Initial ISPF panel _____

Note: The Test Profile DSN cannot be the same ISPF profile data set
      that is in use by ASG-SmartTest (DDNAME=ISPPROF). Enter HELP
      for more information on the ISPF
OPTIONS.
    
```

- 2 Enter the initial load module to be tested in the Load module field.
- 3 Enter any appropriate option(s).

Options

Option	Description
R	Initiates the ISPF test session. This option is the same as issuing a RUN primary command (see "Initiating an ISPF Test Session" on page 78).
S	Displays the Load Module Intercept List pop-up (see "Specifying Programs to be Tested" on page 70).
V	Displays the ISPF Allocation pop-up (see "Specifying ISPF File Allocation Information" on page 70).

Fields

Field	Description
Execution	
Load module	The initial load module to be tested.
Options	
Break on entry (Y/N)	YES causes the test session to stop at the start of the test session. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
Break CSECT/pgm id	Entering a program name causes the test session to stop on entry to the specified CSECT in a statically linked module.
ISPF Options	
Test Profile DSN	The dataset that contains your ISPF profiles for the application to be tested (ISRPROF, VIASPROF, etc.). This must not be the same dataset that is being used by ISPF for the SmartTest test session. If none is entered, a temporary dataset is allocated, so each application profile is created when accessed. To access existing profiles, use ISPF's 3.2 option to allocate a library similar to your ISPPROF library, then use ISPF's 3.3 option to copy the appropriate application profile from ISPPROF to the new library. This dataset name is optional.
NEWAPPL	The ISPF application ID to use for the test (optional).
Initial ISPF panel	The first ISPF screen to be displayed by the test session. If none is entered, SmartTest's VPPISPFT screen displays. If VPPISPFT cannot be located in ISPF's ISPLIB concatenation, then ISPF's Dialog Manager Test screen displays. This screen ID is optional.

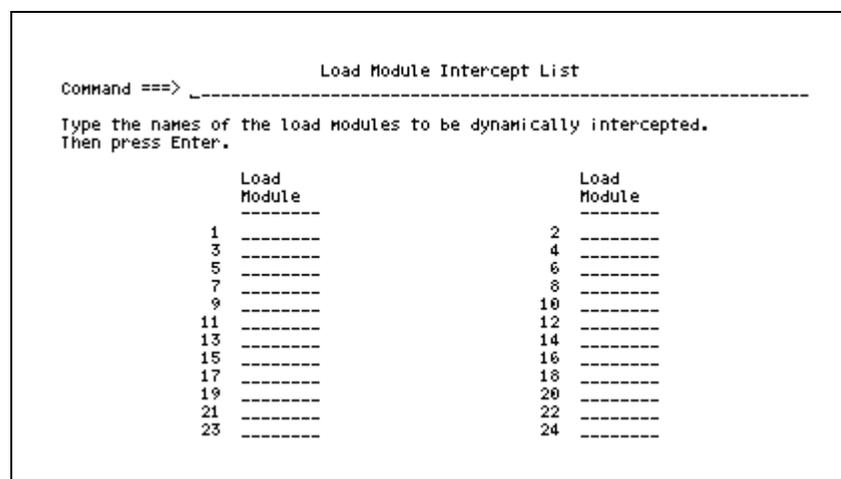
Specifying Programs to be Tested

Use the Load Module Intercept List pop-up to list load modules that may be intercepted by SmartTest for testing. A load module that is LINKed, ATTACHed, or invoked by an ISPEXEC SELECT program may be entered on the Load Module Intercept List screen. Entries must be made before the start of the test.

To list load modules that may be intercepted by SmartTest for testing

- 1 Type S in the primary command input area on the ISPF Session Setup screen and press Enter to display the Load Module Intercept List pop-up shown in [Figure 42](#).

Figure 42 • Load Module Intercept List Pop-up



- 2 Enter the load modules to be dynamically intercepted in the Load Module field and press Enter.
- 3 Press PF3/PF15 to return to the ISPF Allocation screen.

Specifying ISPF File Allocation Information

After selecting an item to be allocated to ISPF, the corresponding screen displays for entry of pertinent information.

Note: _____

To obtain a list of files allocated to your session, type `TSO VIASALCL`.

To select the ISPF datasets and libraries that are to be used by the ISPF test

- 1 Type V in the primary command input area on the ISPF Session Setup screen and press Enter to display the ISPF Allocation pop-up shown in [Figure 43](#).

Figure 43 • ISPF Allocation Pop-up

```

                                ISPF Allocation
Command ==> _____
1 - ISPF Program Load Library (required)
2 - ISPF Panel/Link Libraries (optional)
3 - ISPF Table/Message/Skeleton Libraries (optional)
4 - ISPF LIST Data Set (required)
5 - ISPF LOG Data Set (required)

A - ALL (Display all of the above in succession)
R - Restore ISPF system variables

```

- 2 Select the appropriate option(s) for items to be allocated to ISPF. Typically, the information on these screens is entered once and need not be re-entered each time a test is performed.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up after you have allocated the ISPF datasets and libraries.

Options

Option	Description
1-ISPF Program Load Library	Displays the ISPF Program Load Library pop-up used to specify the location of the load module ISPMMAIN. (See "ISPF Program Load Library" on page 72.)
2-ISPF Panel/Link Libraries	Displays the ISPF Panel/Link Library pop-up used to specify panel and link libraries used by the ISPF test. (See "ISPF Panel/Link Library" on page 73.)
3-ISPF Table/Message/Skeleton Libraries	Displays the ISPF Table/Message/Skeleton Library pop-up used to specify table, message, and skeleton libraries used by the ISPF test. (See "ISPF Table/Message/Skeleton Libraries" on page 74.)
4-ISPF LIST Data Set	Displays the ISPF List Data Set Allocation pop-up used to specify the ISPF list dataset that is used for the ISPF test. (See "ISPF List Allocation" on page 75.)
5-ISPF LOG Data Set	Displays the ISPF Log Data Set Allocation pop-up used to specify the ISPF log dataset that is used for the ISPF test. (See "Allocating the ISPF Log Dataset" on page 77.)

Option	Description
A	Displays the ISPF allocation pop-ups in succession.
R	Executes the VIAPUSPF CLIST that restores the ISPF system variables to the site defaults.

Note: _____
See the discussion on modifying installed CLIST libraries in the *ASG-SmartTest Installation Guide* for detailed information on the VIAPUSPF CLIST.

ISPF Program Load Library

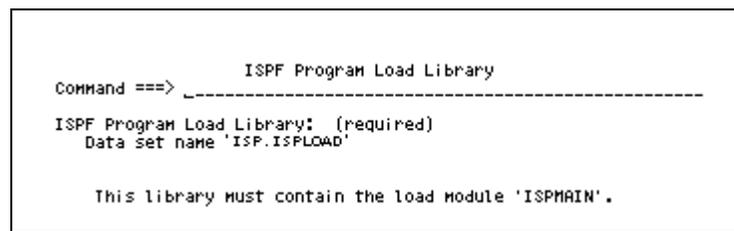
To specify the load library containing the member ISPMAIN

Note: _____

This information is required.

- 1 Type 1 in the command input area on the ISPF Allocation pop-up and press Enter to display the ISPF Program Load Library pop-up shown in [Figure 44](#).

Figure 44 • ISPF Program Load Library Pop-up



- 2 Enter the ISPF load library dataset name that contains the program ISPMAIN in the ISPF Program Load Library field.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up.

ISPF Panel/Link Library

ISPPPLIB and ISPLLIB are the defaults. Typically, no entries are needed on this pop-up. If left blank, the existing ISPF-defined libraries are used. If data is entered on this pop-up, all libraries that are accessed during a test session must be supplied. These entries override the existing definitions.

To specify panel and link libraries for the test

- 1 Type 2 in the primary command input area on the ISPF Allocation pop-up and press Enter. The ISPF Panel/Link Libraries pop-up, shown in [Figure 45](#), displays.

Figure 45 • ISPF Panel/Link Libraries Pop-up

```

ISPF Panel/Link Libraries
Command ==> _____
ISPF panel libraries: (corresponds to ISPPPLIB)
_____-
_____-
_____-
_____-
_____-
ISPF link libraries: (corresponds to ISPLLIB)
_____-
_____-
_____-
_____-
_____-

```

- 2 Enter the ISPF panel library datasets that will be used for the ISPF test in the ISPF panel libraries field.
- 3 Enter the ISPF link library datasets that will be used for the ISPF test in the ISPF link libraries field.
- 4 Press PF3/PF15 to return to the ISPF Allocation pop-up.

ISPF Table/Message/Skeleton Libraries

Note:

ISPTLIB, ISPMLIB, and ISPSLIB are the defaults. Typically, no entries are needed on this pop-up. If left blank, the existing ISPF-defined libraries are used. If data is entered on this pop-up, all libraries that are accessed during a test session must be supplied. These entries override the existing definitions.

To specify table, message, and skeleton libraries for the test

- 1 Type 3 in the primary command input area on the ISPF Allocation pop-up and press Enter. The ISPF Table/Message/Skeleton Libraries pop-up, shown in [Figure 46](#), displays.

Figure 46 • ISPF Table/Message/Skeleton Libraries

```
ISPF Table/Message/Skeleton Libraries
Command ==> _
ISPF Table libraries: (corresponds to ISPTLIB)
-----
-----
-----
ISPF Message libraries: (corresponds to ISPMLIB)
-----
-----
-----
ISPF Skeleton libraries: (corresponds to ISPSLIB)
-----
-----
-----
```

- 2 Enter the ISPF table library datasets that will be used for the test in the ISPF Table libraries field.
- 3 Enter the ISPF message library datasets that will be used for the test in the ISPF Message libraries field.
- 4 Enter the ISPF skeleton library datasets that will be used for the test in the ISPF Skeleton libraries field.
- 5 Press PF3/PF15 to return to the in the ISPF Allocation pop-up.

ISPF List Allocation

To specify an ISPF list file for the test

- 1 Type 4 in the primary command input area on the ISPF Allocation pop-up and press Enter to display the ISPF List Data Set Allocation pop-up shown in [Figure 47](#).

Figure 47 • ISPF List Data Set Allocation Pop-up

```

ISPF List Data Set Allocation
Command ==> _____
Enter Data set name, DUMMY, TEMP, TERM or SYSOUT:
Name . . . SYSOUT

SYSOUT . . X          Dest _____
DSN DISP  ___        (New, Old, or Shr)
Unit . . . _____
Volume . . _____

Space:
Units . . _____ (Cylinder, Track, or Block)
Primary  _____
Secondary _____

DCB:
RECFM . . F
LRECL . . 133
BLKSIZE 133

```

- 2 Enter the appropriate information in the fields and press Enter.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to a terminal, and SYSOUT can be specified to allocate the dataset to JES.
SYSOUT	Specifies a JES output class. An entry in this field is valid only if SYSOUT is specified in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, MOD, or SHR.
Dest	Specifies a JES destination of the SYSOUT output. This can be a remote ID or an NJE ID.
Unit	Specifies the generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK.
Primary	Specifies the number of primary cylinders, tracks, or blocks to be allocated.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks to be allocated.
DCB	
RECFM	Specifies the record format of the list dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the list dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the list dataset.

Note: _____

The SYSOUT and DEST entries are only allowed when the NAME entry is specified as SYSOUT.

Allocating the ISPF Log Dataset

To specify an ISPF log file for the test

- 1 Type 5 in the primary command input area on the ISPF Allocation pop-up and press Enter to display the ISPF Log Data Set Allocation pop-up shown in [Figure 48](#).

Figure 48 • ISPF Log Data Set Allocation Pop-up

```

ISPF Log Data Set Allocation
Command ==> _____
Enter Data set name, DUMMY, TEMP, TERM or SYSOUT:
Name . . . SYSOUT

SYSOUT . . X      Dest _____
DSN DISP  ___      (New, Old, or Shr)
Unit . . . _____
Volume . . _____

Space:
Units . . _____ (Cylinder, Track, or Block)
Primary  _____
Secondary _____

DCB:
RECFM . . F
LRECL . . 133
BLKSIZE 133

```

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to a terminal, and SYSOUT can be specified to allocate the dataset to JES.
SYSOUT	Specifies a JES output class. An entry in this field is valid only if SYSOUT is specified in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, MOD or SHR.
Dest	Specifies a JES destination of the SYSOUT output. This can be a remote ID or an NJE ID.
Unit	Specifies the generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK.
Primary	Specifies the number of primary cylinders, tracks, or blocks to be allocated.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks to be allocated.
DCB	
RECFM	Specifies the record format of the list dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the list dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the list dataset.

Note: _____

The SYSOUT and DEST entries are only allowed when the NAME entry is specified as SYSOUT.

Initiating an ISPF Test Session

After all information for testing a program is specified, you must have completed these tasks:

- Selected the ISPF Dialog Manager environment.
- Specified all appropriate testing options.
- Specified the Load module and AKR.
- Specified programs to be tested.
- Specified ISPF file allocation information.

When test initiation is complete, the test session is active and waiting for a command. Program View displays program source or disassembled object code.

To initiate a test session, follow this step:

- ▶ Type R in the primary command input area on the ISPF Session Setup screen and press Enter.

Or

Type RUN or STEP on any SmartTest screen or pop-up with a primary command input area. You may also use the PF4 (RUN) or PF6 (STEP) keys.

IMS/DB Programs in TSO Foreground

This section details the setup process for testing IMS/DB programs in TSO foreground.

Specifying IMS/DB Setup Information

To set up the IMS/DB environment for testing

- 1 Select IMS/DB on the Environment Selection pop-up to display the IMS/DB Session Setup screen shown in [Figure 49](#).

Figure 49 • IMS/DB Session Setup Screen

```

                                IMS/DB Session Setup
Command ==> _____

      R - Begin IMS test session (RUN)      C - Convert batch JCL to CLIST
                                           U - Verify IMS allocations

Execution:                                Options:
Load module _____                    Break on entry (Y/N) YES
PSB Name  . . _____                  Break CSECT/pgm id  _____

Data base region type:                    DB2 parameters:
DLI/DBB/BMP  DLI                          Plan name  . . . . . _____
                                           Subsystem name . . . . . _____

File allocation CLIST:
Data set name _____
Member  . . . _____                    Deallocate after test NO
    
```

- 2 Specify these fields:
 - a Required. Enter the load module in the Load module field.
 - b Enter the CLIST dataset name and member in the Data set name and Member fields, respectively.
 - c Enter any appropriate options.
- 3 Press PF3/PF15 to exit.

Note: _____

If a CLIST has already been created for the program to be tested, see ["Initiating a Test Session" on page 61](#). If a CLIST does not exist for the program to be tested, see ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#).

Options

Option	Description
R	Initiates the IMS/DB test session. This is the equivalent of entering the RUN command.
C	Displays the Convert Batch JCL screen used to convert batch JCL to an allocation CLIST. The converted allocation CLIST can be used to establish the IMS/DB test session.
V	Displays the IMS/DB Allocation Selection pop-up that is used to select the datasets, libraries, and parameters to be defined for IMS/DB.

Fields

Field	Description
Execution	
Load module	Specifies the initial load module to be tested. This should be the name of the program that is executed by IMS.
PSB Name	Specifies the IMS Program Specification Block associated with the program being tested.

Field	Description
Options	
Break on entry (Y/N)	Specifying YES causes the test session to stop on initial entry at the start of the load module. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
Break CSECT/pgm id	Specifying a program name causes the test session to stop on entry at the start of the specified CSECT in a statically linked module.
Database region type	
DLI/DBB/BMP	<p>Specifies the mode of the IMS/DB test session.</p> <ul style="list-style-type: none"> • DLI Uses private databases with database access through the TSO region; uses DBDLIB and PSBLIB. • DBB Uses private databases with database access through the TSO region; uses ACBLIB. • BMP Uses public databases with database access through the IMS Control Region.
DB2 parameters	
Plan name	Specifies the DB2 Plan that was generated for the program to be tested when the BIND was performed. If the type of test specified is BMP, this field is not needed.
Subsystem name	Specifies the name assigned to DB2 when it was installed in the MVS environment. If the type specified is BMP, this field is not necessary.
File allocation CLIST	
Data set name	Specifies the dataset containing the allocation CLIST for allocating files to be used during IMS/DB testing. If the Convert Batch JCL facility is used, the dataset specified is shown here.

Field	Description
Member	Specifies the name of the allocation CLIST. If the Convert Batch JCL facility is used, the member name specified is shown here.
Deallocate after test	Specifying YES causes the CLIST processor to be invoked automatically to deallocate the test files at the end of the transaction test session, or after a CANCEL command is entered. Issuing another RUN command causes the datasets to be reallocated. By default, the CLIST is automatically invoked before exiting SmartTest. The default is NO.

Specifying IMS File Allocation Information

To specify the IMS datasets, libraries, and parameters to be defined to IMS

- 1 Type V in the primary command input area on the IMS/DB Session Setup screen and press Enter to display the IMS Allocation Selection pop-up shown in [Figure 50](#).

Figure 50 • IMS Allocation Selection Pop-up

```

                    IMS Allocation Selection
Command ==> _____
1 - DFSRESLB/DFSESL
2 - PROCLIB/DFSUSAMP
3 - PSB/DBD Libraries
4 - ACB Libraries
5 - IMSMON
6 - IEFORDER

B - IMS Parms (BMP)
P - IMS Parms (DLI or DBB)
A - ALL (Display All Of The Above In Succession)
R - Restore IMS system variables and parms.
    
```

- 2 Select the appropriate option(s) for items to be allocated to IMS/DB. Typically, the information on these screens is entered once and need not be re-entered each time a test is performed.
- 3 Press PF3/PF15 to return to the IMS Allocation Selection screen.

Note: _____

If you choose option A, PF3/PF15 displays the next screen.

Options

Option	Description
1 - DFSRESLB/DFSESL	Displays the IMS DFSRESLB/DFSESL Allocation pop-up that is used to specify the IMS load library dataset.
2 - PROCLIB/DFSVSAMP	Displays the IMS DFSVSAMP/PROCLIB Allocation pop-up that is used to specify the VSAM buffer pool dataset and any datasets to be concatenated, and to specify the IMS procedure library dataset.
3 - PSB/DBD Libraries	Displays the IMS PSB/DBD Allocation pop-up that is used to specify the PSB and DBD libraries used by IMS.
4 - ACB Libraries	Displays the IMS ACB Allocation pop-up that is used to specify the ACB libraries used by IMS.
5 - IMSMON	Displays the IMS IMSMON Allocation pop-up that is used to specify the monitor dataset used by IMS to log run-time activities.
6 - IEFRDER	Displays the IMS IEFRDER Allocation pop-up that is used to specify the dataset that invokes the IMS logging facility.
B - IMS Parms (BMP)	Displays the IMS BMP Parms pop-up that is used to specify IMS BMP execution parameters.
P - IMS Parms (DLI or DBB)	Displays the IMS DLI/DBB Parms pop-up that is used to specify the IMS execution parameters for DLI and DBB programs.
A - ALL	Displays the pop-ups described above in succession.
R - Restore IMS system variables and parms	Executes the VIAPUIMS CLIST that restores the IMS system variables and parameters to their site defaults.
	<p>Note: _____</p> <p>See the section on modifying installed CLIST libraries in the <i>ASG-SmartTest Installation Guide</i> for detailed information on the VIAPUIMS CLIST.</p>

IMS DFSRESLB/DFSESL Allocation

Use the IMS DFSRESLB/DFSESL Allocation pop-up to specify the IMS and DB2 load library datasets. All IMS load modules, including DL/I, are expected to be accessed through DFSRESLB.

This is an example of typical DFSRESLB and DFSESL DD statements from batch execution JCL:

```
//DFSRESLB DD DSN=IMS.RESLIB,DISP=SHR
//DFSESL DD DSN=DSN.DSNLOAD,DISP=SHR
```

See your batch execution JCL and PROCs for the DFSRESLB and DFSESL dataset names used at your site.

To specify the IMS and DB2 load library datasets

- 1 Type 1 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS DFSRESLB/DFSESL Allocation pop-up shown in [Figure 51](#).

Figure 51 • IMS DFSRESLB/DFSESL Allocation Pop-up

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
Enter IMS Load Library Data Set Names	Specifies the IMS load library datasets to be allocated to the DFSRESLB DDNAME.
Enter DB2 Load Library Data Set Names	Specifies the DB2 load library datasets to be allocated to the DFSESL DDNAME.

IMS PROCLIB/DFSVSAMP Allocation

Use the IMS PROCLIB/DFSVSAMP Allocation pop-up to specify the datasets for VSAM buffer pools and IMS procedure libraries. The DFSVSAMP dataset must be allocated if databases are allocated using VSAM. Cataloged procedures used by IMS are contained in the PROCLIB dataset.

This is an example of typical DFSVSAMP and PROCLIB DD statements from batch execution JCL:

```
//PROCLIB DD DSN=IMS.PROCLIB, DISP=SHR
//DFSVSAMP DD DSN=IMS.PROCLIB(DFSVSAMP), DISP=SHR
```

See your batch execution JCL and PROCs for the DFSVSAMP and PROCLIB dataset names used at your site. The PROCLIB datasets may often be concatenated in a STEPLIB DD statement.

To specify the datasets for VSAM buffer pools and IMS procedure libraries

- 1 Type 2 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS PROCLIB/DFSVSAMP Allocation pop-up shown in [Figure 52](#).

Figure 52 • IMS PROCLIB/DFSVSAMP Allocation Pop-up

```

IMS PROCLIB/DFSVSAMP Allocation
Command ==> -----
Enter IMS PROCLIB Data set names:
'IMS.PROCLIB'
-----
Enter DFSVSAMP Data set names:
'IMS.PROCLIB(DFSVSAMP)'
-----
-----
-----
-----
-----
-----

```

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
Enter IMS PROCLIB Data Set Names	Specifies the procedure library datasets to be allocated to the PROCLIB DDNAME.
Enter DFSVSAMP Data Set Names	Specifies the datasets to be allocated to the DFSVSAMP DDNAME. Datasets to be concatenated to the DFSVSAMP dataset can be entered on the remaining lines.

IMS PSB/DBD Allocation

Use the IMS PSB/DBD Allocation pop-up to specify the PSB and DBD library datasets. PSB and DBD libraries must be allocated when the Database Region Type field on the IMS Session Setup pop-up contains DLI.

This is an example of typical IMS DD statements from batch execution JCL:

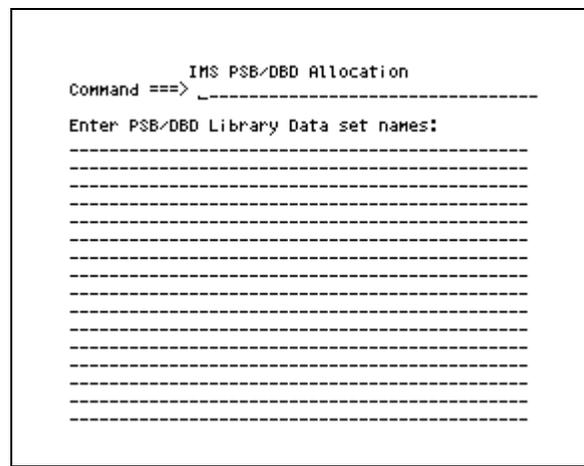
```
//IMS      DD  DSN=DSN=USER.IMS.PSBLIB,DISP=SHR
//         DD  DSN=DSN=USER.IMS.DBDLIB,DISP=SHR
//         DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
```

See your batch execution JCL and PROCs for the PSB and DBD dataset names used at your site. These datasets may often be concatenated in an IMS DD statement.

To specify the PSB and DBD library datasets

- 1 Type 3 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS PSB/DBD Allocation pop-up shown in [Figure 53](#).

Figure 53 • IMS PSB/DBD Allocation Pop-up



- 2 Enter the PSB and DBD dataset names to be allocated for use by IMS in the Enter PSB/DBD Library Data set names field.

Note:

The PSB and DBD datasets are concatenated to the IMS DDNAME.

- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

IMS ACB Allocation

Use the IMS ACB Allocation pop-up to specify the ACB library datasets. ACB libraries contain the combined PSB and DBD information about the program. ACB libraries should be allocated when the Database Region Type field on the IMS Session Setup screen contains DBB.

This example shows a typical IMSACB DD statement from batch execution JCL:

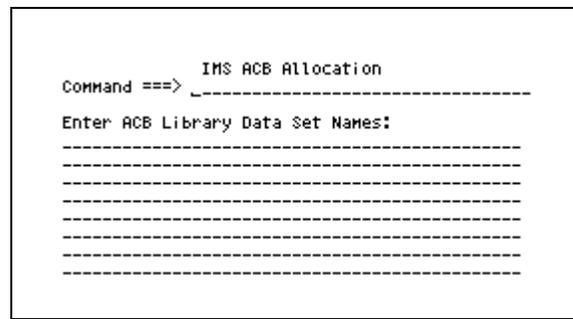
```
//IMSACB      DD  DSN=USER.IMS.ACBLIB,DISP=SHR
//            DD  DSN=IMS.IMS.ACBLIB,DISP=SHR
```

See your batch execution JCL and PROCs for the ACB dataset name used at your site.

To specify the ACB library datasets

- 1 Type 3 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS ACB Allocation pop-up shown in [Figure 54](#).

Figure 54 • IMS ACB Allocation Pop-up



- 2 Enter the ACB dataset names to be allocated to IMSACB for DBB testing in the Enter ACB Library Data Set Names field.
- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

IMS IMSMON Allocation

Use the IMS IMSMON Allocation pop-up to specify the DB monitoring datasets for output from the IMS monitor.

This is an example of a typical IMSMON DD statement from batch execution JCL:

```
//IMSMON DD DUMMY
```

See your batch execution JCL and PROCs for the IMSMON dataset name and attributes used at your site.

To specify the DB monitoring datasets

- 1 Type 5 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS IMSMON Allocation pop-up shown in [Figure 55](#).

Figure 55 • IMS IMSMON Allocation Pop-up

```

                    IMS IMSMON Allocation
Command ==> -----
Enter Data set name, DUMMY, TEMP, or blank:
Name . . . -----
DSN DISP ---      (New, Old, or Shr)
Unit . . . SYSDA
Volume . . . -----
Space:
Units . . . CYL   (Cylinder, Track, or Block)
Primary 1
Secondary 1
DCB:
RECFM . . . UB
LRECL . . . 2044
BLKSIZE 2048
```

- 2 Enter the appropriate information in the fields.

Note: _____

If the IMS monitor is to be used, it is necessary to allocate either a temporary dataset or permanent dataset in the Name field. Additionally, if the IMS monitor is to be invoked, it is necessary to specify Y in the MON field on the IMS DLI/DBB Parameters pop-up.

- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies the device type for the SYSOUT output such as SYSDA. A device type is only specified for new or temporary datasets.
Volume	Specifies the volume serial number containing the IMSMON dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK.
Primary	Specifies the number of primary cylinders, tracks, or blocks to be allocated.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks to be allocated.
DCB	
RECFM	Specifies the record format of the IMSMON dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the IMSMON dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the IMSMON dataset.

IMS IEFRDER Allocation

Use the IMS IEFRDER Allocation pop-up to specify an IEFRDER dataset to provide backout and recovery for your IMS databases.

This example shows a typical IEFRDER DD statement from batch execution JCL:

```
//IEFRDER DD DSN=IMSLOG, DISP=(NEW,CATLG),
//          UNIT=SYSDA, SPACE=(TRK,(3,2),RLSE),
//          DCB=(RECFM=VB,LRECL=2044,BLKSIZE=2048)
```

See your batch execution JCL and PROCs for the IEFRDER dataset name and attributes used at your site.

To specify an IEFRDER dataset

- 1 Type 6 in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS IEFRDER Allocation pop-up shown in [Figure 56](#).

Figure 56 • IMS IEFRDER Allocation Pop-up

```

IMS IEFRDER Allocation
Command ==> -----
Enter Data set name, DUMMY, TEMP, or blank:
Name . . . TEMP

DSN DISP  ___      (New, Old, or Shr)
Unit . . . SYSDA
Volume . . . -----

Space:
Units . . . CYL      (Cylinder, Track, or Block)
Primary 1
Secondary 1

DCB:
RECFM . . . UB
LRECL . . . 1916
BLKSIZE 1920
    
```

- 2 Enter the appropriate information in the fields.

Note: _____

Typing DUMMY in the Name field causes the IMS backout process to fail, should there be a necessity to back out updates to your applications databases.

- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies the device type for the SYSOUT output such as SYSDA.
Volume	Specifies the volume serial number containing the IEFORDER dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK.
Primary	Specifies the number of primary cylinders, tracks, or blocks to be allocated.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks to be allocated.
DCB	
RECFM	Specifies the record format of the IEFORDER dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the IEFORDER dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the IEFORDER dataset.

IMS BMP Parameters

Use the IMS BMP Parameters pop-up to specify execution parameters for IMS BMP programs.

This is an example of typical parameters from batch execution JCL:

```
//STEPNAME EXEC PGM=DFSRR00, PARM= (BMP, &MBR, &PSB, &IN,
//                               &OUT, OPT&SPIE&TEST&DIRCA, &PRLD,
//                               &TIMER&, &CKPTID, &PARDLI, &CPUTIME,
//                               &NBA, &OBA, &IMSID, &AGN, &SSM, &PREINIT)
```

See your batch execution JCL and PROCs for the parms used at your site.

To specify execution parameters for IMS BMP programs

- 1 Type B in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS BMP Parameters pop-up shown in [Figure 57](#).

Figure 57 • IMS BMP Parameters Pop-up

```

                                IMS BMP Parameters
Command ==> _____
Enter IMS BMP Execution parameters:
IN . . ----- (Input transaction code)
OUT . . ----- (Transaction code or logical terminal for output)
OPT . . C      (Operator option N, W, or C)
SPIE  0       (SPIE option 0 or 1)
TEST  0       (Validity check call list address 0 or 1)
DIRCA 000     (Region interregion communication area)
PRLD  --      (DFSMPLEX suffix or leave blank)
STIMER --     (Timer to be set 0 or 1)
CKPTID ----- (Checkpoint ID for restart or leave blank)
PARDLI 1      (Parallel DL/I option 0 or 1)
CPUTIME 0     (CPU time for IMS)
NBA . . ---   (Number Fast Path data buffers)
OBA . . ---   (Number additional page-fixed buffers)
INSID  ---    (Subsystem identifier)
AGN . . ----- (Application Group Name)
SSM . . ----- (DB2 subsystems member)
PREINIT --    (DFSINTXX suffix or leave blank)
APARM  ----- (APARM value or blank)
    
```

- 2 Enter the appropriate information in the fields.

Note:

See the *IBM IMS System Definition Reference Manual* for additional information on each of the IMS BMP execution parameters.

- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
IN	Use this parameter when the application program will be accessing the message queues. The OUT parameter is ignored when you specify this parameter.
OUT	Indicates the transaction code or logical terminal name to which an output message is to be sent. This parameter is needed when the application program sends output without accessing the input queues.
OPT	Indicates the action to be performed when the IMS control region is unavailable. Valid options are N (notify), W (wait), or C (cancel). The default value is N.

Field	Description
SPIE	Indicates whether control is passed when a program exception (OC1-OCF) occurs, thus allowing the program to correct the problem without an abend occurring. The default value is 0 (zero), which indicates control is passed.
TEST	Indicates whether the address in the user call list is checked for validity. The address in the user call list must be greater than the high address of the MVS nucleus and less than the highest virtual storage address of the machine. 1 indicates the address is to be checked. The default value is 0 (zero).
DIRCA	Specifies the interregion communication area of storage that is used by IMS to communicate with the test. The default value is 000.
PRLD	Specifies a suffix for DFSMPL that can be two alphabetic characters. The specified suffix is used to preload modules in the region. This field can be left blank if a suffix is not needed.
STIMER	Specifies whether the timer is to be set. If CPUTIME= <i>n</i> is specified, the STIMER value must be 1. STIMER=1 results in performance degradation and should only be specified when gathering statistics. 0 (zero) specifies that the timer is not to be set.
CKPTID	Specifies the checkpoint/restart ID used to restart a program.
PARDLI	Indicates where DL/I processing is to be performed. 0 (zero) specifies that DL/I processing is to be performed within the BMP region. 1 specifies that all DL/I processing is to be performed in the IMS control region.
CPUTIME	This parameter must be 0 (zero) for SmartTest.
NBA	Specifies the number of Fast Path data buffers. This field can be left blank if Fast Path databases are not used.
OBA	Specifies the number of additional page-fixed buffers for Fast Path applications when the standard buffers are all used.
IMSID	Indicates the subsystem identifier for the operating system being used. This identifier is used instead of the IMS identifier specified when the system was defined.
AGN	Indicates the Application Group Name used for resource access security.

Field	Description
SSM	Specifies a site-specific value that is used to allow access to selected DB2 subsystems under IMS. This field can be left blank or 1 to 4 alphanumeric characters can be entered.
PREINIT	Specifies a suffix for DFSINT that can be two alphabetic characters. DFSINT _{xx} contains a list of preinitialization modules to which control is to be given. This field can be left blank if a suffix is not needed.
Execution parameters (APARM)	Specifies the application parameter string. This is for IMS/ESA only.

Note:

See the *IBM IMS System Definition Reference Manual* for additional information on each of the IMS BMP execution parameters.

IMS DLI/DBB Parameters

Use the IMS DLI/DBB Parameters pop-up to specify execution parameters for IMS DLI/DBB programs.

This example shows typical parameters from batch execution JCL:

```
//STEPNAME EXEC PGM=DFSRR00, PARM=(DLI, &MBR, &PSB, &BUF,  
//          &SPIE&TEST&EXCPVR&RST, &PRLD, &SRCH,  
//          &CKPTID, &MON, &LOGA, &FMTO, &IMSID,  
//          &SWAP, &DBRC, &IRLM, &IRLMNM, &BKO, &IOB)
```

See your batch execution JCL and PROCs for the parms used at your site.

To specify execution parameters for IMS DLI/DBB programs

- 1 Type P in the primary command input area on the IMS Allocation Selection pop-up and press Enter to display the IMS DLI/DBB Parameters pop-up shown in [Figure 58](#).

Figure 58 • IMS DLI/DBB Parameters Pop-up

```

Command ==> _____
                                     IMS DLI/DBB Parameters
-----
Enter IMS DLI/DBB Execution Parameters:
BUF      8      (ISAM/OSAM buffer pool size)
SPIE     0      (SPIE option 0 or 1)
TEST     0      (Validity check call list address 0 or 1)
EXCPVR   0      (Long term fix of buffer pool 0 or 1)
RST      0      (UCF restart 0 or 1)
PRLD     --     (DFSMPLEX prefix or leave blank)
SRCH     0      (Module search 0=standard, 1=JPA and LPA first)
CKPTID   ----- (Checkpoint ID for restart or leave blank)
MON      N      (DB Monitor active Y or N)
LOGA     0      (BSAM or OSAM logging 0 or 1)
FMTO     N      (Formatted dump option T, P, or N)
IMSID    ----   (Subsystem identifier)
SWAP     Y      (Address space swappable or non-swappable Y or N)
DBRC     -      (Data Base Recovery Control)
IRLM     N      (Y or N to use IRLM)
IRLMNM   ----   (IRLM subsystem name or leave blank)
BKO      N      (Dynamic backout Y or N)
APARM    ----- (APARM value or blank)

```

- 2 Enter the appropriate information in the fields.

Note: _____

See the *IBM IMS System Definition Reference Manual* for additional information on each of the IMS execution parameters.

- 3 Press PF3/PF15 to return to the IMS Allocation Selection pop-up.

Fields

Field	Description
BUF	Specifies the ISAM/OSAM buffer pool size. The default value is 8.
SPIE	Indicates whether control is passed when a program exception (OC1-OCF) occurs, thus allowing the program to correct the problem without an abend occurring. The default value is 0 (zero), which indicates control is passed.
TEST	Indicates whether the address in the user call list is checked for validity. The address in the user call list must be greater than the high address of the MVS nucleus and less than the highest virtual storage address of the machine. 1 indicates the address is to be checked. The default value is 0 (zero).

Field	Description
EXCPVR	Indicates if real storage is to be reserved for use by IMS ISAM/OSAM buffers. The default value is 0 (zero), which indicates storage is not reserved.
RST	Indicates if the UCF (Utility Control Facility) is to be used for restarts. The default value is 0 (zero), which indicates UCF is not to be used.
PRLD	Specifies a suffix for DFSMPL that can be two alphabetic characters. The specified suffix is used to preload modules in the region. This field can be left blank if a suffix is not needed.
SRCH	Indicates where the system is to search for modules. 0 (zero) specifies the search is first in the JOBLIB/STEPLIB, then LINKLST, and then LPA. 1 specifies that the search begins first in JPA/LPA, then JOBLIB/STEPLIB, and then LINKLST. LPA (Link Pack Area) modules are loaded into a high storage area that is available for use by all jobs on the machine. JPA (Job Pack Area) modules are loaded into storage for a job. The default value is 0 (zero).
CKPTID	Specifies the checkpoint/restart ID used to restart a program.
MON	Indicates if the IMS monitoring option is active. The default value is N.
LOGA	Specifies the logging access method. This parameter is no longer used and is ignored if specified.
FMTO	Specifies whether formatted dump output is to be produced. T indicates IMS data areas are formatted and other areas are suppressed by the FDDL (Formatted Dump Delete List). P indicates no areas are suppressed and IMS data areas are formatted. N suppresses the formatted dump output.
IMSID	Indicates the subsystem identifier for the operating system being used. This identifier is used instead of the IMS identifier specified when the system was defined.
SWAP	Indicates if the address space can be swapped when the System Resource Manager (SRM) determines that an overload exists. An overload occurs when the CPU utilization or the paging rate is too high. The default value is Y (YES), which indicates the address space can be swapped.

Field	Description
DBRC	Indicates whether Database Recovery Control will be used for this execution of IMS. Y specifies that Database Recovery Control is to be used and must be entered if Y is specified for the IRLM option. N specifies that Database Recovery Control is not to be used for this execution of IMS. C is used only for batch backout type runs of IMS.
IRLM	Indicates if the IRLM (IMS Resource Lock Manager) is to be used. Y specifies that the IRLM is to be used. The default value is N, which specifies that the IRLM is not to be used.
IRLMNM	Specifies the name of the IRLM subsystem if IMS is sharing the database with other IMS systems. The IRLM subsystem name is first specified in the IMSCTRL macro that controls the IMS system. This field can be left blank if the IRLM option is not used.
BKO	Specifies whether database updates are backed out when an abend occurs. The default value is N, which indicates the dynamic backout option is not active.
Execution parameters (APARM)	Specifies the application parameter string. This is for IMS/ESA only.

BTS in TSO Foreground

Specifying BTS Setup Information

To specify the TSO test session parameters for BTS, and to initiate a test session

- 1 Select BTS on the Environment Selection pop-up to display the BTS Session Setup screen shown in [Figure 59](#).

Figure 59 • BTS Session Setup Screen

```

                                     BTS Session Setup
Command ==> _____
-----
R - Begin BTS test session (RUN)      C - Convert batch JCL to CLIST
P - Preview BTSIN data set           U - Verify BTS/IMS allocations

Execution:                            Options:
Load Module      _____          Break on entry (Y/N) YES
                                           Break CSECT/pgm id  _____

Data base region type:
DLI/DBB/BMP    DLI

File allocation CLIST:
Data set name  _____
Member . . .   _____          Deallocate after test NO

BTSIN:
Data set name  _____
Member . . .   _____

```

- 2 Specify these fields:
 - a Enter the load module in the Load module field. This is a required field.
 - b Enter the CLIST dataset name and member in the Data set name and Member field, respectively.
 - c Enter any appropriate options.

Note: _____

If a CLIST has already been created for the program to be tested, see ["Initiating a Test Session" on page 61](#). If a CLIST does not exist for the program to be tested, see ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#).

Options

Option	Description
R	Initiates the BTS test session. This is the equivalent of entering the RUN command.
P	Displays the Preview BTSIN Data Set pop-up that is used to select the BTS transactions and programs to be tested. Before selecting this option, specify the BTS dataset name and member.
C	Displays the Convert Batch JCL screen used to convert batch JCL to an allocation CLIST. The converted allocation CLIST can be used to establish the BTS test session.
V	Displays the BTS Allocation Selection pop-up that is used to select the datasets, libraries, and parameters to be defined for BTS and IMS.
Execution: Load module	Specifies the initial load module to be tested. This should be the name of the program that is executed by IMS.
Options: Break on entry (Y/N)	Specifying YES causes the test session to stop at the start of the test session. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
Break CSECT/pgm id	Specifying a program name causes the test session to stop on entry to the specified CSECT in a statically linked module.
Database region type: DLI/DBB/BMP	Specifies the mode of the IMS/DB test session. DLI - Uses private databases with database access through the TSO region. Uses DBDLIB and PSBLIB. DBB - Uses private databases with database access through the TSO region. Uses ACBLIB. BMP - Uses public databases with database access through the IMS Control Region.

Option	Description
File allocation CLIST: Dataset name	Specifies the dataset containing the allocation CLIST generated by the Convert Batch JCL facility.
Member	Specifies the name of the allocation CLIST generated by the Convert Batch JCL facility.
Deallocate after test	Specifying YES causes the CLIST processor to be automatically invoked to deallocate the test files at the end of the test session, or after a CANCEL command is entered. Issuing another RUN command causes the datasets to be reallocated. By default, the CLIST is automatically invoked to deallocate the test files before exiting SmartTest. The default value is NO.
BTSIN	
Dataset name	Specifies the BTS input control statement dataset that contains the PSBs and transactions available during the test session.
Member	Specifies the BTSIN dataset member.

Selecting BTS Transactions to Monitor

Use the Preview BTSIN Data Set screen to display `./T` commands residing in the BTSIN dataset. Select the transactions to be tested with SmartTest. Selected transactions are saved from session to session. The BTSIN data may need altering to reflect these items.

- Specify a `PSB=` parameter on the appropriate `./T` statement, even if the PSB has the same name as the program to be tested, for example:

```
./T TC=TRNX MBR=TEST1 PSB=TEST1 PLC=25 LANG=CBL TYPE=DLI
```

- Add the `TSO=NO` parameter on a `./O` statement to prevent BTS from prompting at the terminal for input during the test, for example:

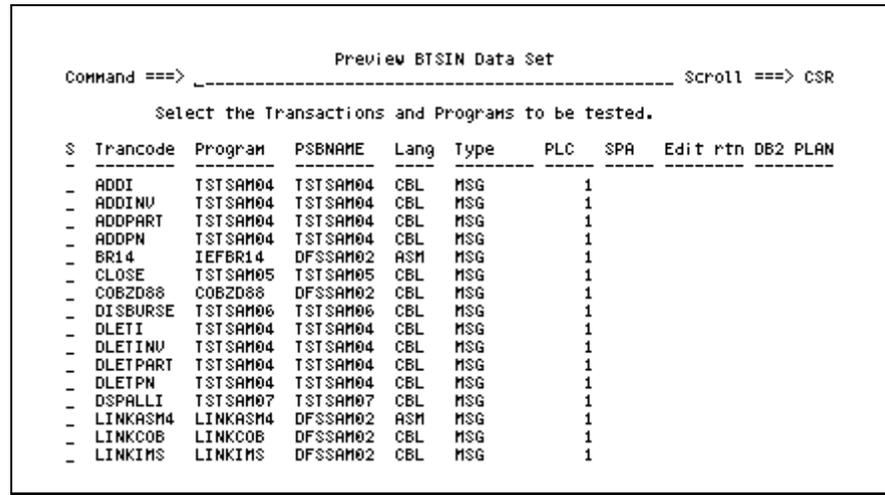
```
./O TSO=NO
```

The BTSIN dataset may not be altered from this screen.

To display /T commands residing in the BTSIN dataset

- 1 Type P in the primary command input area on the BTS Session Setup screen and press Enter to display the Preview BTSIN Data Set screen shown in [Figure 60](#).

Figure 60 • Preview BTSIN Dataset Screen



- 2 Select the transaction(s) to be tested by typing S in the S line command area.
- 3 Press PF3/PF15 to return to the BTS Session Setup screen.

Fields

Field	Description
S	Selects transaction for testing.
Trancode	Specifies the transaction code name of the primary or secondary transaction.
Program	Specifies the load module of the application program that processes the transaction named by the TC= operand.
PSBNAME	Specifies the alphanumeric PSB name to be used when processing the transaction named by the TC= operand.
Lang	Specifies the programming language of the module named by the MBR= operand. The default is ASM.
Type	Specifies the type of application program being defined, or the alternate logical terminal type. The default is MSG.

Field	Description
PLC	Specifies the processing limit count for this transaction. The default is 1.
SPA	Defines the size of the scratch pad area, in bytes, for the transaction named by the TC= operand.
EDIT rtn	Specifies the member name of the user-written transaction code (input) edit routine that is called to edit each input message segment.
DB2 PLAN	Specifies the DB2 plan name for the corresponding transaction.

Note:

If the program uses DB2, the DB2 name should be specified on the BTSIN transaction './T' cards. The subsystem name should be defined to BTS or specified in the BTSIN Patch './P' cards.

Specifying BTS File Allocation Information

To select the BTS datasets, libraries, and parameters to be defined to IMS

- 1 Type V in the primary command input area on the BTS Session Setup screen and press Enter to display the BTS Allocation Selection pop-up shown in [Figure 61](#).

Figure 61 • BTS Allocation Selection Pop-up

```

                    BTS Allocation Selection
Command ==> _____
1 - BTS Load library
2 - FORMAT
3 - QIOPCB
4 - QALTPCB
5 - QALTRAN
6 - BTSOUT
7 - BTSPUNCH
8 - BTSDDEBUG
9 - BTSSNAP

I - IMS allocations and parms
A - ALL (Display All Of The Above In Succession)
R - Restore BTS and IMS system variables and parms.

```

- 2 Select the appropriate option(s) for items to be allocated to BTS. Typically, the information on these screens is entered once and need not be re-entered each time a test is performed.
- 3 Press PF3/PF15 to return to this screen or to display the next screen.

Options

Option	Description
1 - BTS LOAD LIBRARY	Displays the BTS Load Library pop-up that is used to specify the BTS load library dataset.
2 - FORMAT	Displays the BTS Format Libraries pop-up that is used to specify the IMS MFS (Message Format Services) datasets.
3 - QIOPCB	Displays the BTS QIOPCB Allocation pop-up that is used to specify the BTS work file dataset.
4 - QALTPCB	Displays the BTS QALTPCB Allocation pop-up that is used to specify the BTS work file dataset for alternate PCBs.
5 - QALTRAN	Displays the BTS QALTRAN Allocation pop-up that is used to allocate the dataset to be used for alternate PCB output.
6 - BTSOUT	Displays the BTS BTSOUT Allocation pop-up that is used to specify the pop-up and message output dataset.
7 - BTSPUNCH	Displays the BTS BTSPUNCH Allocation pop-up that is used to specify the dataset used to capture all BTS input.
8 - BTSDEBUG	Displays the BTS BTSDEBUG Allocation pop-up that is used to specify the dataset used to capture SNAP dumps of the Trace table and various control blocks taken by BTS.
9 - BTSSNAP	Displays the BTS BTSSNAP Allocation pop-up that is used to specify the dataset used to capture all other SNAP dumps taken by BTS.
I - IMS allocations and parms	Displays the IMS Allocation Information pop-up that is used to specify the datasets, libraries, and parameters that are to be defined to IMS.
	<p>Note: _____</p> <p>See "Specifying IMS File Allocation Information" on page 82 for more information on the IMS Allocation Selection pop-up.</p>

Option	Description
A - ALL	Displays the pop-ups described above in succession.
R - Restore BTS and IMS system variables and parms	Executes the VIAPUBTS CLIST that restores the BTS and IMS system variables and parameters to their site defaults

Note:

See the section on modifying installed CLIST libraries in the *ASG-SmartTest Installation Guide* for detailed information on the VIAPUBTS CLIST. See the IBM IMS Batch Terminal Simulator: Program Reference/Operations Manual for more information on BTS datasets and parameters.

BTS Load Library

Use the BTS Load Library pop-up to specify the BTS load library and STAX indicator for BTS.

This is an example of a typical BTS load library DD statement from batch execution JCL:

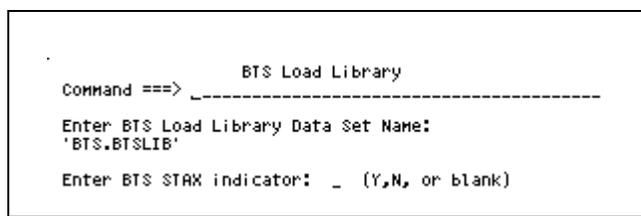
```
//STEPLIB DD DSN=BTS.BTSLIB,DISP=SHR
```

See your batch execution JCL and PROCs for the BTS load library dataset used at your site.

To specify the BTS load library and STAX indicator

- 1 Type 1 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS Load Library pop-up shown in [Figure 62](#).

Figure 62 • BTS Load Library Pop-up



- 2 Enter the load library dataset that contains the programs used by BTS.
- 3 Type Y or N in the Enter BTS STAX indicator field, or leave it blank.

Entering Y specifies that BTS is to process the TSO terminal attention exit. When this field is left blank, BTS does not allow the selection of the TSO terminal attention exit.

If the BTS0015A INVALID KEYWORD ON PARM STRING, 'DLI' ASSUMED message displays when the BTS test session is initiated, the STAX indicator contains the wrong value. If this message displays, change the value in this field to Y, N, or blank.

- 4 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

BTS Format Libraries

Use the BTS Format Libraries pop-up to specify the format datasets from the IMS/ESA Message Format Service (MFS) library.

This is an example of a typical FORMAT DD statement from batch execution JCL:

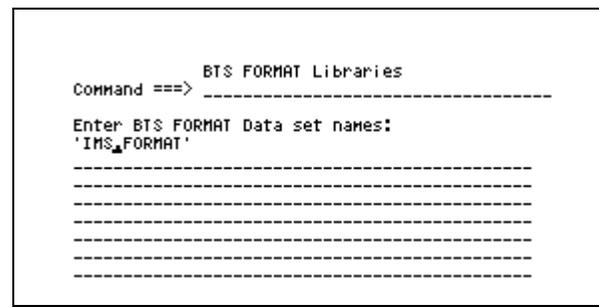
```
//FORMAT DD DSN=USER.TEST.FORMAT, DISP=SHR
// DD DSN=IMS.FORMAT, DISP=SHR
```

See your batch execution JCL and PROCs for the Format dataset name used at your site.

To specify the format datasets from the IMS/ESA MFS

- 1 Type 2 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS Format Libraries pop-up shown in [Figure 63](#).

Figure 63 • BTS FORMAT Libraries Pop-up



- 2 Enter the MFS datasets used to format screen message in the Enter BTS FORMAT Data set names field.
- 3 When all necessary information is specified, press PF3/PF15 to return to the BTS Allocation Selection pop-up.

BTS QIOPCB Allocation

Use the BTS QIOPCB Allocation pop-up to allocate the dataset containing the output message queue for BTS.

This is an example of a typical QIOPCB DD statement from batch execution JCL:

```
//QIOPCB DD UNIT=SYSDA,SPACE=(CYL,1,1),RLSE),
// DCB=(RECFM=FB,LRECL=1024,BLKSIZE=3072)
```

See your batch execution JCL and PROCs for the QIOPCB dataset name and attributes used at your site.

To allocate the dataset containing the output message queue

- 1 Type 3 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS QIOPCB Allocation pop-up shown in [Figure 64](#).

Figure 64 • BTS QIOPCB Allocation Pop-up

```

BTS QIOPCB Allocation
Command ==> -----
Enter Data set name, DUMMY, or TEMP:
Name . . . TEMP

DSN DISP      ___      (New, Old, or Shr)
Unit . . . SYSDA
Volume . . . -----

Space:
Units . . . CYLINDERS (Cylinder, Track, or Block)
Primary 1
Secondary 1

DCB:
RECFM . . . UBS
LRECL . . . 512
BLKSIZE 3072
    
```

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP indicates a temporary dataset is to be allocated. No allocation is performed if this field is left blank.
DSN disp	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.

Field	Description
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset. The default value is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default value is 3072.

BTS QALTPCB Allocation

Use the BTS QALTPCB Allocation pop-up to allocate the alternate message queue dataset used by BTS.

This is an example of a typical QALTPCB DD statement from batch execution JCL:

```
//QALTPCB DD UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
// DCB=(RECFM=FB,LRECL=1024,BLKSIZE=3072)
```

See your batch execution JCL and PROCs for the QALTPCB dataset name and attributes used at your site.

To allocate the alternate message queue

- 1 Type 4 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS QALTPCB Allocation pop-up shown in [Figure 65](#).

Figure 65 • BTS QALTPCB Allocation Pop-up

```

                                BTS QALTPCB Allocation
Command ==> _____
Enter Data set name, DUMMY, or TEMP:
Name . . . TEMP

DSN DISP   ___      (New, Old, or Shr)
Unit . . . SYSDA
Volume . . . _____

Space:
Units . . CYLINDERS (Cylinder, Track, or Block)
Primary 1
Secondary 1

DCB:
RECFM . . UBS
LRECL . . 512
BLKSIZE 3072
    
```

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP indicates a temporary dataset is to be allocated. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.

Field	Description
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset. The default value is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default value is 3072.

BTS QALTRAN Allocation

Use the BTS QALTRAN Allocation pop-up to allocate the dataset used for alternate PCB output.

This is an example of a typical QALTRAN DD statement from batch execution JCL:

```
//QALTRAN DD UNIT=SYSDA,SPACE=(CYL(1,1),RLSE),
// DCB=(RECFM=U,LRECL=1024,BLKSIZE=536)
```

See your batch execution JCL and PROCs for the QALTRAN dataset name and attributes used at your site.

To allocate the dataset used for alternate PCB output

- 1 Type 5 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS QALTRAN Allocation pop-up shown in [Figure 66](#).

Figure 66 • BTS QALTRAN Allocation Pop-up

```

                                BTS QALTRAN Allocation
Command ==> _____
Enter Data set name, DUMMY, or TEMP:
Name . . . TEMP

DSN DISP   ___      (New, Old, or Shr)
Unit . . . SYSDA
Volume . . _____

Space:
Units . . CYLINDER (Cylinder, Track, or Block)
Primary 1
Secondary 1

DCB:
RECFM . . U
LRECL . . 512
BLKSIZE 536
    
```

- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP indicates a temporary dataset is to be allocated. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.

Field	Description
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset. The default value is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default value is 3072.

BTS BTSOUT Allocation

Use the BTS BTSOUT Allocation pop-up to allocate the dataset used for BTS program output. Output includes input verification and formatted IMS call information.

This is an example of a typical BTSOUT DD statement from batch execution JCL:

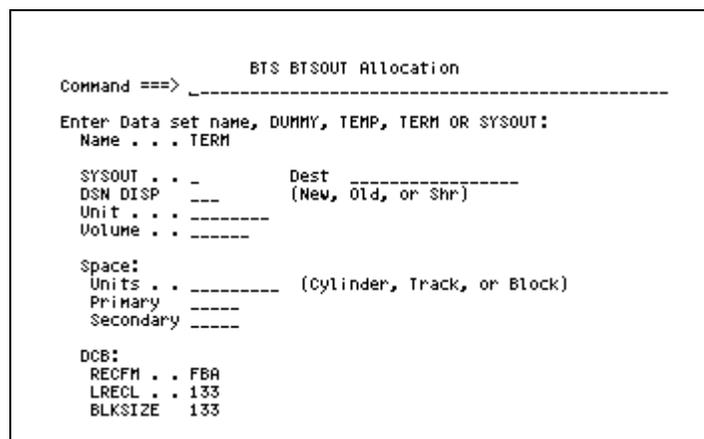
```
//BTSOUT DD SYSOUT=*,
//          DCB=RECFM=FBA,LRECL=133,
```

See your batch execution JCL and PROCs for the BTSOUT dataset name and attributes used at your site.

To allocate the dataset used for BTS program output

- 1 Type 6 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS BTSOUT Allocation pop-up shown in [Figure 67](#).

Figure 67 • BTS BTSOUT Allocation Pop-up



- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to a terminal, and SYSOUT can be specified to allocate the dataset to the output spool.
SYSOUT	Specifies a SYSOUT class value such as A. An entry in this field is valid only if SYSOUT is specified in the Name field.
DEST	Specifies the destination of the SYSOUT output such as R1 or RSCS.ID. An entry in this field is valid only if SYSOUT is entered in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.
Volume	Specifies the volume serial number containing the allocated dataset.

Field	Description
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. If BTSOUT is allocated to SYSOUT, a block size must be specified. A fixed record format (RECFM=F) and a block size of 133 (BLKSIZE=133) outputs an unblocked file to the spool. A BLKSIZE must be specified if the allocation is specified as anything other than TERM.

BTS BTSPUNCH Allocation

Use the BTS BTSPUNCH Allocation pop-up to allocate the dataset containing all input received by BTS.

This is an example of a typical BTSPUNCH DD statement from batch execution JCL:

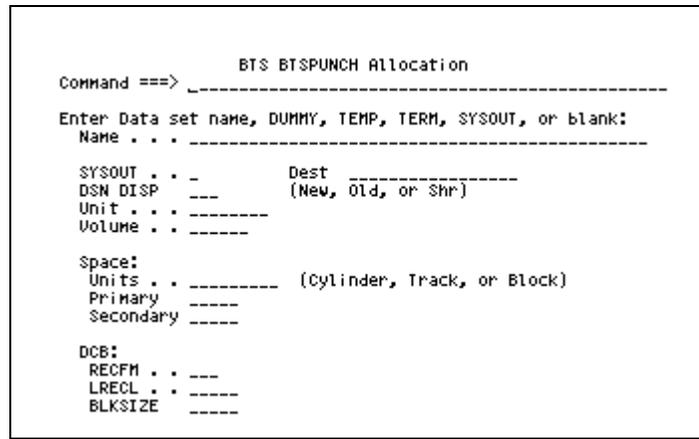
```
//BTSPUNCH DD DSN=USER,BTSPUNCH,DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(10,2),RLSE)
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6800)
```

See your batch execution JCL and PROCs for the BTSPUNCH dataset name and attributes used at your site.

To allocate the dataset containing all input received by BTS

- 1 Type 7 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS BTSPUNCH Allocation pop-up shown in [Figure 68](#).

Figure 68 • BTS BTSPUNCH Allocation Pop-up



- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to the terminal, and SYSOUT can be specified to allocate the dataset to the output spool.
SYSOUT	Specifies a SYSOUT class value such as B. An entry in this field is valid only if SYSOUT is entered in the Name field.
Dest	Specifies the destination of the SYSOUT output such as R1 or RSCS.ID. An entry in this field is valid only if SYSOUT is entered in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset.

BTS BTSDEBUG Allocation

Use the BTS BTSDEBUG Allocation pop-up to allocate the dataset used for SNAP dumps of the Trace table and various control blocks taken during BTS execution.

This is an example of a typical BTSDEBUG DD statement from batch execution JCL:

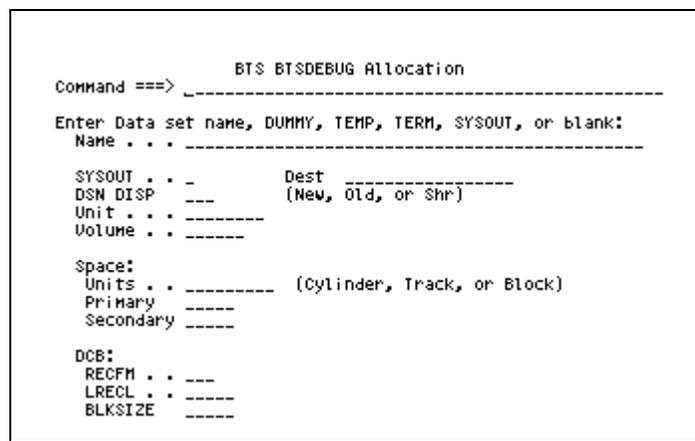
```
//BTSDEBUG DD DUMMY
```

See your batch execution JCL and PROCs for the BTSDEBUG dataset name and attributes used at your site.

To allocate the dataset used for SNAP dumps

- 1 Type 8 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS BTSDEBUG Allocation pop-up shown in [Figure 69](#).

Figure 69 • BTS BTSDEBUG Allocation Pop-up



- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to the terminal, and SYSOUT can be specified to allocate the dataset to the output spool.
SYSOUT	Specifies a SYSOUT class value such as A. An entry in this field is valid only if SYSOUT is entered in the Name field.
Dest	Specifies the destination of the SYSOUT output such as R1 or RSCS.ID. An entry in this field is valid only if SYSOUT is entered in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset.

BTS BTSSNAP Allocation

Use the BTS BTSSNAP Allocation pop-up to allocate the dataset containing SNAP dumps taken by BTS during execution.

This is an example of a typical BTSSNAP DD statement from batch execution JCL:

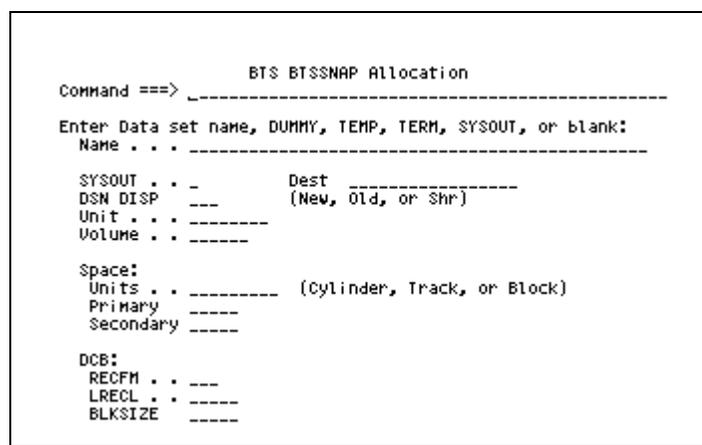
```
//BTSSNAP DD SYSOUT=*
```

See your batch execution JCL and PROCs for the BTSSNAP dataset name and attributes used at your site.

To allocate the dataset containing SNAP dumps

- 1 Type 9 in the primary command input area on the BTS Allocation Selection pop-up and press Enter to display the BTS BTSSNAP Allocation pop-up shown in [Figure 70](#).

Figure 70 • BTS BTSSNAP Allocation Pop-up



- 2 Enter the appropriate information in the fields.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

Fields

Field	Description
Name	Specifies the name of the dataset. TEMP can be specified to allocate a temporary dataset, TERM can be specified to allocate the dataset to the terminal, and SYSOUT can be specified to allocate the dataset to the output spool.
SYSOUT	Specifies a SYSOUT class value such as A. An entry in this field is valid only if SYSOUT is entered in the Name field.
Dest	Specifies the destination of the SYSOUT output such as R1 or RSCS.ID. An entry in this field is valid only if SYSOUT is entered in the Name field.
DSN DISP	Specifies the disposition of the dataset. Disposition can be NEW, OLD, or SHR.
Unit	Specifies a generic name used to allocate the dataset if the dataset name specified in the Name field is not cataloged or if TEMP was specified in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Space	
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default value is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default value is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default value is 1.
DCB	
RECFM	Specifies the record format of the allocated dataset. Record format can be specified as F (fixed) or V (variable).
LRECL	Specifies the record length of the allocated dataset.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset.

Specifying IMS File Allocation Information

Select option I on the BTS Allocation Selection pop-up to display the IMS Allocation Selection pop-up shown in [Figure 71](#). Use this screen to select the IMS datasets, libraries, and parameters to be defined to IMS.

Figure 71 • IMS Allocation Selection Pop-up

```
          IMS Allocation Selection
Command ==> -----
1 - DFSRESLB/DFSESL
2 - PROCLIB/DFSUSAMP
3 - PSB/DBD Libraries
4 - ACB Libraries
5 - IMSMON
6 - IEFORDER

B - IMS Parns (BMP)

P - IMS Parns (DLI or DBB)

A - ALL (Display All Of The Above In Succession)

R - Restore IMS system variables and parms.
```

Note: _____

If your IMS file allocations were not entered previously or you need to change them, see ["Specifying IMS File Allocation Information" on page 82](#). See the *IBM IMS Batch Terminal Simulator: Program Reference/Operations Manual* for more information about BTS about BTS datasets and parameters.

DB2 Programs in TSO Foreground

Specifying DB2 Setup Information

To specify the TSO test session parameters for DB2 programs and initiate a test session

- 1 Select DB2 on the Environment Selection pop-up to display the DB2 Session Setup screen shown in [Figure 72](#).

Figure 72 • DB2 Session Setup Screen

```

                                DB2 Session Setup
Command ==> _____

R - Begin DB2 test session (RUN)    C - Convert batch JCL to CLIST

Execution:                          Options:
Load module      _____          Break on entry (Y/N) YES
DB2 Plan name    _____          Break CSECT/pgm id  _____
DB2 Subsystem    _____

Execution parameters: (quotes are optional)
_____
_____

File allocation CLIST:
Data set name    _____
Member . . . . . _____          Deallocate after test NO

```

- 2 Specify these fields:
 - a Enter the load module in the Load module field. This is a required field.
 - b Enter the CLIST dataset name and member in the Data set name and Member field, respectively. This is an optional field.
 - c Enter any appropriate options. This is an optional field.

Note: _____

If a CLIST has already been created for the program to be tested, see ["Initiating a Test Session" on page 61](#). If a CLIST does not exist for the program to be tested, see ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#).

Options

Option	Description
R	Initiates the DB2 test session. This is the equivalent of typing RUN.
C	Displays the Convert Batch JCL pop-up used to convert batch JCL to an allocation CLIST. The converted allocation CLIST can be used to establish the DB2 test session.

Fields

Field	Description
Execution	
Load module	Specifies the initial load module to be tested. This should be the name from the TSO RUN PROGRAM command.
DB2 plan name	Specifies the DB2 Plan that was generated for the program to be tested when the BIND was performed.
DB2 subsystem	Specifies the name assigned to DB2 when it was installed in the MVS environment.
Options	
Break on entry (Y/N)	Specifying YES causes the test session to stop at the start of the test session. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
Break CSECT/pgm id	Specifying a program name causes the test session to stop on entry to the specified CSECT in a statically linked module.
Execution parameters	Specifies any required application parameters.
File allocation CLIST	
Data set name	Specifies the dataset containing the allocation CLIST generated by the Convert Batch JCL facility.

Field	Description
Member	Specifies the name of the allocation CLIST generated by the Convert Batch JCL facility.
Deallocate after test	Specifying YES causes the CLIST processor to be automatically invoked to deallocate the test files at the end of the test session, or after a CANCEL command is entered. Issuing another RUN command causes the datasets to be reallocated. By default, the CLIST is automatically invoked to deallocate the test files before exiting SmartTest. The default value is NO.

DB2 Stored Procedure Testing Option

SmartTest has an optional feature that enables a programmer/analyst to interactively test and debug a DB2 Stored Procedure.

A Stored Procedure is a user-written program that resides on a DB2 server. An SQL CALL interface allows an SQL requester to invoke the stored procedure at a DB2 server. The Stored Procedure is a Language Environment compliant program, written in COBOL, C/370, Assembler, or PL/I. The procedure name is defined to DB2 in a table, SYSIBM.SYSPROCEDURES. When the client executes an SQL CALL, DB2 searches this table for the procedure name contained in the SQL CALL.

Normally, a stored procedure executes in a special DB2 address space. However, SmartTest tests the stored procedure in TSO foreground.

Requirements

Before setting up a DB2 Stored Procedure test, make sure these preliminary operations have been performed:

- The SmartTest DB2 option is installed at your site.
- The Stored Procedure name is defined to DB2 in the SYSIBM.SYSPROCEDURES table.
- The Stored Procedure is bound as a plan or a package. (If the Stored Procedure is bound as a package, the package must be bound within a plan.)
- The Stored Procedure program is Analyzed and stored in an AKR.

Note: _____

If you are unsure of the completion of any of the above tasks, check with your DB2 Database Administrator.

Setting Up the DB2 Stored Procedure Test

To specify the test session parameters for a DB2 Stored Procedure program and initiate a test session

- 1 Select DB2 Procedure on the Environment Selection pop-up to display the DB2 Stored Procedures Setup screen shown in [Figure 73](#).

Figure 73 • DB2 Stored Procedures Setup Screen

```

                                DB2 Stored Procedures Setup
Command ===> -----

R - Begin DB2 Stored Procedure session (RUN)
C - Convert batch JCL to CLIST
D - Display Parameters

Execution:                                Options:
Load Module . . . -----                Break on entry (Y/N)   YES
DB2 Plan name . . . -----                Break CSECT/pgm id . . . -----
DB2 Subsystem . . . -----                Use RRSF (Y/N) . . . NO
DB2 Schema name . VIA123
DB2 Procedure name -----

File allocation CLIST:
Data set name -----
Member . . . . -----                Deallocate after test NO
    
```

- 2 Under Execution, make these required entries:
 - a Enter the initial Load module to be tested; the entry must be the name of the DB2 stored procedure to be tested.
 - b Enter the DB2 Plan name that was generated for the Stored Procedure program to be tested.
 - c Enter the DB2 Subsystem where the Stored Procedure program is to run.
 - d Enter the DB2 Schema name, which is the name of the DB2 entry in the SYSIBM.SYSROUTINES table that contains the information about the store procedure program. This field defaults to your TSO user ID.
 - e Enter the DB2 Procedure name – that is, the DB2 entry in the SYSIBM.SYSPROCEDURES table that contains the information about the Stored Procedure program.

- 3** Under Options:
- a** In the Break on entry (Y/N) field:
Type YES to stop the test session at the start of the test execution (such as, to change test data values). YES is the default. Type NO to run the test execution to completion (end or abend).

Additional break options are available. See ["Tailoring a Test Session by Program" on page 179](#).
 - b** Type the program name in the Break on CSECT/pgm id field to stop the test session on entry to a specified CSECT in a statically linked module.
 - c** Type Y in the Use RRSF field to perform resource recovery using the Recoverable Resource Manager.

The Use RRSF field is valid only if you have DB2 Version 5.1 or later.
- 4** If non-DB2 resources are used by the stored procedure, convert that JCL to a CLIST and make the CLIST available to the test. To allocate a CLIST:
- a** Make these entries under File allocation CLIST:
 - Enter the dataset name for the CLIST.
 - Enter the member name of the CLIST.
 - To deallocate the test files at the end of this test session or after a CANCEL command, type YES in Deallocate after test. To deallocate the test files at the end of the SmartTest session, type NO. (NO is the default.)
 - b** To display the Convert Batch JCL pop-up, type C. See ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#) for more information.
- 5** Enter an action:
- a** Type D to display parameters:
 - If the stored procedure is expecting input parameters.
 - To review output.
 - b** Type R to begin the test session. Proceed to ["Initiating the DB2 Stored Procedures Test" on page 129](#).

Reviewing DB2 Stored Procedure Parameters

To review input and output parameters

- 1 On the DB2 Stored Procedures Parameters screen, shown in [Figure 74](#), review the parameters displayed. Each parameter specified in the PARMLIST column on the SYSIBM.SYSPROCEDURES table displays.

Figure 74 • DB2 Stored Procedures Parameters Screen

DB2 Stored Procedures Parameters						
Command ==>					Scroll ==>	CSR
Name	Type	Address	Structure	Value	New Value	
CUSTNO	IN	000D5148	CHARACTER(8)	A1234	C1F1F2F3 F4404040	
CUSTNM	OUT	000D5198	CHARACTER(20)	JONES	D1D6D5C2 C2C9D3	
CUSTBAL	OUT	000E0000	SMALLINT	5000	1388	
ERROR	OUT	000E1000	CHARACTER(255)	00000000 00000000	
***** BOTTOM OF DATA *****						

This table describes the fields on the DB2 Stored Procedures Parameters screen:

Column	Description
Name	Specifies the optional parameter name that can be used by DB2 for diagnostic messages.
Type	Defines the parameter as IN (input), OUT (output), or INOUT (I/O).
Address	Specifies the main storage location acquired by SmartTest for the data.
Structure	Provides information on the parameter's data attributes: character, integer, floating point, decimal, or variable character data.
Value	Specifies the character format (maximum 8 characters) of the data at that storage location.
Hex Value	Specifies the hexadecimal representation of the data at that storage location.

- 2 To select a parameter, type S in the line command area to the left of the parameter Name and press Enter.
 - a If the data is defined as numeric, the DB2 Stored Procedures Numeric Display screen displays.
 - b If the data is not defined as numeric, the Memory Display screen displays.
 - c Go to ["Changing Test Data Values" on page 127](#).

Changing Test Data Values

If the parameter you selected on the DB2 Stored Procedures Parameters screen is defined as numeric, the DB2 Stored Procedures Numeric Display screen displays. If the parameter you selected is non-numeric, the Memory Display screen displays.

To review/change numeric values

- 1 Review the selected parameter on the DB2 Stored Procedures Numeric Display screen, shown in [Figure 75](#).

Figure 75 • DB2 Stored Procedures Numeric Display Screen

DB2 Stored Procedures Numeric Display			
Command ==> _____			
Numeric Data	Address	Type	Length
-----	-----	-----	-----
+0	000E0000	SMALLINIT	

This table describes the fields on the DB2 Stored Procedures Numeric Display screen:

Column	Description
Numeric Data	Displays the current numeric value for the parameter. You may change the value in this field.
Address	Specifies the hexadecimal address of this parameter.
Type	Provides the parameter's data definition to DB2 (i.e., integer, decimal, float, etc.).
Length	Specifies the data length defined to DB2 (used only if the Type is decimal).

- 2 If desired, change the numeric value for the parameter in the Numeric Data field.
- 3 Press Enter to save your change.

- 4 Press PF3 twice to return to the DB2 Stored Procedures Setup screen.
- 5 Proceed to ["Initiating the DB2 Stored Procedures Test" on page 129.](#)

To review/change non-numeric values

- 1 Review the selected parameter on the Memory Display screen, shown in [Figure 76.](#)

Figure 76 • Memory Display Screen

Memory Display										
Command ==>								Scroll ==>		CSR
Area	000D5148					Offset	000000	Length	000050	
MEMBER	DATA	NAME=								
000D5148	C1F1FCFS	F4404040	00000000	00000000	00000000	00000000	*	A1234 *	
000D5158	00000000	00000000	00000000	00000000	00000000	00000000	* *		
000D5168	00000000	00000000	00000000	00000000	00000000	00000000	* *		
000D5178	00000000	00000000	00000000	00000000	00000000	00000000	* *		
000D5188	00000000	00000000	00000000	00000000	00000000	00000000	* *		

This table describes the fields on the Memory Display screen:

Column	Description
Area	Displays the hexadecimal address of this parameter.
Offset	Displays the relative address of this parameter.
Length	Displays the data length as defined to DB2.

To make a change to these fields, type over the values displayed.

- 2 Press Enter to save your change.
- 3 Press PF3 twice to return to the DB2 Stored Procedures Setup screen.
- 4 Proceed to [Initiating the DB2 Stored Procedures Test.](#)

Initiating the DB2 Stored Procedures Test

Your DB2 Stored Procedures test is set up and you are ready to initiate the test session. The Break on entry option stops the session for interactive data entry. All SmartTest functions (e.g., breakpoints, session tailoring, pseudo code, searching, etc.) can be used during your DB2 test session.

To initiate the DB2 Stored Procedures test

- 1 On the DB2 Stored Procedures Setup screen, type R to begin DB2 Stored Procedure session (RUN).
- 2 Watch the test session execute the Stored Procedure in Program View.
- 3 If you did not set breaks during set up, the program executes until:
 - An abend error condition
 - An address stop
 - Completion of the stored procedure
- 4 If you set Break on entry, the test session stops at the beginning of PROCEDURE DIVISION (and at other breakpoints, as specified).

Note: _____

You must execute the PROCEDURE DIVISION or entry statement prior to using the ZD (ZOOM DATA) command.

- 5 To use additional SmartTest functions, see these sections:
 - a ["Testing Techniques" on page 143](#)
 - b ["Program Analysis Features" on page 187](#)
 - c ["Additional Testing Features" on page 217](#)
- 6 At the breakpoint at the end of the test, press PF4 (RUN) to return to the DB2 Stored Procedures Setup screen (see ["Setting Up the DB2 Stored Procedure Test" on page 124](#)).
- 7 On the DB2 Stored Procedures Setup screen, type D to review output parameters on the DB2 Stored Procedures Parameters screen (see ["Reviewing DB2 Stored Procedure Parameters" on page 126](#)).

Testing Programs in a Batch Region

Batch Connect Facility

The Batch Connect facility in SmartTest allows your original JCL to be executed in a batch region. Use of this facility provides significant reductions in setup requirements and TSO resource consumption. The tradeoff is the job remains active in the batch region for the entire test session.

Consider using the SmartTest Batch Connect when these conditions occur:

- The program(s) to be tested requires too much region for execution in TSO foreground.
- The test requires media not accessible through TSO foreground (tape datasets).
- The test must consist of multiple interdependent steps.
- The test requires the use of special output forms.
- Running a DB2 test under Batch where there is a PROC in the JCL. The SYSTSIN DD statement must be in the PROC and may not be overridden after the execute of the Procedure. The DD for SYSTSIN in the Procedure may use the DDNAME parameter to allow the DD statement to be placed after the execute.
- There is a limit of one program to test for each SmartTest batch execution.

The Batch Session Setup screens are used to establish a test session in a batch region. Once the test session is established, an online session can be connected to it. Batch program testing can be performed interactively. Note that symbolic support may be unavailable for programs loaded from read protected libraries.

Specifying Batch Connect Setup Information

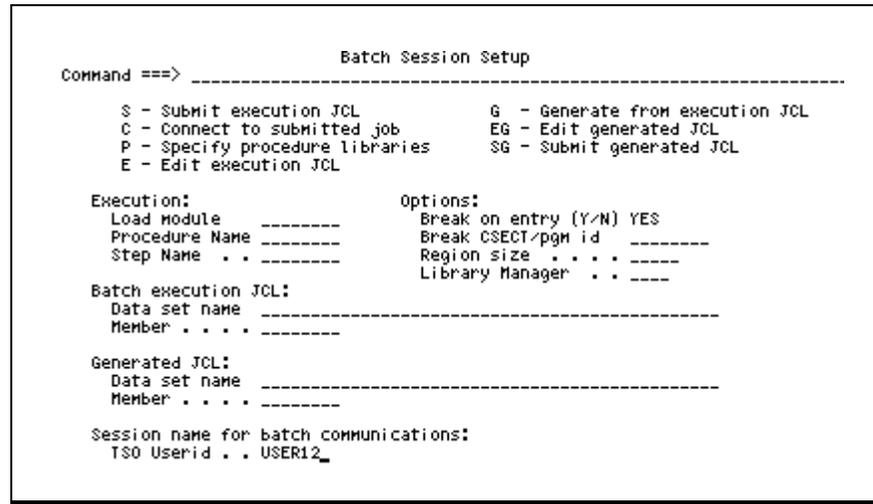
Because the MVS, IMS, BTS, and DB2 Batch Session Setup screens are identical except for the screen title, this discussion pertains to setup in all Batch environments. MVS Batch is used as the example.

Use the Batch Session Setup screen to specify the information necessary to establish a batch test session in an MVS batch region. Once the batch session is established, an online interactive TSO test session can be connected to it.

To establish a batch test session

- 1 Select MVS BATCH on the Environment Selection pop-up to display the Batch Session Setup screen shown in [Figure 77](#).

Figure 77 • Batch Session Setup Screen



- 2 Specify these required fields:
 - a Enter the load module in the Load module field.
 - b Enter the batch execution JCL dataset name and member in the Batch Execution JCL, Data set name and Member fields, respectively.
 - c Enter any appropriate options.

Options

Option	Description
S	Submits the specified execution JCL. The JCL dataset name and member name must be entered in the Batch Execution JCL fields.
C	Displays the Connect to Job pop-up used to connect the online test session with a batch test session.
P	Displays the Procedure Libraries pop-up used to specify procedure libraries. If you do not enter any procedure libraries on this screen, the libraries listed in the PROCLIBS entry in VIA\$PRMS are used to search for procedures.
E	Invokes the TSO/ISPF editor with the batch execution JCL prior to SmartTest changes.

Option	Description
G	Generates the batch JCL and saves the output in the specified GENERATED JCL member without executing it. The SG (Submit Generated JCL) option can be used to pass the generated output directly to the operating system for execution. Job submission does not occur.
EG	Generates batch connect JCL and invoke the TSO/ISPF editor with the generated JCL. Job submission does not occur.
SG	Submits the specified generated JCL. The generated JCL dataset name and member name must be entered in the Generated JCL fields.

Fields

Field	Description
Execution	
Load module	Specifies the initial load module to be tested.
Procedure Name	Specifies the PROC name of the procedure containing the execute statement for the load module to be tested.
Step Name	Specifies the step name in the batch job that executes the load module to be tested.
Options	
Break on entry (Y/N)	Specifying YES causes the test session to break on entry at the start of the load module. Additional break on entry options are available on the Session Tailoring screen for each program to be tested. The default value is YES.
BREAK CSECT/pgm id	Specifying a program name causes the test session to stop on entry at the start of the specified CSECT in a statically linked module.
Region size	Modifies the region size in the execution JCL. The region size of the step containing the program to be tested is changed to the value specified in this field. The default is 4 megabytes.
Library Manager	Specifies the library manager. If the JCL to be converted is stored in a source library that is installed with a subsystem option, enter the name of that subsystem. Also, the dataset and member name used by the source library management system must be specified in the Batch execution JCL field.

Field	Description
Batch execution JCL	
Data set name	Specifies the partitioned dataset containing the member to be submitted or edited, or from which JCL is to be generated. If the Subsystem option is to be used, specify the subsystem library dataset name.
Member	Specifies the member to be submitted or edited, or from which JCL is to be generated.
Generated JCL	
Data set name	The partitioned dataset that will contain the generated JCL.
Member	Specifies the member that will contain the generated JCL.
Session name for batch communications	
TSO Userid	Specifies the TSO user ID that is placed in the SmartTest disk file used to transfer data to and from the TSO address space. This is the ID with which the batch session communicates and typically is the user ID of the person submitting the job. Another user ID can be specified if desired. If another user ID is specified, note that the AKR and the test load libraries are those specified by the user submitting the batch job(s). The AKR and load libraries for another user ID could be different from those being used by the person submitting the job.

Note: _____

When you select option S or G, SmartTest modifies the specified batch execution JCL for the first occurrence in the JCL of the program to be tested.

To test a later occurrence

- 1 Generate the JCL.
- 2 Edit the generated JCL and move the changes to the step to be tested.
- 3 Submit the edited JCL.

These JCL changes are made by SmartTest:

- A STEPLIB DD statement for the step to be tested is added or updated to concatenate the SmartTest load library to the user load library.
- The REGION size is changed to the value specified on this screen.
- A DD VIAQUEUE statement is added for the SmartTest queue dataset used during communication between the TSO and Batch test sessions.
- The EXEC PGM is altered for the step to be tested to allow SmartTest to gain control at the appropriate step.
- PROCs are expanded using the procedure libraries specified on the Procedure Libraries pop-up.

When submitting execution JCL or submitting generated execution JCL, the TSO SUBMIT command is issued from a CLIST so the system standard TSO SUBMIT exit receives control and can process the JCL. The submitted JCL must have a JOB card with a job name that will not be changed by the exit.

If you use the EG (edit generated JCL) option to generate the JCL to the specified dataset, the JCL can then be edited and the job name can be changed in the generated JCL.

You should verify that the work file allocation information entered on the Options - Product Allocations pop-up is valid for the batch test session. Inappropriate work file allocations can cause unpredictable results during the batch test session.

Note: _____

See the online help for detailed information on the Options - Product Allocations pop-up.

Submitting and Connecting to a Batch Job

A batch job can be connected with the online portion of a test session by submitting it for execution, then connecting to it.

Submit a Job

To submit the batch job for execution

- 1 Type S or SG on the primary command input area on the Batch Session Setup screen and press Enter.

Note:

Record the TSO message displaying the job name and assigned job number, for future reference.

Figure 78 • Connect To Job Screen

```
+ASG2076I Batch JOB <jobname> is waiting for connection by ASG-SmartTest
***
```

- 2 Press Enter to return to the Batch Session Setup screen.

Connect to Job

To connect the online TSO test session with a batch test session

- 1 Type C in the primary command input area on the Batch Session Setup screen and press Enter to display the COnnect to Job screen shown in [Figure 79](#).

Figure 79 • Connect to Job Screen

```

                                Connect to Job
Command ==>> _____ Scroll ==>> CSR

TSO Userid TSOUSERD      (Required for TSO/ISPF monitor)

Enter 'S' to select a job to be tested.
Enter 'P' to purge a job from the ASG-SmartTest queue file.

Select      Jobnum      Jobname      Status
-----
-           J8244      TSOUSERB    BEGINNING EXECUTION
-           J8268      TSOUSERC    WAITING FOR CONNECTION
-           J8286      TSOUSERD    CONNECTED, TEST ACTIVE
-           J8301      TSOUSERA    JOB SUBMITTED TO BATCH

```

- 2 Select a job by typing S in a Select field with the status WAITING FOR CONNECTION and pressing Enter.

Options

Option	Description
S	Connects to an available batch session. The selected job becomes active in SmartTest. (See "Batch Test Initiation" on page 136.)
P	Purges (deletes) a batch session or submitted job.
Status	
CONNECTED, TEST ACTIVE	Indicates that the job is currently connected to the online test session by the specified TSO user ID.
BEGINNING EXECUTION	Indicates that the job has a batch initiator, but is not ready for connection.
JOB SUBMITTED TO BATCH	Indicates that the job is waiting for a batch initiator.
TEMPORARILY SUSPENDED	Indicates that the job was previously connected to the online test session by the specified TSO user ID. A job can be suspended by pressing the PA1 or ATTN key.
WAITING FOR CONNECTION	Indicates that the job has a batch initiator and is ready to be connected to the online test session.

Batch Test Initiation

After all information for testing a program in the batch environment has been specified on the appropriate Batch Session Setup screens, test initiation can be invoked by connecting to the batch job.

When test initiation is complete, the program displays in Program View; the test session is active and waiting for a command. Program View displays program source or disassembled object code, as shown in [Figure 80 on page 137.](#)

To display Program View, you must connect to the batch job. For details, see ["Connect to Job" on page 135](#).

Figure 80 • Batch Connect Program View Screen

```

File View Test Search Logic List Options Help
-----
                                Program View                VIAMERGE.VIAMERGE -A (A)
                                SCROLL ==> CSR
====>
000163 001-MAIN-LOGIC.
(B)>>>   PERFORM 1000-INITIALIZE THRU 1000-INITIALIZE-X.
000165   PERFORM 2000-PROCESSING-LOOP THRU 2000-PROCESSING-LOOP-X
000166           UNTIL FINISHED-READING-ALL-FILES.
000167   PERFORM 9000-TERMINATION THRU 9000-TERMINATION-X.
000168   GOBACK.                                           PGM EXIT
000169 1000-INITIALIZE.
000170   OPEN INPUT  INFILE1,
000171           INFILE2,
000172           INFILE3.
000173   OPEN OUTPUT OUTFILE,
000174           OUTRPT.
000175   MOVE ZEROS TO  COMPARISON-KEY-1, COMPARISON-KEY-2,
000176           INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,
000177           MASTER-EOF-SWITCH,
000178           HASH-TEST-A, HASH-TEST-B, HASH-TEST-C,
(C)-----+
| STATUS: BREAK AT START OF TEST SESSION  PROGRAM: VIAMERGE  DATE: DDMMYYYY |
| STMT: 000164  OFF: 001450                MODULE: VIAMERGE  TIME: HH:MM:SS |
| SOURCE: PERFORM 1000-INITIALIZE THRU 1000-INITIALIZE-X.                |
+-----+

```

Screen Description

(A) VIAMERGE.VIAMERGE -A. The short message reflects the module and program (*module.program*) being viewed and indicates the program is in an active (-A) test session.

(B) PERFORM 1000-INITIALIZE. The next statement to be executed is highlighted and chevrons appear in the number column.

(C) STATUS: BREAK. The Status Box displays, by default, when a test session is initiated. This is the information shown in the Status Box:

Screen Field	Description
STATUS	Indicates the current status of the test.
STMT	Specifies the statement number of the next statement to be executed.
OFF	Specifies the offset into the CSECT/load module where the next statement to be executed is located.
SOURCE	Specifies the source for the next statement to be executed.
PROGRAM	Specifies the name of the program/CSECT currently active in the test.
MODULE	Specifies the name of the load module currently active in the test.

Screen Field	Description
DATE	Specifies the current date.
TIME	Specifies the time the last program statement was executed.

Testing DL/I in the Batch Environment

To use the IMS Batch environment

- 1 Select the IMS BATCH environment on the Environment Setup Menu to display the IMS Batch Session Setup screen shown in [Figure 81](#).

Figure 81 • IMS Batch Session Setup Screen

```

                                IMS Batch Session Setup
Command ==> -----
S - Submit execution JCL          G - Generate from execution JCL
C - Connect to submitted job      EG - Edit generated JCL
P - Specify procedure libraries   SG - Submit generated JCL
E - Edit execution JCL

Execution:                          Options:
Load module -----                Break on entry (Y/N) YES
Procedure Name -----             Break CSECT/pgm id -----
Step Name . . -----              Region size . . . . -----
                                   Library Manager . . ----

Batch execution JCL:
Data set name -----
Member . . . . -----

Generated JCL:
Data set name -----
Member . . . . -----

Session name for batch communications:
TSO Userid . . USER12_
    
```

- 2 Submit the JCL from the Batch Submit screen.
- 3 Connect to the batch session using the batch Connect to Job option.

Testing BTS in the Batch Environment

To use the BTS Batch environment

- 1 Select the BTS BATCH environment on the Environment Setup Menu to display the BTS Batch Session Setup screen shown in [Figure 82](#).

Figure 82 • BTS Batch Session Setup Screen

```

                                BTS Batch Session Setup
Command ==> -----

S - Submit execution JCL          G - Generate from execution JCL
C - Connect to submitted job      EG - Edit generated JCL
P - Specify procedure libraries   SG - Submit generated JCL
E - Edit execution JCL

Execution:                          Options:
Load module -----                Break on entry (Y/N) YES
Procedure Name -----             Break CSECT/pgm id -----
Step Name . . . -----            Region size . . . -----
                                   Library Manager . . . -----

Batch execution JCL:
Data set name -----
Member . . . . -----

Generated JCL:
Data set name -----
Member . . . . -----

Session name for batch communications:
TSO Userid . . USER12_

```

- 2 Submit the JCL from the Batch Submit screen.
- 3 Connect to the batch session using the Connect to Job option.

Testing DB2 in the Batch Environment

To test DB2 using the Batch Submit screen

- 1 Select DB2 BATCH as the environment on the Environment Setup Menu to display the DB2 Batch Session Setup screen shown in [Figure 83](#).

Figure 83 • DB2 Batch Session Setup Screen

```
DB2 Batch Session Setup
Command ==> -----
$ - Submit execution JCL      G - Generate from execution JCL
C - Connect to submitted job  EG - Edit generated JCL
P - Specify procedure libraries SG - Submit generated JCL
E - Edit execution JCL

Execution:                      Options:
Load Module -----           Break on entry (Y/N) YES
Procedure Name -----        Break CSECT/pgm id -----
Step Name . . . . .          Region size . . . . .
                               Library Manager . . . . .

Batch execution JCL:
Data set name -----
Member . . . . .

Generated JCL:
Data set name -----
Member . . . . .

Session name for batch communications:
TSO Userid . . USER12_
```

- 2 Submit the JCL from the Batch Submit screen.
- 3 Connect to the batch session using the Connect to Job option.

Note: _____

There is a limit of one program to test for each SmartTest batch execution.

When running a DB2 test under Batch, the JCL that is submitted is automatically changed. If there is a PROC in the JCL, the SYSTSIN DD statement must be in the PROC and may not be overridden after the execution of the Procedure. The DD for SYSTSIN in the Procedure may use the DDNAME parameter to allow the DD statement to follow the execute PROC statement in the JCL.

Note: _____

SmartTest does not recognize the LIB() field of the SYSTSIN dataset. Load libraries specified in the LIB() field must be specified in the STEPLIB DD concatenation.

Testing DFHDRP in the Batch Environment

To test DFHDRP using the Batch Submit screen

- 1 Select MVS Batch as the environment on the Environment Setup Menu.
- 2 Edit the JCL from the Batch Session Setup screen and change the program to be tested to VIAPDLI. This is done by changing the PGM=*pgmname* parameter in the PARM statement to PGM=VIAPDLI.
- 3 Submit the JCL from the Batch Submit screen.
- 4 Connect to the batch session using the Connect to Job option.

DFHDRP Batch JCL

[Figure 84](#) shows the DFHDRP batch JCL before the modifications are made to test DB2 using the Submit Job screen.

Figure 84 • DFHDRP Batch JCL Before Modifications

```
//ASG JOB (ASG), 'ASG-SMARTTEST DFHDRP'
/*ROUTE PRINT DEST
/*
//DFHDRPB PROC SYSOUT=A
/*
//STEP01 EXEC PGM=DFHDRP, REGION=1M,
// PARM= 'SSA=254, PGM=TPGM22, PSB=TPGM22, CICS=CICS1, CMPAT=Y, LANG=C'
//STEPLIB DD DSN=USER.PGMLIB, DISP=SHR
// DD DSN=IMS.RESLIB, DISP=SHR
// DD DSN=CICS.RESLIB, DISP=SHR
//IMS DD DSN=USER.PSBLIB, DISP=SHR
// DD DSN=USER.DBDLIB, DISP=SHR
//DFHLIB DD DSN=CICS.LOADLIB, DISP=SHR
//SYSUDUMP DD SYSOUT=&SYSOUT
// PEND
//DRPTEST EXEC DFHDRPB
//SYSOUT DD SYSOUT=&SYSOUT
//REPORT DD SYSOUT=&SYSOUT
//D121PART DD DSN=APRT.DB10V09, DISP=SHR
```

[Figure 85](#) shows the DFHDRP batch JCL after the modifications are made to test DB2 using the Submit Job screen.

Figure 85 • DFHDRP Batch JCL After Modifications

```
//ASG JOB (ASG), 'ASG-SMARTTEST DFHDRP'
/*ROUTE PRINT DEST
/*
//DFHDRPB PROC SYSOUT=A
/*
//STEP01 EXEC PGM=DFHDRP, REGION=2M,
//
PARM= 'SSA=254, PGM=VIAPDLI, PSB=TPGM22, CICS=CICS1, CMPAT=Y, LANG=C'
//STEPLIB DD DSN=ASG.VIACENxx.LOADLIB, DISP=SHR,
//        DCB=BLKSIZE=32760
//        DD DSN=USER.PGMLIB, DISP=SHR
//        DD DSN=IMS.RESLIB, DISP=SHR
//        DD DSN=CICS.RESLIB, DISP=SHR
//IMS    DD DSN=USER.PSBLIB, DISP=SHR
//        DD DSN=USER.DBDLIB, DISP=SHR
//DFHLIB DD DSN=CICS.LOADLIB, DISP=SHR
//SYSUDUMP DD SYSOUT=&SYSOUT
//        PEND
//DRPTEST EXEC DFHDRPB
//SYSOUT DD SYSOUT=&SYSOUT
//REPORT DD SYSOUT=&SYSOUT
//D121PART DD DSN=PART.DB10V09, DISP=SHR
//VIAQUEUE DD DSN=ASG.VIACENxx.QUEUE, DISP=SHR
```

4

Testing Techniques

This section presents common SmartTest testing and debugging functions available with SmartTest, and contains these sections.

Topic	Page
Learning the SmartTest Commands	144
Controlling Program Execution Using SmartTest	144
Controlling Program Execution	148
Viewing and Changing Test Session Data	162
Execution History (Backtrack)	167
Using Pseudo Code	173
Using Multiple Programs	178
Setting Test Session Options	180
Displaying Program and Test Information	181
Printing Displayed Information (LPRINT Command)	182
Linking to Alliance	183

You may re-create the examples presented by using the VIAMERGE program for the TSO environment.

Note: _____

This chapter assumes that you have completed the appropriate setup and analyze procedures.

Learning the SmartTest Commands

Most procedures in this section emphasize the CUA implementation of SmartTest functions. As you gain experience, you may find the SmartTest command syntax more convenient for frequently used functions.

To assist you in learning commands, SmartTest includes a learn mode showing primary commands equivalent to your CUA selections.

To turn learn mode on, follow this step:

- ▶ Type `SET LEARN ON` in the command input area and press Enter. The short message area displays the message `LEARN ON`.

To turn learn mode off, follow this step:

- ▶ Type `SET LEARN OFF` in the command input area and press Enter. The short message area displays the message `LEARN OFF`.

Controlling Program Execution Using SmartTest

SmartTest allows you to control execution by walking-through the program statement by statement, or executing the program continuously while checking for problems. You may end a test session at any time without waiting for end-of-job. You may perform maintenance and debugging tasks interactively during the test, rather than waiting until the test is over and the output is delivered.

Controlling execution consists of these procedures:

- Continuous program execution until an interrupt condition occurs. (See ["Executing the Program Continuously Using RUN" on page 148.](#))
- Executing a specified number of statements. (See ["Executing a Specified Number of Statements Using STEP" on page 149.](#))
- Specifying the statement to execute next rather than following the normal execution sequence. (See ["Changing Program Execution Sequence Using GO" on page 149.](#))
- Interrupting execution. (See ["Interrupting Test Execution Using Keystrokes" on page 150.](#))
- Inserting breakpoints to interrupt execution. (See ["Inserting Breakpoints to Interrupt Execution" on page 153.](#))
- Locating the next executable statement. (See ["Locating the Next Executable Statement" on page 160.](#))
- Ending the test session. (See ["Exiting a SmartTest Test Session" on page 162.](#))

Testing with SmartTest

At the start of a test session, SmartTest attaches an execution monitor to run the application program as a subtask to SmartTest's user interface code. Execution of the application program is controlled by commands entered into SmartTest, which in turn controls the application code.

SmartTest has two distinct methods for controlling execution of the application program, MONITOR and NOMONITOR. Each method has a strength and a weakness and is designed to help get the most out of debugging sessions. SmartTest allows mixed methods during a single test session to offer benefits of each technique for individual test situations.

Both MONITOR and NOMONITOR are specified by an operand of the RUN function. The SET MONITOR command determines the default. Guidelines for choosing either MONITOR or NOMONITOR are included in this manual (see ["Guidelines for Using the MONITOR/NOMONITOR Methods" on page 146](#)).

MONITOR AND NOMONITOR can also be controlled on the program level using the Session Tailoring option.

Testing Using the MONITOR Method

The MONITOR method is designed to help during the detailed debugging stage, after a problem has been isolated to a specific program or to a specific subroutine within a program.

Using the MONITOR method, SmartTest executes the application program one instruction at a time. Executing instructions individually gives SmartTest the ability to create an execution history (LIST TRACKING), to count each statement as it executes (LIST COUNTS), and to provide storage protection for user-specified locations (LIST ADSTOP) by monitoring instructions that modify memory.

Additional advantages to the MONITOR method include minimal test setup requirements and detection of INVALID BRANCH situations.

Since monitoring requires execution of several instructions by SmartTest for each application instruction, the CPU overhead of monitoring a large application (e.g., processing millions of records) may become prohibitive. If performance becomes an issue, use the NOMONITOR method (RUN NOMON).

Testing Using the NOMONITOR Method

The NOMONITOR method is designed to execute the application at native speed with negligible CPU overhead added by SmartTest. In the NOMONITOR method, SmartTest gives control to the application program and waits for an application abend, a breakpoint, or completion of the test.

In the NOMONITOR method, certain facilities are not available (LIST TRACK, LIST COUNTS, etc.). However, abends are intercepted, and COBOL-compatible pseudo code can be inserted to control the flow of the test program.

In the NOMONITOR method, SmartTest places intentional abend instructions (breakpoints) at various points in the object code and gains control when the code abends (S0C1) during execution. Because this technique is used, each program tested using the NOMONITOR method must be link edited with the REUS parameter. This parameter is automatically supplied by the SmartTest Compile and Analyze process, but can be overridden with an option on the List-Analyze Submit pop-up.

Guidelines for Using the MONITOR/NOMONITOR Methods

In general, you should use the MONITOR method for most testing and debugging. The CPU overhead in relation to the value of the debugging information is acceptable.

The Execution Tracking screen shows all of the statements that have been executed during the test in the reverse sequence. This screen can be used to follow the execution path backward to see the execution path to the current statement.

To reduce the overhead of a test session, place a breakpoint at a paragraph where a suspected problem exists and then execute using the NOMONITOR method. The test executes at native speed until it reaches the breakpoint set. Then, use the STEP primary command or RUN MONITOR primary command to find the problem. After finding the problem, either CANCEL the test or RUN NOMONITOR to completion. This technique uses the strengths of both methods, while reducing the overall CPU utilization.

Note: _____

The NOMONITOR method makes problem determination difficult when the application program issues its own SPIE/ESTAE. This intercepts the intentional abends before SmartTest can examine them, yielding unpredictable results.

To test such programs, disable the SPIE and restart the test using the NOMONITOR method. If there is no way to disable the SPIE, it will be necessary to run the application using the MONITOR method.

Testing LINKed, ATTACHed, and CALLEd Load Modules

SmartTest includes the Load Module Intercept List screen for specifying load module intercepts to allow support for load modules that are LINKed, ATTACHed, or dynamically loaded and CALLEd. The Load Module Intercept List screen allows up to 24 load modules intercept to be specified. Entries must be made before the start of the test.

When running using the MONITOR method, it is not necessary to list load modules on the Load Module Intercept List screen since SmartTest detects when an external module is called and maintains control of the test session. When running using the NOMONITOR method, SmartTest cannot determine when control is transferred from one load module to another. If you are using the NOMONITOR method and want SmartTest to detect when a particular load module is entered, the load module must be listed on the Load Module Intercept List screen.

These considerations apply to load module intercepts:

- It is necessary to specify the exact load module name to be intercepted. Generic entries (e.g., VIA*) are invalid on the Load Module Intercept screen.
- Load module intercepts must be entered before the first RUN command of the test.
- It is necessary to specify the load module name for any load modules to be tested that are ATTACHed.
- Load module intercepts are not necessary for programs that are LINKed to (SVC 6), as long as SmartTest is running MONITORed. Enable monitoring of LINKed modules by using the SET LINK ON primary command.
- Only one subtask may be tested at a time. If you wish to test a subtask, the primary task must run NOMONITOR and may not include breakpoints or pseudo code.
- Testing of Dialog Management programs requires entries on the Load Module Intercept List screen if you are testing different load modules than the one specified on the setup screen. Selecting 'S' from the setup screen automatically displays the Load Module Intercept List screen.
- Once a load module is intercepted, SmartTest starts running MONITORed. Use LIST TAILOR entries to Break on Entry to the intercepted load modules.

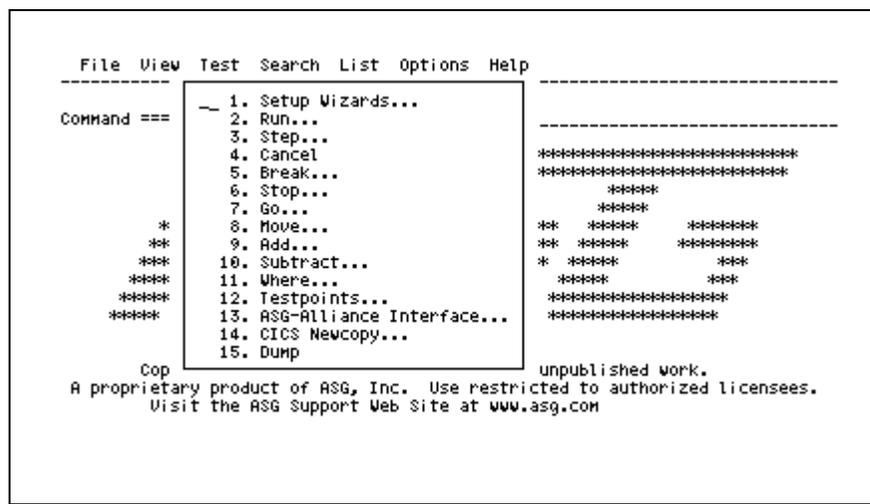
Note: _____

For more information on load module intercepts, see the List section of the *ASG-SmartTest Reference Guide*.

Controlling Program Execution

You can control execution using the Test pull-down or by using primary commands. The Test pull-down is shown in [Figure 86](#).

Figure 86 • Test Pull-down Menu



Selecting Test Environment Using the Setup Wizards

Use the Setup Wizards option to save time in setting up common types of test environments. Setup wizards are available for IMS, CICS, and programs in TSO foreground or using batch connect. These wizards are self-explanatory and walk you through the process of setting up the appropriate environment.

Executing the Program Continuously Using RUN

Use the Run action to begin test execution or to resume execution after an interrupt. An interrupt results from any of these situations:

- Abend error condition
- Address Stop (see ["Setting Program Address Stops" on page 152](#))
- Breakpoint (see ["Inserting Breakpoints to Interrupt Execution" on page 153](#))
- Completion of program

You can enter the RUN command any time program execution is suspended.

The RUN MON command allows all execution features to be available. The primary command RUN TO 1432 establishes line number 1432 as the stopping point. Execution continues to line 1432 unless other interrupts occur before the line is reached.

Note: _____
See the *ASG-SmartTest Reference Guide* for additional information.

Executing a Specified Number of Statements Using STEP

Use the STEP primary command to execute a specified number of statements. When there are no abends or breakpoints, the program executes the number of verbs specified or executes until program completion.

The STEP command executes the next statement and stops. For example, STEP 5 executes the next 5 statements before stopping. The STEP AUTO command executes one statement and pauses for the time period specified using the SET DELAY command.

Note: _____
See the *ASG-SmartTest Reference Guide* for additional information.

Changing Program Execution Sequence Using GO

Use the Go action on the Test pull-down, the GO line command, or the GO primary command to change execution sequence during a test. When you type GO for a statement, the chevrons indicating the next executable statement are moved to that statement. Typing RUN or STEP causes execution to begin with the newly specified statement.

Note: _____
To use the GO command, you must be in an active test session.

You can only enter a GO command for executable source lines (i.e., not on COMMENT lines).

Note: _____
Altering execution flow may cause some statements to be re-executed or by-pass statements that initialize data fields, resulting in abend conditions or unpredictable results.

It is recommended you use the GO command within the paragraph or block currently being executed.

See the *ASG-SmartTest Reference Guide* for GO command descriptions.

Interrupting Test Execution Using Keystrokes

Use the PA1/ATTN key to interrupt program execution during a test session.

For foreground testing, when you press PA1/ATTN after a RUN MONITOR primary command is issued, the RUN command is changed to a STEP command and the program is suspended at the next COBOL verb or Assembler instruction.

When you press PA1 or ATTN after a STEP *n* command is issued, the remaining steps are not executed and the program is suspended at the next COBOL verb or Assembler instruction.

Program execution may be interrupted at any time. If after initiating execution of a program with a RUN command you realize the test dataset being used contains more records than you wanted to process, you can stop execution by pressing PA1/ATTN.

Note: _____

The interrupt keys do not apply to CICS or IMS/DC, or when executing using the NOMONITOR method.

For Batch testing, the ATTN key suspends the test. This frees your TSO User ID, allowing you to view other programs or perform other work. The program continues execution until a breakpoint, abend, or completion of the test. At this point, the message `...WAITING FOR CONNECTION...` displays. You can then reconnect to the batch session.

Intercepting Program Abends

SmartTest automatically intercepts all program abend and suspends test execution. When the test is suspended, you may take action to correct or bypass the error condition. It is possible to detect multiple abend conditions in a single test session and continue program execution, by correcting data or coding temporary work-arounds.

In [Figure 87](#), a Data Exception (0C7) abend condition has been detected, and the program has been suspended. You may not continue program execution until the abend condition is corrected or bypassed.

Figure 87 • S0C7 Abend Screen

```

File View Test Search List Options Help
-----
Program View          VIAMERGE.VIAMERGE -A
Command ==>          Scroll ==> CSR

000432      MOVE ZERO TO READ-INFILE1-SWITCH.
(A)>>>      ADD 1 TO END-FILE-COUNT.
(B)''' +-----+
|          | 10 END-FILE-COUNT          PIC 9(8)          ADDR 000B9450 |
|          | VALUE > VMERGE              < * INVALID NUMERIC * |
|          |-----+-----+
000434      READ INFILE1 INTO INFILE1-WORK-REC
000435          AT END                                FALLTHRU
000436          MOVE INFILE1-EOF-MSG TO OUTRPT-WORK-DATA
000437          MOVE ZERO TO OUTRPT-WORK-KEY
000438          WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA
000439          MOVE 1 TO INFILE1-EOF
000440          MOVE EOF-KEY TO INFILE1-WORK-KEY.          FALLTHRU
000441      3100-READ-INFILE1-X.
(C)-----+
|STATUS: DATA EXCEPTION (0C7)          PROGRAM: VIAMERGE  DATE: DDMMYYYY |
| STMT: 000433  OFF: 0028D2  AMODE: 24  MODULE: VIAMERGE  TIME: HH:MM:SS |
|SOURCE: ADD 1 TO END-FILE-COUNT.      |
+-----+

```

This table describes the sections highlighted on the Program View screen:

Screen Section	Description
(A) Chevrons(>>>>>>)	The chevrons indicate the next statement to be executed. Execution of this statement caused an 0C7 abend.
(B) 10 END-FILE-COUNT	For data related abends (i.e., 0C7, 0C4), a data window displaying the data field values displays below the statement causing the abend.
(C) STATUS	The STATUS line in the status box specifies the Abend condition.

You may use SmartTest techniques described in later sections to modify incorrect data. See ["Inserting Breakpoints to Interrupt Execution" on page 153](#) and ["Viewing and Changing Test Session Data" on page 162](#).

This table describes the sections highlighted on the Program View screen:

Screen Section	Description
(A) Chevrons(>>>>>>)	The chevrons indicate the next statement to be executed is the MOVE. This statement modifies the address specified in the address stop.
(B) STATUS: STOPPED BEFORE ADDRESS MODIFIED	STATUS message indicates the program is suspended before an address modification.

See the *ASG-SmartTest Reference Guide* for more information on the Address Stop Entry screen and the STOP command.

Inserting Breakpoints to Interrupt Execution

These two methods can be used to insert breakpoints into your program to interrupt execution:

- Manually, at specific lines
- Automatically, based on program knowledge

Breakpoints remain active until deactivated or deleted. You may save breakpoints on the AKR for later test sessions with the program.

Manually Inserting Breakpoints in the Program

Use the BR line command to insert a breakpoint before a specific source code statement. You may use the BR line command to set unconditional interrupts in any program. When you type BR on a line containing a paragraph name or label, the BREAK is inserted before the first executable statement in the paragraph.

When you type BR on a non-executable line, the BREAK is inserted before the next executable statement in the program.

Automatically Inserting Breakpoints in the Program

Use the Break option on the Test pull-down or the BREAK command to set breakpoints before or after statements containing a specified target. The Break facility uses the Intelligent Search Function to insert multiple breakpoints with a single command.

Breakpoints can be inserted BEFORE or AFTER the specified target. The target may be limited or extended by the use of the operands such as USE, MOD, NOALIAS, ALL, NEXT, and PREV.

The BREAK facility has several options to specify the target where breakpoints are to be inserted. For example, to insert a breakpoint before paragraph 1000-INITIALIZE is executed, type this command:

```
BREAK 1000-INITIALIZE
```

You may check variables, change data values, or enter pseudo code whenever program execution is suspended.

[Figure 90](#) shows the Program View screen after 33 breakpoints have been inserted using the BREAK primary command.

Figure 90 • Break Before all Paragraphs Results Screen

```

File View Test Search List Options Help
-----
Program View                               33 BREAK(S) INSERTED (A)
Command ==>                               Scroll ==> CSR

(B)''1      BREAK.
000178      OPEN INPUT  INFILE1 INFILE2 INFILE3.
000179      OPEN OUTPUT OUTFILE OUTRPT.
(B)''1      BREAK.
000180      MOVE 'VMERGE ' TO BEGIN-PROGRAM-NAME.
000181      ACCEPT BEGIN-DATE FROM DATE.
000182      WRITE OUTRPT-REC FROM BEGIN-MESSAGE.
000183      IF BEGIN-DATE = 0 THEN
000184          MOVE +16 TO ABEND-CODE
000185          PERFORM 9999-ABEND-IT
000186          MOVE 0 TO END-FILE-COUNT.                                DEADCODE
000187          MOVE ZEROS TO  COMPARISON-KEY-1, COMPARISON-KEY-2,
000188                          INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,
000189                          MASTER-EOF-SWITCH,
+-----+
|STATUS: BREAK AT START OF TEST SESSION  PROGRAM: VIAMERGE  DATE: DDMMYYYY |
| STMT: 000165 OFF: 0014CC  AMODE: 24    MODULE: VIAMERGE  TIME: HH:MM:SS |
|SOURCE: PERFORM 1000-INITIALIZE THRU    |
+-----+

```

This table describes the sections highlighted on the Program View screen:

Screen Sections	Description
(A) 33 BREAK(S) INSERTED	The short message indicates 33 breakpoints have been inserted.
(B) ""1 BREAK	The screen has scrolled to the first executable statement in the first paragraph in the program, and a pseudo code BREAK statement has been inserted. Scrolling through the program shows the other BREAKs inserted by this command.

Note: _____
 BREAKs inserted by other techniques remain.

Inserting Breakpoints with Impact Datasets

SmartTest provides facilities for automatically setting breakpoints with lists of data items. The lists are called impact datasets. You can create impact datasets yourself or import them from AutoChange, Alliance, or Estimate.

Impact Datasets from Other Programs

Execution JCL supplied with these ESW products is used to run batch jobs for creating impact datasets:

- AutoChange—VIAMEXPJ
- Alliance—VIAMEXPJ
- Estimate—VIAMEXPJ

Note: _____

Please consult the *ASG-AutoChange User's Guide*, the *ASG-Alliance User's Guide*, or the *ASG-Estimate User's Guide* for information on creating impact datasets.

User-supplied Impact Datasets

The name of an impact dataset must match the name of the program to which it will be applied. When creating an impact dataset, create it as a member of a PDS and name it appropriately.

[Figure 91](#) is an example of a user-supplied impact dataset. The member contains a list of fully qualified datanames, one fully qualified dataname per line. Each dataname must begin in column ten. Lines beginning with an asterisk are comment lines. All lines that are not comment lines must contain an X in each of the first four columns.

Figure 91 • User-supplied Impact Dataset

```
*   THIS IS A COMMENT LINE
XXXX   TEST-DATE
XXXX   NUMBER1 OF TEST-NUM1
XXXX
XXXX
```

Steps to Set Breakpoints with Impact Datasets

To set breakpoints with impact datasets

- 1 Type TESTPOINT on the command line, or choose TESTPOINTS from the TEST menu. The Test - SmartTest Testpoint Generation pop-up displays.
- 2 Edit the field entries in the Test - SmartTest Testpoint Generation pop-up. When finished, press Enter. The Test - SmartTest Testpoint Generation pop-up is shown in [Figure 92](#).

Figure 92 • Test - SmartTest Testpoint Generation Pop-up

```

                                Test - ASG-SmartTest Testpoint Generation
Command ==> -----
Enter Program, Attribute, Impact type and PDS, then ENTER to generate point
Press END to return.

Generate Testpoints:
Program(s)..... *
Data name Attribute. MOD      (MOD/REF)
Impact type..... AUTO      (AUTO/ALL/EST/USER)
PDS..... 'ASGUSR.REL.TESTPNT'
Member..... -----
    
```

These are the fields on the Test - SmartTest Testpoint Generation pop-up:

Field	Description
Program(s)	Selects programs listed in the impact dataset. Enter the name of a single program or use an asterisk to select all of them. The names you select must match the names of the programs to which you will apply the breakpoints.
Data Name Attribute	Specifies where to set breakpoints. One of two entries is allowed, MOD or REF. MOD sets a breakpoint in the code wherever an impacted data item is modified. REF sets a breakpoint in the code wherever the impacted data item is referenced.
Add to Track List	Allows you to associate specific programs with a TCA plan if the TCA option for SmartTest is installed. If the TCA option is not installed, this field defaults to NO.

Field	Description
TCA AKR Name	If the TCA option for SmartTest is installed, this field contains the name of a TCA AKR. If the TCA option is not installed, this field defaults to N/A.
TCA Plan Name	If the TCA option for SmartTest is installed, this field contains the name of the TCA Plan you want to use. If the TCA option is not installed, this field defaults to N/A.
Impact Type	Indicates the source of the imported impact dataset. These are the valid values for this field: AUTO: AutoChange ALL: Alliance EST: Estimate USER: User-supplied
PDS	Specifies the PDS containing the impact dataset to be imported.
Member	Lists the PDS member name containing the impact dataset for Impact Types of ALL, EST, or USER,

- 3 After appropriately editing the field entries in the Test - SmartTest Testpoint Generation pop-up and pressing Enter, the Test - Testpoint Qualified Program List pop-up shown in [Figure 93](#), displays.

Figure 93 • Test - Testpoint Qualified Program List Pop-up

```

Test - Testpoint Qualified Program List
Command ==> _____ $scroll ==> CSR
Select program(s) from the list, then Press ENTER to generate points. Press
END to return.

PDS..... 'ASQUSER.REL.TESTPNT'
Program Specification: *

  Selected Program      Status
  -----
- CD05C384             READY
- TESTC06              READY
- TESTC06A             READY
- TESTC067             READY
***** BOTTOM OF DATA *****

```

- 4 Type S next to the names of those programs you want to select and press Enter. The Breakpoints List screen, shown in [Figure 94 on page 158](#), displays for the first program you selected.

5 Make adjustments in the Breakpoints List screen as necessary.

Figure 94 • Breakpoint Lists Screen

```

                                Breakpoints List
                                CD05C384.CD05C384
Command ==> _____ Scroll ==> CSR

Set pseudo . . ON      Pseudo active: 5      Inactive: 0
Set breaks . . ON      Breaks active: 5       Inactive: 0
Set whens . . . ON     Whens active: 0       Inactive: 0

$ Line Pseudo Code (breakpoints highlighted)      A Count
-----
- 000379          BREAK.                          Y 000000
- 000380          ENTRY 'CD05C'          USING L-PARM-AREA
-----
- 000607          IF L-BACS-DATE = SPACES
- 000608          BREAK.
- 000608          MOVE 'NOT INPUT' TO WPCD-NARR
-----
- 000609          ELSE
- 000610          BREAK.
- 000610          MOVE L-BACS-DD TO WPCD-DD
-----
- 000610          MOVE L-BACS-DD TO WPCD-DD
- 000611          BREAK.
- 000611          MOVE L-BACS-MM TO WPCD-MM
-----

```

The Breakpoints List screen allows you to make these adjustments:

- Change the status of SET PSEUDO, SET BREAKS, or SET WHENS.
- Deactivate individual breakpoints.
- Remove individual breakpoints.

When finished, exit the Breakpoints List screen. The Program View screen displays.

- 6 The Program View screen, shown in [Figure 95](#), displays breakpoints and the source code or disassembled object code and data. Inspect the breakpoints settings in the Program View screen.

Figure 95 • Program View Screen

```

File View Test Search List Options Help
-----
Command ==> Program View CD05C384.CD05C384
                Scroll ==> CSR

''''1 BREAK.
000380 ENTRY 'CD05C' USING L-PARM-AREA
000381 L-ERROR-REC.
000382 AA10-TEST-CALL.
000383 IF L-CALL-TYPE = 0
000384 PERFORM BA-INITIALISE.
000385 IF L-CALL-TYPE = 1
000386 PERFORM BB-ERROR-REPT.
000387 IF L-CALL-TYPE = 2
000388 PERFORM BC-CNTRL-REPT.
000389 SKIP1
000390 AA99-END.
000391 GOBACK. PGM EXIT
000392 EJECT
000393 BA-INITIALISE SECTION.
000394 *****
000395 *** THIS SECTION OPENS THE PRINT FILES AND SETS UP THE ***
000396 *** TITLE LINES. ***
000397 *****

```

- 7 Press PF3/PF15 to exit the Program View screen. You are returned to the Test - Testpoint Qualified Program List pop-up.

After successfully applying an impact dataset to a program, you see an IMPACT APPLIED message in the Test - Testpoint Qualified Program List pop-up, indicating that breakpoints have been saved into the program's source code in the AKR. You have these options:

- If you selected multiple programs in [step 3](#), those programs are still selected. Press Enter and repeat [step 4](#) and [step 6](#) for the next selected program.
- Select one or more program, press Enter, and proceed.
- Exit.

Inserting Breakpoints at Date-related Data Items

To set breakpoints at date-related data items, use the EXECUTE command to run the STFINDAT script while in a test. The STFINDAT script sets breakpoints wherever it finds date identifiers and special keywords unique to date processing across IMS (PCB date fields), CICS (EXEC CICS asktime), and COBOL (CURRENT_DATE, etc.).

When executed, the STFINDAT script:

- Finds all data items listed in the script.
- Highlights the data items.
- Issues a BREAK HI ALL command.

Removing Breakpoints Individually

Use the D line command to delete individual breakpoints from a program. This command operates in the same manner as the D (delete) line command in TSO/ISPF.

Removing All Breakpoints

Use the View - Reset Request pop-up or the RESET BREAKS primary command to remove all breakpoints from a program. Breakpoints included in blocks of pseudo code are only deleted if the BREAK statement is the first statement in the block.

Generating a Dump

Use the DUMP command, or select Test ► Dump, to generate a symptom and a snap dump of the program being tested, along with its control blocks. The symptom dump includes general program information and REGISTER information. The REGISTER information contained in the snap dump is limited to the current values at the time you issue the dump and might not include the user's register values. The snap dump consists of all PDATA and some SDATA SNAPX options and is appended to the symptom dump data. All dump data is copied to the log file. See "[Log/List/Punch Processing Options](#)" on [page 30](#) for information about processing the log file.

Canceling a Test Session

Use the Cancel action on the Test pull-down or the CANCEL command to terminate the current test session at any time. When you execute the Cancel action, testing ends and the program remains displayed in Program View. Use RUN or STEP to restart the test if desired.

To terminate a test session, follow this step:

- Type CANCEL in the command input area and press Enter.

Any open test files are closed and any allocated storage freed. The short message reflects the cancellation of the test session. The data window and status box are removed from the Program View Screen.

Note: _____

For more information regarding controlling execution and these commands, see the Test section in the *ASG-SmartTest Reference Guide*.

Exiting a SmartTest Test Session

To terminate a test session and exit from SmartTest, follow this step:

- ▶ Select File ▶ Exit.

Any pseudo code, marks, or breaks are saved in the AKR if the online operation parameters have Save as the default. You can also end SmartTest by pressing PF3 until you exit the program.

Viewing and Changing Test Session Data

SmartTest provides the ability to interactively view and change data, allowing you to test all program conditions, including conditions not exercised by the established test data. You can view and change data by opening data windows within the program and typing over the values presented in the data window.

Use the ZD, ZH, ZG, ZGH, ZOOMDATA, K, and KEEP commands to perform these functions.

Viewing Test Session Data Values

You can open windows to view the program data any time program execution is suspended.

Data windows can be displayed inline in the Procedure Division (where the data is used), in the Data Division (where the data is defined), or at the top of your screen. You may open windows on individual data items or on group level data and in display or hex format.

You can specify the size of each data window to conserve space on the screen, or let SmartTest size data windows based on the amount of data to be displayed. When a data window cannot accommodate all the data to be displayed, you may scroll the data window.

The values shown in the data window are always current and change as the program executes.

Viewing Test Session Data Items Inline

Use the ZD line command to open display format data windows to show the value and address of data items. Use the ZH line command to open hexadecimal format data windows to show the value and address of data items.

The result of a ZD or ZH line command is a data window positioned at the line where the command was entered. When the screen is scrolled, the data window moves with the display.

When you type ZD or ZH on a subscripted or indexed data item, the occurrence number is shown to allow you to change the entry being displayed.

You may use the *n* (number) operand to display one data item in a statement containing multiple data items. Data items are assigned relative numbers from left to right in the statement, and the number entered represents the relative number of the data item to be displayed.

The data window, shown in [Figure 97](#), is opened after the last line of the statement where you entered the ZD or ZH line command.

The VALUE area of the data window contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. If the length of the data item is greater than fifty bytes, additional lines are displayed with +50, +100, +150 and so on preceding them.

Figure 97 • Sample Zoom Data Screen - Data Division

```

File View Test Search List Options Help
-----
Command ==> Program View VIAMERGE.VIAMERGE -A
Scroll ==> CSR
000095 05 OUTRPT-WORK-DATA PIC X(117).
+-----+
| 05 OUTRPT-WORK-DATA PIC X(117) ADDR 000E158F |
| VALUE > ..... < |
| +50 > ..... < |
| +100 > ..... < |
+-----+
000096 01 OUTFILE-WORK-AREA.
000097 05 OUTFILE-WORK-KEY PIC 9(10).
000098 05 OUTFILE-WORK-DATA-1 PIC X(70).
000099 05 OUTFILE-WORK-DATA-2 PIC X(70).
000100 05 OUTFILE-WORK-DATA-3 PIC X(70).
000101 COPY VIAMSGS.
000102 01 MESSAGE-DEFINE-AREA.
+-----+
|STATUS: BREAK AT START OF TEST SESSION PROGRAM: VIAMERGE DATE: DDMMYYYY |
| STMT: 000165 OFF: 0014CC AMODE: 24 MODULE: VIAMERGE TIME: HH:MM:SS |
|SOURCE: PERFORM 1000-INITIALIZE THRU |
+-----+

```

An effective picture clause displays on the dataname line. For example:

- A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
- An alphanumeric field PIC XXXX displays as X(4).
- A field defined as COMP displays as COMP-4.
- A field with a long dataname defined as COMP-3 displays as C3.

When you enter another ZD or ZH line command on the same statement, the existing data window is replaced.

Data windows may be removed, individually or globally, from the program being viewed. (See ["Removing Zoom Data Windows" on page 165.](#))

Viewing Group Level Data Items Inline

Type ZG to open display format data windows to show the levels, value, and address of group data items and all of their subordinate data items.

Type ZGH to open hexadecimal format data windows to show the levels, value, and address of group data items and all of their subordinate data items.

The result of a ZG or ZGH line command is a data window positioned where the command was entered. When the screen is scrolled, the data window moves with the display.

When you type ZG or ZGH on a subscripted or indexed data item, the occurrence number is shown to allow you to change the entry being displayed.

You may use the *n* (number) operand to display one data item in a statement containing multiple data items. Data items are assigned relative numbers from left to right in the statement, and the number entered represents the relative number of the data item to be displayed.

The VALUE area of the data window contains the value of the data item. Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. If the length of the data item is greater than fifty bytes, additional lines are displayed with +50, +100, +150 and so on preceding them.

When you enter another ZG or ZGH line command on the same statement, the existing data window is replaced.

A ZD or ZH command is executed when you enter a ZG or ZGH line command for a data item with no subordinate data items.

Viewing Data Values in the DATA DIVISION

Use the ZOOMDATA primary command to reposition the screen to the definition of a specified data item, and open a data window to display the data item and its current value.

The result of a ZOOMDATA primary command is a data window positioned in the Data Division. When the screen is scrolled, the data window moves with the display.

Alphanumeric values are displayed in character format and numeric values are displayed in decimal format with leading zeroes suppressed. If the length of the data item is greater than fifty bytes, additional lines are displayed for each group of fifty bytes.

The effective picture clause displays on the dataname line:

- A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
- An alphanumeric field with PIC XXXX displays as X(4).
- A field defined as COMP displays as COMP-4.
- A field with a long dataname defined as COMP-3 displays as C3.

These are the abbreviations for the ZOOMDATA command:

- ZD *dataname* for display format.
- ZH *dataname* for hex format.
- ZG *dataname* for group level items in display format.
- ZGH *dataname* for group level items in hex format.

Removing Zoom Data Windows

To remove individual Zoom Data windows, follow this step:

- ▶ Type ZO to close a data window opened as a result of a ZOOM line or primary command.

To remove all Zoom Data windows, follow this step:

- ▶ Type RESET ZOOM to close all data windows opened as a result of a ZOOM line or primary command.

Viewing Data Values at the Top of the Screen

Use the Keep option from the View pull-down, the K line command, or the KEEP primary command to retain the value and address of data items in a KEEP window at the top of the screen. These are the variations of the K line command:

- K for data value in display format
- KH for data values in hex format
- KG for group level items in display format
- KGH for group level items in hex format

The result of the Keep option, KEEP command, or the K, KH, KG, or KGH line commands is a data window positioned at the top of Program View. When the screen is scrolled, the data window remains at the top of the screen.

When you enter the K, KH, KG, or KGH command for a subscripted or indexed data item, the occurrence number is shown to allow you to change the entry being displayed.

When entering the line commands, you may use the *n* (number) operand to display one data item in a statement containing multiple data items. Data items are assigned relative numbers from left to right in the statement, and the number entered represents the relative number of the data item to be displayed.

New data items are added to an existing data window after the last data item placed in the window.

When you enter another K, KH, KG, or KGH line command on the same statement, the information for those data items is added to the data window.

Alphanumeric values are shown in character format and numeric values are shown in decimal format with leading zeroes suppressed. If the length of the data item is greater than fifty bytes, additional lines are displayed for each group of fifty items.

An effective picture clause displays on the dataname line. For example:

- A numeric edited field with PIC ZZ,ZZ9 displays as PIC X(6).
- An alphanumeric field PIC XXXX displays as X(4).
- A field defined as COMP displays as COMP-4.
- A field with a long dataname defined as COMP-3 displays as C3.

The entire KEEP data window may be removed, or individual data items may be removed from the data window. (See ["Removing KEEP Data Windows from the Display" on page 167.](#))

You can issue the KEEP command using these abbreviations:

- K for data value in display format.
- KH for data values in hex format.
- KG for group level items in display format.
- KGH for group level items in hex format.

As an example of the KEEP command, KG COMPARISON-KEYS displays the levels, values, and addresses of the data item COMPARISON-KEYS and its subordinate data items.

Removing KEEP Data Windows from the Display

To remove individual data items from a Keep Data window, follow this step:

- ▶ Type D beside the data item you want to remove from the display.

To remove all Keep Data windows at one time, follow this step:

- ▶ Use the View - Reset Request pop-up or type RESET KEEP to close the entire Keep data window.

Changing Test Session Data Values

The value(s) shown in the data window is always current and changes as the program executes. You may alter data values by typing over the value displayed in any data window. The new values take effect immediately.

Data values can be changed to allow you to perform these functions:

- Modify the results of calculations.
- Alter the execution path by changing the value of counters and/or switches.
- Test error handling code by forcing bad data.
- Correct the data exceptions that are encountered.

Execution History (Backtrack)

Program execution history for COBOL programs may be displayed on the Program View screen during an active test session using the SmartTest Backtrack facility. This facility allows you to step backward and forward through the executed code whenever execution is suspended and to view data values as they existed at the time the statement was executed.

The SmartTest Backtrack Facility has two distinct operating modes - recording and reviewing.

Recording Program Execution History

Type SET BACKTRACK (abbreviated SET BA) to toggle the Backtrack Recording mode ON or OFF.

The Backtrack Facility recording mode can be turned on or off whenever an active test session is suspended - at entry to the program, a breakpoint, an address stop, or an ATTN key. Toggling the Backtrack Recording mode OFF causes the recorded history to be discarded.

When a test is canceled using the CANCEL command or reaches END OF TEST, Backtrack Recording mode is automatically turned OFF and the Backtrack recorded history is discarded. The Backtrack Recording facility is OFF by default when SmartTest is entered.

The SET BACKTRACK primary command may also include the desired size of the Backtrack Recording memory buffer where data is collected (i.e., SET BACKTRACK 2 MB for a 2 Meg buffer). This buffer is in memory with a default size of 1 Megabyte (100 KB for CICS). The buffer size affects how much statement execution history is available to the Backtrack Review facility.

After setting Backtrack mode on, continue the active test using the normal RUN or STEP primary commands. When the test execution is suspended by an error condition, a breakpoint, an address stop, an ATTN key, or the end of the program, you can review the Backtrack history as desired.

Reviewing Backtrack History

After turning Backtrack Recording Mode ON and executing the program until a breakpoint is encountered, you can review Backtrack history using either of these methods:

- Enter the Backtrack Review mode using the RUN or STEP function. (See ["Backtrack Recording/Review Session" on page 168.](#))
- Type LIST BACKTRACK to display statements which modified a selected variable. (See ["Using the BackTrack Variable History Function" on page 172.](#))

Execution history can be reviewed using Backtrack Review Mode in either a backward or a forward direction. After switching to Backtrack Review Mode, the status box at the bottom of the Program View screen displays the direction of the review: FWD or BWD. While in Backtrack Review Mode, you can open KEEP and ZOOM windows to view the contents of modified data fields at the time a particular statement is executed.

Note: _____

Before experimenting with this example, make sure all breakpoints and pseudo code statements are deleted by issuing the RESET PSEUDO primary command.

Backtrack Recording/Review Session

To review the recorded Backtrack history

- 1 Initiate a SmartTest test session. This example uses the program VIAMERGE.
- 2 Turn Backtrack Recording Mode ON and change the Backtrack history buffer size from the default to one megabyte by typing SET BACKTRACK 1M in the primary command input area and press Enter.

The amount of Backtrack buffer used depends upon the number of instructions executed, how much data is modified, number of file IOs, and the amount of data read during the recording of Backtrack history. When the buffer fills, SmartTest begins overwriting the oldest information, thus saving the more recent execution history.

Note:

The default buffer size for TSO is 1 MB and for CICS is 100 KB. The maximum buffer size is 8 MB for TSO and 1 MB for CICS. Your site can customize the maximum and default Backtrack sizes during product installation. For more information, see your Systems Programmer.

For the demonstration program VIAMERGE, 1 MB is sufficient to record the history of the entire program.

You can also set the Backtrack mode on using the CUA implementation:

- a Select Options ► Modes and press Enter. The Options - Modes screen displays.
- b Move the cursor to the Set field on the line containing BACKTRACK and type ON and press Enter.
- c Press PF3/PF15.

The status box at the bottom of the Program View screen indicates that Backtrack Recording Mode is ON as shown in [Figure 98](#).

Figure 98 • Program View in Backtrack Recording Mode

```

File View Test Search List Options Help
-----
                                Program View                VIAMERGE.VIAMERGE -A
Command ===> _____ Scroll ===> CSR

000164 PROCEDURE DIVISION.
>>>>>> PERFORM 1000-INITIALIZE THRU
000166          1000-INITIALIZE-X.
000167 PERFORM 2000-PROCESSING-LOOP THRU
000168          2000-PROCESSING-LOOP-X
000169          UNTIL FINISHED-READING-ALL-FILES.
000170 PERFORM 9000-TERMINATION THRU
000171          9000-TERMINATION-X.
000172 IF END-FILE-COUNT = 0 THEN
000173     MOVE +8 TO ABEND-CODE
000174     PERFORM 9999-ABEND-IT.
000175 GOBACK.                                PGM EXIT
000176
000177 1000-INITIALIZE.
-----+-----+
|STATUS: BREAK AT START OF TEST SESSION  PROGRAM: VIAMERGE  BACKTRACK: (A)|
| STMT: 000165 OFF: 0014CC AMODE: 24     MODULE: VIAMERGE    RECORDING  |
|SOURCE: PERFORM 1000-INITIALIZE THRU    |
+-----+-----+

```

BACKTRACK: RECORDING. The usual Date and Time fields are replaced to indicate that Recording Mode is ON.

- 3 Begin recording Backtrack history by typing RUN TO *nnnn* to execute the test to line number *nnnn*.

When the execution is suspended because the program reached line *nnnn*, you can review the Backtrack history and view the value of data items at the time each statement was executed.

- 4 Begin reviewing Backtrack history by selecting Test on the action bar and pressing Enter.
- 5 Select Test ▶ Run and press Enter to display the Test - Run Request pop-up as shown in [Figure 99](#).

Figure 99 • Test - Run Request (BackTrack Recording Mode Active) Pop-up

```
File View Test Search List Options Help
-----
Test - Run Request (Backtrack Recording Mode Active)
-----
Enter Direction and any "Run To" choices, as needed. Then press Enter.
NOTE: * indicates an option that invokes BackTrack Review Mode.
Direction:
2_ 1. Forward (Default)
   2. *Backward
Run To Choices:
*Run To Top . . . . - Enter non-blank to select
Run to Line Number -
*Run to Data Name . END-FILE-COUNT _____
000177 1000-INITIALIZE.
+-----+
| STATUS: BREAK AT START OF TEST SESSION   PROGRAM: VIAMERGE  BACKTRACK: |
| STMT: 000165 OFF: 0014CC AMODE: 24      MODULE: VIAMERGE  RECORDING |
| SOURCE: PERFORM 1000-INITIALIZE THRU     |
+-----+
```

- 6 Select Backward in the Direction field.
- 7 Type END-FILE-COUNT in the Run to Data name field and press Enter.

These specifications direct SmartTest to scroll backward through the executed statements until the first statement in reverse order which modifies the field END-FILE-COUNT. The status box at the bottom of the Program View screen indicates that the first statement found was 13 statements backward from the statement where execution is suspended as shown in [Figure 100](#).

Figure 100 • Reviewing Program Execution in Program View

```

File View Test Search List Options Help
-----
                                Program View                                VIAMERGE.VIAMERGE -A
Command ==> _____ Scroll ==> CSR

000179      OPEN OUTPUT OUTFILE OUTRPT.
BKTR=>      MOVE 'VMERGE ' TO BEGIN-PROGRAM-NAME.
000181      ACCEPT BEGIN-DATE FROM DATE.
000182      WRITE OUTRPT-REC FROM BEGIN-MESSAGE.
000183      IF BEGIN-DATE = 0 THEN
000184          MOVE +16 TO ABEND-CODE
000185          PERFORM 9999-ABEND-IT
000186      MOVE 0 TO END-FILE-COUNT.                                DEADCODE
000187      MOVE ZEROS TO      COMPARISON-KEY-1, COMPARISON-KEY-2,
000188          INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,
000189          MASTER-EOF-SWITCH,
000190          HASH-TEST-A, HASH-TEST-B, HASH-TEST-C,
000191          HASH-TOTAL,
000192          COMPARISON-CODES,
+-----+
|STATUS: * REVIEWING BACKTRACK HISTORY * PROGRAM: VIAMERGE DIRECTION: BWD |
| STMT: 000180 OFF: 001692 AMODE: 24 MODULE: VIAMERGE SEQ# -13          |
|SOURCE: MOVE 'VMERGE ' TO BEGIN-PROGRAM-NAME.                          |
+-----+

```

BKTR==>. This indicator is placed on the statement found as the result of the Run backward action.

DIRECTION: BWD SEQ# -13. This indicates that Backtrack Review Mode is in effect and that the review direction is backward. The current statement (with the BKTR==> indicator is 13 statements back from the statement where execution is suspended.

The Run function can be repeated to scroll to each statement in turn which modified the field END-FILE-COUNT. The same result can be obtained using the RUN BACK TO END-FILE-COUNT MOD primary command.

To move forward in the execution history, use the Run function as shown in [Figure 99 on page 170](#) specifying Forward as the Direction.

To return to the statement where execution is suspended, type RUN TO *. This scrolls the display to the statement where execution is suspended and turns Backtrack Review Mode OFF. Back Recording Mode remains ON.

Note: _____

For more information on BackTrack commands, see the *ASG-SmartTest Reference Guide*.

Using the BackTrack Variable History Function

To review BackTrack execution history using the **LIST BACKTRACK** primary command

- 1 Type **LIST BACKTRACK** to display the List - BackTrack Variable History pop-up shown in [Figure 101](#).

Figure 101 • List - BackTrack Variable History Pop-up

```

List - BackTrack Variable History

Type a fully qualified data name with any applicable subscripts
in the Target area. For a list of data names, type a "pattern"
in the Target area. Then press Enter.

Target  END-FILE-COUNT
    
```

- 2 On the List - BackTrack Variable History pop-up, enter a dataname to search the BackTrack execution history for statements which modified the specified data item. You may enter a pattern to select from a list of datanames.

Press Enter to display the List - BackTrack Variable History screen, shown in [Figure 102](#), showing the statements which modified the selected data item.

Figure 102 • List - BackTrack Variable History Screen

```

List - BackTrack Variable History          VIAMERGE.VIAMERGE -A
Command ===>                               Scroll ===> CSR
Variable: END-FILE-COUNT                    Desc: PIC 9(8)
Index:

S RELMOD VALUE                               LINE  STATEMENT CONTENTS
-----
***** TOP OF DATA *****
VIAMERGE.VIAMERGE
-   -1 .....                                000180  MOVE 'VMERGE ' TO BEGIN-PROGRAM-NAME.
   0 VMERGE                                  >>>>>>  <CURRENT VALUE OF VARIABLE>
***** BOTTOM OF DATA*****

+-----+
|STATUS: DATA EXCEPTION (0C7)                PROGRAM: VIAMERGE  BACKTRACK:      |
| STMT: 000433  OFF: 0028D2  AMODE: 24      MODULE: VIAMERGE  RECORDING  |
|SOURCE: ADD 1 TO END-FILE-COUNT.            |
+-----+
    
```

- 3 The List - BackTrack Variable History screen lists the statements which modified the selected data item. You may view a statement in the program source by selecting the statement and pressing Enter.

You may review other statements in BackTrack execution history by typing **LIST BACKTRACK** and repeating the steps shown above.

After completing your review of the selected statements, you may continue the test by pressing PF3/PF15 on the List - BackTrack Variable History screen or by typing RUN TO * on the Program View screen.

Using Pseudo Code

SmartTest provides the ability to interactively modify the logic of a program while it is being tested. This gives you the opportunity to temporarily fix problems and test hypothetical situations.

Pseudo Code Concepts

To interactively modify the logic of a program, SmartTest provides COBOL compatible statements called pseudo code. You may enter these statements into COBOL and Assembler source code. Inserted code may change the execution sequence of the program, temporarily fix a data problem.

You can enter pseudo code statements in the Procedure Division of any COBOL or Assembler program displayed in Program View. Each pseudo code statement is associated with the executable source statement (COBOL verb or Assembler instruction) following it. Multiple pseudo code statements may be associated with an executable source statement to form a pseudo code block. Pseudo code statements follow standard COBOL syntax rules.

The LIST PSEUDO command displays the pseudo code List screen showing all pseudo code for the program currently in Program View. Pseudo code remains active until deactivated or deleted and may be saved with any program stored on the AKR.

Note: _____

For more setup options on automatically saving pseudo code, see ["Introduction" on page 1](#).

Pseudo Code Statements Available

SmartTest provides these pseudo code statements:

- ADD
- BREAK
- GO TO
- IF/THEN/ELSE
- MOVE
- SUBTRACT
- WHEN
- pslabel
- 77 (item)
- &COUNT

You can also use the ADD, SUBTRACT, MOVE, GO TO, and WHEN statements as primary commands.

Pseudo Code Statements

Code	Description
ADD	Adds the value contained in or represented by the first operand to the value contained in or represented by the second operand. The value is converted to the proper format for the specified data item.
BREAK	Forces a breakpoint before a specific statement. A pseudo code BREAK statement causes an unconditional interrupt in the program execution.
GO TO	Transfers control to the statement containing the specified COBOL paragraph name, pseudo code label, or line number.
IF/THEN/ELSE	Tests conditional expressions. Nested IF statements are supported. Supported condition tests include Relation, Class and Sign.
MOVE	Assigns the value contained in or represented by the first operand to the area represented by the second operand. The value is converted to the proper format for the specified data item, if possible. If the value cannot be converted to the proper format, program execution stops and an error message displays.

Code	Description
SUBTRACT	Subtracts the value contained in or represented by the first operand from the value contained in or represented by the second operand. The value is converted to the proper format for the specified data item.
WHEN	<p>Tests conditional expressions in the same manner as COBOL conditional expressions. Use of the WHEN statement causes the test to be performed after the execution of every COBOL verb that modifies data.</p> <p>Note: Processing WHEN statements is resource intensive. Use sparingly.</p>
pslabel	Defines a pseudo code paragraph name. These names are entered in Area A and are used by GO TO pseudo code statements. A pseudo code label can consist of one to thirty alphanumeric characters, the first being alphabetic. The specified name cannot be an existing COBOL data or label name. Pseudo code labels cannot be defined in WHEN statements.
77	Defines a pseudo code data item. These data items are entered in Area A. Each 77 level dataname must be unique and cannot be qualified. 77 level data definitions can be entered in either the Data Division or the Procedure Division. Pseudo code data items must be defined before being referenced and cannot be defined in WHEN commands.
&COUNT	<p>Specifies the &COUNT SmartTest operand which is provided for use in pseudo code statements. This variable is a counter used to indicate the number of times each pseudo code statement has been executed. &COUNT is useful in tests for specifying alternative logic based on its value. The value of &COUNT is automatically initialized to zero each time the pseudo code line is modified or entered. It is incremented before the corresponding verb is executed.</p> <p>&COUNT can be referenced as required in any pseudo code statement, but cannot be modified (e.g., MOVE 2 TO &COUNT is invalid).</p>

COBOL RESERVED WORDS in Pseudo Code

Many standard COBOL reserved words can be used in pseudo code. These reserved words are used in pseudo code statements in the same manner they are used in standard COBOL programs. These COBOL reserved words are supported by SmartTest pseudo code:

ALL	ELSE	PIC
ALPHABETIC	EQUAL	PICTURE
COMP	FROM	POSTIVE
COMP-1	GREATER	SPACE
COMP-2	IS	SPACES
COMP-3	LESS	THAN
COMP-4	LOW-VALUE	THEN
COMPUTATIONAL	LOW-VALUES	TO
COMPUTATIONAL-1	NEGATIVE	USAGE
COMPUTATIONAL-2	NEXT SENTENCE	ZERO
COMPUTATIONAL-3	NOT	ZEROS
COMPUTATIONAL-4	NUMERIC	ZEROES

You can use these symbols in conditional pseudo code statements:

- = (equals)
- < (less than)
- > (greater than)

Entering and Editing Pseudo Code

Pseudo code statements can be entered inline with existing COBOL or Assembler source code while viewing a program in Program View. During a test session, pseudo code statements can be entered whenever program execution is suspended. Pseudo code statements are in standard COBOL format.

The C (copy), D (delete), I (insert), M (move), and R (repeat) line commands are available to you for entering and manipulating pseudo code statements in a program. You can also use the C (copy) and R (repeat) line commands with existing lines of COBOL source code (the new lines are then considered pseudo code). These commands operate in the same manner as in TSO/ISPF, including block format.

Pseudo code statements can contain numeric and non-numeric literals. If the data element being tested is a character field (non-numeric), the second operand in the pseudo code statement must be in quotes. For example:

```
IF OUT_CNT_4 = '001' THEN BREAK.
```

If the data element being tested is a decimal field (numeric), then the second operand in a pseudo code statement does not need to be in quotes or zero padded. For example:

```
IF OUT_CNT_3 = 1 THEN BREAK.
```

Executing Pseudo Code in a Test Session

Pseudo code is executed as part of the program logic when any execution control commands (RUN or STEP) are issued.

Each statement is associated with the source statement (COBOL verb or Assembler instruction) following it, which means it gets executed before the next verb. For example, when a breakpoint occurs, processing stops and the chevrons are positioned in the line number area of the next statement to be executed. If a block of pseudo code is inserted immediately before this statement, the RUN function starts executing with the first pseudo code statement and the STEP function repositions the chevrons to the first pseudo code statement.

Pseudo code is syntax-checked when a command is entered referencing a pseudo code statement or variable, and when saving the pseudo code to the Application Knowledge Repository (AKR). These commands reference pseudo code: RUN, STEP, ZOOM commands, LIST BREAKS, LIST PSEUDO, LIST WHENS, LPRINT PSEUDO, QUALIFY.

Pseudo code statements with correct syntax are executed as part of the program logic. These statements are not validated to determine if they are logically correct. When a pseudo code statement causing a program execution error is encountered, the current test is interrupted and an error message displays.

Viewing Pseudo Code in a Test Session

Use the pseudo code action on the List pull-down or the LIST PSEUDO primary command to display the Pseudo Code List showing all pseudo code for the program in Program View.

Pseudo code statements may be activated/deactivated individually or globally on the Pseudo Code List.

You may also use the SET PSEUDO command to globally activate/deactivate pseudo code. (See ["Setting Test Session Options" on page 180.](#))

Removing Pseudo Code from a Program

You can remove pseudo code from a program individually or globally.

To remove individual lines of pseudo code, follow this step:

- ▶ Use the D line command. This command operates in the same manner as the D (delete) line command in TSO/ISPF, including block format.

To remove all pseudo code, follow this step:

- ▶ Select View ▶ Reset, or type RESET using the PSEUDO operand to delete all pseudo code, including breakpoints, within a program.

Note: _____

For more information regarding pseudo code and related commands, see the View section in the *ASG-SmartTest Reference Guide*.

Using Multiple Programs

Use the Open option on the File pull-down or the QUALIFY primary command to display another program in Program View. Programs to be displayed in source code format must reside on the AKR specified for the current session. Programs not stored on the AKR may be displayed in disassembled object code.

The program currently being tested remains the active program. If a STEP, RUN, GO, or CANCEL primary command is issued, it is executed for the active program being tested, not the program being viewed; and, the status box changes to indicate the status of the active program being tested. All other commands are performed on the displayed program.

The QUALIFY * primary command is used to redisplay the active program. The QUALIFY CANCEL primary command is used to remove the program being viewed from qualification for viewing.

These are two examples of the QUALIFY primary command:

QUALIFY *loadmod.program*, which brings a program from a different load module into Program View.

Q CANCEL ALL, which removes all programs from qualification and exit Program View.

Tailoring a Test Session by Program

The Test Session Tailoring screen, shown in [Figure 103](#), is used to turn SmartTest features selectively ON or OFF for a set of modules and/or programs. This screen displays using one of these methods:

- Selecting TAILOR on the Test Facilities List screen
- Entering the LIST TAILOR command on any SmartTest screen
- Selecting Test session tailoring on the List pull-down

Session tailoring can reduce the execution time for a test providing options at the individual program level.

A sample input line is provided on the first data line displayed to indicate how to specify information for a program. Use the I (Insert), R (Repeat), and D (Delete) line commands when editing the list of programs. Block forms of these commands are not supported.

Note: _____

For details of the Test session tailoring screen, see the *ASG-SmartTest Reference Guide*.

Figure 103 • Test Session Tailoring Screen

```

Command ==> _____ Test Session Tailoring _____ Scroll ==> CSR
      Module.Program id  Monitor Track Count Break Break Break Pseudo Single
      -----          Act    Act    Act    Act  Entry  Rtn  Act    Step
**** ***** TOP OF DATA *****
''' TESTCOBB.TESTCOBB  YES    NO    YES    NO    NO    NO    NO    NO
''' TESTCOBA.TESTCOBA  YES    NO    YES    NO    NO    NO    NO    NO
**** ***** BOTTOM OF DATA *****

```

Setting Test Session Options

Use the Modes action on the Options pull-down or the SET primary command with no operands to display the Options - Modes screen. This facility allows you to enable or disable any mode on the list.

Typing SET with an operand functions as a toggle switch for the mode specified.

The Options - Modes screen displays when the SET command is entered without operands. This screen shows the current setting for each test session mode. The Options - Modes screen can be used to change the setting for any mode by entering the desired value in the Set field. Modes changed from the default setting are highlighted on the Options - Modes screen.

The current setting for each option is saved between sessions in your PROFILE member on the AKR, with the exception of BREAK, PSEUDO, SCRIPT, and WHENS. Default values for these four options are restored when a new session is initiated.

This is how the test session options can be grouped:

Modes	Options
Capture/Replay	SCRIPT
Data Display	DATA, HEX, KEEP, OPERANDS, SCALE, VALUES, ZEROFILL
Test Execution	ASM, BACKTRACK, BREAKS, DELAY, LINK, MAIN, MONITOR, PSEUDO, TRACK, WHENS, STOPEXEC (CICS only), STOPHAND (CICS only)
Program View	ASMVIEW, AUTOQUAL, CUA, GENERATED, OUTLINE, REFRESH, SHADOW, XMODE
Status Box	ASM, FLOATING, REGISTERS, STATUS

For example, change the SCRIPT mode by typing SET SCRIPT. Reset all test session options to their default values by typing SET DEFAULTS.

Note: _____

For more information regarding the test session options, see the Options section in the *ASG-SmartTest Reference Guide*.

Displaying Program and Test Information

To select program and test information about the program in Program View, for display, use the actions on the List pull-down. You may also use the LIST primary command with no operands to display the Test Facilities List screen as shown in [Figure 104](#).

Figure 104 • TSO Test Facilities List Screen

```

                                Test Facilities List
Command ==> ----- Scroll ==> CSR
ASG2343I USE THE DOWN COMMAND FOR ADDITIONAL TEST SESSION OPTIONS.
Enter $ before the keyword to select a list category

-----
Keyword      Description
-----
- AKRMEM      List the AKR MEMBERS in the concatenated data bases
- ACCESS      List the ACCESS REGISTERS for the active program
- ADSTOP      List ADDRESS STOPS for the current test session
- BACKTRACK   List BACKTRACK variable history by variable name
- BREAKS      List BREAKpoints in the current program
- CALLS       List programs that are CALLED by the current program
- COMPILE     List COMPILER or Assembler options for the current program
- COUNTS      List statement execution COUNTS and histogram
- EQUATES     List EQUATES for the current program
- FLOATING    List the FLOATING point registers for the active program
- INTERCEPT List the load modules to be INTERCEPTed
- MEMORY      List an area of MEMORY in a dump format
- MODULES     List directory of load MODULES in current load libraries
- PERFORMS    List PERFORM ranges in the current COBOL program
- PROFILE     List the members of the PROFILE data set
- PROGRAMS    List PROGRAMS and subprograms in the current COBOL program
- PSEUDO      List PSEUDO code in the current program

```

The LIST command is used to display program related information in a consolidated format. Entering the LIST command with an operand results in the display of the specified list screen.

The COUNTS, FLOATING, MEMORY, and REGISTERS operands pertain to the active program, not necessarily the program being viewed in Program View.

This is how the test facilities lists may be grouped:

List Type	Operands
COBOL Verb List	SUBSET
CICS Environment Lists	EIB, FILE, LIMITS, TABLE (USER/GLOBAL)
IMS/DC Environment List	QUEUE
Program Knowledge Lists	CALLS, COMPILE, EQUATE, PERFORMS, PROGRAMS

List Type	Operands
Test Control Lists	ADSTOP, AKRMEM, BREAKS, INTERCEPTS, PSEUDO, TAILOR, WHEN
Test Execution Lists	ADSTOP, BACKTRACK, COUNTS, FLOATING, MEMORY, MODULES, REGISTERS, TRACKING

Note: _____

For more information on lists, see the LIST command in the *ASG-SmartTest Reference Guide*.

Printing Displayed Information (LPRINT Command)

Issuing the LPRINT * primary command results in the entire virtual screen (all data viewed by scrolling forward and backward) to be copied to the List File. All excluded lines are copied to the List file as excluded lines (as they appear on the screen at the time the LPRINT * command is entered).

The LPRINT * primary command can be entered on these screens to copy them to the List File:

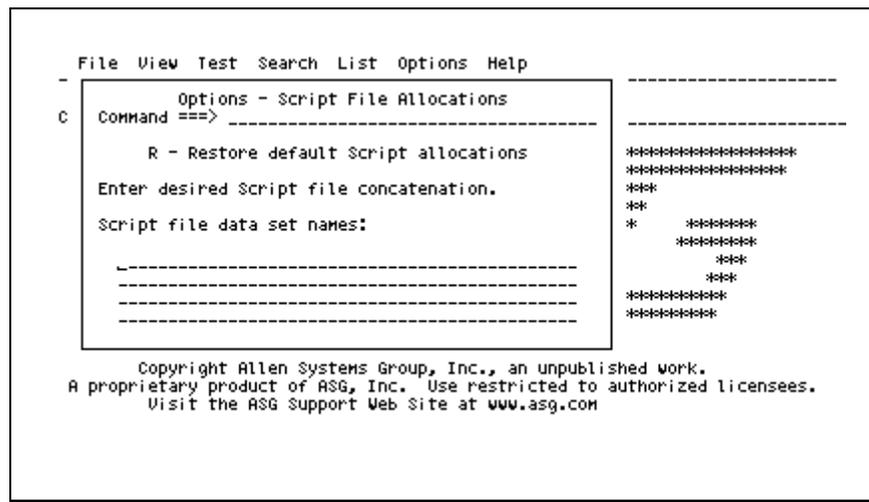
- BackTrack Variable History screen
- Breakpoints List screen
- Execution Counts screen
- Execution Tracking screen
- List - CALL Statements pop-up
- List - COBOL Subsets Names pop-up
- List - Equates pop-up
- List - Perform Range Names pop-up
- List - Program/Subprogram Names pop-up
- List - User Marks pop-up
- Pseudo Code List screen
- When Conditions List screen

Note: _____

For more information regarding this command, see the Commands section in the *ASG-SmartTest Reference Guide*.

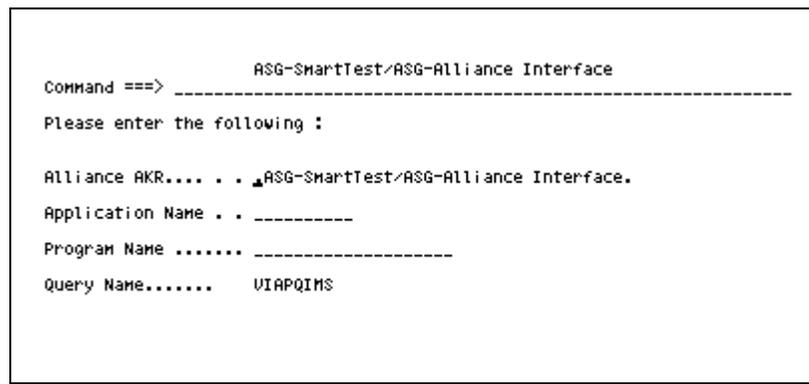
- In the Options - Script File Allocations pop-up, enter the name of the user-defined PDS containing the VIAPALSC and VIAPQ* members. The Options - Script File Allocations pop-up is shown in [Figure 106](#).

Figure 106 • The Options - Script File Allocations Pop-up



- Exit to the SmartTest primary panel. On the command line, issue the ALLIANCE command. The SmartTest/Alliance Interface pop-up shown in [Figure 107](#) displays.

Figure 107 • The SmartTest/Alliance Interface Pop-up



- Press Enter, verify the information, and press Enter again. The VIAPALSC script executes and runs the query listed in the SmartTest/Alliance Interface pop-up. A series of screens flash by, ending with an Alliance query display appropriate for the load module, JCL, transactions, and current SmartTest execution environment.
- To exit the query display, type END on the command line. The Alliance primary panel displays. Exit the Alliance primary panel to return to the SmartTest primary panel.

Fields

Field	Description
Alliance AKR	Specifies the AKR containing the analysis information produced by Alliance for the specified application and program.
Application Name	Specifies the name of the application containing the program to be processed in Alliance.
Program Name	Specifies the name of the program to be processed in Alliance.
Query Name	Specifies a query name other than the one SmartTest selects from the PDS specified in the Options Script File Allocations pop-up. The Query Name field is automatically filled with the name of a query script appropriate for the current environment, however, you can specify another.

5

Program Analysis Features

This chapter presents the basic problem investigation features available with SmartTest. Examples are provided for many of the commands shown. You may re-create most of these examples using the VIAMERGE program.

Topic	Page
COBOL Intelligent Search Function	187
Finding Program Information Using the Search Function	195
Printing Program Information	209
Repositioning the Display	210
Following Branching Logic	211
Searching the Program in Execution Sequence	215

COBOL Intelligent Search Function

The COBOL Intelligent Search function uses SmartTest's built-in understanding of COBOL. The search function can be used to search the program for relevant information about COBOL verbs, data items, labels, and other program details. The search function recognizes and does not search COMMENT statements, unless specifically requested.

The six primary commands that use the COBOL Intelligent Search Function are BREAK, EXCLUDE, FINDXTND, HIGH, LPRINT, and SCROLL. The BREAK command is described in ["Testing Techniques" on page 143](#). Descriptions of the other commands and examples of command usage are contained in later sections.

Search for COBOL Subsets

SmartTest classifies COBOL statements into COBOL subsets by grouping together COBOL verbs of a similar nature. For a complete list of the verbs included in each of the COBOL subset names, select List ► COBOL intelligence features and Subsets from the List COBOL Features pop-up, 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	Includes statements that assign a value to a data item; e.g. MOVE, ADD, or COMPUTE.
CALL	Includes statements related to subprogram calls such as CALL and CANCEL.
CIcs	Includes any CICS or DL/I command level command.
COBOLII	Includes COBOL II, including CONTINUE, END, and INITIALIZE verbs.
COBOL/370	Includes statements and clauses unique to COBOL/370, such as intrinsic function calls, procedure pointers, and calls to LE/370 run-time environment.
COMment	Includes statements having no run-time effect such as all lines with an * (asterisk) in column 7, the entire IDENTIFICATION DIVISION, and NOTE statements.
CONditional	Includes statements or parts of statements that conditionally change the flow of control in a program such as IF, ELSE, and WHEN.
COPy	Includes COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, -INC statements.
DB2/SQL	Includes EXEC SQL statements.
DDL	Includes SQL Data Definition Language statements, such as CREATE, ALTER, DECLARE, and DROP.
DEAD	Includes statements containing dead code and dead data.
DEADCode	Includes statements containing code that cannot be executed under any conditions.

COBOL Subset	Description
DEADData	Includes DATA DIVISION statements containing datanames and their aliases that are not referenced in the PROCEDURE DIVISION.
DEBug	Includes statements containing a DEBUG, EXHIBIT, ON, READY, or RESET verb, as well as statements containing a D in column 7.
DEFinition	Includes declaratives of data items including the SPECIAL-NAMES paragraph in the ENVIRONMENT DIVISION, as well as the entire DATA DIVISION.
DIRective	Includes statements that direct the compiler to take specific actions during compilation such as BASIS, EJECT, and TITLE.
DL/I DL/1	Includes EXEC DL/I commands, ENTRY 'DLITCBL', CALL 'CBLTDLI'.
DML	Includes SQL Data Manipulation Language statements, such as SELECT, UPDATE, INSERT, and COMMENT.
ENtry	Specifies the PROCEDURE DIVISION statement and all ENTRY statements.
EXit PGMExit	Includes statements containing a STOP RUN, GOBACK, or EXIT PROGRAM verb, as well as CALL statements that are indicated as NORET (non-returning).
FALLthrough	Includes statements of PERFORMed paragraphs or units that fall through to the next paragraph.
GOto	Includes statements containing an ALTER or GOTO verb.
IDMS	Includes IDMS statements.
INClude	Includes COPY, COPY IDMS, SQL INCLUDE, ++INCLUDE, -INC statements.
INPUTOutput IO Input Output	Includes COBOL IO statements (IO, Input, or Output) including CALL statements that are indicated as containing IO, Input, or Output.
LABel DIVision PARagraph SECTion	Includes 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.

COBOL Subset	Description
MAINline	Includes mainline code statements that are reachable from the PROCEDURE DIVISION line to the program units by following FALLTHROUGHS and GO TOs, but not PERFORMs.
PERform	Includes statements containing the PERFORM, SORT or MERGE verbs.
RETurn	Includes statements of a PERFORMed paragraph ranges that return control.
SStructure	Includes a group of COBOL subsets that together help show the general structure of the program. These COBOL subsets include CALL, PERFORM, DIVISION, SECTION, PARAGRAPH, EXIT, and GO TO.
TESted	Identifies the lines of code that have been tested based on information created and updated with TCA reports.
UNTested	Identifies the lines of code that have not been tested based on information created and updated with TCA reports.

Screen Subsets

Screen subsets generally are the result of an interactive command. To specify one of these subsets, type the entire name or the minimum abbreviation:

- Highlighted or HI
- NONHighlighted or NHI
- Excluded or X
- NONExcluded or NX

Tag Subsets

Tag subsets are displayed in columns 73 through 80 on the Program View screen, which SmartTest uses to provide immediate information about the source code. Command results show tags in these columns. Data items that are never referenced, statements containing dead code, and statements containing program exits are also tagged in columns 73 through 80. PERFORMed paragraphs use columns 73 through 80 to indicate if they fall through or return. Specific tags are provided for the TRACE option facility.

These tags can be used in commands that accept subsets as targets:

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

When a SUBSET is specified as the target of these commands, the entire program is searched for all occurrences of verbs associated with the SUBSET. All information pertaining to a COBOL function may be presented with a single command.

Search for Dataname References

When a dataname is specified as the target, the program is searched for all occurrences of the specified field. This search includes all aliases of the dataname, by default. The targets found are highlighted and tagged as a reference.

Reference tags are placed in columns 73 through 80 indicating the type of target found. These are the valid dataname reference tags:

Tag	Description
DATA MOD	Specifies the value of the data item is being set or altered.
DATA USE	Specifies the value of the data item is being tested or used.
DATA DEF	Specifies the definitions of a data item or its aliases.

Search for Indirect Dataname References

Using the SIZE operand with the search function locates all datanames directly or indirectly affected by a change in the size of the specified dataname. This results in a complete list of all data items to be reviewed for a size change of the specified dataname. By including the LEVELS operand, the indirect impact of a size change to a dataname can be seen one level at a time.

Using the VALUE operand with the search function locates all occurrences of a dataname directly or indirectly affected by a change in the value of the specified dataname.

Limit the Search Scope

The search function may be limited to direct dataname references with the NOALIAS operand. ALIAS is the default.

The MOD, USE, and DEF operands limit the search function to include only the specified data item reference type. REF is the default and includes MOD, USE, and DEF.

The search function may be limited in direction using the operands NEXT and PREV to start the search from the current position and locate the closest occurrence in the specified direction. The FIRST and LAST operands locate the first and last occurrence in the program respectively. ALL is the search function default, with the exception of the BREAK command where the default is NEXT.

The search function can be limited to a particular statement type through the use of the IN clause.

Excluding Lines from the Display

To examine only desired information on the screen, you can remove unneeded statements from the display. The exclude function can suppress the display of statements meeting specified criteria in Program View.

Select Test ► Exclude or use the EXCLUDE command to remove all lines from the display that meet the specified criteria. Lines may be excluded from the display before or after issuing other commands.

Excluded lines are represented by shadow lines stating *n* LINE(S) NOT DISPLAYED and a line of dashes. The dashed line can be suppressed with the SET SHADOW command (see the Options section of the *ASG-SmartTest Reference Guide*).

Excluded lines may be redisplayed as desired. (See ["Redisplaying Excluded Lines" on page 194.](#))

The EXCLUDE primary command may be abbreviated as X. For example, to remove all non-highlighted lines from Program View, type this command:

```
X NHI
```

Removing Lines from the Display

To remove all lines from the display using the program VIAMERGE

- 1 Type EXCLUDE in the primary command input area and press Enter.

Or

Select View ► Exclude and press Enter. The View - Exclude Request pop-up, shown in [Figure 108](#), displays.

Figure 108 • View - Exclude Request Pop-up

```

View - Exclude Request

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

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

String _____

```

- 2 Specify All lines in the Options field and press Enter to remove all lines from the display.

Figure 109 • Program View Lines of Code Omitted

```

File View Test Search List Options Help
-----
Command ==> Program View 2525 LINES EXCLUDED
                Scroll ==> CSR
-- ... - 2525 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

This table describes the messages displayed on the Program View screen:

Message	Description
2525 LINES EXCLUDED	Reflects the number of lines excluded from the display.
-- ... - 2525 LINES NOT DISPLAYED	Indicates how many lines are not displayed at that point in the program. In Figure 109 , the entire program has been removed from the display.

Redisplaying Excluded Lines

Use the F (First), H (Hidden), L (Last), and S (Show) line commands to redisplay lines excluded from the display.

The F and S line commands redisplay a specified number of lines in a block of excluded lines, starting from the top of the block. The H or L line command redisplay the last lines of a block of excluded lines.

For example, the command F2 in the line command area of the shadow line to display the first two lines of an excluded block. The command L5 in the line command area of the shadow line to display the last five lines of an excluded block.

Displaying All Excluded Lines

To redisplay all excluded lines

- 1 Select View ► Reset and press Enter to display the View - Reset Request pop-up.
- 2 Type / in the Excluded lines field and press Enter as shown in [Figure 110](#).

Or

Type RES X in the primary command area and press Enter to redisplay all excluded lines.

Figure 110 • View - Reset Request Pop-up

```
View - Reset Request
To reset display features, select an Option. Then press Enter.
Reset Options
/ All (Excluded lines, Highlighted lines, Tags, Messages)
- All Lines
- Excluded Lines
- Highlighted Lines
- Tag Comments
- Messages
- Labels
- Keep Windows
- Zoom Windows
- Pseudo Code (Including Breaks and Whens)
- Breaks
- Whens
```

Finding Program Information Using the Search Function

The search function commands are used to locate statements that meet specified criteria. These commands can be used to display and highlight specific program information, datanames, set of lines or patterns.

Use the Search pop-ups or the FINDXTND primary command to invoke the search function to find one or all occurrences of the specified target. Statements that meet the criteria are displayed, tagged and highlighted.

This function searches the program in source code sequence for occurrences of the specified target. A target is the object of a search and can be a set of lines, a dataname, or a pattern. Several target types and operands are available.

Use of the FINDXTND command results in all lines containing the specified target being tagged and highlighted with the cursor positioned on the first target. If lines containing targets have been excluded from the screen, they are redisplayed. Targets highlighted as a result of a previous command are reset, so only the results of the current FINDXTND command are highlighted. Reference tags are placed on the source code lines in columns 73 through 80, specifying the type of target found.

The FINDXTND primary command may be abbreviated as FINDX or FX.

Finding All Input and Output Statements

To remove all statements from the display and redisplay, tag, and highlight all statements containing Input and Output verbs using the program VIAMERGE

- 1 Type X;FX IO in the primary command input area and press Enter. See [Figure 113 on page 197](#) for an illustration of the result.

Or

Select Options ► Modes to display the Options - Modes screen.

- 2 Scroll down to the XMODE option and set to ON, as shown in [Figure 111](#).

Figure 111 • XMODE Option

```

Options - Modes
Command ==> _____ Scroll ==> CSR
CD05A384

-----
Option  Set  Description
-----
OPERANDS  OFF  Display values for the data items on the current statement
OUTLINE   ON   Display outline around items on the Program View screen
PROMPT    OFF  Prompt to save Environment profile
PSEUDO    ON   The PSEUDO code facility is enabled
REFRESH   ON   Refresh full ISPF screen after each STEP or RUN command
REGISTERS OFF  Display the general registers in the status box
SCALE     OFF  Display scale line above data values
SCRIPT    OFF  The SCRIPT facility is disabled
SHADOW    ON   Display a dashed line for excluded lines
STATUS    ON   Display the current status box on screens
TRACK     0   Number of execution tracking entries (0-9999 or 0K-512K)
VALUES    AUTO Number of lines for displaying data values, or AUTO
WHENS     ON   The WHEN condition facility is enabled
WRAP      OFF  Break when tracking table full
XMODE     ON   Exclude all lines before executing Primary Commands
ZEROFILL  OFF  Display numeric data items with leading zeros
***** BOTTOM OF DATA *****

```

- 3 Press PF3 to exit and return to the SmartTest main screen.

Note:

You can also use the SET XMODE ON primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

To search for all statements containing Input and Output verbs

- 1 Select Search ▶ Subset and press Enter to display the Search - COBOL Subset Name pop-up, shown in [Figure 112](#).

Figure 112 • 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 _____

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

```

- 2 Complete these fields:
 - a Type IO in the Subset name field.
 - b Specify All in the Direction field.
 - c Make certain the Options field is blank.
 - d Specify Find in the Action field and press Enter.

Figure 113 shows the results of a COBOL subset search.

Figure 113 • Program View with Input and Output Search Result

```

File View Test Search List Options Help
----- (A)
Program View                26 STATEMENTS FOUND
Command ==>                Scroll ==> CSR
-----
-- 177 LINES NOT DISPLAYED
000178 OPEN INPUT INFILE1 INFILE2 INFILE3. (B) IO
000179 OPEN OUTPUT OUTFILE OUTRPT. IO
-- 1 LINE NOT DISPLAYED
000181 ACCEPT BEGIN-DATE FROM DATE. IO
000182 WRITE OUTRPT-REC FROM BEGIN-MESSAGE. IO
-- 51 LINES NOT DISPLAYED
000234 WRITE OUTRPT-REC FROM END-MESSAGE. IO
000235 CLOSE INFILE1 INFILE2 INFILE3 OUTFILE OUTRPT. IO
-- 136 LINES NOT DISPLAYED
000372 WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA. IO
-- 3 LINES NOT DISPLAYED
000376 WRITE OUTFILE-REC FROM OUTFILE-WORK-AREA. IO
-- 5 LINES NOT DISPLAYED
000382 WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA. IO
-- 3 LINES NOT DISPLAYED
000386 WRITE OUTFILE-REC FROM OUTFILE-WORK-AREA. IO
-- 5 LINES NOT DISPLAYED
000392 WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA. IO

```

This table describes the messages displayed on the Program View screen:

Message	Description
(A) 26 STATEMENTS FOUND	Reflects 26 Input/Output statements were found in the program.
(B) IO	Each Input/Output statement is highlighted and tagged as IO.

With a single search all Input and Output verbs such as: OPEN, READ, WRITE, CLOSE, ACCEPT, and DISPLAY statements are highlighted with all non-search related line excluded.

- 3 To redisplay all excluded lines, type `RES X` in the primary command area and press Enter.

Note: _____

For the CUA method of redisplaying all excluded lines, see ["Displaying All Excluded Lines" on page 194](#).

Determining References to a Data Field

To display, tag, and highlight all statements that reference a data field or its aliases

Note: _____

This example uses the program VIAMERGE.

- 1 Type `X ; FX HOW-MANY-FILES-READ` in the primary command input area and press Enter.

Or

Select Options ► Modes to display the Options - Modes screen. Scroll down to the XMODE option and set to ON, as shown in [Figure 111 on page 196](#). Press PF3 to exit and return to the SmartTest main screen.

Note: _____

You can also use the `SET XMODE ON` primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

- 2 Select Search on the action bar and press Enter.

- 3 Select Search ▶ Data and press Enter to display the Search - Data Name pop-up as shown in [Figure 114](#). (See "[Determining References to a Data Field](#)" on page 198.)

Figure 114 • 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  HOW-MANY-FILES-READ

References      Indirect impact      Size change
1  1. All        2  1. None              levels . . . 1__
   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

```

- 4 Complete these fields:
- a Type HOW-MANY-FILES-READ in the Data name field.
 - b Specify All in the References field.
 - c Specify None in the Indirect Impact field.
 - d Make certain the Size change levels blank.
 - e Specify All in the Direction field.
 - f Make certain the Options field blank.

- g Specify Find in the Action field and press Enter to display the result as shown in Figure 115. (See ["Determining References to a Data Field" on page 198.](#))

Figure 115 • Program View with Data Item Information Search Result

```

File View Test Search List Options Help
-----
                                Program View                                VIAMERGE.VIAMERGE
Command ==>                                Scroll ==> CSR
ASG0443I 10 DATA REFS: 5 DEFS, 3 USES, 2 MODS, FOUND FOR HOW-MANY-FILES-READ. (A)
-----
000187      MOVE ZEROS TO      COMPARISON-KEY-1, COMPARISON-KEY-2,
000188      INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,
000189      MASTER-EOF-SWITCH,
000190      HASH-TEST-A, HASH-TEST-B, HASH-TEST-C,
000191      HASH-TOTAL,
000192      COMPARISON-CODES,
000193      HOW-MANY-FILES-READ.                                DATA MOD
-----
                                                    22 LINES NOT DISPLAYED
000216      COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF          DATA MOD
000217      + INFILE2-EOF
000218      + INFILE3-EOF.
000219      IF READ-3-FILES                                DATA USE
-----
                                                    2 LINES NOT DISPLAYED
000222      ELSE IF READ-2-FILES                            DATA USE
-----
                                                    2 LINES NOT DISPLAYED
000225      ELSE IF READ-1-FILE                            DATA USE
-----
                                                    243 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

(A) ASG0443I 2 DATA MODS FOUND. The long message reflects the number of references and a breakdown of the type of references to the data field in which they were found.

These DATA modes apply to statements:

- Statements tagged with DATA MOD may change the value of the data field indicated.
- Statement tagged with DATA USE are an Alias of the specified dataname.
- Statements tagged with DATA REF are uses and modifications of the dataname specified.
- Statements tagged with DATA DEF are definitions of the specified dataname or an alias.

Determining Where a Data Field is Modified

To display, tag, and highlight statements that change the value of a data item

Note: _____

The example uses the program VIAMERGE.

- 1 Type `X;FX HOW-MANY-FILES-READ MOD` in the primary command input area and press Enter.

Or

Select Options ► Modes to display the Options - Modes screen. Scroll down to the XMODE option and set to ON, as shown in [Figure 111 on page 196](#). Press PF3 to exit and return to the SmartTest main screen.

Note: _____

You can also use the SET XMODE ON primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

- 2 Select Search ► Data and press Enter to display the Search - Data Name pop-up as shown in [Figure 114](#). (See ["Determining Where a Data Field is Modified" on page 201](#).)
- 3 Complete these fields:
 - a Type `HOW-MANY-FILES-READ` in the Data name field.
 - b Specify `MODS` in the References field.
 - c Specify `NONE` in the Indirect impact field.
 - d Make certain the Size change levels is blank.
 - e Specify `ALL` in the Direction field.
 - f Make certain the Options field is blank.

- g Specify Find in the Action field and press Enter to display the result shown in Figure 116. (See ["Determining Where a Data Field is Modified" on page 201.](#))

Figure 116 • Program View with Search Result

```

File View Test Search List Options Help
-----
Command ==>
Program View
VIAMERGE.VIAMERGE
Scroll ==> CSR
ASG0443I 2 DATA MODS FOUND FOR HOW-MANY-FILES-READ. (A)
-----
000187 MOVE ZEROS TO COMPARISON-KEY-1, COMPARISON-KEY-2, 186 LINES NOT DISPLAYED
000188 INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,
000189 MASTER-EOF-SWITCH,
000190 HASH-TEST-A, HASH-TEST-B, HASH-TEST-C,
000191 HASH-TOTAL,
000192 COMPARISON-CODES,
(B)193 HOW-MANY-FILES-READ. DATA MOD
-----
(B)216 COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF DATA MOD
000217 + INFILE2-EOF DATA MOD
000218 + INFILE3-EOF.
-----
***** 250 LINES NOT DISPLAYED *****
***** BOTTOM OF DATA *****

```

This table describes the messages displayed on the Program View screen:

Message	Description
(A) ASG0443I 2 DATA MODS FOUND FOR	Reflects the number of statements found.
(B) DATA MOD	Statements that modify the value of the specified data item.

Determining if a Data Field is Used in Conditional Logic

To display, tag, and highlight conditional statements containing a specified data item

Note: _____

The example uses the program VIAMERGE.

- 1 Type `X;FX HOW-MANY-FILES-READ IN COND` in the primary command input area and press Enter.

Or

Select Options ► Modes to display the Options - Modes screen. Scroll down to the XMODE option and set to ON, as shown in [Figure 111 on page 196](#). Press PF3 to exit and return to the SmartTest main screen.

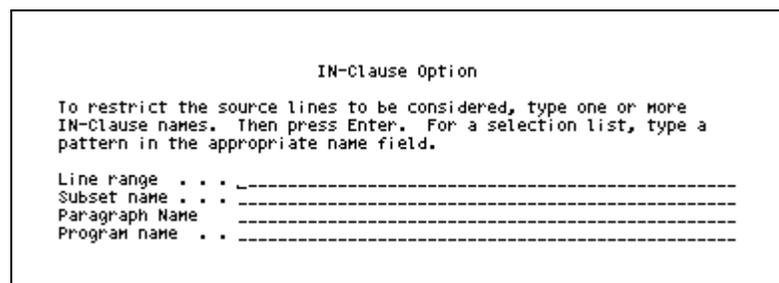
Note: _____

You can also use the SET XMODE ON primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

- 2 Select Search on the action bar and press Enter.
- 3 Select Search ► Data and press Enter to display the Search - Data Name pop-up as shown in [Figure 114 on page 199](#). (See ["Determining if a Data Field is Used in Conditional Logic" on page 203](#).)
- 4 Complete these fields:
 - a Type `HOW-MANY-FILES-READ` in the Data name field.
 - b Specify `All` in the References field.
 - c Specify `None` in the Indirect impact field.
 - d Make certain the Size change levels field blank.
 - e Specify `All` in the Direction field.
 - f Select IN-clause by typing `/` or any non-blank character in the Options field.

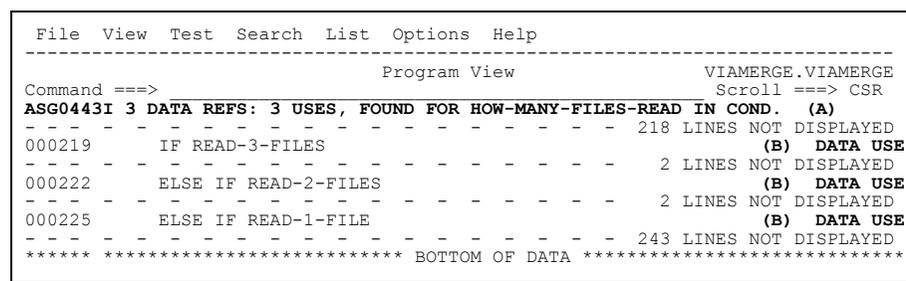
- g Specify Find in the Action field and press Enter to display the IN-Clause Option pop-up. (See ["Determining if a Data Field is Used in Conditional Logic" on page 203.](#))

Figure 117 • IN-Clause Option Pop-up



- 5 Type COND in the Subset field as shown in [Figure 117](#) and press Enter. [Figure 118](#) shows the result of a COBOL dataname search for dataname modifications. (See ["Determining if a Data Field is Used in Conditional Logic" on page 203.](#))

Figure 118 • Program View with Search Result



This table describes the messages displayed on the Program View screen:

Message	Description
(A) ASG0443I 3 DATA REFS: ...	Reflects the number of references found.
(B) DATA USE	Conditional statements that use Aliases of the specified data item.

Determining the Impact of a Data Field Size Change

To display, tag, and highlight the first level of data items affected by a change in the size of a specified data field

Note:

The example uses the program VIAMERGE.

- 1 Type `X;FX HOW-MANY-FILES-READ SIZE LEVEL 1` in the primary command input area and press Enter.

Or

Select Options ► Modes to display the Options - Modes screen. Scroll down to the XMODE option and set to ON, as shown in [Figure 111 on page 196](#). Press PF3 to exit and return to the SmartTest main screen.

You can also use the SET XMODE ON primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

- 2 Select Search on the action bar and press Enter.
- 3 Select Search ► Data and press Enter to display the Search - Data Name pop-up as shown in [Figure 119](#). (See ["Determining the Impact of a Data Field Size Change" on page 205](#).)
- 4 On the Search - Data Name pop-up:
 - a Type `HOW-MANY-FILES-READ` in the Data name field.
 - b Select All in the References field.
 - c Select Of size change in the Indirect impact field.
 - d Type 1 in the Size change levels field.
 - e Select All in the Direction field.
 - f Leave the Options field blank.
 - g Select Find in the Action field and press Enter to display the results shown in [Figure 120](#). (See ["Determining the Impact of a Data Field Size Change" on page 205](#).)

Figure 119 shows the Search - Data Name pop-up for determining the impact of a change in size in the field HOW-MANY-FILES-READ.

Figure 119 • Search Data Name Pop-up

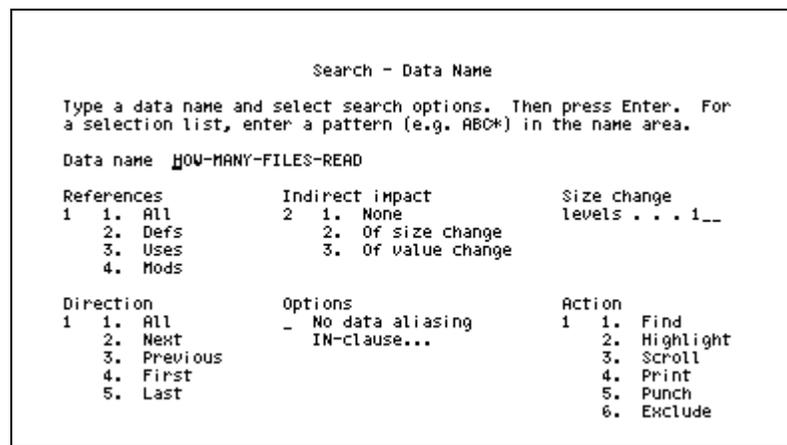
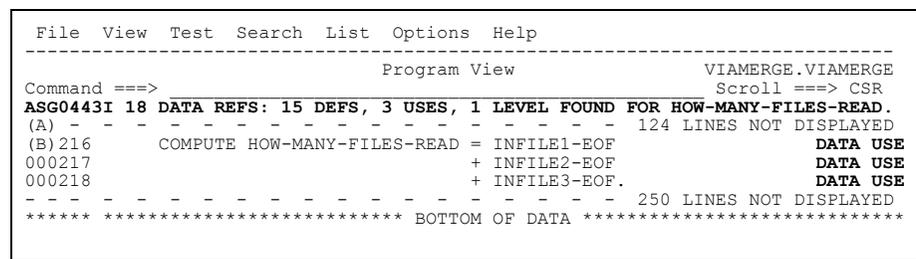


Figure 120 shows the result of a search for the impact of a size change in the field HOW-MANY-FILES-READ.

Figure 120 • Program View Showing the Effect of a Change in Size



This table describes the messages displayed on the Program View screen:

Message	Description
(A) ASG0443I 18 DATA REFS: ...	Reflects the number and type of references found.
(B) DATA USE	Indicates that the data fields INFILE1-EOF, INFLE2-EOF, and INFILE3-EOF are affected by changing the size of the data field, HOW-MANY-FILE-READ.

Determining the Impact of a Data Field Value Change

To display, tag, and highlight all statements affected by a change in the value of a specified data field

Note:

The example uses the program VIAMERGE.

- 1 Type `X;FX HOW-MANY-FILES-READ MOD VALUE` in the primary command input area and press Enter.

Or

Select Options ► Modes to display the Options - Modes screen. Scroll down to the XMODE option and set to ON, as shown in [Figure 111 on page 196](#). Press PF3 to exit and return to the SmartTest main screen.

You can also use the SET XMODE ON primary command. This command causes SmartTest to exclude all lines that do not meet the criteria of the search to be performed. For the CUA method of excluding all lines from the display, see ["Excluding Lines from the Display" on page 192](#).

- 2 Select Search on the action bar and press Enter.
- 3 Select Search ► Data and press Enter to display the Search - Data Name pop-up shown in [Figure 120 on page 206](#).
- 4 Complete these fields:
 - a Type `HOW-MANY-FILES-READ` in the Data name field.
 - b Select Mod in the References field.
 - c Select Of value change in the Indirect impact field.
 - d Leave the Size change levels field blank.
 - e Select All in the Direction field.
 - f Leave the Options field blank.
 - g Select Find in the Action field and press Enter to display the result. (See ["Determining the Impact of a Data Field Value Change" on page 207](#).)

[Figure 121](#) shows the Search - Data Name pop-up for determining the impact of a change in the value of the data item HOW-MANY-FILES-READ.

Figure 121 • 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  HOW-MANY-FILES-READ

References          Indirect impact          Size change
1  1. All           2  1. None                levels . . . 1__
   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
   5. Last
   5. Last          3. Scroll
   5. Last          4. Print
   5. Last          5. Punch
   5. Last          6. Exclude

```

[Figure 122](#) shows the result of a search to determine the impact of a change in the value of the data item HOW-MANY-FILES-READ.

Figure 122 • Program View Showing Modification of a Data Item

```

File View Test Search List Options Help
-----
Program View                                VIAMERGE.VIAMERGE
Command ==>                                Scroll ==> CSR
ASG0443I 8 DATA MODS FOUND FOR HOW-MANY-FILES-READ. (A)
000188      INFILE1-EOF, INFILE2-EOF, INFILE3-EOF,      DATA MOD
000189      MASTER-EOF-SWITCH,
000190      HASH-TEST-A, HASH-TEST-B, HASH-TEST-C,
000191      HASH-TOTAL,
000192      COMPARISON-CODES,
000193      HOW-MANY-FILES-READ.                          DATA MOD
-----
22 LINES NOT DISPLAYED
000216      COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF      DATA MOD
000217      + INFILE2-EOF
000218      + INFILE3-EOF.
-----
220 LINES NOT DISPLAYED
(B) 439      MOVE 1 TO INFILE1-EOF                          DATA MOD
-----
11 LINES NOT DISPLAYED
000451      MOVE 1 TO INFILE2-EOF                          DATA MOD
-----
11 LINES NOT DISPLAYED
000463      MOVE 1 TO INFILE3-EOF                          DATA MOD
-----
5 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

```

This table describes the messages displayed on the Program View screen:

Message	Description
(A) ASG0443I 8 DATA MODS FOUND FOR HOW-MANY-FILES-READ	Reflects the number of statements found.
(B) DATA MOD	Statements modifying the specified data field or fields used to value the data field (see the previous COMPUTE statement).

Highlighting Search Results

Select Highlight on the Search pop-ups or use the HIGH primary command to search the program in source code sequence and display, tag and highlight all occurrences of the specified target. Previously highlighted lines remain highlighted.

A target is the object of a search function and can be a set of lines, a dataname, or a pattern. Several target types and operands are available. (See "[Command Targets](#)" on [page 19](#).)

The HIGH function tags and highlights all lines containing the specified target. The cursor is positioned on the first target. If lines containing targets have been excluded from the screen, they are redisplayed.

The HIGH primary command may be abbreviated as HI. For example, this command displays, highlights, and tags all COBOL Assignment statements:

```
HI ASSIGN
```

Note: _____

For more information on the HIGH command, see the *ASG-SmartTest Reference Guide*.

Printing Program Information

Select Print on the Search pop-up or use the LPRINT primary command to copy one or all occurrences of the specified target to the List File. To print the List File use the Option - Log/List/Punch Definition pop-up.

The LPRINT primary command may be abbreviated as LP.

As an example, the command LP PERFORM copies all PERFORM statements in the program to the List File for subsequent printing.

The command LPRINT * primary command copies the current virtual screen (all displayed lines that can be viewed by scrolling forward and backward) to the List File. Excluded line messages are copied to the List file as they appear on the screen at the time the LPRINT * command is entered.

The LPRINT * primary command can be entered on these screens to copy those screens to the List File:

- BackTrack Variable History Screen
- Breakpoints List screen
- Execution Counts screen
- Execution Tracking screen
- List - CALL Statements pop-up
- List - COBOL Subsets Names pop-up
- List - Equates pop-up
- List - Perform Range Names pop-up
- List - Program/Subprogram Names pop-up
- List - User Marks pop-up
- Pseudo Code List screen
- View - Paragraph Cross Reference
- When Conditions List screen

Note: _____

For more information regarding this command, see the *ASG-SmartTest Reference Guide*.

Repositioning the Display

Select Search ► Scroll or use the SCROLL primary command to position the display to the first line containing the specified target. The most frequent use of the SCROLL command is to position the display to the next occurrence of a highlighted statement.

You can abbreviate the SCROLL command as SC.

As an example, the command SC HI NEXT positions the display to the next occurrence of a highlighted statement.

Note:

For more information regarding the COBOL intelligent Search Function and these commands, see the Commands section in the *ASG-SmartTest Reference Guide*.

Following Branching Logic

Select Search ► Branch or use the BRANCH primary command to position the display at the specified target paragraph. This command can be used to scroll from a statement such as a PERFORM, to the paragraph being PERFORMed. Use the BRANCH BACKUP command to redisplay the statement from which a BRANCH was issued.

If the target paragraph is already on the screen, the cursor is placed on the first line of the paragraph. If the target paragraph is not on the screen, the specified paragraph is scrolled to the top of the screen.

The BRANCH function is cursor sensitive. If the cursor is positioned in excluded lines, the first line of the excluded block is selected.

The BRANCH function can be used to track branching logic several levels deep into PERFORMed code and then return to each PERFORM statement.

The screen position is saved for use with the BACKUP operand if a LABEL name is entered on the command line or the cursor is placed on a GOTO or PERFORM statement.

The PROCEDURE DIVISION label displays when the BRANCH command is entered with no operands and the cursor is positioned in a part of the program other than the PROCEDURE DIVISION, or when the BRANCH PROC command is entered.

By default, PF10/PF22 is assigned as the BRANCH command. PF11/PF23 is the default for the BRANCH BACKUP command.

The BRANCH primary command may be abbreviated as B or BRA.

Using the Branch Function

To position the display to the first statement in a specified paragraph

Note:

This example uses the paragraph 2000-PROCESSING-LOOP in the demonstration program VIAMERGE.

- 1 Type BRANCH 2000-PROCESSING-LOOP in the primary command input area and press Enter to display the result shown in [Figure 123](#). (See ["Using the Branch Function" on page 212](#).)

Or

Select Search ► Branch and press Enter to display the Search - Branch Request pop-up shown in [Figure 122](#). (See ["Using the Branch Function" on page 212](#).)

- 2 Select Branch to target in the Option field.
- 3 Type 2000-PROCESSING-LOOP in the Target name field.
- 4 Select Label name in the Target type field and press Enter. (See ["Using the Branch Function" on page 212](#).)

[Figure 123](#) shows the Search - Branch Request pop-up for branching through program logic.

Figure 123 • Search - Branch Request Pop-up

```
Search - Branch Request

To branch to another area of the program, select the Option desired.
For Option 1, type the branch location (Target) information. Then
press Enter. For a name selection list (for Target type 2, 3 or 4),
type a pattern (e.g. ABC*) in the name field.

Option
1  1. Branch to target
   2. Return to previous "Branch to target" location
   3. Branch to transfer(s) of control to cursor position

Target name -----

Target type
1  1. None - use cursor
   2. Label name
   3. Perfrange name
   4. Program name
```

[Figure 124](#) shows the result of a Branch function.

Figure 124 • Program View with Cursor Positioned by Branch Function

```

File View Test Search List Options Help
-----
Program View                                CURSOR POSITIONED (A)
Command ==> _____ Scroll ==> CSR

(B) 204 2000-PROCESSING-LOOP.
000205     IF NOT-END-INFILE1 AND READ-INFILE1
000206         PERFORM 3100-READ-INFILE1 THRU
000207             3100-READ-INFILE1-X.
000208     IF NOT-END-INFILE2 AND READ-INFILE2
000209         PERFORM 3200-READ-INFILE2 THRU
000210             3200-READ-INFILE2-X.
000211     IF NOT-END-INFILE3 AND READ-INFILE3
000212         PERFORM 3300-READ-INFILE3 THRU
000213             3300-READ-INFILE3-X.
000214     MOVE SPACES TO OUTFILE-WORK-AREA,
000215         OUTRPT-WORK-AREA.
000216     COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF
000217         + INFILE2-EOF
000218         + INFILE3-EOF.
000219     IF READ-3-FILES
000220         PERFORM 2100-COMPLEX-MERGE THRU
000221             2100-COMPLEX-MERGE-X
000222     ELSE IF READ-2-FILES
FALLTHRU

```

(A) CURSOR POSITIONED. Message area indicates the cursor has been positioned on the specified label.

(B) 2000-PROCESSING-LOOP. The target paragraph is scrolled to the top of the screen. If the specified paragraph was already on the screen, the cursor would be moved to the paragraph and the display would not be scrolled.

Using the **BRANCH** Command

*To indicate the label to be the target of the **BRANCH** function*

- 1 To position the display on paragraph 3100-READ-INFILE1 using the cursor location, type **BRANCH** in the command area, place the cursor on the statement containing the paragraph name **PERFORM 3100-READ-INFILE1**, and press Enter.

If you use the standard SmartTest PF Key settings, you can also place the cursor on the statement containing the label desired and press PF10.

[Figure 125](#) shows the result of a Branch to the paragraph indicated by cursor placement.

Figure 125 • Program View with Cursor Positioned by Branch Function Key

```

File View Test Search List Options Help
-----
Program View
Command ==> _____
(CURSOR POSITIONED)
Scroll ==> CSR

(B)431 3100-READ-INFILE1.
000432     MOVE ZERO TO READ-INFILE1-SWITCH.
000433     ADD 1 TO END-FILE-COUNT.
000434     READ INFILE1 INTO INFILE1-WORK-REC
000435     AT END                                     FALLTHRU
000436         MOVE INFILE1-EOF-MSG TO OUTRPT-WORK-DATA
000437         MOVE ZERO TO OUTRPT-WORK-KEY
000438         WRITE OUTRPT-REC FROM OUTRPT-WORK-AREA
000439         MOVE 1 TO INFILE1-EOF
000440         MOVE EOF-KEY TO INFILE1-WORK-KEY.     FALLTHRU
000441 3100-READ-INFILE1-X.
000442     EXIT.                                     RETURN
000443 3200-READ-INFILE2.
000444     MOVE ZERO TO READ-INFILE2-SWITCH.
000445     ADD 1 TO END-FILE-COUNT.
000446     READ INFILE2 INTO INFILE2-WORK-REC
000447     AT END                                     FALLTHRU
000448         MOVE INFILE2-EOF-MSG TO OUTRPT-WORK-DATA
000449         MOVE ZERO TO OUTRPT-WORK-KEY

```

(A) CURSOR POSITIONED. Message area indicates the cursor has been positioned on the specified label.

(B) 3100-READ-INFILE1. The target paragraph is scrolled to the top of the display (because it was not already on the display).

- To return the display to the paragraph where the last BRANCH command was issued, type BRANCH BACKUP in the command input area and press Enter. The Program View screen, shown in [Figure 126](#) displays. If you use the standard SmartTest PF Key settings, you can also press PF11 to execute a BRANCH BACKUP command.

Figure 126 • Program View with Cursor Positioned by Branch Function

```

File View Test Search List Options Help
-----
Program View
Command ==> _____
VIAMERGE.VIAMERGE
Scroll ==> CSR

000204 2000-PROCESSING-LOOP.
000205     IF NOT-END-INFILE1 AND READ-INFILE1
(A)206         PERFORM 3100-READ-INFILE1 THRU
000207             3100-READ-INFILE1-X.
000208     IF NOT-END-INFILE2 AND READ-INFILE2
000209         PERFORM 3200-READ-INFILE2 THRU
000210             3200-READ-INFILE2-X.
000211     IF NOT-END-INFILE3 AND READ-INFILE3
000212         PERFORM 3300-READ-INFILE3 THRU
000213             3300-READ-INFILE3-X.
000214     MOVE SPACES TO OUTFILE-WORK-AREA,
000215             OUTRPT-WORK-AREA.
000216     COMPUTE HOW-MANY-FILES-READ = INFILE1-EOF
000217             + INFILE2-EOF
000218             + INFILE3-EOF.
000219     IF READ-3-FILES
000220         PERFORM 2100-COMPLEX-MERGE THRU
000221             2100-COMPLEX-MERGE-X
000222     ELSE IF READ-2-FILES
FALLTHRU

```

(A) 2000-PROCESSING-LOOP. The previous BRANCH target paragraph is scrolled to the top of the display.

Note: _____

For more information regarding branching, see the BRANCH command in the Commands chapter of the *ASG-SmartTest Reference Guide*.

Searching the Program in Execution Sequence

This section describes how to use the FLOW primary command to display, tag, and highlight specific targets in the execution sequence of the program. The FLOW command requires that the program be analyzed using the Extended Analysis option. The program must be the active program.

Note: _____

If Insight is installed with SmartTest, the Flow function is available as an action on the Logic pull-down.

By default, the cursor location is used as the starting point for the search. When the cursor is in the command input area, the search starts with the source line at the top of the display. If the cursor is positioned on a non-executable COBOL statement, the starting point of the search is the first executable statement physically prior to the cursor position.

The starting point for the FLOW command may be specified using the FROM operand. The FLOW command accepts datanames as targets. The operand MOD indicates a search for statements that modify the specified dataname. The operand PREV indicates the search is performed for data item modifications in reverse execution sequence.

The FLOW command locates the first target on each path in the program.

The FLOW primary command may be abbreviated as FL.

For example, if the program VIAMERGE is suspended at an 0C7abend, you want to search the program in reverse execution sequence and displaying the statements where the specified data field may have been previously modified.

To begin the search, type FLOW END-FILE-COUNT MOD PREV in the primary command area and press Enter to display the result shown in [Figure 127](#).

Note:

The result indicates that multiple statements modifying the data field are reachable. This means that there are multiple execution paths in the program that access the field.

Figure 127 • Highlighted Statements in Reverse Execution Sequence

```

File View Test Search List Options Help
----- (A)
Command ==> Program View 4 MOD(S) REACHABLE
Scroll ==> CSR

(B)180 MOVE 'VMERGE ' TO BEGIN-PROGRAM-NAME. DATA MOD
----- 252 LINES NOT DISPLAYED
>>>>> ADD 1 TO END-FILE-COUNT. DATA MOD
|-----+
| 10 END-FILE-COUNT PIC 9(8) ADDR 000C8450 |
| VALUE > VMERGE < * INVALID NUMERIC * |
|-----+
----- 11 LINES NOT DISPLAYED
000445 ADD 1 TO END-FILE-COUNT. DATA MOD
----- 11 LINES NOT DISPLAYED
000457 ADD 1 TO END-FILE-COUNT. DATA MOD
----- 11 LINES NOT DISPLAYED
***** ***** BOTTOM OF DATA *****

+-----+
|STATUS: DATA EXCEPTION (0C7) PROGRAM: VIAMERGE DATE: DDDMMYYYY |
| STMT: 000433 OFF: 0028D2 AMODE: 24 MODULE: VIAMERGE TIME: HH:MM:SS |
|SOURCE: ADD 1 TO END-FILE-COUNT. |
+-----+

```

(A) 4 MODS REACHABLE. The short message reflects the number of lines containing the target that are reachable from the Start Point.

(B) MOVE 'VMERGE' TO BEGIN-PROGRAM-NAME. The lines that modify the specified data item are highlighted and tagged.

Note:

For more information regarding execution sequence processing, see the FLOW command in the Commands section in the *ASG-SmartTest Reference Guide*.

6

Additional Testing Features

This chapter describes additional testing features and contains these sections:

Topic	Page
Capturing and Replaying Command Sequences	217
Locating the Next Executable Statement (LOCATE * Command)	220
Simplifying Commands	221
Other Primary Commands	223
Other Line Commands	224
Commands Available Only with Insight	224

Capturing and Replaying Command Sequences

SmartTest provides a Script facility to capture and replay primary command sequences. The captured commands are saved in a script file that you can execute later to replay the recorded primary commands.

SmartTest assigns a dataname for the script dataset and displays it in the long message area when you issue the SET SCRIPT ON primary command. The Script facility allocates and opens a dataset to record all SmartTest primary commands as you enter them. To stop recording the primary commands and close the script file, issue the SET SCRIPT OFF primary command. Note the dataset name for future reference; you must specify this dataset name when you request script file playback.

Script files are sequential datasets or members of partitioned datasets. Script files can be used to initialize a session, set default values, set up a session, execute or re-execute a session, or execute a predefined command sequence. The name of the dataset displays when the dataset is opened or closed. This name is in the format:

TSOUSERID.STTnnnnn.VIASCRIP

where *nnnnn* is a number beginning with 00001 that is incremented by 1 for each new script file. After the dataset is closed, you can use it as an input dataset name with the EXECUTE command.

Note: _____

If a TSO prefix is not set to the user ID, the prefix is added to the beginning of the dataset name.

Comments can be included in script files by entering an asterisk (*) in the first column of the command by the comment text.

The next section describes how to execute the script file.

Note: _____

Functions executed using CUA and line commands are not recorded in the script file.

For information on the primary commands, see the *ASG-SmartTest Reference Guide*.

Replaying a Script File

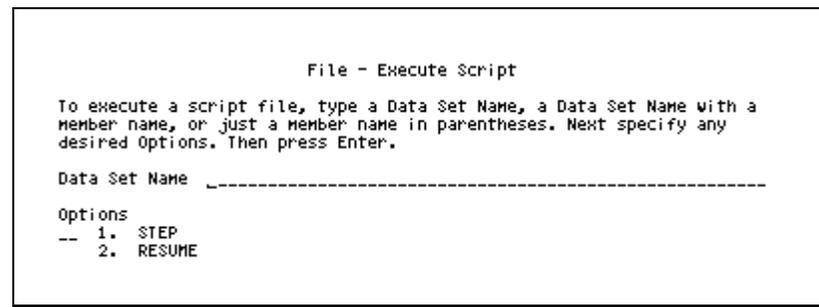
Use the Execute facility to replay a primary command sequence stored in a script file.

The Execute facility reads and executes a script file. Each script file can contain EXECUTE commands that execute lower level (nested) script files. Script files that create loops are recognized, and an error message displays.

To execute a script file using the CUA method

- 1 Select File ► Execute and press Enter. The File - Execute Script pop-up, shown in [Figure 128](#), displays.

Figure 128 • File - Execute Script



- 2 Type the name of the script file in the Data Set Name field and press Enter:
- 3 Select the STEP option to invoke the debug option that allows you to step through each primary command in the script file.

The primary commands are displayed in the primary command input area, where you can change or erase each displayed command. You must press Enter to execute the displayed primary command. Continue processing of the script file in this manner until all primary commands have been displayed and executed.

While stepping through the script file, select the RESUME option to execute the remaining script file primary commands without stepping through each one.

To execute a script file using the command method

- 1 Type EXECUTE (or EXEC) and the name of the script file dataset and press Enter.
For example:

```
TSOUSERID.STTnnnnn.VIASCRIP
```

Note: _____

The STEP operand allows step-by-step execution of each script file, one command at a time. When you use the STEP operand, each command displays in the command input area.

- 2 Press Enter to execute the displayed primary command. You can also modify or erase any displayed command.

While in STEP script execution mode, you can issue an EXECUTE command with the RESUME operand to execute the remaining script commands without stepping through them.

For more information about the EXECUTE primary command, see the *ASG-SmartTest Reference Guide*.

Locating the Next Executable Statement (LOCATE * Command)

Use the LOCATE * (or L *) primary command during a test session to position the display at the next statement to be executed. The next executable statement is denoted by the chevrons (>>>>>>).

During a test session, chevrons are placed in the line number area of the Program View screen to indicate the next statement to be executed. If the display has been scrolled and the chevrons are not visible, the LOCATE * command positions the display to the statement containing the chevrons.

You can use this command only from the Program View screen during an active test session.

Simplifying Commands

This section describes how to reduce the number of keystrokes needed to process commands. You can use cursor substitution, or the EQUATE, & (RETAIN), or RECALL commands to reduce the keystrokes you type.

Using the Cursor Substitution Character

When you are using the command syntax, use the Cursor Substitution character while in the Program View screen to bypass typing datanames and paragraph names as command targets.

Search Function commands allow you to type the Cursor Substitution character in place of a dataname or paragraph name. You place the cursor in the displayed source code on the name to be used as the target. When the command is processed, the substitution character is replaced with the name at the cursor location.

By default, the Cursor Substitution character is an % (percent sign). For example, typing `FX % MOD`, moving the cursor to `END-FILE-COUNT`, and pressing Enter finds all statements that modify this data item.

Note: _____

You can substitute multiple tokens (i.e., qualified datanames such as `MYDATA OF MYSTRUCTURE`) by specifying multiple cursor substitution characters, each separated by a blank. When substituting multiple tokens, the tokens must be contiguous in the source code.

Creating Short Names for Character Strings (EQUATE Command)

Use the EQUATE (or % primary command) to define a substitute name for any character string.

Equated names are used in the same manner as any dataname is used. They can be saved in the AKR, and may be viewed on the Equates List by typing `LIST EQUATES`.

To delete an equate, follow this step:

- ▶ Type EQUATE using the equate name without any other operands.

This command may be abbreviated as EQ, for example:

```
EQ FILES HOW-MANY-FILES-READ
```

assigns the substitution short name FILES to the long field name HOW-MANY-FILES-READ. Each time you need to reference the long field name use the short name.

Keep Commands in the Command Input Area (& Command)

Use the & (retain) primary command in conjunction with any other primary command to reuse the command specified.

The & command is useful when the same primary command is to be executed repeatedly, or if minor changes to a command are desired. It removes the necessity of re-entering the entire command. For example:

```
& FX END-FILE-COUNT USE
```

executes a FINDXTND command targeting field uses and leaves the command displayed in the command input area.

You can type over the USE with MOD to execute a FINDXTND command for the same field targeting field modifications.

Recall Primary Commands and Messages (RECALL Command)

Use the RECALL primary command to redisplay previously issued primary commands or long messages or pop-ups.

The last 20 primary commands and the last 20 long messages, are stacked in a recall buffer. Any commands or messages in the buffer may be displayed using the RECALL command. After a command displays in the command input area, it can be executed with or without modification.

Use the MESSAGE operand to redisplay messages instead of primary commands. The NEXT operand can be used to move forward rather than backward through the stack.

When the RECALL command has passed through the entire command or message stack, further RECALL commands process the stack again. After you type a RECALL command with operands, subsequent RECALL commands with no operand use the last operand entered until you specify a different operand or another command is executed.

PF12/PF24 is the default installation for the RECALL command. You can abbreviate this command REC.

For example, this command displays the last message in the long message area:

```
REC MSG
```

This command redisplay the last pop-up:

```
REC POPUP
```

These commands are not stacked for use by the RECALL command:

ANALYZE	KEYS	PRODLVL
RESET	RIGHT	UPDATE
DOWN	LEFT	RECALL
RETURN	RSCROLL	END
PLOG	REDO	RFIND
HELP	PLIST	REPEAT
RIGHT	UP	

Displaying Product Release and Level Numbers (PRODLVL Command)

Use the PRODLVL primary command to display the current SmartTest product level.

The PRODLVL command displays the product name, operating system, product release number, and release level on the message line, in this format:

```
ASG1554I ASG-SMARTTEST-OS(390) Rn.n AT Lnnn, ASG-Center Rn.n AT Lnnn,
```

where:

Rn.n is the release number.

Lnnn is the release level.

This information is requested if you need to contact the ASG Service Desk for assistance.

Other Primary Commands

DISPLAY	FIND	LPUNCH
MERGE	PREF	REDO
REFRESH	REPEAT	RFIND
RHIGH	RPREF	RSCROLL
SAVE	WHERE	ZOOMIN
ZOOMOUT	KEEP	RESET

Other Line Commands

.label

H (highlight)

ZA (zoom Assembler)

Commands Available Only with Insight

COPY

DELETE

MARK

RENAME

RTRACE

SELECT

TRACE

FLOW

EQUATE

7

Analyze

This chapter describes how to set up a SmartTest test session for the TSO execution environment and contains these sections:

Topic	Page
Analyzing a COBOL Program	225
The Analyze Process	227
Using the File - Analyze Submit Pop-up	228
Using ISPF	231
Using ISPF/PDF Edit	232
Automatic JCL Modifications	237
Analyze Summary Report	244
Adding Analyze Facilities to a Standard Compile Mechanism	246
Assembler Analyzer	248
Analyze Options	252

Analyzing a COBOL Program

A program must be analyzed before ESW products can provide intelligent information about it. The analyze process gathers information about the program, including program relationships, logic, data and execution paths, and stores this information in the Application Knowledge Repository (AKR). After the analyze information is placed in the AKR, it is available to ESW products in online and batch environments, where it is accessed to provide valuable information about the design and operation of user systems.

You can test a program using SmartTest if it has not been analyzed, however, SmartTest is not able to show the COBOL source statements or display on datanames. The program displays in disassembled object with SET ASMVIEW ON. You can set breakpoints and view memory areas.

The analyze process is similar to a COBOL compile. The process has these three primary inputs:

- Source COBOL program (including copybooks)
- JCL used to compile and link the COBOL program
- Options and features that tailor the analyze steps

Analyze Input Descriptions

Input 1 - COBOL Source Program

The analyze process requires these basic program standards:

- The COBOL language as specified in the *COBOL II Language Reference* manual is accepted by the analyze job. It correctly processes any program that can be compiled without warnings or errors by the IBM COBOL II compilers.
- COBOL II and later programs that receive error (E), severe (S), or unrecoverable (U) messages from the IBM compiler cannot be successfully analyzed.

These are the Program Analyzer resource estimates to process COBOL programs of various sizes:

Version:	ASG-Center Release 6.0
CPU Type:	3090-600 running OS/390
Disk Type:	3390
Analyze Parms:	AKR_Buffer_Mask=4096
Compiler Parms:	BUF=256 KB,SIZ=1024 KB

Input 2 - Compile/Link JCL

The compile and link JCL should be the complete JCL used to compile the program. Specifically, the JCL should contain steps to fetch the source from the source manager (such as Librarian or Panvalet), execute the preprocessor, invoke the compiler with the appropriate options and COPY libraries, then invoke the linkage editor.

The SmartTest analyze forces some COBOL compiler options to certain values so that sufficient data is available for the use of all product features. For a description of the COBOL compiler options used by the SmartTest analyze, see ["COBOL Compiler Options" on page 289](#).

Input 3 - Analyze Options and Features

The analyze features indicate the type of analysis to be performed.

- An Encore analysis provides the information required for code extraction and execution flow capabilities.
- An Insight analysis provides logic and execution flow capabilities.
- A SmartTest analysis provides the testing and debugging information required by SmartTest.
- A SmartTest Extended analysis provides an Insight analysis.
- A SmartDoc analysis provides the information required for SmartDoc reports.
- An Extended SmartDoc analysis provides data flow analysis.
- A Recap analysis provides the inventory analysis data required for Recap reports.
- An Alliance analysis provides the entity-relationship data required for impact assessment.

Default options for the analyze process are established at installation time. Options that are to be overridden are specified when submitting the analyze job.

The Analyze Process

The analyze process consists of setting up and executing a batch job. There are three methods used to invoke the analyze process. The method used depends primarily on the environment from which the analyze process is invoked, but may depend on the access method containing the compile/link JCL. These are the three methods used to invoke the analyze process:

Method	Description
Analyze Option or Command	Select File ► Analyze or type ANALYZE from any screen, to display the File - Analyze Submit screen. Enter the required input and output information, then submit the job.

Method	Description
ISPF	<p>From any ISPF screen, execute the VIASUBDS CLIST. This CLIST is executed by typing this command:</p> <pre>TSO VIASUBDS dsn parms</pre> <p>where:</p> <p><i>dsn</i> is a PDS member or sequential dataset containing the compile JCL</p> <p><i>parms</i> represents any of the available execution parameters described in the table in "Using ISPF/PDF Edit" on page 232.</p>
ISPF/PDF Edit	<p>Execute the VIASUB PDS edit macro. This edit macro is executed by entering <code>VIASUB parms</code>, where <i>parms</i> represents any of the available execution parameters described in the table in "Using ISPF/PDF Edit" on page 232.</p>

Using the File - Analyze Submit Pop-up

To begin an analyze, follow this step:

- ▶ Select File ▶ Analyze or type ANALYZE on any screen. The File - Analyze Submit pop-up displays as shown in [Figure 129](#).

Figure 129 • File - Analyze Submit Pop-up

```

                                     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)
    
```

These are the default analyze values on the Analyze Submit screen:

Default	Analyze Option
Y	Insight
Y	SmartTest
N	SmartDoc
N	Encore

Options

Option	Description
E - Edit JCL	<p>Displays the compile/analyze JCL to review or change the JCL, if necessary. When the E option is selected, the JCL to be edited is generated from the JCL member specified in the Data Set Name field, applying the rules outlined in the Automatic JCL Modifications section. The generated JCL is then displayed on the Edit screen.</p> <p>When editing is complete, type <code>ISPF SUBMIT</code> to submit the edited JCL for execution. Optionally, the edited JCL can be saved in a partitioned dataset by using the <code>CREATE</code> command. Otherwise, any changes made at this time are not saved.</p>
S - Submit JCL	<p>Submits the JCL to compile/analyze the specified program. The JCL submitted is generated from the JCL member specified in the Data Set Name field, applying the rules outlined in the Automatic JCL Modifications section.</p>
D - SmartDoc Options	<p>Displays only if SmartDoc is installed. Type <code>D</code> to display the File - SmartDoc Options pop-up that is used to request an Extended SmartDoc analysis and to specify which reports (if any) are to be generated.</p>

Fields

Field	Description
Data Set Name	Specifies the PDS member or sequential dataset containing the JCL to compile and link the program. If the JCL resides in a source manager such as Librarian or Panvalet, use the VIASUB edit macro to submit the compile/analyze job.
Analyze Features	An analyze feature product field is listed for each ESW product installed.
SmartTest	Displays only if SmartTest is installed. Y indicates that a SmartTest compile/analysis is to be performed. This type of analysis provides the testing and debugging information required by SmartTest. If SmartTest is the only product installed, this field contains Y and cannot be changed. The default is Y.
Extended Analysis	Displays only if SmartTest is installed. This type of analysis provides comprehensive program analyzing capabilities in addition to the testing and debugging of SmartTest. The default is N.
AKR Data Set Name	Specifies the AKR that is to contain the information for the analyzed program.
AKR Program Name	Specifies an alias name used by the analyze process to save its results in the AKR. If a value is not entered in this field, the analyze job uses the program name from the PROGRAM-ID statement in the COBOL source as the name under which to save results in the AKR.
	<p>Note: _____</p> <p>This field is only used for the AKR program name and does not change the COBOL program name in the source.</p> <p>_____</p>
Analyze Options	Specifies the analyze options that are to be overridden. Default options for the analyze job are established at installation time. Analyze options that can be entered in this field are described in the "Input 3 - Analyze Options and Features" on page 227 .

Field	Description
Compile?	Indicates whether the program is to be compiled. A program need not be compiled if Insight, Encore, or SmartDoc are the only features specified. The compile step can be suppressed by specifying N in this field. This field is forced to a value of Y if SmartTest and/or Extended analysis is selected.
Link load module reusable	Tests a program under SmartTest that is dynamically loaded. It is necessary to mark the load module as reusable so that the breakpoints are retained across calls. The default is Y. If the JCL used for an analyze includes a linkedit step, the REUS parameter is inserted.

Using ISPF

From any ISPF screen, the VIASUBDS CLIST can be used to submit the analyze job. This is the syntax for VIASUBDS:

```
TSO VIASUBDS input.jcl.dsn parms
```

where:

Parameter Value	Description
<i>input.jcl.dsn</i>	Specifies the dataset containing the compile/link JCL. This must be a sequential dataset or a member of a PDS.
<i>parms</i>	Specifies one or more parameter that controls the operation of VIASUBDS. Typically, the PANEL parameter is entered to display the Analyze Submit Parameters screen for entry of any necessary parameters. The parameters are saved in the ISPF profile and used as defaults for the next analyze submission. The table in "Using ISPF/PDF Edit" on page 232 contains a list of these parameters, with default parameters underlined.

Note:

Using the VIASUBDS CLIST requires the ESW CLIST library to be available through the standard SYSPROC allocations.

Using ISPF/PDF Edit

From the ISPF/PDF Edit screen, the VIASUB edit macro can be used to submit the analyze job. This is the syntax for VIASUB:

```
VIASUB parms
```

where *parms* is one or more parameter that controls the operation of VIASUB. Typically, the PANEL parameter is entered to display the Analyze Submit Parameters screen for entry of any necessary parameters. The parameters are saved in the ISPF profile and used as defaults for the next analyze submission.

Note:

Using the VIASUB edit macro requires the ESW CLIST library to be available through the standard SYSPROC allocations.

This table contains a list of these parameters, with default parameters underlined:

Parameter	Description
AKR(<u>xxxxxx</u>)	Indicates the AKR where the results of the analyze job will be placed. The specified name must conform to the standard TSO dataset naming conventions.
AOPT(<u>xxxxxx</u>)	Specifies the options to be supplied to the analyze job. The COBOLII option is automatically added if the compiler specified in the input JCL is COBOL II. When specifying more than one analyze option, the options should be separated by commas and enclosed in single quotes; for example: <pre>AOPT ('XMEM, RECUR, SUBSYS=D239')</pre> <p>See "Analyze Options" on page 252, for information on each analyze option.</p>
CMPL NOCMPL	CMPL indicates a COBOL compile and an analysis is to be executed by the new JCL. NOCMPL indicates the new JCL is to bypass the compile step and only execute the analyze job. When NOCMPL is specified, a return code of 1000 (decimal) greater than the analyze return code is produced. This causes the subsequent job steps (e.g., a link edit) to be bypassed based on a successful compilation. NOCMPL cannot be specified if a SmartTest analysis is being executed.

Parameter	Description
DSCHK NODSCHK	DSCHK indicates datasets needed by the resulting JCL are to be verified to ensure they exist. Specifically, the AKR and the load library containing VIASMNTR are checked. When NODSCHK is specified, the AKR and the load library need not exist at the time VIASUB or VIASUBDS is executed. NODSCHK is useful when the JCL is being prepared for submission on another system, or for delayed execution when an AKR does not yet exist. Note that the cataloged procedure libraries must exist and be accessible to VIASUBDS or VIASUB.
EDIT	EDIT specifies that the resulting JCL is not to be submitted for batch processing. The PDF editor is invoked for the resulting JCL. Any desired changes can be made and then the JCL can be submitted by typing SUBMIT. You must enter EDIT each time it is needed. Note that the edits made to the JCL are not saved. The CREATE command must be used to save the modified JCL elsewhere. The EDIT option is ignored if the Analyze Submit Parameters screen displays. In this case, the E command must be entered to edit the JCL.
ENS NOENS	ENS specifies that an Encore analysis is to be performed.
INS NOINS	INS specifies that an Insight analysis is to be performed.
OUTPUT(XXXXXX)	Specifies that the resulting JCL is not to be submitted for batch processing. The JCL is written to the specified dataset. The specified name must conform to the standard TSO dataset naming conventions. A dataset is created if it does not already exist. OUTPUT must be entered each time it is needed.
PANEL NOPANEL	PANEL indicates that the Analyze Submit Parameters screen is to be displayed for entry of parameters for the analyze job. The Analyze Submit Parameters screen displays even if a valid AKR name is specified as a parameter, or can be obtained from the ISPF profile when PANEL is specified.
PGM(XXXXXX)	Specifies a name to be used when storing the program in the AKR. This name overrides the program name in the PROGRAM-ID paragraph.
PROONLY	Indicates the JCL contains only a cataloged procedure rather than a complete job. PROONLY suppresses the generation of the VIAIN DD statement. PROONLY must be entered each time it is needed.

Parameter	Description
REUS NOREUS	REUS specifies that when the program is tested using SmartTest, it is dynamically loaded and tested with RUN NOMONITOR.
SD NOSD	SD specifies that a SmartDoc analysis is to be performed.
SDR NOSDR	SDR specifies that SmartDoc reports will be run.
SDX NOSDX	SDX specifies that a SmartDoc Extended analysis is to be performed.
ST NOST	ST specifies that a SmartTest analysis is to be performed.
STX NOSTX	STX specifies that a SmartTest Extended analysis is to be performed. When the INS and ST parameters are specified, a SmartTest Extended analysis is automatically performed.

Analyze Submit Parameters Screen

Figure 130 shows the Analyze Submit Parameters screen. This screen displays when the PANEL parameter is specified when executing VIASUBDS or VIASUB, or when the NOPANEL option is used and an error condition is detected. You can also access this screen by selecting File ► Compile/Analyze.

Figure 130 • Analyze Submit Parameters 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

```

These are the default analyze options:

Default	Analyze Option
N	Extended Analysis
Y	SmartTest

Options

Option	Description
E - Edit JCL	<p>Displays the compile/analyze JCL for edit. When you select option E, the JCL to be edited is generated from the JCL specified when the VIASUBDS CLIST or VIASUB edit macro was invoked, applying the rules outlined in the Automatic JCL Modifications section of this topic. The generated JCL is then displayed on the Edit screen.</p> <p>When editing is complete, type <code>ISPF SUBMIT</code> to submit the edited JCL for execution. Optionally, the edited JCL can be saved in a partitioned dataset by using the <code>ISPF CREATE</code> command. Otherwise, any changes made at this time are not saved.</p>
S - Submit JCL	<p>Submits the JCL to compile/analyze the specified program. The JCL submitted is generated from the JCL specified when the VIASUBDS CLIST or VIASUB edit macro was invoked, applying the rules outlined in the Automatic JCL Modifications section of this topic.</p>
D - SmartDoc Options	<p>Displays only if SmartDoc is installed. Select option D to display the SmartDoc Options screen that is used to request an Extended SmartDoc analysis and to specify which reports (if any) are to be generated.</p>

Fields

Field	Description
Analyze features	<p>An analyze feature product field is listed for each ESW product installed.</p>
SmartTest	<p>Displays only if SmartTest is installed. Y indicates that a SmartTest compile/analysis is to be performed. This type of analysis provides the testing and debugging information required by SmartTest. If SmartTest is the only product installed, this field contains YES and cannot be changed. The default is Y. This field is optional.</p>

Field	Description
Extended Analysis	Displays only if SmartTest is installed. This type of analysis provides comprehensive program analyzing capabilities in addition to the testing and debugging of SmartTest. The default is N.
AKR dataset name	Specifies the dataset name of the AKR that is to contain the information for the analyzed program. This field is required.
AKR program name	Specifies the alias name to be used by the analyze job to save its results in the AKR. If a value is not entered in this field, the analyze job uses the program name from the PROGRAM-ID statement in the COBOL source as the name under which to save results in the AKR. Note: _____ This field is only used for the AKR program name and does not change the COBOL program name in the source. _____
Analyze options	Specifies analyze options that are to be overridden. Default options for the analyze job are established at installation time. Analyze options that can be entered in this field are described in the Analyze Options topic.
Compile?	Indicates whether to perform a compile. A program need not be compiled if Insight, Encore, or SmartDoc are the only features specified. The compile step can be suppressed by typing N in this field. This field is forced to a value of Y if SmartTest is selected.
Link load module reusable?	Tests a program under SmartTest that is dynamically loaded. It is necessary to mark the load module as reusable so that the breakpoints are retained across calls. The default is Y. If the JCL used for an analyze includes a linkedit step, the REUS parameter is inserted.
Display this panel by default in the future?	Determines whether the ISPF profile is updated to display this screen whenever subsequent executions of VIASUBDS or VIASUB are invoked. When this field contains N, this screen is not displayed on subsequent executions of VIASUBDS or VIASUB unless an error condition is encountered.

Automatic JCL Modifications

The analysis process automatically modifies the JCL based on the specified parameters and analyze options. If problems arise, this procedure can be used as a checklist to perform the analyze process manually until the problem can be determined and resolved.

To make changes to the JCL, the compile procedure, or a copy of the compile procedure

- 1 Replace the PGM= parameter in the compile step(s) as shown:

PGM= parameter: Replace with:

PGM=IGYCRCTL	PGM=VIACOBII
PGM=CPXUPTSM	PGM=VIAOPT3
PGM=CAOTSMON	PGM=VIAOPTII
PGM=CAOMSON	PGM=VIAOPTII

- 2 Add DD statements to the compile step(s) for these datasets (VIAUT2 DD statement is Encore only):

```
//VIALOG DD SYSOUT=*
//VIAMRPT DD SYSOUT=*
//VIAPRINT DD SYSOUT=*
//VIAAKR DD DSN=[specified AKR name],DISP=SHR
```

- 3 If the SYSIN DD statement contains FREE=CLOSE, change it to FREE=END.
- 4 Ensure that the ESW load libraries are available to the modified step by adding a //STEPLIB DD statement specifying the ESW load libraries, or by concatenating these libraries to an existing STEPLIB DD.
- 5 Ensure that the JOB and the modified STEP EXEC statements have a minimum of REGION=4096K.
- 6 Add a VIAIN DD statement that designates the features and options to be used during analysis. This is the format for this statement:

```
//VIAIN DD *
* ANALYZE FEATURES:
ST,STX
/*
```

You can also modify the COBOL parameter string to include the appropriate ESW parameter by using this command:

```
V Parm=(vopt,vopt,vopt...)
```

where *vopt* can be:

INS	Specifies an Insight only analysis (no COBOL compile).
ST	Specifies a SmartTest only analysis (no Extended analysis).
STX	Specifies a SmartTest Extended analysis.
SD	Specifies a SmartDoc analysis.
SDX	Specifies a SmartDoc Extended analysis.
SDR	Specifies a SmartDoc report generation.
ENS	Specifies an Encore analysis (no COBOL compile).
[<i>analyze parms</i>]	Specifies valid analyze parameters (using the standard analyze options)
CMPL	Specifies a COBOL compile (forces a COBOL compile and an analysis to be executed by the JCL).
NOCMPL	Suppresses the COBOL compile (JCL bypasses the compile and executes only an analyze job).
(NO)SYSPRINT	Creates separate compiler output file.
(NO)VIADCOMP	Creates SmartDoc intermediate compiler output file. The intermediate compiler output file is used to produce the SmartDoc Compiler Output.
D Parm	Specifies SmartDoc run-time parameters.

Note: _____

If you do not specify a feature (i.e., INS, ST, SD, SDX, SDR, or ENS), all processing is suppressed. This means the procedure executes a compile as it did before.

[Figure 131](#) is an example of compile JCL with Panvalet as it might appear in a dataset at your site.

Figure 131 • Compile JCL with Panvalet

```
// ASG JOB (ASG), 'PANVALET COMPILE'
/*ROUTE PRINT DEST
/** PANVALET EXTRACT
/**
//PANEXT EXEC PGM=PAN#1, REGION=256K
//PANDD1 DD DSN=COBOL.PANLIB, DISP=SHR
//PANDD2 DD DSN=&&COBIN, UNIT=SYSDA, SPACE=(CYL, (1, 1)),
// DISP=(NEW, PASS), DCB=(RECFM=FB, LRECL=80, BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
++WRITE WORK, VIASDDMO
/*
/**
/** COBOL COMPILE
/**
//COBCOMP EXEC PGM=IKFCBL00, REGION=1024K, COND=(8, LT, PANEXT),
// PARM='SIZE=512K, BUF=128K, LANGLVL(2), LIB, DYNAM'
//STEPLIB DD DSN=SYS1.COBOLII.COMPIILER, DISP=SHR
//SYSIN DD DSN=&&COBIN, DISP=(OLD, DELETE)
//SYSLIB DD DSN=COBOL.COPYLIB, DISP=SHR
//SYSLIN DD DSN=&&LINKIN, UNIT=SYSDA, SPACE=(CYL, (1, 1)),
// DISP=(NEW, PASS), DCB=(RECFM=FB, LRECL=80, BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=121, BLKSIZE=1573)
//SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
//SYSUT2 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
//SYSUT3 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
//SYSUT4 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
//SYSUT5 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
/**
/** LINK EDIT
/**
//LINKED EXEC PGM=IEWL, REGION=1024K, COND=(8, LT, COBCOMP)
//SYSLIB DD DSN=SYS1.COBOLII.COBLIB, DISP=SHR
//SYSLMOD DD DSN=USER.LOADLIB, DISP=OLD
//SYSPRINT DD SYSOUT=*, DCB=(RECFM=FBA, LRECL=121, BLKSIZE=1573)
//SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (1, 1))
//SYSLIN DD DSN=&&LINKIN, DISP=(OLD, DELETE)
// DD *
NAME VIASDDMO(R)
/*
```

Figure 132 is an example of the compile JCL, as it would appear after the Panvalet/compile/analyze JCL has been generated according to the rules in this section. Statements that have been added or modified are tagged to the right with ASG NEW and ASG MOD.

Figure 132 • Generated JCL after the Panvalet/Compile/Analyze

```

//ASG JOB ( ),'VIASANLZ'
//*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
//*****
//* THIS JCL HAS BEEN MODIFIED BY THE ASG ANALYZE      *
//* SUBMIT FACILITY, WHICH CONVERTS COMPILE JCL INTO *
//* COMPILE AND ANALYZE JCL. NEW OR MODIFIED LINES  *
//* CONTAIN 'VIA' IN COLUMNS 74 THROUGH 76.         *
//*****
//VIAIN EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT2 DD DSN=&&VIAIN,DISP=(,PASS),UNIT=SYSDA,
//          SPACE=(TRK,(1,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=7440)
//SYSUT1 DD *
* ANALYZE FEATURES:
  ST,STX
* ANALYZE OPTIONS:
  NORET=(ABENDPGM),SEQ
/*
//* PANVALET EXTRACT
//*
//PANEXT EXEC PGM=VIASPAN1,REGION=4096K
//PANDD1 DD DSN=COBOL.PANLIB,DISP=SHR
//PANDD2 DD DSN=&&COBIN,UNIT=SYSDA,SPACE=(CYL,(1,1)),
//          DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
++WRITE WORK,VIASDDMO
/*
//STEPLIB DD DSN=ASG.VIACENxx.LOADLIB,
//          DISP=SHR,DCB=BLKSIZE=19069
//          DD DSN=DB2TEST.DSNLOAD,DISP=SHR
//VIAINCLS DD DSN=&&VIAINCLS,DISP=(MOD,PASS),
//          UNIT=SYSDA,SPACE=(CYL,(1,1))
//VIALOG DD SYSOUT=*
//VIAMRPT DD SYSOUT=*
//VIAPRINT DD SYSOUT=*
//*
//* COBOL COMPILE
//*
//COBCOMP EXEC PGM=VIACOBVS,REGION=4096K,COND=(8,LT,PANEXT),
//          PARM='SIZE=512K,BUF=128K,LANGLVL(2),LIB,DYNAM'
//STEPLIB DD DSN=SYS1.COBOLII.COMPIER,DISP=SHR
//          DD DSN=ASG.VIACENxx.LOADLIB,DISP=SHR
//          DD DSN=DB2.DSNLOAD,DISP=SHR
//SYSIN DD DSN=&&COBIN,DISP=(OLD,DELETE)
//SYSLIB DD DSN=USER.COPYLIB,DISP=SHR
//SYSLIN DD DSN=&&LINKIN,UNIT=SYSDA,SPACE=(CYL,(1,1)),
//          DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)

```

```

//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//VIAINCLS DD DSN=&&VIAINCLS,DISP=(MOD,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(1,1))
//VIALOG DD SYSOUT=*
//VIAMRPT DD SYSOUT=*
//VIAPRINT DD SYSOUT=*
//VIAAKR DD DSN=ASG.VIACENxx.AKR,DISP=SHR
//VIAIN DD DSN=&&VIAIN,DISP=(OLD,PASS)
//*
//* LINK EDIT
//*
//LINKED EXEC PGM=IEWL,REGION=1024K,COND=(8,LT,COBCOMP)
//SYSLIB DD DSN=SYS1.COBOLII.COBLIB,DISP=SHR
//SYSLMOD DD DSN=USER.LOADLIB,DISP=OLD
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1573)
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIN DD DSN=&&LINKIN,DISP=(OLD,DELETE)
// DD *
NAME VIASDDMO(R)
/*

```

[Figure 133](#) is an example of compile JCL with CICS as it might appear in a dataset at your site.

Figure 133 • Compile JCL with CICS

```

//ASG JOB (ASG),'CICS COBOL '
//*
//* *****
//* * DFHEITCL PROC INVOCATION *
//* * *
//* * STANDARD CICS COBOL COMMAND LEVEL PROCEDURE FOR *
//* * TRANSLATING, COMPILING AND LINK EDITING SOURCE. *
//* * *****
//*
//COBOLC EXEC DFHEITCL,
// PARM.LKED='LET'
//TRN.SYSIN DD DSN=ASG.VIACENxx.CNTL(VIACDEMO),DISP=SHR
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
NAME VIACDEMO(R)
/*

```

Figure 134 is an example of the compile JCL as it would appear after the CICS/compile/analyze JCL has been generated according to the rules in this section. Statements that have been added or modified are tagged to the right with ASG NEW and ASG MOD. Similar additions and modifications are made when DB2 and CA-IDMS precompilers are used.

Figure 134 • Generated JCL after CICS/Compile/Analyze

```

//ASG JOB ( ),'VIASANLZ'
//*   INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
//*****
//* THIS JCL HAS BEEN MODIFIED BY THE ASG ANALYZE      *
//* SUBMIT FACILITY, WHICH CONVERTS COMPILE JCL INTO *
//* COMPILE AND ANALYZE JCL. NEW OR MODIFIED LINES  *
//* CONTAIN 'VIA' IN COLUMNS 74 THROUGH 76.        *
//*****
//VIAIN EXEC PGM=IEBGENER
//SYSIN DD DUMMY
//SYSPRINT DD DUMMY
//SYSUT2 DD DSN=&VIAIN,DISP=(,PASS),UNIT=SYSDA,
//          SPACE=(TRK,(1,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=7440)
//SYSUT1 DD *
* ANALYZE FEATURES:
  ST,STX
//*
//*          *****
//*          * DFHEITCL PROC INVOCATION *
//*          * *
//*          * STANDARD CICS COBOL COMMAND LEVEL PROCEDURE FOR *
//*          * TRANSLATING, COMPILING AND LINK EDITING SOURCE. *
//*          *****
//*
//DFHEITCL PROC SUFFIX=1$,
// INDEX='CICS',
// INDEX2='CICS',
//   OUTC=A,
//   REG=4096K,
//   LNKPARM='XREF',
//   WORK=SYSDA,VIAPGMA=VIACICS
//*
//TRN EXEC PGM=&VIAPGMA,
//   REGION=&REG
//STEPLIB DD DSN=&INDEX2..LOADLIB,DISP=SHR,DCB=BLKSIZE=32760
//   DD DSN=ASG.VIACENxx.LOADLIB,DISP=SHR
//   DD DSN=DB2.DSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC
//SYSPUNCH DD DSN=&&SYSCIN,
//   DISP=(,PASS),UNIT=&WORK,
//   DCB=BLKSIZE=400,
//   SPACE=(400,(400,100))
//VIATIN DD DSN=&VIATIN,DISP=(MOD,PASS),SPACE=(CYL,(1,1))
//   UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7440)
//VIAPRINT DD SYSOUT=*
//VIAPGM DD DSN=&&DFHECP&SUFFIX,DISP=(NEW,DELETE),
//   UNIT=SYSDA,SPACE=(TRK,(1,1))

```

```

//COB      EXEC PGM=VIACOBVS,REGION=&REG,                                ASG MOD
//          PARM='NOTRUNC,NODYNAM,LIB,SIZE=256K,BUF=16K,APOST,DMAP,XREF'
//SYSLIB DD DSN=&INDEX..COBLIB,DISP=SHR
//          DD DSN=SYS1.COBCOMP,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//SYSLIN DD DSN=&&LOADSET,DISP=(MOD,PASS),
//          UNIT=&WORK,SPACE=(80,(250,100))
//SYSUT1 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT2 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT3 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT4 DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5 DD UNIT=&WORK,SPACE=(460,(350,100))
//STEPLIB DD DSN=ASG.VIACENxx.LOADLIB,                                ASG NEW
//          DISP=SHR,DCB=BLKSIZE=19069                                ASG NEW
//          DD DSN=DB2.DSNLOAD,DISP=SHR                                ASG NEW
//VIALOG DD SYSOUT=*                                                    ASG NEW
//VIAMRPT DD SYSOUT=*                                                    ASG NEW
//VIAPRINT DD SYSOUT=*                                                  ASG NEW
//VIAAKR DD DSN=ASG.VIACENxx.AKR,DISP=SHR                               ASG NEW
//VIAIN DD DSN=&&VIAIN,DISP=(OLD,PASS)                                   ASG NEW
//COPYLINK EXEC PGM=IEBGENER,COND=(7,LT,COB)
//SYSUT1 DD DSN=&INDEX..COBLIB(DFHEILIC),DISP=SHR
//SYSUT2 DD DSN=&&COPYLINK,DISP=(NEW,PASS),
// DCB=(LRECL=80,BLKSIZE=400,RECFM=FB),
// UNIT=&WORK,SPACE=(400,(20,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN DD DUMMY
//LKED EXEC PGM=IEWL,REGION=&REG,
//          PARM=&LNKPARM,COND=(5,LT,COB)
//SYSLIB DD DSN=&INDEX2..LOADLIB,DISP=SHR
//          DD DSN=SYS1.COBLIB,DISP=SHR
//SYSLMOD DD DSN=&INDEX2..LOADLIB,DISP=SHR
//SYSUT1 DD UNIT=&WORK,DCB=BLKSIZE=1024,
//          SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSLIN DD DSN=&&COPYLINK,DISP=(OLD,DELETE)
//          DD DSN=&&LOADSET,DISP(OLD,DELETE)
//          DD DDNAME=SYSIN
//          PEND                                                        ASG NEW
//COBOLC EXEC DFHEITCL,
//          PARM.LKED='LET'
//TRN.SYSIN DD DSN=ASG.VIACENxx.CNTL(VIACDEMO),DISP=SHR
//COB.VIATIN DD DSN=&&VIATIN,DISP=(MOD,DELETE),SPACE=(CYL,(1,1)),      ASG NEW
//          UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7440)          ASG NEW
//LKED.SYSLMOD DD DSN=USER.LOADLIB,DISP=SHR
//LKED.SYSIN DD *
//NAME VIACDEMO(R)
/*

```

Analyze Summary Report

Information about the analyzed program is placed in the AKR when the analyze job completes. A summary report of the run-time statistics and diagnostic messages is also produced. This report varies depending on whether the SOURCE or NOSOURCE option was specified when the analyze job was submitted.

[Figure 135](#) shows the Analyze Summary Report with the SOURCE option used.

Figure 135 • Analyze Summary Report with the Source Option

```

(A)
00001 000100 IDENTIFICATION DIVISION.
00002 000200 PROGRAM-ID. VIASDDMO.
00003 000300 AUTHOR. WRITTEN BY ASG IN LANG LEVEL 2.
00004 000400*
00005 000500 ENVIRONMENT DIVISION.
00006 000600 INPUT-OUTPUT SECTION.
00007 000700 FILE-CONTROL.
00008 000800 SELECT INFILE1 ASSIGN TO UT-S-INFILE1.
00009 000900 SELECT INFILE2 ASSIGN TO UT-S-INFILE2.
00010 001000 SELECT INFILE3 ASSIGN TO UT-S-INFILE3.

*STATISTICS* SOURCE RECORDS = 466 DATA DIVISION STATEMENTS = 120 PROCEDURE
DIVISION STATEMENTS = 220
*OPTIONS IN EFFECT* SIZE = 1048576, BUF = 262144, LINECNT = 54, SPACE1, FLAGW, SEQ
*OPTIONS IN EFFECT* SOURCE, DMAP, PMAP, NOCLIST, SUPMAP, NOXREF, NOSXREF, LOAD, NODECK
*OPTIONS IN EFFECT* APOST, NOTRUNC, NOFLOW, NOTERM, NONUM, NOBATCH, NONAME, COMPILE=0
*OPTIONS IN EFFECT* NOSTATE, RESIDENT, DYNAM, LIB, NOSYNTAX, NOOPTIMIZE, NOSYMDMP
*OPTIONS IN EFFECT* NOTEST, VERB, ZWB, SYST, NOENDJOB, NOMIGR, NOLVL, DUMP, NOADV
*OPTIONS IN EFFECT* NOLST, NOFDECK, NOCDECK, LCOL1, L120, NOFDECK, NOCDECK, LCOL1
*OPTIONS IN EFFECT* L120, DUMP, NOADV, NOPRINT, NOCOUNT, NOVBSUM, NOVBREF, LANGLVL(1)

(B)
ASG1534I PROGRAM VIASCOPR STARTED
ASG1519I PROGRAM VIASCOPR COMPLETED WITH RETURN CODE 0000
ASG1534I PROGRAM IKFCBL00 STARTED
ASG1519I PROGRAM IKFCBL00 COMPLETED WITH RETURN CODE 0000
ASG1534I PROGRAM VIASSYMB STARTED
ASG1519I PROGRAM VIASSYMB COMPLETED WITH RETURN CODE 0000
ASG1534I PROGRAM VIASANLZ STARTED
ASG1519I PROGRAM VIASANLZ COMPLETED WITH RETURN CODE 0000
ASG1025I THE PRODUCT LEVEL FOR ASG-CENTER-OS (XA) Rx.x IS 000.
ASG1435I ASG-CENTER-OS(XA) Rx.x LVL000 -SUMMARY REPORT- PROGRAM=VIASDDMO
ASG1399I OPTIONS IN EFFECT ARE: SOURCE, NODMAP, NOPMAP.
ASG1394I SUMMARY OF COBOL II SYMBOLS EXTRACTED FROM VIASDDMO.
ASG1395I 98 DATA NAME SYMBOLS PROCESSED.
ASG1396I 33 PROCEDURE SYMBOLS PROCESSED.
ASG1397I 131 TOTAL SYMBOLS.
ASG1436 DIAGNOSTICS: 0 TOTAL - 0 WARNING, 0 ERROR, 0 SEVERE, 0 CATASTROPHE
ASG1437i ASG-CENTER-OS (XA) Rx.x LVL000 - END OF SYMBOL EXTRACTION FOR VIASDDMO

(C)
ASG CENTER-OS (XA) Rx.x LVL000 PROGRAM: VIASDDMO DDMMYYYY HH:MM:SS PAGE 1

(D)
LINE ERROR MESSAGE
ASG0237I 131 SYMBOLS PROCESSED.
ASG0238I 131 SYMBOLS MATCHED.
ASG0240I 199 VERBS PROCESSED.

(E)
DIAGNOSTICS LINES: 0 TOTAL - 0 WARNINGS, 0 CONDITIONALS, 0 ERRORS, 0 DISASTERS

(F)
SOURCE LINES: 466 TOTAL - 120 DATA DIVISION STATEMENTS, 220 PROCEDURE
DIVISION STATEMENTS

(G)
PARAMETERS PASSED: NOCOBOLII, LANGLVL(1), FEATURES=(I,S,X)

(H)
OPTIONS IN EFFECT: BUFMAXK=2000K, FEATURES=(INSIGHT, SMARTTEST,EXTENDED), FLAG(W),
LINECNT=60, NORECUR, NOSEQ, NOSOURCE, SPACE1, LANGLVL(1),

```

```

      (I)
ENTRY POINTS:   VIASDDMO
      (J)
EXTERNAL CALLS: VIASUB
      (K)
END OF PROCESSING: DDMMYYYY HH:MM:SS

```

This table describes the areas highlighted in the JCL:

Notes	Description
(A)	<p>A complete listing of the program is produced and shows statement numbers generated by the analyze job in the first six columns. These numbers are referenced in diagnostic messages. These notations can also appear on the source listing:</p> <ul style="list-style-type: none"> • C Statement was inserted with a COPY statement. • ** Original source statement number is out of sequence. • I Statement was inserted with an INSERT statement.
(B)	<p>This portion of the Analyze Summary is the report from the ESW monitor facility. The job steps that were executed by the monitor facility are listed along with the return codes produced.</p>
(C)	<p>The Center (Analyze) release and product level is shown along with the date and time the analysis was performed.</p>
(D)	<p>LINE and ERROR MESSAGE - This information is shown only if there were error conditions encountered. If so, this area lists the line number on which the error occurred and the error message.</p>
(E)	<p>DIAGNOSTICS LINES - Indicates the total number of messages issued with subtotals for each type.</p>
(F)	<p>SOURCE LINES - Indicates the number of source lines in the program. The number of statements within the DATA DIVISION and PROCEDURE DIVISION are also shown. Each level number is counted as one statement in the DATA DIVISION. Each verb is counted as one statement in the PROCEDURE DIVISION.</p>
(G)	<p>PARAMETERS PASSED - Lists all of the analyze options specified for this analyze job.</p>
(H)	<p>OPTIONS IN EFFECT - Lists all options in effect, including default and user-specified options.</p>
(I)	<p>ENTRY POINTS - Lists the entry points in this program.</p>

Notes	Description
(J)	EXTERNAL CALLS - Lists the programs that this program CALLs.
(K)	END OF PROCESSING - Lists the day, month, year, and time the analyze job completed. This date and time is also reflected in the online AKR statistics.

Adding Analyze Facilities to a Standard Compile Mechanism

CLIST Compile Mechanism

The Analyze Submit Facility can be incorporated into many existing in-house compile mechanisms. This provides users with access to ESW analysis without the complications of learning a new or different compile sequence.

Most ISPF Dialogs that use combinations of panels, CLISTs and the TSO SUBMIT command are easily adapted to invoke the Analyze Submit facility.

The first step is to add a line in a panel to query whether an ESW analysis should be performed. The compile panel can be used for this purpose. The new line in the panel definition may look like this example:

```
+   ASG ANALYZE      ==>_Z+      (%Y+OR%N+)
```

Add this line to the)INIT section of the panel definition:

```
.ZVARS = ' (VSVANLYZ) '
```

If there is already an assignment to .ZVARS, VSVANLYZ should be added to the list in the appropriate place. This is assuming the VSVANLYZ variable will be used to record the response to this question. If another variable name is to be used, the description below would need to change so it corresponds.

It may be appropriate to perform a VGET (VSVANLYZ) PROFILE before the panel displays, and a VPUT (VSVANLYZ) PROFILE after it displays.

The CLIST should then be changed to invoke VIASUBDS instead of SUBMIT, if the VSVANLYZ variable has a Y value. Assume the CLIST contains this code:

```
IF &E = Y THEN DO
    WRITE EDITED JCL WILL NOT BE SUBMITTED BY CLIST
    ISPEXEC EDIT DATASET (&ZUSER..CNTL(MEMBERX))
    END
IF &E ≠ Y THEN DO
    SUBMIT &ZUSER..CNTL(MEMBERX)
    END
```

The code should be changed to read as shown in this example:

```
IF &VSVANLYZ = Y THEN DO
    IF &E = Y THEN DO
        WRITE EDITED JCL WILL NOT BE SUBMITTED BY CLIST
        %VIASUBDS &ZUSER..CNTL(MEMBERX) EDIT
        END
    ELSE DO
        %VIASUBDS &ZUSER..CNTL(MEMBERX)
        END
    END
ELSE DO
    IF &E = Y THEN DO
        WRITE EDITED JCL WILL NOT BE SUBMITTED BY CLIST
        ISPEXEC EDIT DATASET (&ZUSER..CNTL(MEMBERX))
        END
    IF &E ≠ Y THEN DO
        SUBMIT &ZUSER..CNTL(MEMBERX)
        END
    END
END
```

Many JCL skeleton generators can have the required Analyze features imbedded easily. The JCL modifications typically done by the ESW JCL converter can be added to existing skeletons to create JCL that executes an ESW analysis. Use the Automatic JCL Modifications described in the next topic as a guide for making these additions.

ISPF Compile Mechanism

The Analyze Submit facility can also be installed to emulate the standard ISPF Compile Option (option 5), by adding this line to the appropriate ISPF menu panel:

```
V5, 'CMD(%ASGSISP5 ISRJPA) NOCHECK'
```

This option functions exactly like the ISPF Compile Option except that the resulting JCL is not submitted directly. Instead, it is passed to the Analyze Submit facility. Review the VIASISP5 CLIST for other required modifications.

If emulation of the ISPF Compile Option is not sufficient for your site, the Analyze Submit facility can also be configured to replace the ISPF option directly. Contact the ASG Service Desk for details.

Assembler Analyzer

The Assembler Analyzer gathers and stores Assembler source code and data information in the AKR.

Assembler source code can be displayed on the Program View screen. The source code displays as output by the Assembler, and can be stepped through at the Assembler instruction level. Data fields can be displayed and modified.

Assembler Analyzer Input

This input is included to the Assembler Analyzer:

- JCL to assemble and link the program
This JCL should be the complete JCL used to assemble the program. Specifically, the JCL should contain steps to fetch the source from the source manager (such as Librarian or Panvalet), invoke the Assembler with the appropriate options, then invoke the linkage editor.
- Assembler Analyzer options
Analyze options are not used by the Assembler Analyzer.
- Assembler Analyzer features
SmartTest is the only available Assembler analyze feature.

An analyze job can be executed using one of these methods:

- Using the File - Analyze Submit pop-up.
- Executing the VIASUBDS CLIST by typing `TSO VIASUBDS <dsn> <parms>`

where:

dsn is a PDS member or sequential dataset containing the Assemble JCL.

parms represents any of the available execution parameters.

- Executing the VIASUB PDF edit macro on an ISPF/PDF edit screen when editing the Assemble JCL from any source manager. This edit macro is executed by typing `VIASUB parms`, where *parms* represents any of the available execution parameters.

Using the VIASUBDS CLIST and the VIASUB edit macro requires the ESW CLIST library to be available through the standard SYSPROC allocations.

Automatic JCL Modifications

The analysis process automatically modifies the JCL based on the specified parameters. If problems arise, this procedure can be used as a checklist to perform the analyze process manually until the problem can be determined and resolved.

To make changes to the JCL, the assemble procedure, or a copy of the assemble procedure

- 1 Replace the PGM parameter in the assemble step(s) as shown:

PGM= parameter: Replace with:

```
PGM=IEV90    VIAASMH
PGM=ASMA90   VIAHLASM
```

- 2 Add DD statements to the assemble step(s) for these datasets:

```
//VIAAKR    DD DSN=[specified AKR name],DISP=SHR
//VIALOG    DD SYSOUT=*
//VIAMRPT   DD SYSOUT=*
//VIAPRINT  DD SYSOUT=*
```

- 3 If the SYSIN DD statement contains FREE=CLOSE, change it to FREE=END.
- 4 Ensure that the ESW load libraries are available to the modified step by adding a STEPLIB DD statement specifying the ESW load libraries, or by concatenating these libraries to an existing STEPLIB DD statement.
- 5 Ensure that the JOB and modified STEP EXEC statements have a minimum of REGION=4096 KB.
- 6 Add a VIAIN DD statement that designates the features and options to be used during analysis. This is the format:

```
//VIAIN     DD *
            ST
            /*
```

Assembler Analyze JCL

[Figure 136](#) shows Assembler/analyze JCL before submitting the Assembler analyze job.

Figure 136 • Assembler/Analyze JCL Before Analyze

```
//ASG JOB (ASG), 'ASG ASSEMBLER'
/*ROUTE PRINT DEST
/*
//ASM EXEC PGM=IEV90, REGION=512K, PARM='OBJ, XREF (SHORT), TEST'
//SYSIN DD DSN=ASG.VIACENxx.CNTL (VIAPASM), DISP=SHR
//SYSLIB DD DSN=SYS1.MACLIB, DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSLIN DD DSN=&&SYSLIN, UNIT=SYSDA, SPACE=(CYL, (1,1)),
// DISP=(MOD, PASS), DCB=(RECFM=FB, LRECL=80, BLKSIZE=2480)
//SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
//SYSUT2 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
//SYSUT3 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
/*
//LINK EXEC PGM=IEWL, PARM='LIST, MAP, CALL, LET', COND=(5, LT, ASM),
// REGION=300K
//SYSLIN DD DSN=&&SYSLIN, DISP=(OLD, DELETE)
// DD DDNAME=SYSIN
//SYSLMOD DD DSN=USER.LOADLIB, DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSDA, SPACE=(CYL, (1,1))
//SYSIN DD *
NAME VIAPASM(R)
/*
```

Analyze Options

The Analyze job uses many of the same options as the IBM COBOL compilers. These compile-time options are available to control the output format and to describe COBOL options. Default options for the Analyze job were established when ESW products were installed.

To override the default installation options, enter the desired options on the File - Analyze Submit pop-up. Separate the options with a comma (,).

If an invalid option is entered, it is ignored by the analyze job. If a valid option is entered more than once, the last one is processed.

Options that accept program names as parameters, with the exception of PROGRAM, accept wildcard characters. The asterisk (*) represents zero or more characters. The question mark (?) represents a single character, for example:

Option	Description
DBA*	All programs that begin with DBA and end with any number of other characters.
D?A*	All programs that begin with D followed by any one character, followed by A, then followed by any number of other characters.
DBA???	All programs that begin with DBA and end with any three characters.

The analyze options are summarized in the rest of this topic. In this summary, abbreviations are shown in uppercase (abbreviations comply with compiler standards). The Analyze Summary Report printed at the end of each analyze job lists the actual options in effect, and the options that were passed to it (the override options).

Buffers

BUF(*nnnnn*K)
BUF=*nnnnn*K

where *nnnnn* is a number from 20 to 20000.

BUF is used only as an override. The amount of main storage allocated to buffers and internal tables is dynamically allocated. Use BUF if an override is necessary. The minimum BUF value is 20K; the maximum is 20000K.

COBOL Level

COBOLMVSVM
COBOLOS390
COBOL370
COBOLII
COB2R3
NOCOBOLII

COBOL370 overrides the LANGLVL option and processes the input program as COBOL/370.

COBOLII overrides the LANGLVL option and processes the input program as COBOL II.

COB2R3 overrides the LANGLVL option and processes the input program as COBOL II Release 3.0 or Release 3.1.

NOCOBOLII overrides the LANGLVL option and processes the input program as VS COBOL.

The default is based on the compiler being used and is determined automatically by the submit process.

DB2 Load Library

DB2LIB=xxxxxxx.yyyyyy.zzzzzz

where xxxxxx.yyyyyy.zzzzzz is the dataset name of the DB2 load library.

DB2LIB specifies the load library that is used to invoke the DB2 preprocessor at your site.

DB2 Application Plan

DB2PLAN=xxxxxxx

where xxxxxxxx is the name of the ESW application plan.

DB2PLAN specifies the ESW application plan that was created at installation time by the VIASBIND job. You can use DB2PLAN to override the default plan name.

Dynamic CALLs

DYNcall
NODYNcall

The minimum abbreviation is DYN or NODYN.

DYNCALL specifies whether the analyzer will use the variable name in dynamic calls as the name of the called program. If NODYNCALL is specified, dynamic calls are not processed by the analyze and information for them is not available to ESW product functions. The default is DYNCALL.

The analyze process for this code proceeds differently, depending on whether DYNCALL is specified:

```
77 MYPROG PIC X(8).  
   CALL MYPROG USING PARM1, PARM2.
```

In this example, if DYNCALL is in effect, the analyze process assumes that the program being called is "MYPROG," regardless of the data value that MYPROG contains at run-time. The analyze process looks up the analysis results of MYPROG in the AKR to determine whether PARM1 and PARM2 are used or modified.

If NODYNCALL is in effect, the analyze process assumes that the program being called could be anything, and treats both PARM1 and PARM2 as used and modified on the call statement.

Note: _____

The DYNCALL option is unrelated to the COBOL compiler option DYNAM.

Flag Messages

fLAGW
fLAGE
fLAG(*x*)

where *x* is a specified message level. The minimum abbreviations are LAGW, LAGE, and LAG(*x*). These are the valid message levels:

- I - Informational
- W - Warning
- E - Error
- S - Severe
- U - Unrecoverable

FLAG specifies the types of messages to be listed for the Analyze job.

FLAGW indicates all warning and diagnostic messages are listed. This is the default.

FLAGE indicates diagnostic messages are listed; all other messages are suppressed.

FLAG(*x*) indicates all messages of the specified level or above are listed.

Note: _____

Some informational messages are produced regardless of the flag setting.

Input

Input(*x*, *x*, . . . *x*)
Input=*x*
NOInput(*x*, *x*, . . . *x*)
NOInput=*x*

where *x* is a program name. The minimum abbreviation is I or NOI, with at least one program name. Wildcard characters are allowed in the program name.

INPUT lists the CALLED programs that contain INPUT statements. When commands that search for INPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

NOINPUT overrides the installation default list of CALLED programs that contain INPUT statements. The specified programs are deleted from the default list.

IO

IO(*x*, *x*, . . . *x*)
IO=*x*
NOIO

where *x* is a program name. Wildcard characters are allowed in the program name.

IO lists the CALLED programs that contain INPUT and OUTPUT statements. When commands that search for INPUT and OUTPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

NOIO overrides the installation default list of CALLED programs that contain INPUT and OUTPUT statements. The specified programs are deleted from the default list.

Language Level

LANGLVL(1)
LANGLVL(2)

LANGLVL specifies whether to use the 1968 or 1974 American National Standard COBOL definitions when analyzing source elements with meanings that have changed.

LANGLVL(1) indicates the 1968 standard is used; LANTLRVL(2) indicates the 1974 standard (X3.23-1974) is used.

The default is based on the compiler being used and is determined automatically by the submit process.

Line Count

lineCNT=*nn*

where *nn* is a number from 01 to 99. The minimum abbreviation is CNT, with a line count number.

LINECNT indicates the number of lines to be printed on each page of the source listing. The default is 60.

Main

MAIN

MAIN is used only as an override. The EXIT PROGRAM statements in COBOL programs are treated as GOBACKs by the Analyze job, because the program is treated as a CALLED subprogram. If the program is the main program, using the MAIN option treats the EXIT program as a fallthrough.

Maximum Number of Errors

MBRERCNT=*nnnn*

where *nnnn* is a number from 01 to 4000. MBRERCNT specifies the maximum number of analysis errors allowed for a member during an analyze job. If this number of errors is exceeded, the analyze terminates processing for that member. The number specified must be between 1 and 4000. The default is set at installation.

Output

Output(*x, x, . . . x*)

Output=*x*

NOOutput(*x, x, . . . x*)

NOOutput=*x*

where *x* is a program name. The minimum abbreviation is O or NOO, with at least one program name. Wildcard characters are allowed in the program name.

OUTPUT lists the CALLED programs that contain OUTPUT statements. When commands that search for OUTPUT are issued, statements that CALL these programs are shown in the command results. The specified programs are in addition to those specified at installation time.

NOOutput overrides the installation default list of CALLED programs that contain OUTPUT statements. The specified programs are deleted from the default list.

Program

```
PROgram(xxxxxxxxxx)
PGM=xxxxxxxxxx
```

where `xxxxxxxxxx` is a program name up to 10 characters. The minimum abbreviation is `PRO`, with a program name.

Analyzed programs are stored in the AKR and identified by the program name coded in the PROGRAM-ID statement. PROGRAM is used to override the name coded in the PROGRAM-ID statement.

Note: _____

SmartTest will not find this member in the AKR. SmartTest searches for the program or CSECT name in the load module.

Recursion

```
RECur
NORECur
```

The minimum abbreviation is `REC` or `NOREC`.

RECUR specifies whether the recursion report should be included in the Analyze Summary Report.

If RECUR is specified and recursion is not found, a message is issued that indicates no recursion was detected. If RECUR is specified and recursion is found, a message is issued and the recursive code is printed on the report.

The default is `NORECUR`.

Return

```
RETurn(x, x, . . . x)
RETurn=x
NORETurn(x, x, . . . x)
NORETurn=x
```

where `x` is a program name. The minimum abbreviation is `RET` or `NORET`, with at least one program name. Wildcard characters are allowed in the program name.

RETURN overrides the installation list of programs or entry points that do not return when CALLED. The system defaults are overridden by listing the desired programs or entry points that are to return when CALLED.

NORETURN lists the additional programs or entry points that are not to return when CALLED. When any of these programs are CALLED by the program being analyzed, they are treated as non-returning CALLs. The specified programs are in addition to the system defaults for programs that do not return when CALLED.

Sequence

SEQ
NOSEQ

SEQ specifies whether the analyze job checks the source module statement number sequence. A warning message is printed if the statements are not in sequence. If the Source option is also specified, a flag (**) is placed between the Analyze job sequence numbers and the source sequence numbers.

The default is SEQ.

Source

SOURce
NOSOURce

The minimum abbreviation is SOU or NOSOU.

SOURCE specifies whether the source program is to be listed. The SOURCE option is specified if a full program listing is desired at Analyze time.

The default is NOSOURCE.

Spacing

spACE1
spACE2
spACE3

The minimum abbreviations are ACE1, ACE2, and ACE3.

SPACE specifies the spacing for the source listing that is generated when the SOURCE option is used.

SPACE1 specifies single spacing.

SPACE2 specifies double spacing; that is, one blank line displays between every source line.

SPACE3 specifies triple spacing; that is, two blank lines appear between every source line.

SPACE1 (single spacing) is the default.

SQL Authorization ID

```
SQLID=nnnnnnnn  
SQLID(nnnnnnnn,nnnnnnnn,nnnnnnnn)
```

where *nnnnnnnn* is an 8 character name.

SQLID specifies the authorization ID or owner that is used by the analyze process to qualify unqualified table and view references in your program.

DB2 Subsystem

```
SUBSYS=xxxx
```

where *xxxx* is the name of the subsystem or location of the DBMS.

SUBSYS specifies the subsystem or location that designates the DBMS in which the tables accessed by a specified program are stored. SUBSYS overrides the name provided at installation time.

Live Exit

```
XLIVE
```

XLIVE is used as an override and should only be used with programs that contain live exits. Live exits are exits from perform ranges that are left dangling by imbedded PERFORMs or GO TOs in the original performed paragraph.

If XLIVE is not used, code that is unprocessed because of the live exit is ignored. If XLIVE is used, unprocessed code is saved.

Note: _____

Using XLIVE can significantly increase resource usage.

Memory

XMEM

XMEM is used only as an override. If a program is extremely large (i.e., 30,000 source lines) and there is insufficient memory, increase the region space. If there is still insufficient memory, enter the XMEM option. This results in more disk I/O and additional CPU usage, but less memory consumption.

8

Additional Language Support

This chapter describes how SmartTest supports Assembler, INTERSOLV APS, and PL/I programming languages, and contains these sections:

Topic	Page
SmartTest and Assembler Language	263
SmartTest and INTERSOLV APS	269
SmartTest and PL/I	273

Note:

For more detailed information on the SmartTest-PLI option, see the *ASG-SmartTest PLI User's Guide*.

SmartTest and Assembler Language

The SmartTest-ASM option is designed to alleviate the burden of testing and debugging Assembler and High Level Assembler Language Code programs by providing an online, interactive testing environment with powerful features suited to the needs of Assembler programmers. The Assembler component is fully integrated with SmartTest. All testing and debugging functions are available with Assembler including: Program View, execution control, breakpoints, monitoring, and changing data andabend processing. The only difference between using SmartTest with COBOL programs and SmartTest with Assembler or High Level Assembler programs is certain analysis and debugging facilities, such as the COBOL intelligent search, are limited by their COBOL orientation.

SmartTest with Assembler enables you to work with the source level display of your programs in Program View. When a lower level of detail is necessary, SmartTest allows viewing and manipulation of memory, input data, and the general purpose and floating point registers.

Analyzing an Assembler Program

The analyze process for Assembler programs is the same as the analyze process for COBOL programs, except the Assembler is invoked instead of the COBOL compiler.

Starting a Test Session

The steps to set up a test of your Assembler program are identical to the steps required for testing COBOL programs. For more information, see ["Test Session" on page 25](#).

Assembler Source Testing and Debugging

SmartTest allows you to test and debug your Assembler programs at the source code level using Program View. Program View, the Status Box, screen manipulation, and command entry methods are no different than those shown in other sections of this guide.

You can review many of the capabilities of SmartTest using the SmartTest Assembler tutorial program VIAPASM.

To set up VIAPASM for a test session

- 1 Assemble/Link and Analyze the ESW program VIAPASM.
- 2 Convert the ESW execution JCL member VIAPASMJ to a CLIST. (See ["Converting Batch Execution JCL to a TSO CLIST" on page 55](#).)
- 3 Setup and run a SmartTest TSO test session and specify ALL in the EXECUTION PARAMETERS entry on the TSO Session Setup screen. (See ["MVS Programs in TSO Foreground" on page 53](#).)

Figure 138 shows VIAPASM suspended at the beginning of the test session.

Figure 138 • Program View Screen with an Assembler Program

```

File View Test Search List Options Help
-----
Program View                                VIAPASM.VIAPASM -A
====> _____ SCROLL ==> CSR

000149 *
>>>>> B BYINFO-VIAPASM(R15) BRANCH AROUND PGM INFORMATION
000152 *-----*
000153 * *
000154 * ***** R E A D T H I S ***** *
000155 * * *
000156 * * PLEASE READ THE INSTRUCTIONS AT THE TOP OF THIS *
000157 * * PROGRAM. THEY WILL HELP YOU GET THE MOST VALUE *
000158 * * FROM THIS DEMONSTRATION. *
000159 * * *
000160 * * ENTER 'UP MAX' ON THE PRIMARY COMMAND LINE. *
000161 * * *
000162 * ***** R E A D T H I S ***** *
000163 * *
000164 *-----*

000165 DC CL8'VIAPASM ' PROGRAM NAME
+-----+
|STATUS: BREAK AT START OF TEST SESSION PROGRAM: VIAPASM DATE: DDMMYYYY |
| STMT: 000151 OFF: 000000 AMODE: 24 MODULE: VIAPASM TIME: HH:MM:SS |
|SOURCE: B BYINFO-VIAPASM(R15) BRANCH AROUND PGM INFORMATION |
+-----+

```

Note:

Follow the tutorial instructions given throughout the program VIAPASM to learn more about using SmartTest with Assembler.

Display Expanded Assembler Macros

Use the SET GENERATED ON primary command to display the generated Assembler source statements. The SET GENERATED OFF primary command display the Assembler macro statements only.

When the GENERATED mode is ON, SmartTest functions such as STEP and BREAK operate at the generated statement level. When the GENERATED mode is OFF, SmartTest functions operate at the macro statement level.

To illustrate the effect of using the SET GENERATED primary command

Note:

These steps use the Assembler tutorial program VIAPASM.

- 1 Scroll the screen to line 878 (the ATTACH macro), shown in [Figure 139](#), by typing L 878 in the primary command input area and pressing Enter.

Figure 139 • Program View with an Assembler Program at the Macro Statement Level

```

File View Test Search List Options Help
-----
Command ==> Program View VIAPASM.VIAPASM -A
Scroll ==> CSR

000877 *-----*
000878 ATTACH ECB=(R2),EPLOC=(R3),SF=(E,ATTACHL)
000889 LTR R15,R15 ENSURE VALID RETURN CODE
000890 BNZ ABENDA AND ABEND IF NO GOOD ???
000891 ST R1,WORKTCB SAVE ADDRESS OF ATTACHED TCB
000892 WAIT 1,ECB=(R2) WAIT FOR ATTACHED TASK TO COMPLETE
000897 PAUSED08 LA R1,WORKTCB POINT TO ADDRESS OF ATTACHED TCB
000898 *-----*
000899 *
000900 * 'SET ASMVIEW OFF' TO TURN OFF THE DISPLAY OF DISASSEMBLED OBJECT. *
000901 *
000902 * ENTER 'RUN' TO CONTINUE THE DEMONSTRATION. *
000903 *
000904 *-----*
+-----+
|STATUS: BREAK AT START OF TEST SESSION PROGRAM: VIAPASM DATE: DDMMYYYY |
| STMT: 000153 OFF: 000000 AMODE: 24 MODULE: VIAPASM TIME: HH:MM:SS |
|SOURCE: B BYINFO-VIAPASM(R15) BRANCH AROUND PGM INFORMATION |
+-----+

```

- 2 Type SET GEN ON in the command input area and press Enter. The Assembler expanded macro statements, shown in [Figure 140](#), are displayed.

Figure 140 • Program View with an Assembler Program at the Generated Statement Level

```

File View Test Search List Options Help
-----
Command ==> Program View GENERATED ON
Scroll ==> CSR

000877 *-----*
000878 ATTACH ECB=(R2),EPLOC=(R3),SF=(E,ATTACHL)
000879 +* /* MACDATE 01/06/86 @L1C*
000880 +* /*
000881 + LA 15,ATTACHL LOAD LIST ADDRESS @G860PX
000882 + ST R3,0(,15) INSERT EPLOC INTO LIST @G860PX
000883 + ST R2,8(,15) ECB INTO LIST @G860PX
000884 + MVI 55(15),72 SET LENGTH OF THIS PARM LIST @G860PX
000885 + OI 8(15),X'80' SET NEW FORMAT BIT @G860PX
000886 + NI 60(15),127 CLEAR UNWANTED BITS @ZA7242
000887 + MVI 61(15),1 SET FORMAT NUMBER @G860PX
000888 + SVC 42 ISSUE ATTACH SVC @G860PX
000889 LTR R15,R15 ENSURE VALID RETURN CODE
000890 BNZ ABENDA AND ABEND IF NO GOOD ???
+-----+
|STATUS: BREAK AT START OF TEST SESSION PROGRAM: VIAPASM DATE: DDMMYYYY |
| STMT: 000153 OFF: 000000 AMODE: 24 MODULE: VIAPASM TIME: HH:MM:SS |
|SOURCE: B BYINFO-VIAPASM(R15) BRANCH AROUND PGM INFORMATION |
+-----+

```

- 3 Type `SET GEN OFF` and press Enter to display only the Assembler macro statements.

Note: _____

See "[Testing Techniques](#)" on page 143 for more information regarding SmartTest testing features.

The Assembler Specific Commands

The USING and DROP commands apply only when working with Assembler programs.

Use the USING command to specify the base register to be used for addressing of data fields within Assembler DSECTs. For example:

```
USING TESTMOD 1
```

uses register 1 as the base of addressability for the DSECT named TESTMOD.

The DROP command ends addressability to any Assembler DSECT currently addressed by base register *n*, where *n* is the register as set by the USING command. For example:

```
DROP 1
```

stops using register 1 as the base of addressability for the DSECT named TESTMOD.

Note: _____

The USING and DROP commands are only available for Assembler H.

Commands with Limited Use in Assembler

Some commands are limited when working with Assembler programs. Certain target types and operands may not be used with these commands, due to their COBOL orientation:

- BREAK
- EXCLUDE
- FINDXTND
- HIGH
- LIST
- LOCATE
- LPRINT

- LPUNCH
- SCROLL

SmartTest-ASM distinguishes between labels with associated executable statements and those without for the purpose of inserting breakpoints and for the Execution Tracking screen.

The only subsets available in SmartTest-ASM as target operands are screen subsets. The screen subsets include; HI, NHI, X, and NX.

Commands Not Available with Assembler

These commands are not available when working with Assembler programs because of their COBOL orientation:

- BRANCH
- COPY
- DELETE
- FLOW
- MARK
- MERGE
- PREF
- RENAME
- RPREF
- RTRACE
- TRACE
- SELECT
- UPDATE
- ZOOMIN

Note: _____

For further information regarding SmartTest test facilities, see ["Testing Techniques" on page 143](#).

SmartTest and INTERSOLV APS

INTERSOLV APS programs may be accessed at generated source code level through SmartTest. The SmartTest-APS option is designed to ease testing and debugging of programs generated by the INTERSOLV APS product, by providing an interactive testing environment able to display both Program Painter code and generated COBOL source code.

All testing and debugging functions are available with APS including: Program View, controlling execution, setting breakpoints, monitoring and changing data, pseudo code, abend processing, and the COBOL intelligent search function.

Analyzing an INTERSOLV APS Program

To invoke the proper SmartTest processing during the compile process, the final jobstream must be submitted using the ESW job submit utility VIASUBDS.

Use the APS Generator Options screen to integrate the SmartTest analyze submit facility into the APS product. The APS Debug field on this screen is used to invoke the analyze submit facility. The APS Generator Options screen is provided by INTERSOLV, Inc.

[Figure 141](#) shows the APS Generator Options screen.

Figure 141 • APS Generator Options Screen

```

----- APS Generator Options -----
OPTION ==>

TARGET OS ==>          (MVS, OS2, PCDOS, OS400, VSE)
  DC ==>              (IMS, CICS, DLG, DDS, MVS, or ISPF (prototyper))
  DB ==>              (IMS, DLI, VSAM, or IDMS)
  SQL ==>             (OS2DM, XDB, SQLDS, DB2)

JOB CLASS ==>          JOB DEST ==>
MSG CLASS ==>          CARDIN MEMBER ==>

LISTGEN ==>          (Yes or No) COBOL-II ==>          (Yes or No)
COBOL ==>            CICS RELEASE ==>          (1.7, 2.1 or 3.1)
OBJECT ==>          IMS RELEASE ==>          (2 or 3)
MFS/BMS ==>        SUPRA ==>          (Yes or No)
GENSRC ==>          EBCDIC ==>          (Yes or No)
APS DEBUG ==>      PC CICS ==>          (IBM, MFOCUS)

APS Parm ==>
COBOL Parm ==>

```

Type YES in the APS Debug field on the APS Generator Options screen to process the APS compile JCL. If you type NO in the APS Debug field, the compile JCL is submitted without performing the SmartTest analyze.

Starting an INTERSOLV APS Test Session

The steps to set up a test of your APS program are identical to the steps required for testing standard COBOL programs. For more information, see ["Test Session" on page 25](#).

APS Testing and Debugging

SmartTest allows you to test and debug your APS programs at either the Program Painter code or generated COBOL source code levels. Program View, the Status Box, screen manipulation, and command entry methods are the same as for other languages and environments.

Displaying Painter Code

Use the SET GENERATED ON primary command to display the generated COBOL source code with the Program Painter code interspersed as COMMENTS. The SET GENERATED OFF primary command displays the Program Painter code only.

When the GENERATED mode is ON, the program functions as a standard COBOL program and all SmartTest features and functions are available. When the GENERATED mode is OFF, the program functions at the Program Painter statement level. The STEP and BREAK commands apply to Program Painter statements, not generated COBOL source.

As a rule, the ZOOM and KEEP commands attempt to parse out valid variable names from the Painter statement. If none are found, it operates on the next generated COBOL source statement, which normally contain the variables in the Program Painter statement. An exception to this rule is the APS macro statement (abbreviated and full format). With this statement, SmartTest parses the abbreviated and full format macro statement and attempt to open a Zoom data window on all referenced datanames.

APS Program

This exercise can be reproduced using the INTERSOLV supplied demonstration program DLGINQ.

[Figure 142](#) shows the APS program DLGINQ suspended at the start of the test session. The default GENERATED mode is OFF. Only the Painter source code displays.

Figure 142 • Program View Screen at Start of Test Session

```

File View Test Search List Options Help
-----
                                Program View                                DLGINQ.DLGINQ -A
Command ==>> _____ Scroll ==>> CSR

000002 SYM1% *****
000004 % SET INFO - DLGINQ - APS1803 - 12/13/89 - INTERSOLV
000006 % *****
000008 % *
000010 % * COPYRIGHT 1986 TO 1988, 1989
000012 % * INTERSOLV, INC.
000014 % * ALL RIGHTS RESERVED.
000016 % *
000018 % *****
000020 SYM1% &DLG-PROGRAM-TRANSFER-OPTION = "SELECT"
000079 REC SAVE-PART-NBR PIC X(8)
>>>>> NTRY
000651 IF END-ON-SEND
000654 TERM

+-----+
|STATUS: BREAK ON ENTRY TO A PROGRAM PROGRAM: DLGINQ DATE: DDMMYYYY |
| STMT: 000082 OFF: 000F64 AMODE: 24 MODULE: DLGINQ TIME: HH:MM:SS |
|SOURCE: NTRY |
+-----+

```

Note:

If a RUN or STEP primary command is issued, the Painter source statement(s) are executed along with the generated COBOL source, between lines 82 and 651, as one statement.

To display the COBOL generated source code and display the next executable statement

- 1 Type SET GEN ON; L * in the primary command input area and press Enter. The Program View screen, shown in [Figure 143](#) displays.

Figure 143 • Program View Screen with Generated COBOL Source Code

```

File View Test Search List Options Help
-----
                                Program View                                DLGINQ.DLGINQ -A
Command ==>> _____ Scroll ==>> CSR

000576 MAIN--SECTION--PARA.
>>>>> PERFORM APS-HOUSEKEEPING-PARA
000578 THRU APS-HOUSEKEEPING-PARA--EXIT.
000579 PERFORM APS-USER-MAIN-PARA THRU APS-USER-MAIN-PARA--EXIT.
000580 PERFORM APS-MAIN-PARA THRU APS-MAIN-PARA--EXIT. DEADCODE
000581 GOBACK. DEADCODE
000582 MAIN--SECTION--EXIT. DEADCODE
000583 EXIT PROGRAM. DEADCODE
000584 MAIN--SECTION--SXIT. DEADCODE
000585 GOBACK. DEADCODE
000586 *
000587 */*** END TP-ENTRY
000588 *
000589 EJECT

+-----+
|STATUS: BREAK ON ENTRY TO A PROGRAM PROGRAM: DLGINQ DATE: DDMMYYYY |
| STMT: 000577 OFF: 000F64 AMODE: 24 MODULE: DLGINQ TIME: HH:MM:SS |
|SOURCE: PERFORM APS-HOUSEKEEPING-PARA |
+-----+

```

The generated COBOL source code statements are displayed. The chevrons appear on the next executable statement.

- 2 To view only the Painter source code, type `SET GEN OFF; L *` and press Enter.
- 3 To execute the Painter code statement, type `STEP` and press Enter. The Program View screen, shown in [Figure 144](#), displays.

Figure 144 • Program View Screen with Program Painter Code

```

File View Test Search List Options Help
-----
Command ==> _____ Program View _____ DLGINQ,DLGINQ -A
Scroll ==> CSR
000079 REC SAVE-PART-NBR PIC X(8)
000082 NTRY
>>>>> IF END-ON-SEND
000654 TERM
000662 ELSE-IF DLGI-ZCMD = '2'
000670 IF DLGI-PART-NBR-INPT > SPACES
000677 $DLG-VDEFINE("01 PARTNO PIC X(08)")
000691 MOVE DLGI-PART-NBR TO PARTNO
000694 XCTL DLGUPD * PARM('&PARTNO')
000721 $DLG-VDELETE("PARTNO")
000742 ELSE
000745 XCTL DLGUPD
000774 SC-CLEAR DLGI
000783 ELSE-IF DLGI-ZCMD = SPACES
+-----+
|STATUS: STOPPED BY STEP REQUEST PROGRAM: DLGINQ DATE: DDMMYYYY |
| STMT: 000651 OFF: 001138 AMODE: 24 MODULE: DLGINQ TIME: HH:MM:SS |
|SOURCE: IF END-ON-SEND |
+-----+

```

To view the generated COBOL source code execution history

- 1 Type `LIST TRACKING` in the primary command input area and press Enter.

Or

Select List ▶ Execution Tracking and press Enter.

Or

Select Source on the List Execution Tracking pop-up and press Enter.

The Execution Tracking screen shows the executed COBOL statements in execution sequence.

- 2 Press PF3/PF15 to return to the Program View screen.

Note:

See ["Testing Techniques" on page 143](#) for more information regarding SmartTest testing features.

Considerations with APS Program Painter Code

When SET GENERATED OFF is specified, the COBOL Intelligent Search commands EXCLUDE, FINDXTND, HIGHLIGHT, and FLOW display the appropriate message for the results at the generated COBOL level, but will not highlight and tag Program Painter statements. The LPRINT command outputs the appropriate COBOL generated statements meeting the criteria.

The counts and tracking facilities operate with the COBOL generated code regardless of the GENERATED display setting. The Statement Counts and Execution Tracking screen reflects information based on the generated COBOL program processing.

Considerations with APS Generated COBOL Code

The primary commands SCROLL and UPDATE function are not available when working with Program Painter code. In general, all SmartTest screens appear the same whether the GENERATED option is ON or OFF. The difference can be seen in the Program View screen.

Note: _____

For further information on the SET GENERATED command, see the *ASG-SmartTest Reference Guide*.

SmartTest and PL/I

SmartTest-PL/I provides an online, interactive testing environment with a full set of features for PL/I applications. It is fully integrated with SmartTest and provides the testing and debugging functions required, such as execution control, breakpoints, abend interrupts, and changing/monitoring data. The only difference between using SmartTest with PL/I rather than COBOL is certain analysis features that utilize COBOL intelligence are limited. If you attempt to use a feature not available under PL/I, you receive a short message notifying you that the data is not available.

SmartTest-PL/I enables you to work with the source level display of your programs in Program View. When a lower level of detail is necessary, SmartTest-PL/I allows viewing and manipulation of data and memory, and it allows execution control through pseudo code.

Note: _____

See the *ASG-SmartTest PL/I User's Guide* for additional information on SmartTest-PL/I.

9

Help Facility

This chapter describes how to set up a SmartTest test session for the TSO execution environment and contains these sections:

Topic	Page
Introduction	275
Help Navigational Commands	277
Screen Help	278
Command Help	279
General Information	281
Specific Information	282
Help Abends	283
Help Messages	284

Introduction

A comprehensive and context sensitive Help facility, including an online Help Tutorial, is provided that answers most questions online. The Help Tutorial contains help information for several types of topics, such as pull-downs, screens, pop-ups, commands, messages, and abends. The Help Tutorial also includes a Table of Contents that describes each major SmartTest function, and a comprehensive Index for viewing specific information.

The SmartTest online help facility can be reached through several means. Selecting Help on the action bar displays the help pull-down. You can also access Help using these methods:

- Typing `H` on the Primary Menu to reach the Help Table of Contents.
- Pressing PF1 or PF13.
- Typing `HELP` or question mark (?) in the command input area on any screen.

This table lists the various online help information provided by SmartTest, and the means of accessing them:

Help Topic	Access
Screen and pop-up help	Help for the current screen is requested by selecting Help ► Current screen. Help for the current screen or pop-up displays by typing <code>HELP</code> , or by pressing PF1/13. No messages can appear on the screen at the time this help is requested.
Command help	Help for a command is requested by selecting Help ► Specific command, or by typing the command in the command input area and pressing PF1/13. Typing <code>HELP COMMANDS</code> displays a long message that lists most of the primary commands. Once this message displays, pressing PF1/13 displays a complete list of all SmartTest commands. This list of commands can also be accessed by selecting Help ► All commands. From the list of commands, information about a particular command can be displayed by selecting the appropriate number.
General Information	General help information is requested by selecting Help ► Table of Contents, or typing <code>TOC</code> on any Help Tutorial screen, to display the Help Table of Contents. Help information can be viewed by pressing Enter or by selecting a menu option.
Specific Information	Help for specific topics is requested by selecting Help ► Index, by selecting option I on the Help Table of Contents, or by typing <code>INDEX</code> from within the Help Tutorial. Help for a specific topic can be viewed by selecting the appropriate index entry.

Help Topic	Access
Abends	Help for ESW user abends is available by selecting Help ► Common abends, or by typing <code>HELP ABENDS</code> . The Help Tutorial Abends screen displays. Select Option 2 on this screen to display the ASG Abend Codes screen, which lists all the ESW user abends, and explanations for each abend.
Messages	Help for a current message displays by selecting Help ► Current message. Help for a short message, displayed in the upper right corner of the screen, is requested by typing <code>HELP</code> or by pressing PF1/13. The corresponding long message displays near the bottom of the screen. Help for a specific message can be displayed by selecting Help ► Specific message, or by typing <code>HELP msg#</code> .

Help Navigational Commands

All of the online help topics listed in this table are contained in the Help Tutorial. Each online help topic can be reached from anywhere within the Help Tutorial by going through the Help Table of Contents or Index. After you access the Help Tutorial, there are several commands available for navigating within the Help Tutorial. These are the commands:

Help Command	Purpose
BACK	Redisplay the previous Help Tutorial screen.
END	Exit the Help Tutorial.
ENTER	Display the next screen in a continuation series.
INDEX	Display the first screen of the Help Index.
SKIP	Go directly to the next subject
TOC	Display the Help Table of Contents
UP	Display the next higher-level subject.
Alpha character	On an Index screen, entering an alphabetic character displays the Index screen corresponding to that character.

Screen Help

You can request help for the current screen by selecting Help ► Current screen, by typing HELP, or by pressing PF1/13. If any messages appear, help for the screen or pop-up displays by typing HELP SCREEN. The Help Tutorial for the current screen displays, as shown in [Figure 145](#).

The Help Tutorial for each screen or pop-up describes all the options available on that screen, lists descriptions of all the fields, and notes any special processing considerations.

Figure 145 • Pop-up Help Example

```
ASG-SmartTest - R6.0 - Options - Product Allocations ----- HELP
==>
The Options - Product Allocations pop-up is used to specify the allocation
parameters for the Log, List, Punch, and Work files. To access this pop-up
from the Options pull-down, select Allocation, or enter the ALLOCDEF command
on any screen.

Note: Management Class, Storage Class, and Data Class provide various
parameters for newly allocated data sets. These parameters apply only if you
have SMS active at your site. Your system administrator determines the valid
entries for these parameters.

The Storage Class and Volume serial parameters are mutually exclusive.

Field      Descriptions
Log File   Specify either the Management Class and Storage Class or the
           Generic unit and volume serial number for the Log file
           that is allocated upon entry into ASG-SmartTest. The Log file
           is used for error messages and log commands. File characteristic
           are specified on the Options - Log/List/Punch Definition pop-up.

(MORE...press ENTER for more information.)
```

Command Help

You can request help for a specific command by selecting Help ► Specific command, by typing the command in the command input area and pressing PF1/13, or by typing HELP followed by the desired command name. A long message describing the command is displayed. Pressing PF1/13 again displays the Help Tutorial screen for that command, as shown in [Figure 146](#). The Help Tutorial for each command displays the command syntax diagram, and gives a description of each operand in the command. Entering the UP command on a command help screen displays the Help Table of Contents.

Figure 146 • Command Help Example

```

ASG-SmartTest - R6.0 ----- RECALL ----- HELI
===> _

The RECALL command displays the previous ASG primary or internal command,
message, or pop-up. The last twenty commands that have been executed and the
last twenty messages that have been displayed are stacked. These commands and
messages can be displayed using RECALL. Once the desired command is displayed,
it can be executed again by pressing ENTER or changed prior to execution.

The RECALL command syntax is:

RECALL ----->
  -COMmand| CMD-   | -NEXT- |
  -MESSage| MSG-   | -PREV- |
  -POPup----->
                                     Minimum Abbreviations are in CAPS
                                     Default operands are highlighted
LEGEND: ---required--->
                                     |-optional-|

The following topic will be presented only if explicitly selected by number:

  1 - Operand Descriptions

```

For help on all SmartTest commands, select Help ► All commands or type HELP COMMANDS. A complete list of all SmartTest commands displays as shown in [Figure 147 on page 280](#). From this list, you can access information about a particular command by selecting the appropriate number.

Figure 147 • ASG-SmartTest Commands Screen

```

ASG-SmartTest - R6.0 ----- ASG-SmartTest COMMANDS ----- HELF
==>
The following will be presented in sequence, or may be selected by number:

  1 - & (retain)    20 - FIND      39 - PROCESS  58 - SET
  2 - ADD           21 - FINDXTND 40 - PRODLVL 59 - SETUP
  3 - ALLIANCE     22 - FLOW     41 - QUALIFY 60 - STEP
  4 - ALLOCDEF     23 - GO       42 - RECALL  61 - STOP
  5 - ANALYZE     24 - HELP     43 - REDO    62 - SUBTRACT
  6 - BRANCH      25 - HIGH    44 - REFRESH 63 - TEST
  7 - BREAK       26 - KEEP    45 - RENAME  64 - TESTPOINT
  8 - CANCEL      27 - KEYS    46 - REPEAT  65 - TOGGLE
  9 - CONVERT     28 - LIST    47 - RESET   66 - TRACE
 10 - COPY        29 - LOCATE  48 - RETURN  67 - UPDATE
 11 - CURRENT     30 - LPRINT  49 - RFINN   68 - USING
 12 - DELETE     31 - LPUNCH  50 - RHIGH   69 - UTILITY
 13 - DISPLAY    32 - MARK    51 - RPREF   70 - VIEW
 14 - DROP       33 - MERGE   52 - RSCROLL 71 - WHEN
 15 - END        34 - MOVE    53 - RTRACE  72 - WHERE
 16 - ENVIRONMENT 35 - PARMDEF 54 - RUN     73 - WIZARD
 17 - EQUATE     36 - PREF    55 - SAVE    74 - ZOOMDATA
 18 - EXCLUDE    37 - PRINTLOG 56 - SCROLL  75 - ZOOMIN
 19 - EXECUTE    38 - PRINTLST 57 - SELECT  76 - ZOOMOUT

----- Cur panel = UPTCHDS Prev panel = UPPPRIME Last msg = -----

```

For some commands, on the Program View screen only, after the long message displays, pressing PF1/13 displays NOTES giving specific examples for using the command. Once the NOTES are displayed, as shown in [Figure 148](#), pressing PF1/13 again displays the Help Tutorial for that command. The Help NOTES displays for those commands where specific examples are helpful.

Figure 148 • Help NOTES Example

```

File View Test Search List Options Help
-----
Program View                                UIAPPLI
Command ==>                                Scroll ==> CSR
ASG4645I RECALL REDISPLAYS THE PREVIOUS COMMAND OR MESSAGE.
=NOTE= +----- Examples ----- RECALL ----- Descriptions -----+
=NOTE= | RECALL | Recalls the last command that was |
=NOTE= | | entered in the primary command |
=NOTE= | | area. |
=NOTE= | REC MSG | Recalls the last message that was |
=NOTE= | | displayed in the message area. |
=NOTE= | RECALL NEXT | Once recalling is started, NEXT |
=NOTE= | | may be used to reverse its |
=NOTE= | | direction. |
+-----+
000001 * PROCESS OPTIONS INSOURCE SOURCE NEST NOMACRO;
000002 * PROCESS AGGREGATE ESD STMT GOSTMT;
000003 * PROCESS MARGINS(2,72,1) MARGINI(' ');
000004 * PROCESS OPT(0) ATTRIBUTES(FULL) XREF(FULL);
000005 * PROCESS GOSTMT,LIST,NEST,LINECOUNT(55),OPTIONS,SOURCE,NOTEST;
000006 * PROCESS NOOPTIMIZE,NOFLOW,ATTRIBUTES;
000007
000008
000009 /*****

```

General Information

Request general help information by selecting Help ► Table of contents, or by typing TOC on any Help Tutorial screen. The Help Table of Contents, shown in [Figure 149](#), displays. Help information can be viewed by pressing Enter or by selecting a menu option.

Figure 149 • Help Table of Contents

```
ASG-SmartTest - R6.0 ----- HELP TABLE OF CONTENTS ----- HELI
==> -
The topics below represent general categories of information about the ASG-ESW
Testing/Debugging component, ASG-SmartTest. To get help for a pull-down,
select the Action Bar topic. This Table of Contents may be redisplayed from
any HELP screen by entering the TOC command.

The following topics will be presented only if explicitly selected by number:

  1 Overview of ASG-SmartTest
  2 Introduction to CUA
  3 The Action Bar
  4 Customer Support
  5 ASG-SmartTest Release 6.0 Summary of Revisions
  6 Index for ASG-SmartTest Help

----- Cur panel = UPTHTOC  Prev panel = UPPPRIME Last msg = ISR2002 -----
```

Specific Information

Help for specific topics is requested by selecting Help ► Index, by selecting option I on the Help Table of Contents, or by typing INDEX from within the Help Tutorial. The Help Index screen, shown in [Figure 150](#) displays. Help for a specific topic can then be viewed by selecting the appropriate Index entry.

On any Index screen, entering an alphabetic character displays the Index screen corresponding to that character.

Figure 150 • Help Index Example

```
ASG-SmartTest - R6.0 ----- INDEX A - B ----- HELI
===> _
To select a topic, enter the two- or three-character identifier.

A1 - ABENDS
A2 - ADD Command
A3 - Address Stop Entry Screen
A4 - AKR Directory
A5 - AKR Utilities
A6 - ALLOCDEF Command
A7 - ALLIANCE Command
A8 - ALTPCB Message Queue List
   Screen
A9 - ALTPCB Segment List Screen
A10 - ANALYZE Command
A11 - Analyze Options

B1 - BackTrack Facility
B2 - BackTrack Variable History Pop-up
B3 - BackTrack Variable History Screen
B4 - BRANCH Command
B5 - BREAK Command
B6 - Breakpoints List Screen

Another index page can be displayed by entering its letter.
```

Help Abends

ESW products include help for these frequently encountered abend codes:

- System abend codes such as 0C1, 0C4, x13, x22
- ESW product abend codes
- IMS abend codes

Help for System, ESW product, and IMS user abend codes is requested by selecting Help ► Common abends, by typing `HELP ABENDS`, or by typing `ABENDS` in the command input area and pressing PF1/13. All of these actions displays the `ABENDS` screen, as shown in [Figure 151](#).

Figure 151 • ABENDS Screen

```
ASG-SmartTest - R6.0 ----- ABENDS ----- HELI
===> _

The following topics will be presented only if explicitly selected by number:

  1 - System Abends
  2 - ASG Abends
  3 - IMS Abends 008 - 648
  4 - IMS Abends 684 - 931
  5 - IMS Abends 932 - 3415
```

On the `ABENDS` selection screen, enter the number of the set of abend codes you want to view. Selecting Option 2 on the `ABENDS` screen displays the `ASG Abend Codes` screen, shown in [Figure 152](#). This screen lists all the ESW user abend messages, and explanations for each message.

Figure 152 • ASGAbend Codes Screen

```
ASG-SmartTest - R6.0 ----- ASG ABEND CODES ----- HELF
===> _

Abend codes in the range 900 - 999 (X'384 - X'3E7') bypass ASG error
recovery, causing the abend to be handled by ISPF or by the system. If the
problem cannot be resolved, call Customer Support.

965 X'3C5'    Unable to intercept program.
967 X'3C7'    The ASG-Center AUTHORIZE password was not specified during
              installation.
968 X'3C8'    An internal error occurred during initialization.
970 X'3CA'    A package load module was called directly.
972 X'3CC'    The ASG Edit Monitor encountered a severe error.
974 X'3CE'    An invalid VIASBASE module was found. The current product
              expects a level of CE050 or greater. Enter HELP 4968 for more
              information.
              (continued)
```

Help Messages

SmartTest messages are displayed in the long message area. This is the format for messages:

ASGnnnnx text

where:

nnnn is the message number.

x is one of the severity levels listed in the table.

text is the long or short message text.

Severity Levels

This table describes the severity levels:

Level	Message Type	Description
I	Informational	Indicates that there is no required action.
W	Warning	Indicates a non-critical error condition exists.
E	Error	Indicates a critical error condition exists.
D	Disaster	Indicates that a serious error condition exists and the product is unable to continue.
T	Termination	Indicates that the product terminated with the specified error.

Short messages are displayed when available. Long messages are displayed if a short message does not exist, or when help is requested immediately after a displayed short message.

Request help for a specific message by selecting Help ► Specific message, or by typing HELP followed by the message number. The Help Explanation and Action Panel for that message displays, as shown in [Figure 153](#).

Figure 153 • Help Explanation and Action Panel

```

                                HELP Explanation and Action Panel
Command ==> _____ Scroll ==> PAGE
Additional support may be found at our Web Site: www.asg.com
ASG0650 END OF PROGRAM; USE BRANCH BACKUP TO FOLLOW OTHER BRANCHES.
EXPLANATION:
  This is a warning message indicating that BRANCH has reached the
  physical end of the program.
ACTION:
  If you wish to branch to other paths, use BRANCH BACKUP to reach
  the desired decision point and then use BRANCH to follow another
  path.
***** BOTTOM OF DATA *****

```

Printing Messages

All SmartTest messages or a range of messages can be printed using the VIASMPRT program. The VIASMPRT program produces a listing of the specified messages that includes:

- Message number
- Short message (if available)
- Long message
- Explanation of the message
- Action (if any)

JCL to execute the VIASMPRT program is in ASG.VIACEN_{xx}.CNTL(VIASMPRT). The entire message file is printed unless a specific range is specified in the PRM parameter. For example:

```
PRM=' START=300, END=499 '
```

prints messages 300 through 499.

The ALL keyword can be specified in the PRM parameter to print all messages.

The default value for START is 1 and the default value for END is 5000. If only the START value is entered, messages print starting at the message number specified and ending with 5000. If only the END value is entered, messages print starting with 1 and ending with the message number specified.

The NOTES keyword specifies that any notes associated with a message are printed. The default is NONOTES. Typically, notes are provided to show Center primary commands.

[Figure 154](#) and [Figure 155 on page 287](#) show the VIASMPRT JCL and the output from the job.

Figure 154 • VIASMPRT JCL

```
//ASG JOB ( ),'ASG-CENTER VIASMPRT'
//* INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
//* *****
//* * ASG, INC.          ASG-CENTER Rx.x      December, 2001      *
//* *                                                           *
//* *           UTILITY TO PRINT ASG MESSAGES                       *
//* *                                                           *
//* *****
//*
//VIASMPRT PROC ASG='ASG', HIGH LEVEL NODE OF ASG DATA SETS
//          CENTER='VIACENxx', MIDDLE NODE OF ASG DATA SETS
//          SYSOUT='*',        PRINT OUTPUT MESSAGE CLASS
//          PRM=' '            PARM FOR MESSAGES TO BE PRINTED
//*
//* *****
//* *
//* *           M E S S A G E   P R I N T   U T I L I T Y           *
//* *                                                           *
//* * THIS PROGRAM WILL PRINT ALL OF THE MESSAGES IN THE ASG      *
//* * MESSAGE FILE AND THE HELP TEXT ASSOCIATED WITH EACH        *
//* * MESSAGE IT WILL PRINT THE ENTIRE FILE BY DEFAULT. YOU MAY  *
//* * SELECT A GIVEN RANGE OF MESSAGES BY SPECIFYING THE OPTION-  *
//* * AL PARAMETER KEYWORDS: START AND END. FOR EXAMPLE:         *
//* *           PRM='START=300,END=499'                            *
//* * WILL PRINT MESSAGES NUMBER 300 THROUGH 499, INCLUSIVE.    *
//* * THE DEFAULT VALUES FOR START AND END ARE 1 AND 99999      *
//* * RESPECTIVELY. CONSEQUENTLY THE PRM VALUE 'END=300' WILL    *
//* * PRINT MESSAGES 1 THROUGH 300, AND THE PRM VALUE            *
//* * 'START=4000' WILL PRINT MESSAGES 4000 THROUGH 99999.      *
//* *                                                           *
//* * AN OPTIONAL KEYWORD, NOTES, WILL ALSO PRINT ANY NOTES     *
//* * ASSOCIATED WITH A MESSAGE.                                  *
//* *                                                           *
//* * ADDITIONALLY, THE KEYWORD 'ALL' WILL EXPLICITLY PRINT ALL  *
//* * MESSAGES.                                                  *
//* *****
//*
//*
//VIAMPRT EXEC PGM=VIASMPRT,REGION=4096K,
//          PARM='&PRM'
//STEPLIB DD DSN=&ASG..&CENTER..LOADLIB,DISP=SHR
//VIAMSGS DD DSN=&ASG..&CENTER..VIAMSGS,DISP=SHR
//SYSPRINT DD SYSOUT=&SYSOUT
//VIAPRINT DD SYSOUT=&SYSOUT
//VIALOG DD SYSOUT=&SYSOUT
//SYSUDUMP DD SYSOUT=&SYSOUT
```

```
/**
//          PEND
/**
//VIASMPRT EXEC VIASMPRT          PRINT MESSAGES
/**
```

Figure 155 • VIASMPRT Output

```
PRINTING MESSAGES FROM 0771 TO 0772.
2 MESSAGES PRINTED.
END OF MESSAGE PRINT PROCESSING.
ASG0771  SUBSET 'COBOLII' IS NOT VALID IN A LANGLVL 1 OR 2 EDIT SESSION.

EXPLANATION:
The COBOL Edit session was selected to view a program as COBOL
LANGLVL1 (COBOL68) or LANGLVL2 (COBOL74), and the command entered
requested a target of SUBSET COBOLII.

ACTION:
If the program is COBOLII, then reenter the Edit screen with COBOLII
selected; then you may enter commands for SUBSET COBOLII.

ASG0772  THE EDITOR PARAMETER '1' IS UNKNOWN.

EXPLANATION:
An invalid parameter was entered in the Editor Parm's field of the
Edit Options panel.

ACTION:
Refer to the Reference Manual or the Reference Card for a list of
valid editor parameters.
```

10

COBOL Compiler Options

Introduction

These tables list the COBOL compiler options used by SmartTest. Each table contains the options that apply to a particular type of compiler, such as COBOL II or later.

Note:

A SmartTest Analyze automatically forces these options to their required settings, so your standard compile/link JCL is usually acceptable. For the SYSPRINT listing, the SmartTest Analyze tries to approximate the options provided by your site defaults and JCL. For example, if you specify NOSOURCE, the SmartTest Analyze forces SOURCE for the compile, then strips the source statements out of the output listing before printing.

This table describes the COBOL II (and later) compiler options:

Required Compiler Option	Related Option	Comments
LIST	NOOFF SET	Required to establish the location of verbs and paragraph/section names.
MAP		Required to establish the location of data items in the user's load module.
NONUM		Required for compiler generated line numbers.
NOOFFSET	LIST	Required because OFFSET overrides LIST.
SOURCE		Required.
NOOPTIMIZE		Highly recommended for testing.

This table describes the CA-OPTIMIZER II compiler options:

Required Compiler Option	Related Option	Comments
LIST	NOOFF SET	Required to establish the location of verbs and paragraph/section names.
MDMAP		Required to establish the location of data items in the user's load module.
NONUM		Required for compiler generated sequence numbers.
NOOFFSET	LIST	Required because OFFSET overrides LIST.
SOURCE		Required.

Compiler Limitations

COPYLIBs With Debug Limitations

Currently, Analyze ignores copybooks (i.e., does not expand) that are flagged as DEBUG statements when the DEBUG option is not active. The COBOL compiler expands these entries, flagging each expansion line as DEBUG (i.e., a comment line since DEBUG is not active). The result is that the line numbers between the COBOL source and the Analyze source are different after the point of the COPYLIB insertion. Such programs produce sequence errors. These programs can be viewed by SmartTest, however, STEPPing, BREAKing, and viewing disassembled code can produce unpredictable results.

TEST Option Limitations

The TEST compiler option adds object code to the program so that the load module produced is usable by the debug tool for the product. If you specify the TEST option, it can adversely effect the SmartTest STEP, LIST COUNTS, BREAK, and ZA commands, as well as the pseudo code commands and statements.

NOTEST is the default and ASG recommends that you do not use the TEST option with SmartTest.

Compiler Optimization Limitations

SmartTest processes COBOL II, and later releases, programs compiled with the OPTIMIZE compiler option, with these limitations:

- When testing with the ASM option ON, the disassembled code that displays for repeated code segments (i.e., embedded PERFORMs or subprograms) are shown only for the first occurrence of such code (i.e., the lowest address).
- When inserting pseudo code into repeated code segments (i.e., embedded PERFORMs or subprograms), you may not insert user labels. This is because all pseudo code variables and labels are global, and insertion of multiple labels (one for each occurrence of the code) causes duplicate labels.
- When inserting pseudo code into repeated code segments (i.e., embedded PERFORMs or subprograms), avoid the use of the &COUNT internal variable. &COUNT maintains a separate instruction count for each separate occurrence of the repeated code. Instead, create a pseudo code variable, increment it on each pass through the pseudo code, and test the pseudo code variable rather than the &COUNT variable.
- When issuing the LIST COUNTS command, the default ordering is by execution address, not line number as stated on the screen. The result is that the same sequence of line numbers displays multiple times, once for each occurrence of repeated code segments; and each repeated line has its own independent execution count.
- Certain source lines may appear to be unexecutable from the Program View screen, and causes unexpected results for commands such as STEP, BREAK, and ZA (Zoom Assembler).

Compiler optimization also affects the reliability of certain SmartTest data display functions. These limitations result from compiling with the OPTIMIZE compiler option:

- A variable value may reside in a register rather than in the storage where the variable has been assigned. SmartTest displays the contents of the storage to which the variable has been assigned.
- In the object code, code associated with one statement can appear to be part of a different statement.
- After optimization, code generated for a statement is dependent on register values loaded by code for preceding statements. Changing the path of flow in a program with the GO command may therefore deprive statements of necessary input.
- A breakpoint may not actually occur at the beginning of the code generated for the statement at which it was set.

For additional information on the effects of compiler optimization, see the Application Programming Guide for your version of COBOL.

Note: _____

ANY compiler optimization can result in relocation, combination, or elimination of underlying code generated for any COBOL statement. The STEP function and insertion of pseudo code in SmartTest can therefore be affected.

ACB

See [application control block \(ACB\)](#).

action bar

The line of keywords at the top of a screen. Each keyword represents a category of actions that may be performed on that screen. An action is selected by moving the cursor to the desired keyword and pressing Enter. See ["Introduction" on page 1](#) for more information.

active program

A program that is being viewed and/or tested on the Program View screen. Also see ["qualified program" on page 299](#).

address command

A command that is entered in the hexadecimal address area within the status box or in any address or offset field displayed on a SmartTest screen. These commands display specific areas of memory such as the current word, 24-bit address, or the 31-bit address. A message can also be displayed that indicates to where the 24-bit or 31-bit address points. See the *ASG-SmartTest Reference Guide* for more information.

address stop

An absolute address and length of storage that is to be monitored during a test session. A program interrupt occurs before the specified area is actually updated.

AKR

See ["Application Knowledge Repository" on page 294](#).

alias

A dataname alias includes a Parent (higher level group item), a Child (lower level item), a RENAMES or REDEFINES, or an 88 level item.

alias name

The name of a program entry point. Alias names are shown on the AKR Directory and Module Directory screens.

alias of

A field on a pop-up listing entries in the AKR. If the analyzed program contains an ENTRY point, Alias Of is the name of the program which contains the ENTRY point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias Of is the name that was entered in the AKR program name field on the File - Analyze Submit pop-up.

analyze

The process used by SmartTest to prepare a COBOL or Assembler program for testing. See ["analyzer" on page 294](#).

analyzer

The batch component of SmartTest that executes complex algorithmic formulas to analyze the complexities of a COBOL or Assembler program. It produces a detailed analysis of the elements and relationships for a program, then places this information in the Application Knowledge Repository (AKR). There are two analyzers. The Program Analyzer analyzes COBOL programs. The Assembler Analyzer analyzes Assembler programs. The PL/I Analyzer analyzes PL/I programs.

analyze options

Run-time options that control the Program Analyzer processing. Many of these options are similar to the COBOL compiler options. Default values are established at installation time and can be overridden by editing the Analyzer JCL or by using the Analyze screens. ["Analyze Options" on page 252](#) contains a complete description of each Analyze option. Analyze Options are not valid for the Assembler Analyzer or the PL/I Analyzer.

analyzer summary

A summary of the run-time statistics and diagnostic messages that are produced when an Analyze job completes.

application control block (ACB)

A control block for IMS/DC which contains the PSB and DBD, and is created by the ACBGEN process.

Application Knowledge Repository

The Application Knowledge Repository component of SmartTest. The AKR is a BDAM or VSAM file organization that contains all analysis information produced by the Program Analyzer, the PL/I Analyzer, or Assembler Analyzer. SmartTest supports concatenated AKRs.

application plan

See [plan name](#).

Assembler analyzer

See [analyzer](#).

Batch terminal simulator (BTS)

An IBM product that allows execution of IMS/VS DB/DC applications in a TSO or batch environment.

Batch test session

A SmartTest test session that is established in an MVS batch region, to which an online test session can be connected. This feature allows batch program testing to be performed interactively.

breakpoint

An interruption that occurs during the execution of a program being tested. Breakpoints result from a BREAK command or an error condition.

BTS

See [Batch terminal simulator \(BTS\)](#).

COBOL subset

COBOL verbs of a similar nature that have been grouped together. For example, READ, WRITE, OPEN, and CLOSE are grouped into the IO subset. The LIST SUBSETS command can be entered to display all subsets online. See the LIST SUBSETS command in the *ASG-SmartTest Reference Guide* for a complete description of each COBOL subset.

command input area

The field on SmartTest screens where primary commands are entered, indicated by ==>> on the second line of the screen.

current cursor location

Refers to the COBOL source statement where command processing begins. The current cursor location can be one of these based on the cursor position:

- If the cursor is in the command input area, the current location is the first source code line on this screen.
- If the cursor is in the line command input area, the current location is the beginning of that line.
- If the cursor is in the source code area, the current location is the cursor position.

cursor character

A substitution character that can be used in commands. Type the command and cursor characters in the command input area, place the cursor on the desired token, then press Enter. SmartTest locates the cursor and reads the specified token as part of the command. The cursor character is set on the Parameter Definition screen. See the *ASG-SmartTest Reference Guide* for more information.

dataname

A standard term for fields defined in the DATA DIVISION of a COBOL program or for a data area label in an Assembler program. Variable names, files, groups, array elements, and fully qualified datanames.

data usage

Defines how a data item is used: DEF indicates the statements in the DATA DIVISION where the data item is defined; USE indicates the statements where the value is used or tested; MOD indicates the statements where the value is set or modified; REF indicates any of the above conditions.

database description (DBD)

A control block that describes the physical structure of a database in an IMS environment.

DBCS

See [Double Byte Character Set \(DBCS\)](#).

DBD

See [database description \(DBD\)](#).

DB2

An IBM relational database management system.

DB2 Stored Procedure

A user-written program that resides on a DB2 server and is invoked by an EXEC SQL CALL statement. A stored procedure is a Language Environment compliant program written in COBOL, C/370, PL/I, or Assembler. The SmartTest DB2 Stored Procedure option enables programmers/analysts to use SmartTest features to interactively test DB2 Stored Procedures.

decision option

Choices listed on the Trace Decision menu. Decision options represent the branches in the path that the TRACE command is following.

decision point

A statement that causes a branch in the execution path of the program. It could be a PERFORM, conditional statement, exception condition, input statement, GO TO, or LABEL.

debug

The process of locating and correcting program errors.

diagnostic message

An informational or error message generated by the online and batch components. Online - A short message displays in the upper right corner of the screen (if available). A long message displays when you type `HELP` or press `PF1/13`. Batch - Messages are included in the Analyzer Summary.

DL/I | DL/1

The database (DB) portion of the IMS system.

Double Byte Character Set (DBCS)

A character set that uses two bytes to represent each character. Various Double Byte Character Sets are used with languages such as Chinese and Japanese which cannot be represented with single byte codes.

equate

A substitution name for a character string. The substitution name is created using the `EQUATE` command. Long commands, patterns, datanames, etc. can be equated to a substitution name.

IMS

An IBM hierarchical database management system.

label name

Any `PROCEDURE DIVISION` paragraph or section name and the `PROCEDURE` and `PROC` literals.

line command

An abbreviated keyword that is entered in the line command area (columns 1 through 6) on the screen.

list file

A file that is allocated the first time the `LPRINT` command is executed. This file is used for `LPRINT` output.

log file

A file that is allocated upon entry into SmartTest. This file is used for error messages and log commands.

long message

A diagnostic or error message that displays near the bottom of SmartTest screens. Long messages are sometimes preceded by short messages that are displayed in the upper right corner of the screen. Pressing `PF1/13` after receiving a short message displays the corresponding long message.

member

A member in a PDS or source manager such as Panvalet or Librarian. This is the alias name found in the Application Knowledge Repository (AKR).

module

A link edited member of a load library PDS. A module can contain several programs.

monitored storage

The data monitored by specifying the absolute addresses and length of storage areas on the Address Stop Entry screen. All storage areas listed on this screen are monitored by SmartTest and a program interrupt occurs before a specified storage area is actually updated.

network

The result of a FLOW command. It consists of all possible executable statements that reach from the starting point of the command to all possible targets. The next FLOW command replaces the previous network with the new results.

path

Any group of executable statements that describe possible execution flows of the COBOL program. FLOW creates paths called NETWORK, and TRACE creates a path called TRACK.

PCB

See [program communication block \(PCB\)](#).

perform range

A perform range consists of the source code contained in a PERFORM statement, and includes all code that is executed as a result of GO TOs, PERFORMs, etc. within that PERFORM.

plan name

A DB2 application plan that contains information relating a program to the data used by the program. A DB2 plan is the output of the BIND process.

PL/I analyzer

See [analyzer](#).

pop-up

A window that displays as the result of selecting an item on a pull-down or pop-up, or as the result of entering certain commands. It is superimposed on the screen to allow entry of information for the requested action. See ["Introduction" on page 1](#) for more information.

primary command

A series of keywords that are entered in the command input area of the screen.

program

Program source member name, the name specified in the IDENTIFICATION DIVISION of a COBOL program, or the CSECT name of a program that is not COBOL.

program analyzer

See [analyzer](#).

program communication block (PCB)

A definition of the database view used by an application program in an IMS environment.

program specification block (PSB)

All Program Communication Blocks (PCBs) are contained in a PSB. PCBs define the view of a database to an application program in an IMS environment.

PSB

See [program specification block \(PSB\)](#).

pseudo code

Statements that are temporarily inserted into program source during a test session. The syntax of this code is compatible with COBOL.

pull-down

The list that displays when an action is selected on the action bar. On a pull-down, actions followed by ... display a pop-up when selected. Actions not followed by ... immediately activate internal commands. See ["Introduction" on page 1](#) for more information.

punch file

A file that is allocated the first time the LPUNCH command is executed. This file is used for LPUNCH output.

qualified program

A program that displays on the Program View screen using the QUALIFY command, while another program is being viewed and/or tested. The program specified with the QUALIFY command is a qualified program; the program being viewed and/or tested is an active program.

SBCS

See [Single Byte Character Set \(SBCS\)](#).

screen subsets

Lines that have been acted upon by an interactive command that have caused them to be in one of these screen display set types:

Highlighted | HI

NONHighlighted | NHI

Excluded | X

NONExcluded | NX

screen transfer feature

This feature can be used on any SmartTest screen that displays a screen number in the upper right corner. Users can immediately transfer to a numbered screen from any other SmartTest screen by entering a = followed by the desired screen number in the command input area.

script file

A sequential file that contains a predefined sequence of commands that can be read and executed.

short message

A diagnostic or error message that displays in the upper right corner of SmartTest screens. Pressing PF1 after receiving a short message displays the corresponding long message.

Single Byte Character Set (SBCS)

A character set that uses one byte to represent each character. Single Byte Character Sets are used with languages such as English where the characters can be represented with a one-byte code.

status box

A box that displays at the bottom of SmartTest screens during a test session. Current status information for the program being tested is shown. This information includes the statement number, offset, source statement, program name, module name, date, and time. When the Assembler mode is set ON, the status box also shows the AMODE value, Assembler instructions, hexadecimal value, and the general register contents. The status box can be removed from the screen using the SET STATUS OFF command.

Storage Management Subsystem (SMS)

An operating environment that helps automate and centralize the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

Stored Procedure

See [DB2 Stored Procedure](#).

subnet

A portion of the NETWORK that reaches from the starting point to one result. Created by the FLOW command as part of the NETWORK. Each portion of the NETWORK that reaches a target is assigned a number (SUB1...SUB n).

subset

A COBOL subset, screen subset, or tagged lines subset. See the *ASG-SmartTest Reference Guide* for detailed information on subsets.

symbol extractor

A post-compile batch processor that extracts COBOL symbol and verb information from the COBOL compiler source listing for input into the Program Analyzer.

tagged lines subsets

Information shown in columns 73 through 80 as the result of a FLOW or TRACE command. These tags can be used as subsets in commands that accept subsets as targets:

- TAGged | TAGS
- TARGet | TGT
- DECision
- STArt
- OPTions

target

An object of a SmartTest primary command.

TCA

See [Test Coverage Analysis](#).

Test Coverage Analysis

SmartTest-TCA, Test Coverage Analysis, is a testing option that measures test suite coverage for a set of programs. It fosters a systematic approach to assess the quality of testing performed in the internal operation of programs. Test Coverage Analysis is a tool for application systems testing and quality assurance testing, to help ensure that the program routines are covered by data execution (i.e., to what extent the data exercises the program). Test Coverage Analysis records the program's reaction to the data presented, without user intervention (in order to preserve the integrity of the results). For detailed information, see the *ASG-SmartTest TCA User's Guide*.

token

A contiguous set of characters preceded by and followed by a blank, period, comma, or parenthesis.

TRACK

The result of a TRACE command. TRACK is the set of lines captured while moving through program logic selecting the branch options.

TSO test session

A SmartTest test session that is executed in a TSO/ISPF environment. Batch JCL is converted to an allocation CLIST and is used to establish the TSO test session.

Symbols

& (retain) command 222
&COUNT command 18

A

abend codes screen 284
abend conditions 151
ACB
 see application control block
ACB allocation 87
action
 Run 148
 STEP 149
action bar
 definition 293
 description 7
activating/deactivating pseudo code 177
active program 6, 293
address
 space 96
 swapped space 96
ADDRESS commands, definition 293
address stop
 definition 293
 setting 152
 unavailable 152
AKR
 allocating 35
 allocation report 37
 dataset name 230
 in analyze process 225, 244
 parameter 234
 utilities 35
alias
 definition 293
 name, program entry point 293
 searching for 192
ALIAS operand 22
Alliance 3, 8, 155
 accessing from ESW screen xvi
 description xiii
 linking xvi, 183

ALLOC parameter 56
allocating
 alternate PCB output dataset 109
 an AKR 35
 BTS program output 111
 BTSDEBUG 115
 BTSPUNCH input dataset 113
 BTSSNAP 117
 IEFRDER dataset 90
 IMS ACB 87
 IMS and DB2 load library datasets 84
 IMS backout and recovery datasets 90
 IMS IMSMON 88
 IMS procedure libraries 85
 ISPF file 70
 product files 29
 PSB and DBD library datasets 86
 QALTPCB 107
 QALTRAN 109
 QIOPCB 106
 SNAP dump dataset 115
 VSAM buffer pools 85
allocation CLIST 56, 122
alternate PCB 103, 109
analysis, Insight 11
analyze
 a COBOL program 225
 adding to compile mechanism 246
 automatic JCL modifications 237
 CICS JCL after modifications 244
 CICS JCL before modifications 242
 compile/link JCL 226
 features 227
 Insight 11
 job submit 228
 methods 39, 227
 options 227
 Panvalet JCL after modifications 241
 Panvalet JCL before
 modifications 239
 parameters 234
 process 227

- program 38
- requirements 226
- source program 226
- Summary report 244
- using File - Analyze Submit
 - pop-up 228
- using ISPF 231
- using ISPF/PDF Edit 232
- verifying results 44
- VIAIN DD statement 237
- VIASUB edit macro 230, 232, 236
- VIASUBDS CLIST 228, 231, 236
- ANALYZE command 228
- analyze option
 - BUF 252
 - COB2R3 253
 - COBOL370 253
 - COBOLII 253
 - DB2LIB 253
 - DB2PLAN 253
 - DYNCALL 254
 - FLAG 255
 - INPUT 255
 - IO 256
 - LANGLVL 256
 - LANGLVL override 253
 - LINECNT 257
 - MAIN 257
 - NOCOBOLII 253
 - NODYNCALL 254
 - NOINPUT 255
 - NOIO 256
 - NOOUTPUT 257
 - NORECUR 258
 - NORETURN 258
 - NOSEQ 259
 - NOSOURCE 259
 - OUTPUT 257
 - PROGRAM 258
 - RECUR 258
 - RETURN 258
 - SEQ 259
 - SOURCE 259
 - SPACE 259
 - SQLID 260
 - SUBSYS 260
 - XLIVE 260
 - XMEM 261
- Analyze Submit Parameters screen
 - example 234
 - options 235
- Analyze Submit screen
 - example 229
 - field descriptions 229
- Analyze Summary report 258
- AOPT parameter 234
- application
 - control block 294
 - group name 93
 - parameters 122
- Application Knowledge Repository
 - see AKR*
- APS Generator Options screen 269
- APS painter code 270
- ASG-SmartTest/ASG-Alliance Interface
 - pop-up
 - example 184
 - field descriptions 185
- Assembler
 - integration 3
 - support 248
- ATTN key usage 136, 150
- AutoChange 155
 - accessing from ESW screen xvi
 - description xiii
- B**
- backtrack
 - buffer size 168
 - execution history 167
 - recording 167
 - reviewing 168
 - set buffer size 168
 - variable history 172
- Batch Connect facility
 - using to test 130
 - when to use 130
- Batch Session Setup screen
 - example 131
 - field descriptions 132
- Batch terminal simulator 295
- Batch test session 134, 295
- BIND 122
- BMP
 - database region type 99
 - parameters 91
 - region 93
- BR (break) command 153
- BRANCH command 12, 211
- branching logic 211
- BREAK command 18, 22, 153–154
- break on entry 99, 122
- breakpoints
 - automatically setting 8
 - automatically setting at date-related data items 159
 - definition 295
 - displaying 160

- inserting manually 153
 - processing 153
 - removing all 161
 - removing individually 161
 - setting automatically 153
 - setting with impact datasets 155
 - Breakpoints List screen 158
 - Bridge
 - accessing from ESW screen xvi
 - description xiii
 - BTS
 - allocating BTS program output dataset 111
 - allocating input dataset 113
 - allocating QALTPCB 107
 - allocating SNAP dump dataset 115, 117
 - alternate PCB output 109
 - definition 295
 - format libraries 105
 - input 103
 - QIOPCB allocation 106
 - specifying load library dataset 104
 - STAX indicator 104
 - system parameters 104
 - system variables 104
 - work file 103
 - BTS Allocation Selection screen
 - example 102
 - options 103
 - BTS BTSDEBUG Allocation pop-up
 - example 116
 - field descriptions 116
 - BTS BTSOUT Allocation pop-up
 - example 112
 - field descriptions 112
 - BTS BTSPUNCH Allocation screen
 - example 114
 - field descriptions 114
 - BTS BTSSNAP Allocation pop-up
 - example 118
 - field descriptions 118
 - BTS Format Libraries pop-up 105
 - BTS Load Library pop-up 104
 - BTS QALTPCB Allocation pop-up
 - example 108
 - field descriptions 108
 - BTS QALTRAN Allocation pop-up
 - example 110
 - field descriptions 110
 - BTS QIOPCB Allocation pop-up
 - example 106
 - field descriptions 106
 - BTS Session Setup screen
 - example 98
 - options 99
 - BTSDEBUG allocation 115
 - BTSIN Dataset 100
 - BTSOUT allocation 111
 - BTSPUNCH allocation 113
 - BTSSNAP allocation 117
 - BUF analyze option 252
 - buffer size
 - backtrack 168
 - buffer size, backtrack 168
 - buffers
 - in analyze job 252
 - page-fixed 93
- C**
- CALL interface 123
 - CALLED programs 255–256
 - CALLED subprogram 257
 - CANCEL command 161
 - Center, description xiii
 - changing data values 167
 - changing execution sequence 149
 - checkpoint/restart ID 93, 96
 - CICS compile/analyze JCL 242
 - CLIST
 - VIAPUBTS 104
 - VIAPUSPF 72
 - VIASUBDS 228, 231
 - CMPL parameter 234
 - CNTL member
 - VIAPALSC 183
 - VIAPQ* 183
 - COB2R3 analyze option 253
 - COBOL
 - CASE generated COBOL support 4
 - COBOL II support 4
 - compiler options 289
 - OF clause 21
 - program structure 12
 - subsets 188
 - verbs 188
 - COBOL II in analyze job 253
 - COBOL II Optimize
 - considerations 18
 - GO command caution 18
 - pseudo code in repeated segments 18
 - pseudo code user labels 18
 - repeated code segments 18
 - unexecutable source code 18
 - COBOL II Release 3 in analyze job 253
 - COBOL Intelligent Search 187
 - COBOL370 analyze option 253

- COBOLII analyze option 253
- command scripts 217
- commands
 - & (retain) 222
 - &COUNT 18
 - ADD 173
 - ALLIANCE 184
 - ANALYZE 228
 - BR (break) 153
 - BRANCH 211
 - BREAK 18, 153–154
 - CANCEL 161
 - DROP (Assembler) 267
 - EQUATE 221
 - EXCLUDE 15, 192
 - EXECUTE 218
 - F (first) 194
 - FINDXTND 13, 195
 - FLOW 215
 - GO 149
 - HIGH 209
 - KEEP 165
 - KG (keep group) 165
 - KGH (keep group hex) 165
 - KH (keep hex) 165
 - L (last) 194
 - line 6
 - LIST 181
 - LIST ADSTOP 145
 - LIST BREAKS 160
 - LIST COUNTS 145
 - LIST PSEUDO 177
 - LIST TAILOR 179
 - LIST TRACKING 145
 - LOCATE 220
 - LOCATE * 160
 - LPRINT 15, 182, 209
 - LPRINT * 182
 - LPUNCH 15
 - MOVE 173
 - PREF 16
 - primary 6
 - PRODLVL 223
 - pseudo code list of 174
 - QUALIFY * 178
 - QUALIFY CANCEL 178
 - RECALL 222
 - RESET 165, 178
 - RESET BREAKS 161
 - RESET KEEP 167
 - reusing results 16
 - RUN 145
 - S (show) 194
 - SCROLL 15, 210
 - SET 217
 - SET ASM 18
 - SET BACKTRACK 167–168
 - SET BACKTRACK #M 168
 - SET DELAY 149
 - SET GENERATED 265, 270
 - SET LE 52
 - SET LEARN 144
 - SET LECOND 52
 - STEP 18
 - STEP AUTO 149
 - STOP 152
 - SUBMIT 134
 - SUBTRACT 173
 - TESTPOINT 156
 - USING (Assembler) 267
 - WHEN 173
 - ZA 18
 - ZD (zoom display) 162
 - ZG (zoom group) 164
 - ZGH (zoom group hex) 164
 - ZH (zoom hex) 162
 - ZO (zoom out) 165
 - ZOOMDATA 164
- compiler
 - COBOL options 289
 - limitations 290
- compiler optimize limitations 291
- Connect to Job screen
 - example 135
 - options 136
- continuous execution 148
- controlling execution
 - cancelling a test session 161
 - changing sequence 149
 - executing continuously 148
 - interrupting by keystroke 150
 - locating next statement 160
 - specifying steps 149
- conventions page xix
- Convert Batch JCL facility 100, 122
- Convert Batch JCL screen 55
- converting JCL to CLIST 55
- COPYLIB debug limitations 290
- CPU usage in analyze job 261
- CUA
 - action bar 7
 - overview 2
 - pop-ups 10
 - pull-downs 8
 - screens 7
 - terms 7
- cursor character 295

D

- DATA DIVISION, viewing data 164
- data element 21
- data item size, impact of changing 191
- data usage 13
- data usage, FINDXTND command 13
- database
 - DB2 support 5
 - dynamic backout 97
 - ISM/DB support 5
 - plan name 298
 - VSAM support 5
- Database Recovery Control 97
- databases supported 5
- dataname
 - alias 293
 - DEFINITION 22
 - description 21
 - fully qualified 21
 - group level 21
 - MODIFICATION 22
 - REFERENCE 22
 - USE 22
- dataname type
 - elementary dataname 21
 - file name 21
 - group name 21
 - special name 21
 - table element name 21
- DB monitoring datasets, allocating 88
- DB2
 - allocating load library 84
 - changing non-numeric values 128
 - changing numeric values 127
 - plan 122
 - plan name 298
 - session setup 121
 - specifying test session parameters 124
 - SQL CALL interface 123
 - subsystems 94
 - test session 122
 - testing in Batch environment 140
- DB2 Batch Session Setup screen 140
- DB2 Session Setup screen
 - example 121
 - field descriptions 122
- DB2 Stored Procedure
 - definition 123, 296
 - displayed parameters 126
 - entering test data 126
 - initiating the test 129
 - requirements 123
 - session setup 123
 - setting up the test 124
 - testing option 123
- DB2 Stored Procedures Numeric Display screen 127
- DB2 Stored Procedures Parameters screen 126
- DB2 Stored Procedures Setup screen 124
- DB2LIB analyze option 253
- DB2PLAN analyze option 253
- DBB
 - database region type 99
 - parameters 94
- DBCS
 - see Double Byte Character Set*
- DBD allocation 86
- DBD libraries 83
- DD statements
 - BTS load library 104
 - BTSDEBUG 115
 - BTSOUT 111
 - BTSPUNCH 113
 - BTSSNAP 117
 - DFSRESLB and DFSESL 84
 - DFSVSAMP and PROCLIB 85
 - FORMAT 105
 - IEFRDER 90
 - IMSACB 87
 - IMSMON 88
 - PSB and DBD 86
 - QALTPCB 107
 - QALTRAN 109
 - QIOPCB 106
 - VIAIN 237
- DEALC parameter 57
- DFHDRP
 - Batch JCL 141
 - testing 141
- DFSESL allocation 84
- DFSRESLB allocation 84
- DFSVSAMP allocation 85
- Dialog Management programs, intercepting 147
- DIRECT operand 22
- displaying memory
 - see ADDRESS command*
- DLI database region type 99
- DLI parameters 94
- Double Byte Character Set 297
- DROP command 267
- DSCHK parameter 234
- dynamic calls in analyze job 254
- DYNCALL analyze option 254

E

- edit macro, VIASUB 232
- EDIT parameter 234
- Encore
 - accessing from ESW screen xvi
 - description xiv
- ending a test session 65
- ENS parameter 234
- entering/editing pseudo code 176
- entry point, alias name 293
- equate 297
- Estimate 155
 - accessing from ESW screen xvi
 - description xiv
- ESW
 - description xii
 - integration 3
 - invoking products xv
 - product integration xvi
- EXCLUDE command 15, 22, 192
- executing programs continuously 148
- executing pseudo code 177
- execution environments supported 4
- execution path 298
- execution sequence search 215
- Execution Tracking screen 146
- Existing Systems Workbench
 - see ESW
- EXIT PROGRAM statements in analyze job 257
- exiting a test session 65
- exiting SmartTest 162

F

- F (first) command 194
- fast path
 - data buffers 93
 - databases 93
- FDDL (Formatted Dump Delete List) 96
- File - Analyze Submit pop-up 252
- File pull-down
 - description 7
 - Open option 178
 - using 45
- FINDXTND
 - command 12, 22, 195
 - displaying data usage 13
 - results 15
- FLAG analyze option 255
- flag messages in analyze job 255
- FLOW command 215
- FLOW, system-generated path 19
- format libraries 105

- Formatted Dump Delete List (FDDL) 96
- formatted dump output 96

G

- GO command
 - COBOL II Optimize 18
 - using 149
- GOBACK statements in analyze job 257
- group item 21

H

- help
 - for abends 283
 - for commands 279
 - for general information 281
 - for messages 284
 - for screens 278
 - for specific information 282
 - index 282
 - NOTES 280
 - online 275
 - printing messages 285
 - requesting 277
- Help Explanation and Action screen 285
- HI screen subset 17
- HIGH command 22, 209

I

- IEFRDER allocation 90
- impact datasets 155
- impact of change
 - data item size 191
 - value of data item 192
- IMS
 - AGN execution parameter 93
 - allocating ACB library datasets 87
 - allocating load library 84
 - allocating procedure libraries 85
 - backout and recovery datasets 90
 - BKO execution parameter 97
 - BMP execution parameters 83
 - CKPTID execution parameter 93, 96
 - control region 92-93
 - CPUTIME execution parameter 93
 - DBB execution parameters 83
 - DBRC execution parameter 97
 - DIRCA execution parameter 93
 - DLI execution parameters 83
 - EXCPVR execution parameter 96
 - execution parameters for DLI/DBB programs 94
 - file allocation 120
 - FMTO execution parameter 96

- identifier 96
- IEFRDER allocation 90
- IMSCTRL macro 97
- IMSID execution parameter 93, 96
- IMSMON allocation 88
- IN execution parameter 92
- IRLM execution parameter 97
- IRLMNM execution parameter 97
- load library 83
- Load library datasets 104
- load library datasets 72, 105
- LOGA execution parameters 96
- logging facility 83
- Message Format Services 103
- MON execution parameter 96
- monitoring 96
- NBA execution parameter 93
- OBA execution parameter 93
- OPT execution parameter 92
- OUT execution parameter 92
- PARDLI execution parameter 93
- PREINIT execution parameter 94
- PRLD execution parameter 93, 96
- RST execution parameter 96
- SPIE execution parameter 93, 95
- SRCH execution parameter 96
- SSM execution parameter 94
- STIMER execution parameter 93
- SWAP execution parameter 96
- system parameters 72, 83, 104
- system variables 72, 83, 104
- TEST execution parameter 93, 95
- IMS ACB Allocation pop-up 87
- IMS Allocation Selection pop-up
 - example 82
 - options 83
- IMS BMP parameters
 - AGN 93
 - CKPTID 93
 - CPUTIME 93
 - DIRCA 93
 - IMSID 93
 - NBA 93
 - OBA 93
 - OPT 92
 - OUT 92
 - PARDLI 93
 - PREINIT 94
 - PRLD 93
 - SPIE 93
 - SSM 94
 - STIMER 93
 - TEST 93
 - using 91
- IMS BMP Parameters pop-up
 - example 92
 - field descriptions 92
- IMS DFSRESLB/DFSESL Allocation
 - pop-up
 - example 84
 - field descriptions 84
- IMS DLI/DBB parameters
 - BKO 97
 - CKPTID 96
 - DBRC 97
 - EXCPVR 96
 - FMTO 96
 - IMSID 96
 - IRLM 97
 - IRLMNM 97
 - LOGA 96
 - MON 96
 - PRLD 96
 - RST 96
 - SPIE 95
 - SRCH 96
 - SWAP 96
 - TEST 95
- IMS DLI/DBB Parameters pop-up
 - example 95
 - field descriptions 95
- IMS IEFRDER Allocation pop-up
 - example 90
 - field descriptions 91
- IMS IMSMON Allocation pop-up
 - example 88
 - field descriptions 89
- IMS PROCLIB/DFSVSAMP Allocation
 - pop-up
 - example 85
 - field descriptions 86
- IMS PSB/DBD Allocation pop-up 86
- IMS Resource Lock Manager (IRLM) 97
- IMS/DB Session Setup screen
 - example 79
 - field descriptions 80
- IMS/OSAM, buffer pool size 95
- IMSCTRL macro 97
- IMSMON allocation 88
- IN clause 16–17
- INDIRECT operand 22
- initiating a test session 61
- INPUT analyze option 255
- INPUT statements in analyze job 255–256
- INS parameter 234
- insert breakpoints 153
- inserting pseudo code 176

- Insight
 - accessing from ESW screen [xvi](#)
 - analysis [11](#)
 - benefits [11](#)
 - description [xiv](#)
 - ESW integration [3](#)
 - using analysis functions [xvi](#)
 - insufficient memory in analyze job [261](#)
 - integration
 - Assembler [3](#)
 - ESW [3](#)
 - intercepts
 - attached programs [147](#)
 - Dialog Management programs [147](#)
 - LINKed to programs [147](#)
 - internal tables in analyze job [252](#)
 - inter-region communication area [93](#)
 - interrupting execution, ATTN key [150](#)
 - IO analyze option [256](#)
 - IRLM (IMS Resource Lock Manager) [97](#)
 - ISAM/OSAM, buffer [96](#)
 - ISPF
 - dialogs [246](#)
 - user interface [ix, 2](#)
 - ISPF allocation [70](#)
 - ISPF List Data Set Allocation pop-up [75](#)
 - ISPF Log Allocation [77](#)
 - ISPF Panel/Link Library pop-up [73](#)
 - ISPF Program Load Library pop-up [72](#)
 - ISPF Session Setup screen
 - example [68](#)
 - field descriptions [69](#)
 - ISPF Table/Message/ Skeleton Library pop-up [74](#)
- J**
- JCL to CLIST conversion [55](#)
 - job pack area (JPA) [96](#)
 - JOBLIB/STEPLIB [96](#)
 - JPA (job pack area) [96](#)
- K**
- K (keep) command [165](#)
 - KEEP command [165](#)
 - Keep window
 - globally remove [167](#)
 - individually remove [167](#)
 - KG (keep group) command [165](#)
 - KGH (keep group hex) command [165](#)
 - KH (keep hex) command [165](#)
- L**
- L (last) command [194](#)
 - label name [20](#)
 - LANGLVL analyze option [256](#)
 - Language Environment testing [52](#)
 - language level in analyze job [256](#)
 - language support
 - Assembler [ix, 2](#)
 - COBOL [ix, 2, 4](#)
 - languages supported [4](#)
 - LE
 - see Language Environment*
 - LIB() field [140](#)
 - limitations
 - compiler [290](#)
 - COPYLIB with debug [290](#)
 - limiting a search [192](#)
 - line count in analyze job [257](#)
 - line range [20](#)
 - LINECNT analyze option [257](#)
 - link pack area (LPA) [96](#)
 - LINKed programs, intercepting [147](#)
 - LINKLST [96](#)
 - List - Analyze Submit pop-up [146](#)
 - List - BackTrack Variable History screen [172](#)
 - List - User Marks pop-up [19](#)
 - LIST ADSTOP command [145](#)
 - LIST BREAKS command [160](#)
 - LIST command [181](#)
 - LIST COUNTS command [145](#)
 - LIST PSEUDO command [177](#)
 - List pull-down [8](#)
 - LIST TAILOR command [179](#)
 - LIST TRACKING command [145](#)
 - live exits
 - in analyze job [260](#)
 - load library datasets for IMS and DB2 [84](#)
 - Load Module Intercept List pop-up [52](#)
 - Load Module Intercept List pop-up [70](#)
 - Load Module Intercept screen [147](#)
 - LOCATE * command [160](#)
 - LOCATE command [220](#)
 - locating next executable statement [160](#)
 - log/list/punch/work file definition [30](#)
 - logging access method [96](#)
 - Logic pull-down [8](#)
 - logical terminal name [92](#)
 - long message area [6](#)
 - LPA (link pack area) [96](#)
 - LPRINT * command [182](#)
 - LPRINT command [15, 22, 182, 209](#)
 - LPUNCH command [15, 22](#)

M

macro
 displaying expanded Assembler 265
 IMSCTRL 97
 VIASUB 39, 228
 MAIN analyze option 257
 main program
 in analyze job 257
 main program in analyze job 257
 main storage in analyze job 252
 Mark name
 copying 19
 deleting 19
 description 19
 listing 19
 merging 19
 renaming 19
 Memory Display screen 128
 memory in analyze job 261
 Message Format Service (MFS) 105
 Message Format Services (MFS) 103
 message queues 92
 messages
 in analyze job 255
 long 6, 284
 printing 285
 short 6, 284
 MFS (Message Format Services) 103
 MOD operand 192
 modifying program logic 173
 MONITOR 52
 execution monitor 145
 guidelines 146
 monitor and change data 162
 monitor dataset 83
 multiple program processing 178
 MVS
 Batch session 295
 environment ix, 2, 4

N

network
 definition 298
 NOALIAS operand 22
 NOCMPL parameter 234
 NOCOBOLII analyze option 253
 NODSCHK parameter 234
 NODYNCALL analyze option 254
 NOENS parameter 234
 NOINPUT analyze option 255
 NOINS parameter 234
 NOIO analyze option 256

NOMONITOR

 execution monitor 145
 guidelines 146
 test session 52
 NOOUTPUT analyze option 257
 NOPANEL parameter 234
 NORECUR analyze option 258
 NORETURN analyze option 258
 NOREUS parameter 234
 NOSD parameter 234
 NOSDR parameter 234
 NOSDX parameter 234
 NOSEQ analyze option 259
 NOSOURCE analyze option 259
 NOST parameter 234
 NOTES in help 280

O

Open option 178
 operating modes 145
 Optimization in COBOL II
 pseudo code in repeated segments 18
 repeated code segments 18
 optimization in COBOL II
 considerations 18
 GO command caution 18
 pseudo code user labels 18
 unexecutable source code 18
 options
 setting 180
 setting and verifying 28
 Options - Allocation Definition pop-up 134
 Options - Script File Allocations pop-up 32
 Options Menu 28
 Options pull-down 8
 OUTPUT
 analyze option 257
 parameter 234
 statements in analyze job 256

P

PA1 key usage 136
 page-fixed buffers 93
 painter code, APS 270
 PANEL parameter 234
 Panvalet compile/analyze JCL 239
 path, execution 298
 pattern string 23
 PCB
 alternate 103
 alternate output 103
 PERFORM range 20
 Perfrange 20

- PF key definitions 33
 - PGM parameter 234
 - PL/I 273
 - plan name 298
 - pop-up
 - AKR Utilities 35
 - example 10
 - List - User Marks 19
 - Log/List/Punch Definition 30
 - Options - Script File Allocations 32
 - PF Key Definition 33
 - Procedure Libraries 51
 - Product Allocations 29
 - Product Parameters 29
 - System Load Libraries 50
 - View - Paragraph Cross Reference 16
 - pop-up field types 10
 - PREF command 16
 - Preview BTSIN Dataset screen
 - example 101
 - field descriptions 101
 - print program information 182, 209
 - printing messages 285
 - PROC, literal 20
 - procedure libraries 51
 - PROCLIB allocation 85
 - PROONLY parameter 234
 - PRODLVL command 223
 - product allocations 29
 - product integration xvi
 - product parameters 29
 - program
 - abends 151
 - analyze 38–39
 - entry point 293
 - VIASMPRT 285
 - program analyze
 - methods 39
 - requirements 38
 - PROGRAM analyze option 258
 - program execution, controlling 148
 - program information, displaying 181
 - program name
 - description 20
 - in analyze job 258
 - program structure, exposing 12
 - Program View screen
 - batch test initiation 136
 - example 5, 7, 159
 - line numbers 20
 - tag subsets 190
 - PROGRAM-ID statement 258
 - PSB allocation 86
 - PSB libraries 83
 - pseudo code
 - &COUNT 175
 - 77 175
 - activating/deactivating 177
 - ADD 174
 - BREAK 174
 - COBOL compatible 3
 - COBOL II Optimize 18
 - COBOL RESERVED WORDS 176
 - command list 174
 - definition 299
 - entering/editing 176
 - executing 177
 - GO TO 174
 - IF/THEN/ELSE 174
 - MOVE 174
 - pslabel 175
 - removing 178
 - SUBTRACT 175
 - symbols allowed 176
 - using in SmartTest 173
 - viewing 177
 - WHEN 175
 - Pseudo Code List screen 177
 - pull-down
 - descriptions of CUA features 9
 - example 9
 - File 7
 - List 8
 - Logic 8
 - Options 8
 - Search 8
 - selecting actions 8
 - Test 8, 148
 - View 7
- Q**
- QALTPCB allocation 107
 - QALTRAN allocation 109
 - QIOPCB allocation 106
 - qualified program 6, 299
 - QUALIFY * command 178
 - QUALIFY CANCEL command 178
 - qualifying a search 192
- R**
- RECALL command 222
 - Recap
 - accessing from ESW screen xvi
 - description xiv
 - recording, backtrack 167
 - Recoverable Resource Manager 125
 - RECUR analyze option 258

- recursion in analyze job 258
- Recursion report 258
- release and level numbers, SmartTest 223
- removing
 - data windows 165
 - Keep windows 167
 - pseudo code 178
- repositioning the display 210, 220
- RESET BREAKS command 161
- RESET KEEP command 167
- RESET PSEUDO Command 178
- RESET ZOOM command 165
- resource consumption, reducing 130
- RETURN analyze option 258
- return of CALL in analyze job 258
- REUS parameter 146, 234
- reusing command results 16
- reviewing execution history 168
- RRSAF 125
- RUN action 148
- RUN command 145

- S**
- S (show) command 194
- scratch pad area 102
- screen
 - Analyze Submit 229
 - Analyze Submit Parameters 234
 - APS Generator Options 269
 - ASG Abend Codes 284
 - ASG-SmartTest/ASG-Alliance Interface 184
 - Batch Session Setup 131
 - Batch Submit 141
 - Breakpoints List 158
 - BTS Allocation Selection 102
 - BTS BTSDEBUG Allocation 116
 - BTS BTSOUT Allocation 112
 - BTS BTSPUNCH Allocation 114
 - BTS BTSSNAP Allocation 118
 - BTS Format Libraries 105
 - BTS Load Library 104
 - BTS QALTPCB Allocation 108
 - BTS QALTRAN Allocation 110
 - BTS QIOPCB Allocation 106
 - BTS Session Setup 98
 - command input area 6
 - Connect to Job 135
 - Convert Batch JCL 55
 - DB2 Batch Session Setup 140
 - DB2 Session Setup 121
 - DB2 Stored Procedures Numeric Display 127
 - DB2 Stored Procedures Setup 124
 - Execution Tracking 146
 - File - Analyze Submit 252
 - format 5
 - general field descriptions 5
 - Help Explanation and Action 285
 - IMS ACB Allocation 87
 - IMS Allocation Selection 82
 - IMS BMP Parameters 91–92
 - IMS DFSRESLB/DFSESL/MODBLKS Allocation 84
 - IMS DLI/DBB Parameters 95
 - IMS IEFORDER Allocation 90
 - IMS IMSMON Allocation 88
 - IMS PROCLIB/DFSVSAMP Allocation 85
 - IMS PSB/DBD Allocation 86
 - IMS/DB Session Setup 79
 - ISPF Allocation 70
 - ISPF List Data Set Allocation 75
 - ISPF Log Data Set Allocation 77
 - ISPF Panel/Link Library 73
 - ISPF Program Load Library 72
 - ISPF Session Setup 68
 - ISPF Table/Message/Skeleton Libraries 74
 - line command input area 6
 - List - BackTrack Variable History 172
 - Load Module Intercept List 70, 147
 - long message area 6
 - Memory Display 128
 - Options - Product Allocation 134
 - Options Menu 28
 - Preview BTSIN Dataset 100
 - Program View 5, 7, 20, 136, 159, 190
 - Pseudo Code List 177
 - short message area 6
 - Stored Procedures Parameters 126
 - subsets 190
 - Test - SmartTest Testpoint Generation 156
 - Test - Testpoint Qualified Program List 157
 - Test Session Tailoring 179
 - TSO Session Setup 53
 - TSO/ISPF Edit 5
 - View - Reset Request 161
- screen subset
 - description 190
 - HI 17
- script
 - files 3
 - STFINDAT 159

- script file allocations
 - selecting option 183
 - setting defaults 32
 - SCROLL command 15, 22, 210
 - scrolling 6
 - SD parameter 234
 - SDR parameter 234
 - SDX parameter 234
 - search
 - IN clause 192
 - intelligent 187
 - limiting 192
 - limiting the scope 16
 - references 192
 - Search pull-down 8
 - SEQ analyze option 259
 - SET ASM command 18
 - SET BACKTRACK #M command 168
 - SET BACKTRACK command 167–168
 - SET DELAY command 149
 - SET GENERATED command 265, 270
 - SET LE command 52
 - SET LEARN command 144
 - SET LECOND command 52
 - SET SCRIPT command 217
 - setting breakpoints automatically 153
 - setup
 - Batch Connect 130
 - BTS in TSO 98
 - considerations 61
 - considerations, TSO foreground 61
 - DB2 in TSO 121
 - DB2 Stored Procedure 123
 - IMS/DB in TSO 79
 - ISPF Dialog Manager 68
 - multiple step job 62
 - MVS in TSO 53
 - TSO foreground 61
 - using wizards 148
 - short message area 6
 - simplifying commands 221
 - SmartDoc
 - accessing from ESW screen xvi
 - description xiv
 - SmartEdit
 - accessing from ESW screen xvi
 - description xv
 - SmartTest
 - accessing from ESW screen xvi
 - description xv
 - displaying release and level 223
 - introduction 1
 - languages 4
 - operation 145
 - supported databases 5
 - supported execution environments 4
 - testing modes 145
 - SmartTest-APS 269
 - SmartTest-Assembler 263
 - SmartTest-PL/I 273
 - SNAP dump 103
 - SOURCE analyze option 259
 - source program listing
 - in analyze job 259
 - source program listing in analyze job 259
 - source statement sequence numbers in analyze job 259
 - SPACE analyze option 259
 - spacing for source listing in analyze job 259
 - SQL CALL interface 123
 - SQLID analyze option 260
 - SRM (System Resource Manager) 96
 - ST parameter 234
 - status box 6
 - STAX indicator 104
 - Step action 149
 - STEP AUTO command 149
 - STEP command 18
 - STFINDAT script 159
 - STOP command 152
 - Stored Procedure 300
 - structure levels 12
 - submitting an analyze job 228
 - subnet 300
 - subset target
 - DECISION 191
 - OPTION 191
 - START 191
 - TAGGED 191
 - TAGS 191
 - TARGET 191
 - subsets
 - COBOL 188
 - description 20
 - screen 190
 - Tag 190
 - SUBSYS analyze option 260
 - subsystem identifier 93, 96
 - system load libraries 50
 - System Resource Manager (SRM) 96
 - system-generated path 19
 - NETWORK 19
 - SUBNETn 19
 - TRACK 19
- T**
- table entry 21
 - Tag subsets 190

- target
 - dataname 21
 - description 19
 - label name 20
 - line range 20
 - Mark name 19
 - pattern string 23
 - program name 20
 - subset name 20
 - TCA definition 301
 - terminating a test session 65, 161
 - terminating SmartTest 162
 - Test - SmartTest Testpoint Generation
 - pop-up 156
 - Test - Testpoint Qualified Program List
 - pop-up 157
 - test initiation in Batch 136
 - Test pull-down
 - description 8, 148
 - example 148
 - test results, displaying 181
 - test session
 - displaying information 181
 - displaying SmartTest
 - release/level 223
 - displaying test results 181
 - ending 65
 - executing continuously 148
 - executing steps 149
 - exiting SmartTest 162
 - finding COBOL subsets 188
 - finding dataname references 191
 - finding screen subsets 190
 - finding tag subsets 190
 - following program branches 211
 - highlighting search results 209
 - impact of data item value change 192
 - initiating 61
 - limiting a search 192
 - modifying program logic 173
 - multiple programs 178
 - options 180
 - positioning the display 210
 - printing displayed information 182
 - printing program information 209
 - recording execution history 167
 - reviewing execution history 168
 - searching in execution sequence 215
 - steps in setting up 44
 - tailoring 179
 - terminating 65
 - using setup wizards 148
 - viewing data 162
 - Test Session Tailoring screen
 - example 52
 - SmartTest features 179
 - testing
 - DB2 Batch environment 140
 - DFHDRP 141
 - Language Environment 52
 - modes 145
 - TESTPOINT command 156
 - token substitution, cursor character 295
 - token/cursor substitution 221
 - Trace table 103
 - TRACE, system-generated path 19
 - transaction code name 101
 - TSO
 - SYSPROC DD statement 57
 - terminal attention exit 104
 - TSO foreground execution 53
 - TSO Session Setup screen
 - example 53
 - field descriptions 54
 - TSO/ISPF environment 4
- U**
- user call list 93, 95
 - user options 28
 - USING command 267
 - Utility Control Facility (UCF) 96
- V**
- variable redefined 21
 - verifying analyze results 44
 - VIAIN DD statement 237
 - VIAPALSC CNTL member 183
 - VIAPDLI program 141
 - VIAPQ* CNTL members 183
 - VIAPUBTS CLIST 104
 - VIAPUIMS CLIST 83
 - VIAPUSPF CLIST 72
 - VIASMPRT program 285
 - VIASUB
 - edit macro 230, 236
 - VIASUB edit macro
 - analyze 232
 - parameters 234
 - VIASUBDS CLIST 228, 231, 236
 - View - Paragraph Cross Reference pop-up
 - description 16
 - example 16
 - field descriptions 16
 - View - Reset Request pop-up 161
 - view breakpoints 160
 - View pull-down 7

- viewing data
 - at top of screen [165](#)
 - changing values [167](#)
 - group items in-line [164](#)
 - in DATA DIVISION [164](#)
 - in-line [162](#)
 - removing data windows [165](#)
 - removing keep windows [167](#)
 - test sessions [162](#)
- viewing multiple programs [178](#)
- viewing pseudo code [177](#)
- VS COBOL in analyze job [253](#)
- VS identifier [96](#)
- VSAM buffer pools [85](#)

W

- wizard, setting up environment [148](#)

X

- XLIVE analyze option [260](#)
- XMEM analyze option [261](#)

Z

- ZA command [18](#)
- ZD (zoom display) command [162](#)
- ZG (zoom group) command [164](#)
- ZGH (zoom group hex) command [164](#)
- ZH (zoom hex) command [162](#)
- ZO (zoomout) command [165](#)
- ZOOMDATA command [164](#)
- ZOOMIN [12](#)
- ZOOMOUT [12](#)

ASG Worldwide Headquarters Naples Florida USA | asg.com