

# ASG-SmartTest™ PLI User's Guide

Version: 6.0

Publication Number: STL0200-60

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](http://asg.com)

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



# ASG Documentation/Product Enhancement Fax Form

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

Company Name	Telephone Number	Site ID	Contact name

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

Enhancement Request:



# ASG Support Numbers

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

## Please have this information ready:

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

## If You Receive a Voice Mail Message:

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

## Severity Codes and Expected Support Response Times

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

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

## ***Business Hours Support***

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

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

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

## **ASG Web Site**

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

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

If you do not have access to the web, FAX your suggestions to product management at (941) 263-3692. Please include your name, company, work phone, e-mail ID, and the name of the ASG product you are using. For documentation suggestions include the publication number located on the publication's front cover.



---

# Contents

---

<b>Preface</b> .....	<b>vii</b>
<b>About this Publication</b> .....	<b>vii</b>
<b>Related Publications</b> .....	<b>viii</b>
<b>ASG-Existing Systems Workbench (ASG-ESW)</b> .....	<b>ix</b>
<b>Invoking ESW Products</b> .....	<b>xii</b>
<b>ESW Product Integration</b> .....	<b>xiii</b>
<b>Examples</b> .....	<b>xiv</b>
<b>Publication Conventions</b> .....	<b>xvi</b>
<b>1 Introduction</b> .....	<b>1</b>
<b>Interactive Testing/Debugging</b> .....	<b>1</b>
<b>Testing Support</b> .....	<b>3</b>
Supported Languages .....	3
Supported Execution Environments .....	3
Supported Databases.....	4
<b>SmartTest-PLI Interfaces</b> .....	<b>4</b>
The CUA Interface .....	4
Command Processing .....	4
Screen Format .....	5
<b>Command Targets</b> .....	<b>7</b>
Label Name Target .....	7
Variable Name Target.....	7
Pattern String Target .....	8
Line Range Command Target .....	8
Procedure Name Target.....	8
Subset Name Target .....	9
<b>PL/I Optimization Considerations</b> .....	<b>9</b>
<b>Limitations</b> .....	<b>9</b>

<b>2 SmartTest-PLI Test Session</b> .....	<b>13</b>
<b>Introducing the Test Session</b> .....	<b>13</b>
<b>Beginning a SmartTest Session</b> .....	<b>14</b>
<b>SmartTest User Options</b> .....	<b>16</b>
Online Operation Parameters .....	17
Log/List/Punch/Work File Allocations .....	18
Log/List/Punch Processing Options .....	19
Script File Allocations .....	24
PF Key Values .....	25
Mode Options .....	26
<b>Application Knowledge Repository (AKR)</b> .....	<b>27</b>
Verifying AKR Allocation Results .....	29
<b>Program Analyze</b> .....	<b>30</b>
Program Analyze Requirements .....	30
Analyzing a Program .....	30
Method 1 - Analyzing a Program Using SmartTest .....	31
Method 2 - Analyzing a Program from an Edit Session .....	32
Method 3 - Analyzing a Program from a User Screen .....	34
Verifying Analyze Results .....	34
<b>Setting Up the Test Session</b> .....	<b>35</b>
Selecting the Test Environment .....	35
Specifying TSO Setup Information .....	38
Converting Batch Execution JCL to a TSO CLIST .....	40
Initiating the Test Session .....	45
Setup Considerations .....	46
<b>Saving the Test Session Setup</b> .....	<b>47</b>
Sharing Test Setups .....	48
Sharing an Alternate Profile Dataset .....	48
<b>Restoring the Test Session Environment</b> .....	<b>49</b>
<b>Terminating a Test Session</b> .....	<b>49</b>
<b>3 Test Session: Additional Environments</b> .....	<b>51</b>
<b>Setting Up the Test Session in Other Environments</b> .....	<b>51</b>
DB2 Stored Procedure Option .....	51
CICS and IMS Environments .....	52
<b>ISPF Dialog Manager</b> .....	<b>52</b>
Specifying ISPF Dialog Manager Information .....	52
Specifying Programs to be Tested .....	54
Specifying ISPF File Allocation Information .....	55

Specifying the ISPF Program Load Library .....	56
Initiating an ISPF Test Session .....	62
<b>IMS/DB Programs in TSO Foreground .....</b>	<b>63</b>
Specifying IMS/DB Setup Information .....	63
Specifying IMS File Allocation Information .....	65
<b>BTS in TSO Foreground.....</b>	<b>79</b>
Specifying BTS Setup Information.....	79
Selecting BTS Transactions to Monitor .....	81
Specifying BTS File Allocation Information.....	83
Specifying IMS File Allocation Information from BTS Allocation Selection .....	99
<b>DB2 Programs in TSO Foreground .....</b>	<b>100</b>
Specifying DB2 Setup Information.....	100
<b>DB2 Stored Procedure Testing Option .....</b>	<b>102</b>
Requirements.....	102
Setting Up the DB2 Stored Procedure Test.....	103
Reviewing DB2 Stored Procedure Parameters .....	105
Initiating the DB2 Stored Procedure Test.....	107
<b>Testing Programs in a Batch Region .....</b>	<b>108</b>
Specifying Batch Connect Setup Information.....	109
Submitting and Connecting to a Batch Job.....	112
Initiating the Batch Test .....	114
<b>Testing DL/I in the Batch Environment .....</b>	<b>116</b>
<b>Testing BTS in the Batch Environment.....</b>	<b>117</b>
<b>Testing DB2 in the Batch Environment.....</b>	<b>118</b>
Support for Changed DB2 Interface .....	119
<b>Testing DFHDRP in the Batch Region.....</b>	<b>119</b>
DFHDRP Batch JCL.....	119
<b>4 Testing Techniques .....</b>	<b>121</b>
<b>Learning the Commands.....</b>	<b>122</b>
<b>Testing with SmartTest .....</b>	<b>122</b>
Testing Using the MONITOR Method.....	122
Testing Using the NOMONITOR Method .....	123
Guideline for Using the MONITOR/NOMONITOR Methods.....	123
Testing Dynamically Called Load Modules .....	124
<b>Controlling Program Execution.....</b>	<b>124</b>
Selecting Test Environment Using the Setup Wizards.....	125
Executing the Program Continuously Using RUN.....	125
Executing a Specified Number of Statements Using STEP .....	126
Changing Program Execution Sequence Using GO .....	127

Interrupting Test Execution Using Keystroke . . . . .	128
Inserting Breakpoints to Interrupt Execution . . . . .	128
Locating the Next Executable Statement . . . . .	133
Displaying Breakpoints . . . . .	134
Generating a Dump . . . . .	134
Canceling a Test Session . . . . .	134
Exiting a SmartTest-PLI Test Session . . . . .	135
<b>Viewing and Changing Test Session Data . . . . .</b>	<b>135</b>
Viewing Test Session Data Items Inline . . . . .	135
Removing Zoom Data Windows . . . . .	138
Viewing Data Values at the Top of the Screen . . . . .	138
Changing Test Session Data Values . . . . .	140
<b>Program Execution History (Backtrack) . . . . .</b>	<b>140</b>
Recording Execution History . . . . .	140
Reviewing Execution History . . . . .	141
Using the BackTrack Variable History Function . . . . .	147
<b>Using Pseudo Code . . . . .</b>	<b>149</b>
Pseudo Code Concepts . . . . .	149
Pseudo Code Statements Available . . . . .	150
Using PL/I Datanames in Pseudo Code . . . . .	152
Using Constants in Pseudo Code . . . . .	152
Entering and Editing Pseudo Code . . . . .	152
Executing Pseudo Code in a Test Session . . . . .	153
Viewing Pseudo Code in a Test Session . . . . .	153
Removing Pseudo Code from a Program . . . . .	153
<b>Making Proposed Changes Permanent . . . . .</b>	<b>154</b>
<b>Using Multiple Programs . . . . .</b>	<b>154</b>
<b>Tailoring a Test Session by Program . . . . .</b>	<b>155</b>
<b>Capturing and Replaying Test Sessions . . . . .</b>	<b>155</b>
Replaying a Script File . . . . .	156
<b>Position the Display at the Next Executable Statement . . . . .</b>	<b>157</b>
<b>Saving Keystrokes . . . . .</b>	<b>158</b>
Token/Cursor Substitution . . . . .	158
Substituting Short Names for Character Strings . . . . .	158
Retain Commands in the Command Input Area . . . . .	159
Recall Commands, Messages, and Pop-ups . . . . .	159
View Release and Level Numbers . . . . .	160
<b>Linking to Alliance . . . . .</b>	<b>160</b>

<b>5 Analysis Features</b> .....	<b>163</b>
<b>The Intelligent Search Function</b> .....	<b>163</b>
Searching for PL/I Language Subsets .....	163
Determining Impact of Change .....	165
<b>Excluding Lines from the Display</b> .....	<b>166</b>
Displaying Individual Excluded Lines .....	166
Displaying All Excluded Lines .....	166
<b>Searching for PL/I Program Information</b> .....	<b>167</b>
Finding All IO Statements in a Program .....	168
Example of Finding Data Field References .....	170
<b>Printing Program Information</b> .....	<b>171</b>
Scrolling the Display .....	172
<b>6 Additional Languages</b> .....	<b>173</b>
<b>SmartTest-PLI and Assembler</b> .....	<b>173</b>
<b>SmartTest-PLI and COBOL</b> .....	<b>173</b>
<b>7 Help Facility</b> .....	<b>175</b>
<b>Introduction</b> .....	<b>175</b>
Help Navigational Commands .....	177
<b>Screen Help</b> .....	<b>177</b>
<b>Command Help</b> .....	<b>178</b>
<b>General Information</b> .....	<b>180</b>
<b>Specific Information</b> .....	<b>180</b>
<b>Help Abends</b> .....	<b>181</b>
<b>Help Messages</b> .....	<b>182</b>
Severity Levels .....	183
<b>Printing Messages</b> .....	<b>184</b>
<b>Glossary</b> .....	<b>187</b>
<b>Index</b> .....	<b>195</b>



---

## Preface

---

This *ASG-SmartTest PLI User's Guide* provides user information about ASG-SmartTest PL/I (herein called SmartTest-PLI). SmartTest-PLI provides testing and debugging processes for applications in the PL/I environment. SmartTest-PLI is fully integrated with ASG-SmartTest (herein called SmartTest). All basic and extended facilities are available including: program view, execution control, Breakpoints, monitoring and changing data, pseudocode, abend processing, Backtrack, and the COBOL intelligent search function.

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-PLI.
- [Chapter 2, "SmartTest-PLI Test Session,"](#) introduces the SmartTest test session.
- [Chapter 3, "Test Session: Additional Environments,"](#) provides information for setting up SmartTest in additional environments.
- [Chapter 4, "Testing Techniques,"](#) presents common SmartTest testing and debugging techniques.
- [Chapter 5, "Analysis Features,"](#) describes the SmartTest-PLI Intelligent Search Function
- [Chapter 6, "Additional Languages,"](#) provides information about SmartTest-PLI with Assembler and COBOL.
- [Chapter 7, "Help Facility,"](#) describes the comprehensive and context-sensitive Help facility, including an online Help Tutorial.

## Related Publications

The documentation library for ASG-SmartTest PLI 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-SmartTest-PLI-TCA (Test Coverage Analysis) option.

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

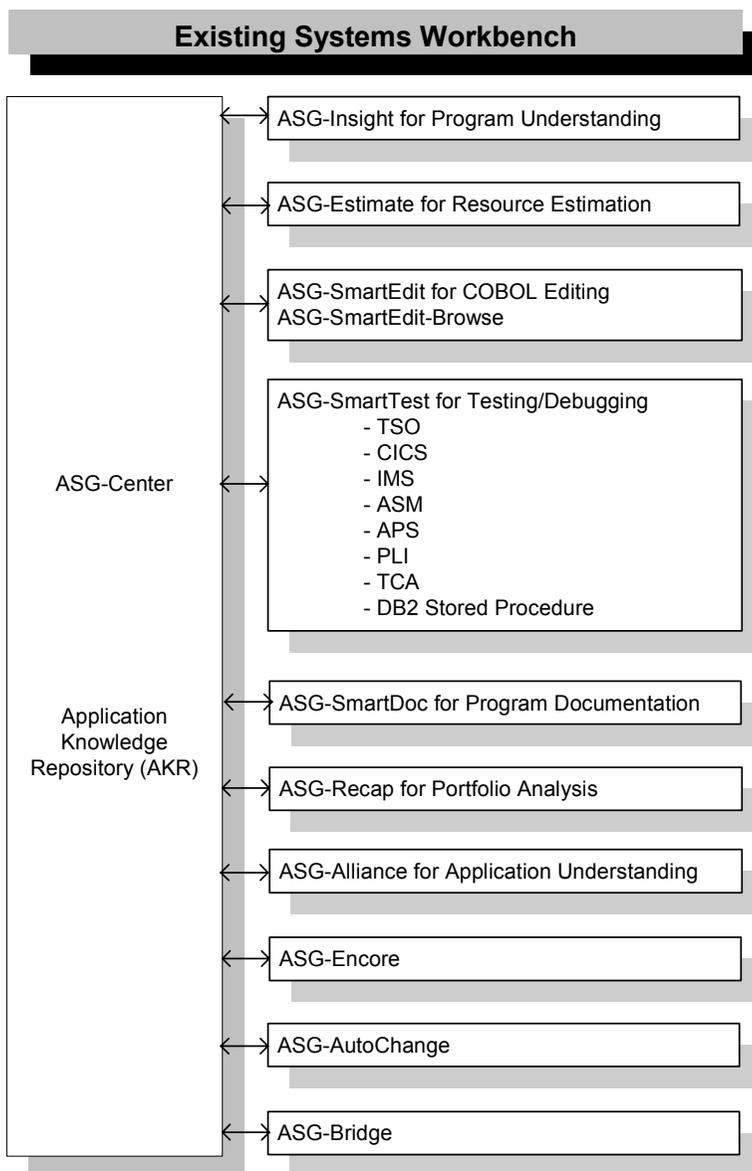
To obtain a specific version of a publication, contact the ASG Service Desk.

---

## 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 vii](#) for more information about using ESW.

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

Product Name	Shortcut	ESW Pull-down Options
Alliance	AL	Understand ▶ Application
AutoChange	CC	Change ▶ Conversion Set
Bridge	BR	Change ▶ ASG-Bridge
Encore (Re-engineer)	EN	Re-engineer ▶ Program
Estimate	ES	Measure ▶ ASG-Estimate
Insight (Understand)	IN	Understand ▶ Program
Recap (Portfolio Analysis)	RC	Measure ▶ Portfolio
SmartDoc (Document)	DC	Document ▶ Program
SmartEdit	SE	Change ▶ Program <b>Or</b> Change ▶ Program with Options
SmartTest	ST	Test ▶ Module/Transaction

## ESW Product Integration

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

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



**Example 2.** [Figure 4](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest directly. [Figure 5](#) shows the File - Analyze Submit pop-up that displays when you access SmartTest through ESW.

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

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

**Figure 4 • File - Analyze Submit Screen**

```

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

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

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

AKR data set name 'USER12.GENERAL.AKR'
AKR program name      (if overriding PROGRAM-ID)

Analyze options:
-----
-----

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

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

```

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

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

Analyze features (Y/N):
  ASG-Insight: Y   ASG-SmartTest: Y   Extended Analysis: N
  ASG-SmartDoc: N   ASG-Encore: N
AKR data set name 'USER12.GENERAL.AKR'
AKR program name      (if overriding PROGRAM-ID)

Analyze options:
-----
-----

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

## Publication Conventions

ASG uses these conventions in technical publications:

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

---

# 1

## Introduction

---

This chapter provides an introduction to SmartTest and contains these sections:

Topic	Page
<a href="#">Interactive Testing/Debugging</a>	<a href="#">1</a>
<a href="#">Testing Support</a>	<a href="#">3</a>
<a href="#">SmartTest-PLI Interfaces</a>	<a href="#">4</a>
<a href="#">Command Targets</a>	<a href="#">5</a>
<a href="#">PL/I Optimization Considerations</a>	<a href="#">9</a>
<a href="#">Limitations</a>	<a href="#">9</a>

SmartTest is the testing and 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

### Interactive Testing/Debugging

PL/I programs are prepared for a SmartTest testing and debugging session using the Analyze feature. Output from the Program Analyze process is stored in the AKR. This information then becomes an integrated component of the test session.

Once a PL/I program is prepared for SmartTest testing, interactive testing and debugging can begin.

SmartTest-PLI enables PL/I programs to be viewed, analyzed, tested, and debugged at the source level. It executes in the TSO/ISPF environment and provides full ISPF compatibility, which facilitates learning and usage through a standard IBM interface.

SmartTest-PLI interfaces features Common User Access (CUA) screens, pull-downs, and pop-ups that are designed to provide easy access to all of the product features. Powerful SmartTest-PLI commands can be used instead of, or in conjunction with, the selections offered from the action bar.

SmartTest-PLI provides these comprehensive testing and debugging 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.
PL/I Intelligence	Dynamic Breakpoints can be set based on PL/I 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	ESW integrates the ESW family of products that support the automation of the software maintenance cycle.
Execution Review (Backtrack)	Execution history may 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.

The dramatic advantage SmartTest-PLI has over conventional program debugging tools is that it is PL/I intelligent. It provides comprehensive analysis information about a program.

SmartTest-PLI 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 are easily recognized and can be quickly corrected.

## Testing Support

SmartTest runs under MVS (XA/ESA) TSO/ISPF (Release 3.x) and supports these languages, execution environments, and databases:

### *Supported Languages*

- COBOL II
- OS PL/I Version 2.3
- PL/I MVS & VM
- COBOL/370
- CASE-generated COBOL
- High-level Assembler
- CA-Optimizer II (COBOL II)
- INTERSOLV APS

### *Supported Execution Environments*

- TSO
- BTS
- DLI
- CICS Version 3.3 through Version 4.1
- CICS/TS 1.1, 1.2, and 1.3
- CICS command level programs
- IMS/DC
- ISPF Dialog Manager
- HOGAN
- LANGUAGE ENVIRONMENT

## Supported Databases

- VSAM
- IMS/DB (DL/1)
- DB2
- IDMS/DB
- SYSTEM 2000
- DATACOM/DB
- TOTAL/TIS

## SmartTest-PLI Interfaces

SmartTest-PLI provides Common User Access (CUA) screens, pull-downs, and pop-ups designed to provide easy access to all of the product features. In addition, all functions are available using primary or line commands.

### The CUA Interface

All CUA screens include action bars. An action bar is the line of keywords displayed at the top of the screen. Each keyword represents actions that are available on that screen. To view the list of actions on a pull-down, move the cursor to the desired keyword, and press Enter. To make a selection, enter the number that corresponds to the desired action, or place the cursor on the desired action and press Enter. Selecting an action followed by an ellipsis (...) displays a pop-up requesting more information.

A pop-up is a window that is superimposed on your screen to allow entry of information for the requested action. Complete the requested action by entering the desired data and options and pressing Enter.

### Command Processing

SmartTest-PLI provides both primary commands, entered in the command input area at the top of the screen, and line commands, entered in the line sequence number area of the screen.

SmartTest-PLI line commands are processed before primary commands.

Commands entered by pressing a PF key are handled as if entered in the primary command input area. If a command is entered by pressing a PF key, the contents of the command input area are appended to the PF key command. The combined text is then executed as a whole.

Multiple commands entered in the command input area are separated by an ISPF command delimiter (usually a semicolon) and are processed in a left-to-right sequence.

**Note:**

The command delimiter is defined for ISPF in the COMMAND DELIMITER field on the ISPF Terminal Characteristics screen.

See the *ASG-SmartTest Reference Guide* for details on CUA and command syntax.

## Screen Format

SmartTest screens are modeled after the TSO/ISPF edit screen. [Figure 6](#) shows a SmartTest Program View screen.

**Figure 6 • SmartTest Screen Example**

```

File View Test Search List Options Help
-----
                                (A)                                (B)
                                Program View                        VIAPPLI.VIAPPLI1 -A
Command ==>>> _____ (C) _____ Scroll ==>> (D) CSR

>>>>>>      DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
|          |-----+-----+-----+-----+-----+-----+-----+-----+
|          | 03 DATA_PACKED_DEC          DEC FIX (15,0)      ADDR 000D6870 |
|          |          VALUE > _____ < * INVALID NUMERIC * |
|          |-----+-----+-----+-----+-----+-----+-----+
000483
- - - - - 20 LINES NOT DISPLAYED
000504
(F)005      SOC7FIX1:
000506          RETURN;
000507
000508      END SOC7_DEMO;
000509
(E)
(G)
+-----+-----+-----+-----+-----+-----+-----+-----+
|STATUS: DATA EXCEPTION (0C7)          PROGRAM: VIAPPLI1  DATE: DMMMYYYY |
| STMT: 000482  OFF: 000C10  AMODE: 31   MODULE: VIAPPLI   TIME: HH:MM:SS |
|SOURCE: DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Screen Section	Description
(A) Program View	Name of the screen.
(B) VIAPPLI.VIAPPLI1	Module/program being viewed (module.program). 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 module.program 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 ==>>>	This is the primary field for entering SmartTest-PLI commands.

Screen Section	Description
(D) Scroll ==>	<p>Scroll specifies the number of lines or columns to scroll the display screen. The Scroll ==&gt; field is omitted from screens that cannot be scrolled. These are the valid values:</p> <p><b>1- 9999:</b> Specifies the number of lines or columns to scroll.</p> <p><b>CSR:</b> 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><b>MAX:</b> Specifies that the top, bottom, right, or left margin is the scroll value.</p> <p><b>HALF:</b> Specifies a scroll value of a half screen.</p> <p><b>PAGE:</b> Specifies a scroll value of one screen.</p> <p><b>DATA:</b> 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	<p>Descriptive, informational, or error messages are displayed in this area. Long messages are displayed when there are no corresponding short messages, or when HELP is entered in the command input area while a short message displays.</p>
(F) Line command input area	<p>SmartTest line commands may be entered over the displayed line numbers in columns 1 through 6 as with other TSO/ISPF editor screens.</p>
(G) Status box at the bottom of the SmartTest screen	<p>This 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.</p>

## Command Targets

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

- Label name
- Variable name
- Pattern string
- Line range
- Procedure name
- Subset name

### Label Name Target

A label name in a program is any section name. A label in PL/I is any PL/I block or label name. A label name in an Assembler program is any Assembler label. A label name in COBOL is any paragraph or section name. The label name specifies all transfers of control to a section.

### Variable Name Target

A variable name can be:

- Elementary variable name - String or arithmetic
- Program control variable - Pointers
- Aggregates - Arrays and structures

Any legal reference for a data element can be specified as a variable name. Any reference to an entry in an array is treated as an individual element. When variable items overlap so a name can refer to parts of multiple variable items, searches are performed on each part and all references are reported.

Fully qualified variable names can be specified by the high- to low-level qualifiers separated by periods, for example:

```
PROCEDURE_NAME.DATA_NAME_ELEMENT
```

Multiple variable names can be located at the same time by concatenating the variable names with a plus sign (+) between them, for example:

```
DATA_NAME1 + DATA_NAME2
```

## Pattern String Target

A pattern string is a sequence of characters. It must be 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.

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

## Line Range Command 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.

## Procedure Name Target

The procedure name is the internal CSECT name assigned by PL/I to the specified PROC statement.

## Subset Name Target

This is one of the PL/I language subsets. See ["Searching for PL/I Language Subsets" on page 163](#) for a table of subsets.

## PL/I Optimization Considerations

SmartTest-PLI may test optimized PL/I programs differently than non-optimized programs because of modifications made to the program code by the optimizer. Testing optimized PL/I programs may produce these results:

- Certain source lines may appear not to be executable from the Program View screen, and cause unexpected results for commands such as STEP, BREAK, and ZA (Zoom Assembler).
- Use the GO command with care. Source code can be altered or even eliminated in the optimization process. As a result, unpredictable or erroneous results can occur with the GO command. See the optimized assembler output generated by the compiler as a navigational tool for testing optimized programs.
- The PL/I Optimizer often moves all or part of the processing for one statement to another statement. When this occurs, you may find that certain variables remain unchanged, even after a statement that modifies the field.

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

For the SmartTest-PLI test session, you should not use the PL/I Optimizer. After completion of testing and correction, you can recompile using the Optimizer to get a more efficient program.

\_\_\_\_\_

## Limitations

These PL/I coding techniques are unsupported in all current and past versions of SmartTest-PLI. ESW is currently unable to support Based variables when the variable is based on a pointer that is defined as part of an array of pointers. For example, this code is supported by SmartTest-PLI:

```
DCL 1 BASIS_BEREICH,          /* Individual pointer declarations */
    3 P1 PTR,
    3 P2 PTR,
    3 P3 PTR,
    3 P4 PTR,
    3 P5 PTR;

DCL BEREICH1 CHAR(20);
DCL BEREICH2 CHAR(20);
DCL BEREICH3 CHAR(20);
```

---

## ASG-SmartTest PLI User's Guide

```
DCL BEREICH4          CHAR(20);
DCL BEREICH5          CHAR(20);

P_BASIS = ADDR(BASIS_BEREICH);

P1      = ADDR(BEREICH1);
P2      = ADDR(BEREICH2);
P3      = ADDR(BEREICH3);
P4      = ADDR(BEREICH4);
P5      = ADDR(BEREICH5);

DCL 1 STRUK1A BASED(P1),
      3 F1      CHAR(10),
      3 F2      CHAR(10);
DCL 1 STRUK2A BASED(P2),
      3 F1      CHAR(10),
      3 F2      CHAR(10);
DCL 1 STRUK3A BASED(P3),
      3 F1      CHAR(10),
      3 F2      CHAR(10);
DCL 1 STRUK4A BASED(P4),
      3 F1      CHAR(10),
      3 F2      CHAR(10);
DCL 1 STRUK5A BASED(P5),
      3 F1      CHAR(10),
      3 F2      CHAR(10);

STRUK1A.F1 = 'A1A1A1A1A1';
STRUK1A.F2 = 'B1B1B1B1B1';
STRUK2A.F1 = 'A2A2A2A2A2';
STRUK2A.F2 = 'B2B2B2B2B2';
STRUK3A.F1 = 'A3A3A3A3A3';
STRUK3A.F2 = 'B3B3B3B3B3';
STRUK4A.F1 = 'A4A4A4A4A4';
STRUK4A.F2 = 'B4B4B4B4B4';
STRUK5A.F1 = 'A5A5A5A5A5';
STRUK5A.F2 = 'B5B5B5B5B5';
STRUK4A.F1 = 'XXXXXXXXXX';
```

However, this variation on the same code is not currently supported by SmartTest-PLI:

```
DCL 1 BASIS_BEREICH,          /* TABLE OF POINTER'S          */
      3 P_(5) PTR;
DCL BEREICH1          CHAR(20);
DCL BEREICH2          CHAR(20);
DCL BEREICH3          CHAR(20);
DCL BEREICH4          CHAR(20);
DCL BEREICH5          CHAR(20);

P_BASIS = ADDR(BASIS_BEREICH);
P_(1) = ADDR(BEREICH1);
P_(2) = ADDR(BEREICH2);
P_(3) = ADDR(BEREICH3);
P_(4) = ADDR(BEREICH4);
P_(5) = ADDR(BEREICH5);

DCL 1 STRUK1 BASED(P_(1)),
      3 F1          CHAR(10),
      3 F2          CHAR(10);
```

```
DCL 1 STRUK2 BASED (P_(2)),
    3 F1          CHAR(10),
    3 F2          CHAR(10);
DCL 1 STRUK3 BASED (P_(3)),
    3 F1          CHAR(10),
    3 F2          CHAR(10);
DCL 1 STRUK4 BASED (P_(4)),
    3 F1          CHAR(10),
    3 F2          CHAR(10);
DCL 1 STRUK5 BASED (P_(5)),
    3 F1          CHAR(10),
    3 F2          CHAR(10);

STRUK1.F1 = 'A1A1A1A1A1';
STRUK1.F2 = 'B1B1B1B1B1';
STRUK2.F1 = 'A2A2A2A2A2';
STRUK2.F2 = 'B2B2B2B2B2';
STRUK3.F1 = 'A3A3A3A3A3';
STRUK3.F2 = 'B3B3B3B3B3';
STRUK4.F1 = 'A4A4A4A4A4';
STRUK4.F2 = 'B4B4B4B4B4';
STRUK5.F1 = 'A5A5A5A5A5';
STRUK5.F2 = 'B5B5B5B5B5';
STRUK4.F1 = 'XXXXXXXXXX';
```

The Dictionary dumps used for analysis do not contain enough information about the Base Pointer when the pointers are contained in an array.



---

# 2

## SmartTest-PLI Test Session

---

This chapter introduces the SmartTest test session and contains these sections:

Topic	Page
<a href="#">Introducing the Test Session</a>	<a href="#">13</a>
<a href="#">Beginning a SmartTest Session</a>	<a href="#">14</a>
<a href="#">SmartTest User Options</a>	<a href="#">16</a>
<a href="#">Application Knowledge Repository (AKR)</a>	<a href="#">27</a>
<a href="#">Program Analyze</a>	<a href="#">30</a>
<a href="#">Setting Up the Test Session</a>	<a href="#">35</a>
<a href="#">Saving the Test Session Setup</a>	<a href="#">47</a>
<a href="#">Restoring the Test Session Environment</a>	<a href="#">49</a>
<a href="#">Terminating a Test Session</a>	<a href="#">49</a>

### Introducing the Test Session

The first time you use SmartTest, verify parameters that are initially set to installation defaults. This section describes how to perform these tasks:

- Begin a SmartTest session.
- Set and verify these user options from the Options pull-down:
  - Online operation parameters. These parameters establish online operational defaults.
  - Log, List, Punch, and Work file allocations. The Log, List, Punch, and Work files are used by SmartTest for recording and printing selected information such as error messages and printed output, and as work space.





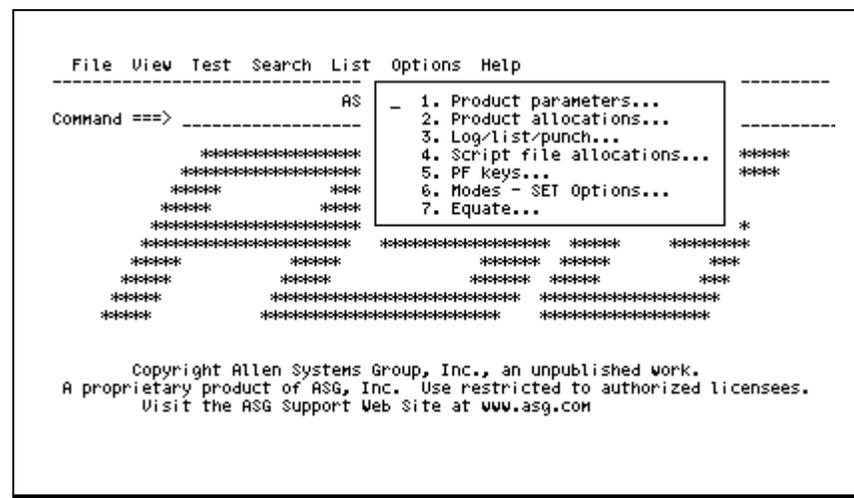
## SmartTest User Options

Upon initial use of SmartTest, options should be customized to reflect the appropriate settings for your situation. Some options are given defaults during installation.

### To display the Options pull-down

- 1 Select Options on the action bar and press Enter. The Options pull-down, shown in [Figure 9](#), displays.

Figure 9 • Options Pull-down



- 2 Select one of these options from the pull-down to customize the SmartTest environment:

Option	Description
Product parameters	Displays the Options - Product Parameters pop-up used to set parameters that affect the online operation of SmartTest.
Product allocations	Displays the Options - Product Allocations pop-up used to specify allocations for the Log, List, Punch, and Work files.
Log/list/punch	Displays the Options - Log/List/Punch Definition pop-up used to set values for allocating, formatting, and processing the SmartTest Log, List, and Punch files.
Script file allocations	Displays the Options - Script File Allocations pop-up used to set the default script libraries.

Option	Description
PF Keys	Displays the Options - PF Key Definition pop-up used to edit PF key values.
Modes	Displays the Options - Modes pop-up used to enable and disable SmartTest processing modes (LEARN, SCRIPT, XMODE).
Equate	Displays the Options - Equate pop-up used 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 <code>LIST EQUATES</code> .

**Note:** \_\_\_\_\_  
If you have Insight installed, this option displays as Scratchpad on the Options pull-down. See the online help or the *ASG-Insight Reference Guide* for more information about the Scratchpad option.

## 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. The Options - Product Parameters pop-up, shown in [Figure 10](#), displays.

**Figure 10 • Options - Product Parameters Pop-up**

```

Command ==> Options - Product Parameters
-----
Alarm . . . . . YES (YES or NO)
Save pseudo code . . YES (YES or NO)
Save equates . . . . YES (YES or NO)
Save marks . . . . . NO (YES or NO)
Cursor character . . % (Token substitution character)

```

- 2 Enter the appropriate information in the fields.

## Fields

Field	Description
Alarm	Controls the audible alarm on the terminal. If YES, the alarm sounds when an error message displays. Some terminals do not support this alarm control.
Save pseudo code	Specifies the default for saving pseudo code (pseudo code, Breakpoints, and When Conditions) when exiting from a program.
Save equates	Specifies the default for saving equates when exiting from a program.
Cursor Character	Sets the cursor token substitution character. The cursor substitution character can be used in any primary command during a test session. Any character can be specified as the token substitution character; however, it should be a character that is rarely used in commands. The default substitution character is the percent sign (%).

## 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. These options are initially set to default values established during product installation by the Systems Programmer and probably only need verifying.

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

- 1 Select Options ► Product Allocations. The Options - Product Allocations pop-up, shown in [Figure 11](#), displays.

Figure 11 • Options - Product Allocations Pop-up with SMS

```

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.

**Fields**

Field	Description
Log file	Specifies the Management and Storage Class or the device type and volume serial number for the Log file that is allocated upon entry into SmartTest. Log files are used for internal messages.
List file	Specifies the Management and Storage Class or the device type and volume serial number for the List file that is allocated the first time a request is made to print output using the LPRINT command.
Punch file	Specifies the Management and Storage Class or the device type and volume serial number for the Punch file that is allocated the first time a request is made to punch output using the LPUNCH command.
Work file	Specifies the Management, Storage, and Data Class or the device type, volume serial number, and space requirements for the Work file that is allocated upon entry into SmartTest. Approximately one cylinder (on a 3380 device) is required for a 5,000 line program.

**Log/List/Punch Processing Options**

Use the Options - Log/List/Punch Definition pop-up to define and set processing values for 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. Some options are initially set to default values established during product installation by the Systems Programmer.

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

- 1 Select Options ► Log/List/Punch. The Options - Log/List/Punch Definition pop-up, shown in [Figure 12](#), displays.

**Figure 12 • Options - Log/List/Punch Definition Pop-up**

```

Options - Log/List/Punch Definition
Command ==> -----
1 - Process log file  2 - Process list file  3 - Process punch file
                   4 - Customized data set name

Options              Log              List              Punch
-----              ---              ---              ---
Process option      . . . . . K              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.

**Fields**

Field	Description
1 - Process log file	Verify or change the Options for the Log file, then select option 1 to process the Log file. A new file is allocated to collect additional data, if required. The Log file is processed when you press Enter.
2 - Process list file	Verify or change the Options for the List file, then select option 2 to process the List file. By default, a new file is allocated to collect additional data, if required. The file name is in this format:  <i>userid.yyyxxxxx.VIALIST</i>  where:  <i>yyy</i> is the product ID.  <i>xxxxxx</i> is a sequential number starting with 00001.  The List file is processed when you press Enter.
3 - Process punch file	Verify or change the Options for the Punch file, then select option 3 to process the Punch file. This file is defined as fixed block with 80-byte records. The Punch file is processed when you press Enter.

Field	Description
4 Customized data set name	Select option 4 to display the Options - Log/List/Punch Name Customization pop-up on which you can specify user-defined dataset names for the Log, List, and Punch files. You can also choose to be prompted for the DSN name during file processing.
Process option	Enter one of the listed processing options. The default is PD for the Log and List files, and PK for the Punch file.
Primary tracks	Enter the number of primary tracks to allocate. A size change does not take effect until the next allocation occurs. The default is 1 for the Log, List, and Punch files.
Secondary tracks	Enter the number of secondary tracks to allocate. A size change does not take effect until the next allocation occurs. The default is 2 for the Log file and 5 for the List and Punch files.
Lines per page	Enter the number of print lines per page. Typical maximum values are 60 for six lines per inch and 80 for eight lines per inch. The default is 56.
Sysout class	Enter the SYSOUT class value. The default is an asterisk (*), which sends the SYSOUT to the destination specified in the MSGCLASS parameter on the JOB statement.
Process options	Lists the available options for the Log, List, and Punch files. These are the valid values: <ul style="list-style-type: none"> <li>• PK. Print and keep.</li> <li>• PD. Print and Delete.</li> <li>• K. Keep without printing.</li> <li>• D. Delete without printing.</li> </ul>
xxxxx FILE IS ALLOCATED	Displays when the Log, List, or Punch file has been properly allocated. If the message does not appear, check the assignments on the Options - Product Allocations pop-up. xxxxx is LOG, LIST, or PUNCH depending on the file that was allocated.
Job statement information	Enter the appropriate JOB statement information for your installation. These JCL statements are required if the PK or PD processing option has been specified for processing the Log, List, or Punch files.

**Note:**

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

If you specify K or PK process option, you can customize the data set where the log, list, or punch file is allocated. By default, SmartTest allocates the Log, List, and Punch files, for example:

```
userid.STTnnnnn.VIAxxxxx
```

where:

*userid* is your TSO user ID.

*nnnnn* is a sequential number from 00001 to 99999.

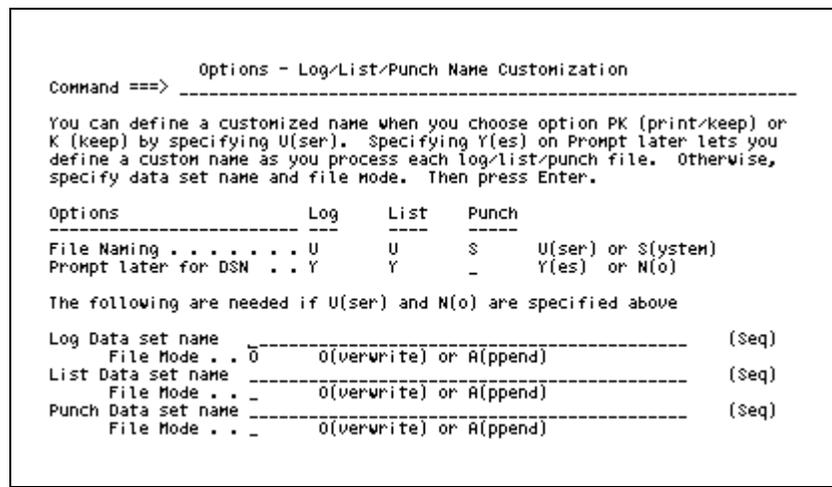
*xxxxx* is LOG for Log, LIST for List, and PUNCH for Punch files.

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 the Log, List, or Punch dataset name**

- 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 13](#).

**Figure 13 • Options - Log/List/Punch Name Customization Pop-up**



- 2 Type U in the File Naming field for Log, List, and/or Punch to indicate a user-defined dataset name. Enter the appropriate values in these fields:

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

If you specify Y in the Prompt later for DSN field, SmartTest prompts you for the dataset name during file processing using the Log Name Customization pop-up, shown in [Figure 14](#).

**Figure 14 • Log Name Customization Pop-up**

```

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

```

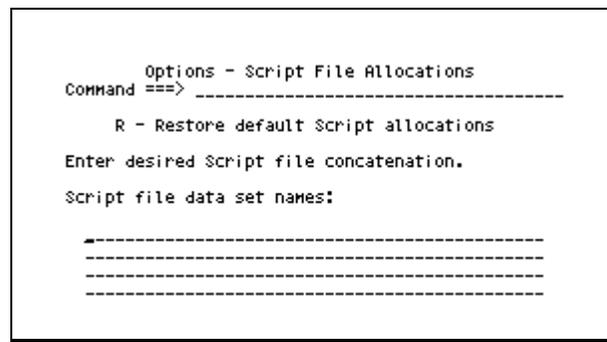
## Script File Allocations

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

### To specify your default script files

- 1 Select Options ► Script file allocations to display the Options - Script File Allocations pop-up, shown in [Figure 15](#).

Figure 15 • Options - Script File Allocations Pop-up



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

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

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

\_\_\_\_\_

## Fields

Field	Description
R	Restores the default Script library allocations set at install time.
Script file dataset names	Enter the dataset names of the desired Script files in the order in which they should be concatenated.

## PF Key Values

*To display and/or redefine PF key values used with SmartTest*

- 1 Select Options ► PF Keys. The Options - PF Key Definition pop-up, shown in [Figure 16](#), displays.

**Figure 16 • Options - PF Key Definition Pop-up**

```

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

Number of PF keys: 24      Terminal type: 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.

The value in the Terminal type field indicates the type of terminal being used. SmartTest supports these 3270 type terminals:

- 3270-2 (24 lines x 80 columns)
- 3270-3 (27 lines x 80 columns)
- 3270-4 (42 lines x 80 columns)
- 3270-5 (27 lines x 133 columns)

## Mode Options

To enable or disable the mode indicated by the specified option

- ▶ Select Options ▶ Modes or type SET with no operands. The Options - Modes pop-up, shown in [Figure 17](#), displays.

Figure 17 • Options - Modes Pop-up

Option	Set	Description
ASM	ON	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

Each mode shown on this pop-up can be directly enabled or disabled using the SET command with the corresponding operand. The current setting for each Test Session option is saved between sessions with the exception of BREAK, PSEUDO, SCRIPT, and WHENS. Defaults for these options are restored when a new session is initiated.

Scroll the screen forward to display modes that do not appear. The STOPEXEC and STOPHAND modes only appear if the current environment is CICS.

For detailed descriptions of options, see the *ASG-SmartTest Reference Guide*.

**Note:**

For testing PL/I programs, the MAIN option must be ON.

## Application Knowledge Repository (AKR)

Before programs can be tested, your program must be analyzed. The results of the analyze are stored in the AKR for use by SmartTest while testing. 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 allow you to perform these tasks:

- Allocate or expand the AKR.
- Rename and delete programs.
- Work with the AKR directory.

### To allocate an AKR

- 1 Select File ► AKR utility or type UTILITY. The File - AKR Utility pop-up, shown in [Figure 18](#), displays.

Figure 18 • 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 Type A in the command input area and press Enter. The File - AKR Allocate/Expand pop-up, shown in [Figure 19](#), displays.

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

```

                                File - AKR Allocate/Expand
Command ==> -----
          S - Submit JCL      E - Edit JCL      C - Specify Catalog
Expand existing AKR . . . NO          (Yes or No)
AKR data set name . . . '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 (ACCT ),
//          MSGCLASS=A
/*          INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
/*
  
```

- 4 Enter the appropriate information in these required fields:
  - a Type NO in the Expand existing AKR field.
  - b Select the AKR dataset name in the AKR dataset name field.
  - c Enter the appropriate SMS classes, if your site is using SMS, or the desired space and volume information.
  - d Enter a valid Job card in the Job statement information field.
- 5 If you need to specify a catalog or password, type C in the command input area and press Enter to display the AKR Catalog Information pop-up. After completing the dataset name and/or password, press PF3 to return to the File - AKR Allocate/Expand pop-up.
- 6 If you need to review or edit the JCL prior to submitting it, type E in the command input area and press Enter.
  - a After making the desired modifications, issue the standard ISPF SUBMIT command.
  - b Return to the File - AKR Allocate/Expand by issuing the END primary command or pressing PF3/PF15. The edited JCL is not saved.

If you do not need to edit the JCL, submit the JCL by typing S in the command input area and pressing Enter.

- 7 Verify the AKR allocation results by examining the job output. Sample job outputs are presented in ["Verifying AKR Allocation Results" on page 29](#).

## Verifying AKR Allocation Results

After the AKR allocation batch job has completed, review the job output to verify successful allocation and initialization. [Figure 20](#), [Figure 21](#), and [Figure 22](#) show output excerpts with messages that indicate successful AKR allocation and initialization.

**Figure 20 • IDCAMS Utility Condition Code Message Output**

```
IDCAMS  SYSTEM SERVICES                TIME: HH:MM:SS      DDMMYYYY  PAGE 1
DEFINE CLUSTER -
  (NAME(USER12.GENERAL.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 21 • AKR Initialization Message Output**

```
ASG-CENTER-OS(ESA)                AKR UTILITY LOG      DDMMYYYY  HH:MM:SS  PAGE 1
INIT DSNAME(USER12.GENERAL.AKR)
ASG1316I AKR "USER12.GENERAL.AKR" INITIALIZED.
ASG1314I *** END OF VIASYSIN ***
```

**Figure 22 • AKR Utility Log Summary Message Output**

```
ASG-CENTER-OS(ESA)                AKR UTILITY LOG - SUMMARY  DDMMYYYY  HH:MM:SS  PAGE 2
ASG1300I      1 AKR(S) INITIALIZED      0 FAILED.
ASG1315I *** END OF SUMMARY REPORT ***
```

## Program Analyze

The Program Analyze extracts knowledge about the program and populates the AKR with this information. Programs can be analyzed if they compile with a return code less than 8 and have only W or I level messages.

### Program Analyze Requirements

These are the input to the Program Analyze:

- The complete JCL to fetch, compile, and link the program.
- Program Analyze parameters indicating the type of analysis to be performed.
- An allocated Application Knowledge Repository to receive Program Analyze information.

### Analyzing a Program

You can submit the Program Analyze job using these methods:

- From the File - Analyze Submit pop-up in SmartTest (see ["Method 1 - Analyzing a Program Using SmartTest" on page 31](#)).
- From a TSO/ISPF edit screen by executing the VIASUB edit macro (see ["Method 2 - Analyzing a Program from an Edit Session" on page 32](#)).
- 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 34](#)).

This table describes when to use each method:

Compile JCL is from	Method for Executing Analyze Job
PDS or sequential dataset	<a href="#">"Method 1 - Analyzing a Program Using SmartTest" on page 31</a> , <a href="#">"Method 2 - Analyzing a Program from an Edit Session" on page 32</a> , or <a href="#">"Method 3 - Analyzing a Program from a User Screen" on page 34</a>
Librarian, Panvalet, or other user source manager when editing the JCL	<a href="#">"Method 2 - Analyzing a Program from an Edit Session" on page 32</a>
Screen-driven submit facility that generates the JCL	<a href="#">"Method 2 - Analyzing a Program from an Edit Session" on page 32</a> or <a href="#">"Method 3 - Analyzing a Program from a User Screen" on page 34</a>

## Method 1 - Analyzing a Program Using SmartTest

### To analyze a program while in SmartTest

- 1 Select File ► Compile/Analyze action. The File - Analyze Submit pop-up, shown in [Figure 23](#), displays.

Figure 23 • File - Analyze Submit Pop-up

```

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

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

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

```

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

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

Additional Analyze Features are shown on this pop-up if additional ESW components are installed at your site.

\_\_\_\_\_

- 2 Enter the appropriate information in these 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 for SmartTest in the Analyze Features (Y/N) field. Extended Analysis is not used for PL/I programs.
  - c Specify the AKR dataset name in the AKR data set name field.

- 3 If you need to review or edit the JCL prior to submitting it, type E in the command input area and press Enter.
  - a After making the desired modifications, type SUBMIT.
  - b Return to the File - Analyze Submit pop-up by typing END or pressing PF3/PF15. The edited JCL is not saved.

If you do not need to review the JCL, submit the compile/link JCL by typing S in the command input area and pressing Enter.

- 4 Verify the analyze results by examining the job output. An excerpt from a sample job output report is presented in ["Verifying Analyze Results" on page 34](#).

## 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.

### To analyze a program while in an edit session

- 1 Initiate an edit session on your standard compile/link JCL. An Edit screen, similar to the one shown in [Figure 24](#), displays.

Figure 24 • Sample Edit Session Screen

```

EDIT ----- ASG.VIACENxx.CNTL(VIAMECII) - 01.03 ----- COLUMNS 001 072
COMMAND ==>                                     SCROLL ==> CSR
***** ***** TOP OF DATA *****
000100 //VIASOFTA JOB (ACCOUNT), 'NAME',
000200 //          MSGCLASS=X
001820 //COMPLNK PROC ASG='ASG',                HIGH LEVEL NODE OF ASG DATA SETS
001830 //          CENTER='VIACENXX',          MIDDLE NODE OF ASG DATA SETS
001840 //          SYSOUT='*',                OUTPUT MESSAGE CLASS
001850 //          SYSDA='SYSDA',            WORK FILE UNIT NAME
001860 //          COMPILR='VIAPLI',          PL/I COMPILER
001870 //          PLICOMP='OSPLI.COMPILER',
001880 //          PLILIB='OSPLI.PLILIB',
001890 //          DYNAM='DYNAM',            DYNAM OR NODYNAM
001891 //          MEMBER='',                SOURCE/LOAD MODULE NAME
001892 //          USERLIB='USER.TEST.LOADLIB'
001893 //*
001894 //PLI EXEC PGM=&COMPILR,
001895 //          PARM='LIB,APOST,RES,&DYNAM'
001896 //STEPLIB DD DSN=&PLICOMP,DISP=SHR
001897 //SYSIN DD DSN=&ASG.&CENTER..CNTL(&MEMBER),DISP=SHR
001898 //SYSLIB DD DSN=&ASG.&CENTER..CNTL,DISP=SHR
001899 //SYSPRINT DD SYSOUT=&SYSOUT,
001900 //          DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1573)
001901 //SYSLIN DD DSN=&&SYSLIN,DISP=(MOD,PASS),

```

- 2 Make any changes necessary, such as job card information, program and/or load module names, or copybook and/or load library names.

- 3 Type VIASUB and press Enter. The Analyze Submit Parameters screen, shown in [Figure 25](#), displays.

**Figure 25 • Analyze Submit Parameters Pop-up**

```

Command ==> _____ File - Analyze Submit
                E - Edit JCL                               S - Submit JCL
Compile and link JCL (PDS or sequential):
Data set name 'USER12.REL.CNTL(HTEST)'
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

```

- 4 Enter the AKR dataset name in the AKR data set name field.

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

The Extended Analysis field is not used when analyzing PL/I programs.

Additional Analyze Features are shown on this pop-up if additional ESW components are installed at your site. Enter the appropriate information in the additional fields, as needed.

\_\_\_\_\_

- 5 If you need to review or edit the JCL prior to submitting it, type E in the command input area and press Enter.
- a After making the desired modifications, type SUBMIT.
  - b Return to the editor screen by typing END or pressing PF3/PF15. The edited JCL is not saved.

If you do not need to review the JCL, submit the compile/link JCL by typing S in the command input area and pressing Enter.

- 6 Verify the analyze results by examining the job output. An excerpt from a sample job output report is presented in ["Verifying Analyze Results" on page 34](#).

### 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 SmartTest systems administrator at your site for these details.

If you use a CLIST to submit jobs, SmartTest is installed with a sample CLIST, VIASUBDS, which submits an analyze job using the specified dataset name. This is the format of the CLIST:

```
TSO EXEC 'ASG.VIACENXX.CLIST(VIASUBDS)' [dataset name]
```

Check with your systems administrator for the exact location of this CLIST.

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 32](#).

Verify the analyze results by examining the job output. An excerpt from a sample job output report is presented in ["Verifying Analyze Results" on page 34](#).

### Verifying Analyze Results

After the Compile/Link - Analyze batch job has completed, review the job output to verify results. Check for this output:

- Acceptable compiler results.
- Acceptable linkage editor results.
- Messages indicating the program has been successfully analyzed and the program information is stored on the AKR.

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

[Figure 26](#) shows an output excerpt indicating a successful analyze.

**Figure 26 • Sample Analyze Output Report**

```
ASG-SMARTTEST-OS (ESA) Rx.x LVL000          PL/I PROGRAM ANALYZER
                                           PROGRAM NAME: VIAPPLI

DIAGNOSTICS LINES: 0 TOTAL - 0 WARNINGS, 0 ERRORS, 0 CATASTROPHES
PARAMETERS PASSED:  FEATURES=(S,PLI,X),NOLIST,SOURCE
OPTIONS IN EFFECT:  FEATURES=(SMARTTEST,PL/I),NOLIST
ENTRY POINTS:
PROGRAM 'VIAPPLI' WAS STORED IN AKR 'ASG.VIACENXX.AKR'.
```

This screen contains these report areas:

Report Area	Description
DIAGNOSTIC LINES	Program Diagnostics.
PROGRAM 'VIAPPLI'	Message indicating the program was stored on the AKR.

## Setting Up the Test Session

This section provides instructions for setting up a test session for the TSO environment. Setup information for other environments is in ["Test Session: Additional Environments" on page 51](#).

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 CLIST (if TSO foreground).
- Prepare input data for testing.
- Activate SmartTest.
- Provide program, module, and testing environment information.
- Initiate test session.

After the first time a program has been set up for testing, subsequent test sessions may no longer require all of these steps.

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 an IMS/DC test session, see the *ASG-SmartTest IMS User's Guide*.

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

The PF keys specified in this manual are set to the SmartTest default values. For more information on the PF keys, see ["PF Key Values" on page 25](#).

## Selecting the Test Environment

If you have previously set up the desired environment and saved the profile, you can restore the profile and skip the setup steps. (See ["Restoring the Test Session Environment" on page 49](#) for more information.)

Use the Environment Selection pop-up to select the SmartTest test environment and to specify the AKR, application load libraries, and procedure libraries for the test session.

**To display the Environment Selection pop-up**

- 1 Select the File ► Setup test environment.
- 2 Select execution environment on the File - Setup Test Environment pop-up and press Enter.

**Or**

Type ENV in the primary command area and press Enter on any SmartTest screen.

The Environment Selection pop-up, shown in [Figure 27](#), displays.

**Figure 27 • Environment Selection Pop-up**

```

                                     Environment Selection
Command ==> -----
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'
-----
    
```

- 3 Select the TSO environment by entering the corresponding environment number. The environment options are defined in [Options](#).

**Options**

Option	Description
A	Displays the System AKR Libraries pop-up.
L	Displays the System Load Libraries pop-up.
P	Displays the Procedure Libraries pop-up used to specify the procedure libraries to be used during the test session. If you do not enter any procedure libraries on this screen, the libraries listed in the PROCLIBS entry in VIASPRMS are used to search for procedures.

Option	Description
D	Displays the Environment - AKR Directory pop-up showing members in the concatenated AKRs.
1	Displays the TSO Session Setup screen. Test MVS programs in TSO foreground (see <a href="#">"Specifying TSO Setup Information" on page 38</a> ).
2	Displays the CICS Session Setup screen. Test programs in the CICS environment. For more information, see the <i>ASG-SmartTest CICS User's Guide</i> .
3	Displays the ISPF Session Setup screen. Test programs through ISPF Dialog Manager (see <a href="#">"ISPF Dialog Manager" on page 52</a> ).
4	Displays the IMS/DC Session Setup screen. Test programs in the IMS/DC environment. For more information, see the <i>ASG-SmartTest IMS User's Guide</i> .
5	Displays the IMS/DB Session Setup screen. Test IMS/DB programs in TSO foreground (see <a href="#">"IMS/DB Programs in TSO Foreground" on page 63</a> ).
6	Displays the BTS Session Setup screen. Test IMS/DC program with BTS in TSO foreground (see <a href="#">"BTS in TSO Foreground" on page 79</a> ).
7	Displays the DB2 Session Setup screen. Test DB2 programs in TSO foreground (see <a href="#">"DB2 Programs in TSO Foreground" on page 100</a> ).
8	If the DB2 Stored Procedure test option is installed at your site, displays the DB2 Stored Procedure Setup screen to test DB2 stored procedures (see <a href="#">"DB2 Stored Procedure Testing Option" on page 102</a> ). If not, a message indicating that this option is not installed at your site displays.
9	Displays the Batch Session Setup screen. Test MVS programs with Batch Connect.(see <a href="#">"Testing Programs in a Batch Region" on page 108</a> ).
10	Displays the IMS Batch Session Setup screen. Test IMS/DB programs with Batch Connect (see <a href="#">"Testing Programs in a Batch Region" on page 108</a> ).
11	Displays the BTS Batch Session Setup screen. Test IMS/DC programs with BTS with Batch Connect (see <a href="#">"Testing BTS in the Batch Environment" on page 117</a> ).

Option	Description
12	Displays the DB2 Batch Session Setup screen. Test DB2 programs with Batch Connect (see <a href="#">"Testing DB2 in the Batch Environment" on page 118</a> ).
Application Knowledge Repository (AKR):	Specifies the AKR dataset name(s). The AKRs is 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. A message indicates the total number of AKRs specified, including those specified on the System AKR Libraries pop-up.
Application load libraries:	Specifies the application load libraries to be used for the test session. Application load libraries are searched in the order entered. A message indicates the total number of load libraries specified, including those specified on the System Load Libraries pop-up.
	<p><b>Note:</b></p> <p>The Application load libraries field does not apply to CICS programs.</p>

## Specifying TSO Setup Information

*To specify the TSO test session parameters for MVS programs and to initiate a test session*

- 1 Select TSO on the Environment Selection pop-up. The TSO Session Setup screen, shown in [Figure 28](#), displays.

**Figure 28 • 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)
-----
-----

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

- 2 Complete these fields:
- a In the Load module field, type the name of the load module.
  - b In the Options group, enter any appropriate options.
  - c In the Execution parameters field, enter any parameters to be passed into the program at run time.
  - d In the Data set name field, type the CLIST dataset name that contains the CLIST member from the JCL conversion process.
  - e In the Member name field, enter the CLIST member name that contains, or will contain, the converted JCL.

**Note:**

If a CLIST has already been created for the program to be tested, see ["Initiating the Test Session" on page 45](#). If a CLIST does not exist for the program to be tested, see ["Converting Batch Execution JCL to a TSO CLIST" on page 40](#).

**Options**

Option	Description
R	Invokes the CLIST to initiate the TSO test session.
C	Displays the Convert Batch JCL pop-up.

**Fields**

Field	Description
Execution: Load module	Specifies the name of the load module to be executed.
Options	
Break on entry (Y/N)	Type YES (the default) to cause the test session to break or pause at the start of the program. Additional break on entry options are available on the Session Tailoring screen for each program to be tested.
Break CSECT/pgm id	Enter a program or CSECT name to cause the test session to stop on entry to the specified CSECT in a statically linked module.

Field	Description
File allocation CLIST	
Data set name	Enter 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 there is shown here.
Member	Enter the name of the allocation CLIST. If the Convert Batch JCL facility is used, the member name specified is shown here.
Execution parameters	Enter application execution parameters in this field.
Deallocate after test	Type YES to cause the allocation CLIST to be automatically executed 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 automatically. Type NO to cause the CLIST to be automatically executed before exiting SmartTest-PLI to deallocate the test files. The default value is NO.

### **Converting Batch Execution JCL to a TSO CLIST**

Executing a program in TSO foreground requires that data files used during execution be allocated to the TSO session. This is accomplished by a TSO file allocation CLIST.

A JCL to CLIST conversion facility is provided in SmartTest to automate CLIST creation. Unless otherwise indicated, all DD statements in the JCL stream are converted regardless of the number of steps. The conversion facility includes a comment statement for each step.

### **Converting Batch Execution JCL**

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

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

If the TSO Session Setup screen is already displayed, bypass [step 1](#) and [step 2](#).

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

- 3 Type C in the primary command input area and press Enter. The Convert Batch JCL pop-up, shown in [Figure 29](#), displays.

Figure 29 • 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.VIACENXX_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 Complete these required fields:
- 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.
  - 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.

Option	Description
F	Displays the Allocations from JCL screen, which enables you to edit the SmartTest allocation CLIST using screens rather than directly editing the CLIST.
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	Enter the partitioned dataset containing the JCL to be converted to a TSO CLIST.
Member	Enter the member to be converted to a TSO CLIST.
Data set name	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	The partitioned dataset member that contains the generated allocation CLIST.
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 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 references a PROC in a procedure library (or libraries), specify the procedure library in this field. Specified procedure libraries must be partitioned datasets.

## Editing Allocation CLIST

To edit the CLIST manually, follow this step:

- ▶ Select E on the Convert Batch JCL screen. The generated allocation CLIST, shown in [Figure 30](#), displays.

**Figure 30 • 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.GOOO6VO1') SHR KEEP
```

To provide a user interface for editing the CLIST, follow this step:

- ▶ Select F on the Convert Batch JCL screen. The Allocations from JCL screen, shown in [Figure 31](#), displays. The basic editor provides a list of DDNAME and dataset information that corresponds to the CLIST allocations.

Figure 31 • 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=NEU' 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)	Use 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 is NO.
DD Name	Specifies 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	Specifies the name of the dataset to be allocated. If DISP=DATA, this field is ignored.
Disp	Specifies 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 the Test Session

After you have completed these tasks, you are ready to begin testing a program in TSO foreground by performing these tasks.

- Select the environment.
- Specify the Load module, AKR, load libraries, and procedure libraries (if needed).
- Generate the CLIST.

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 either method of initiating the test session, SmartTest executes the CLIST to allocate the files for the program. When the CLIST has completed, the test session is active with the program source displayed in Program View.

## **Setup Considerations**

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-PLI.
- JCL sysout designations of \* (i.e., `//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:
  - 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 (e.g., `SORT`, `IDCAMS`). Those programs serving to allocate or delete files (e.g., `IEFBR14`) may be skipped and the appropriate statements manually added to the CLIST after conversion.

## Saving the Test Session Setup

Frequently, there is a need to test several programs in an application at the same time. These programs may run in multiple environments requiring many test session setup changes. SmartTest provides the LIST PROFILE primary command 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 test session setup.

### *To save the test session 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 LIST PROFILE command can be abbreviated LI PR.
- 2 In the line command area beside any line that indicates AVAILABLE, type w. In the User profile description field, enter any text that reminds you of the purpose of the setup being saved. For example, you may include the name of the program being tested, the application or project ID, and the date as shown in [Figure 32](#).

**Figure 32 • Profile Dataset Member List Screen Example**

```

                                Profile Data Set Member List
Command ==> -----
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

If your site security allows file sharing in the TSO/ISPF environment, you can also copy setup information from other profile datasets.

### *To copy setup information*

- 1 In the Profile data set name field on the Profile Data Set Member List screen, enter the fully qualified dataset name of the profile dataset containing the desired setup. The saved profile list is refreshed to display the profiles from the specified 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 copied to your profile dataset and may be later modified.

## 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 an FB 80 (fixed block, LRECL=80) 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, enter 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, enter the name of the common, alternate dataset to receive the profiles, for example:

```
'USER.TSO.CNTL'
```

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

The alternate must be an FB 80 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, enter 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, enter 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 Test Session 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 the desired profile by typing S in the line command area of the line containing the profile.

The saved test session setup is restored. You can start the test session using the R command.

## Terminating a Test Session

***To terminate a test session and exit from SmartTest***

- 1 Select Test ► Cancel to cancel the active test session.
- 2 Select File ► Exit to end SmartTest. Any pseudo code, marks, or breaks are saved in the AKR if SAVE is specified on the Options -Product Parameters pop-up. See ["Online Operation Parameters" on page 17](#) for more information on setting these parameters.

As an alternative, exit SmartTest by pressing PF3 until you exit the product.



---

# 3

## Test Session: Additional Environments

---

This chapter describes setting up the SmartTest-PLI test sessions in additional environments and contains these sections:

Topic	Page
<a href="#">Setting Up the Test Session in Other Environments</a>	<a href="#">51</a>
<a href="#">ISPF Dialog Manager</a>	<a href="#">52</a>
<a href="#">IMS/DB Programs in TSO Foreground</a>	<a href="#">63</a>
<a href="#">BTS in TSO Foreground</a>	<a href="#">79</a>
<a href="#">DB2 Programs in TSO Foreground</a>	<a href="#">100</a>
<a href="#">DB2 Stored Procedure Testing Option</a>	<a href="#">102</a>
<a href="#">Testing Programs in a Batch Region</a>	<a href="#">108</a>
<a href="#">Testing DL/I in the Batch Environment</a>	<a href="#">116</a>
<a href="#">Testing BTS in the Batch Environment</a>	<a href="#">117</a>
<a href="#">Testing DB2 in the Batch Environment</a>	<a href="#">118</a>
<a href="#">Testing DFHDRP in the Batch Region</a>	<a href="#">119</a>

### Setting Up the Test Session in Other Environments

#### *DB2 Stored Procedure Option*

SmartTest offers an option that supports testing a DB2 stored procedure (the DB2 Release 4.1 Stored Procedure feature). See [“DB2 Stored Procedure Testing Option” on page 102](#) for more information.

## CICS and IMS Environments

Two additional environments are also available as options for SmartTest. These are discussed in the *ASG-SmartTest CICS User's Guide* and the *ASG-SmartTest IMS User's Guide*.

## ISPF Dialog Manager

### Specifying ISPF Dialog Manager Information

#### To initiate a test session

- 1 From the Environment Selection pop-up, select ISPF Dialog Manager. The ISPF Session Setup screen, shown in [Figure 33](#), displays.

Figure 33 • ISPF Session Setup Screen

```
Command ==> _____ ISPF Session Setup
                        -----
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 -----
NEWRPPL . . . . . ----
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 Complete these fields:
  - a Enter the initial load module to be tested in the Load module field.
  - b Select an option by typing R, S, or V on the command line.
  - c Enter the appropriate information in the fields. See ["Fields" on page 53](#) for a description of the fields on this screen.

**Options**

Option	Description
R	Initiates the ISPF test session. This option is the same as issuing a RUN primary command.
S	Displays the Load Module Intercept List pop-up.
V	Displays the ISPF Allocation pop-up.

**Fields**

Field	Description
Execution: Load module	Specifies the initial load module to be tested.
Break on entry (Y/N)	Specifies whether the test session stops 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 is YES.
Break CSECT/pgm id	Specifies the program name that causes the test session to stop on entry to the specified CSECT in a statically linked module.
Test Profile DSN	Specifies the dataset that contains your ISPF profiles for the application to be tested (i.e., 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	Specifies the ISPF application ID to use for the test (optional).
Initial ISPF panel	Specifies 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 any load modules that may be intercepted by SmartTest for testing. A load module that is linked, attached, or dynamically loaded and called may be entered on the Load Module Intercept List screen. This includes modules that are invoked by an ISPEXEC Select program. Entries must be made before the start of the test.

### To display the Load Module Intercept List pop-up

- 1 Type S in the primary command input area on the ISPF Session Setup screen and press Enter. The Load Module Intercept List, shown in [Figure 34](#), displays.

Figure 34 • Load Module Intercept List Pop-up

```
Load Module Intercept List
Command ==> -----
Type the names of the load modules to be dynamically intercepted.
Then press Enter.

Load Module          Load Module
-----            -----
1 -----            2 -----
3 -----            4 -----
5 -----            6 -----
7 -----            8 -----
9 -----            10 -----
11 -----           12 -----
13 -----           14 -----
15 -----           16 -----
17 -----           18 -----
19 -----           20 -----
21 -----           22 -----
23 -----           24 -----
```

- 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 Session Setup screen.

## Specifying ISPF File Allocation Information

Use the ISPF Allocation pop-up to specify datasets and libraries to be used by the test which are not already allocated to your TSO session. To obtain a list of files allocated to your session, type `TSO VIASALCL`.

### Note:

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 display the ISPF Allocation pop-up

- 1 Type `V` in the primary command input area on the ISPF Session Setup screen and press Enter. The ISPF Allocation pop-up, shown in [Figure 35](#), displays.

Figure 35 • 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 the resulting pop-ups is entered once and retained by SmartTest-PLI.
- 3 Press PF3/PF15 to return to the ISPF Session Setup screen.

## Options

Option	Description
1-ISPF Program Load Library	Displays the ISPF Program Load Library pop-up that is used to specify the location of the load module ISPMMAIN.
2-ISPF Panel/Link Libraries	Displays the ISPF Panel/Link Library pop-up that is used to specify panel and link libraries used by the ISPF test.
3-ISPF Table/Message/Skeleton Libraries	Displays the ISPF Table/Message/Skeleton Library pop-up that is used to specify table, message, and skeleton libraries used by the ISPF test.

Option	Description
4-ISPFLIST Data Set	Displays the ISPF List Data Set Allocation pop-up that is used to specify the ISPF list dataset that is used for the ISPF test.
5-ISPFLG Data Set	Displays the ISPF Log Allocation pop-up that is used to specify the ISPF log dataset that is used for the ISPF test.
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 *ASG-SmartTest Installation Guide* for information on the VIAPUSPF CLIST.

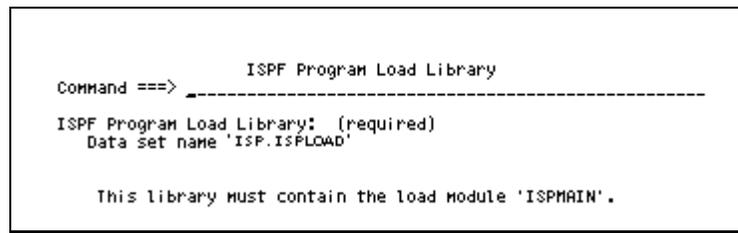
### Specifying the ISPF Program Load Library

**Note:** This procedure is required.

#### To display the ISPF Program Load Library containing the ISPMMAIN member

- 1 Type 1 in the primary command input area on the ISPF Allocation pop-up and press Enter. The ISPF Program Load Library pop-up, shown in [Figure 36](#), displays.

Figure 36 • ISPF Program Load Library Pop-up



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

### Specifying the ISPF Panel/Link Libraries

Use the ISPF Panel/Link Libraries pop-up to specify panel and link libraries for the test. ISPPLIB and ISPLLIB are the defaults.

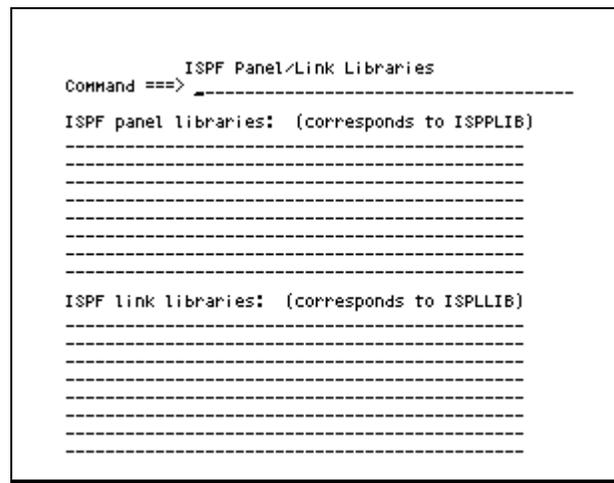
**Note:**

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 the ISPF panel and link libraries

- 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 37](#), displays.

Figure 37 • ISPF Panel/Link Libraries Pop-up



- 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.

### Specifying the ISPF Table/Message/Skeleton Libraries

Use the ISPF Table/Message/Skeleton Libraries pop-up to specify table, message, and skeleton libraries for the test. ISPTLIB, ISPMLIB, and ISPSLIB are the defaults.

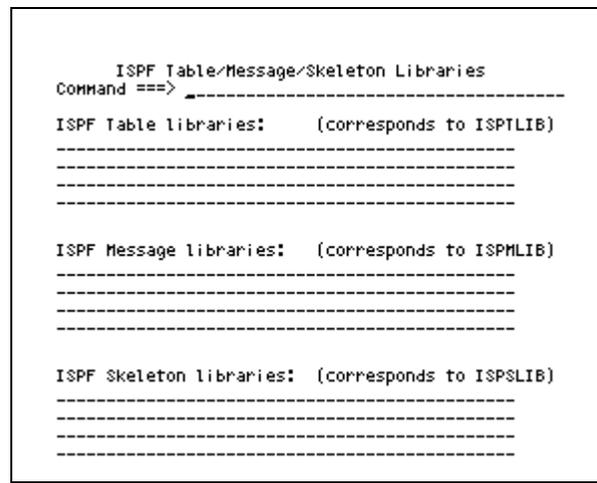
**Note:**

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 the ISPF Table/Message/Skeleton Libraries pop-up

- 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 38](#), displays.

Figure 38 • ISPF Table/Message/Skeleton Libraries Pop-up



- 2 Complete these fields:
  - a Enter the ISPF table library datasets that will be used for the test ISPF Table libraries.
  - b Enter the ISPF message library datasets that will be used for the test ISPF Message libraries.
  - c Enter the ISPF skeleton library datasets that will be used for the test ISPF Skeleton libraries.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up.

## Allocating the ISPF List Dataset

### 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. The ISPF List Data Set Allocation pop-up, shown in [Figure 39](#), displays.

Figure 39 • 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.
- 3 Press PF3/PF15 to return to the ISPF Allocation pop-up.

## Fields

Field	Description
Name	Specifies the name of the dataset. You can specify TEMP to allocate a temporary dataset, TERM to allocate the dataset to a terminal, and SYSOUT to allocate the dataset to JES.
SYSOUT	Specifies a JES output class. An entry in this field is valid only if you specified SYSOUT in the Name field.
DSN DISP	Specifies the disposition of the dataset. The disposition can be NEW, OLD, MOD, or SHR.
Dest	Specifies a JES destination of the SYSOUT output. This can be a remote ID or a 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 you specified TEMP in the Name field.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Units	Specifies the type of space to be allocated for the dataset. You can specify space as CYLINDER, TRACK, or BLOCK.
Primary	Enter the number of primary cylinders, tracks, or blocks to be allocated.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks to be allocated.
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:**

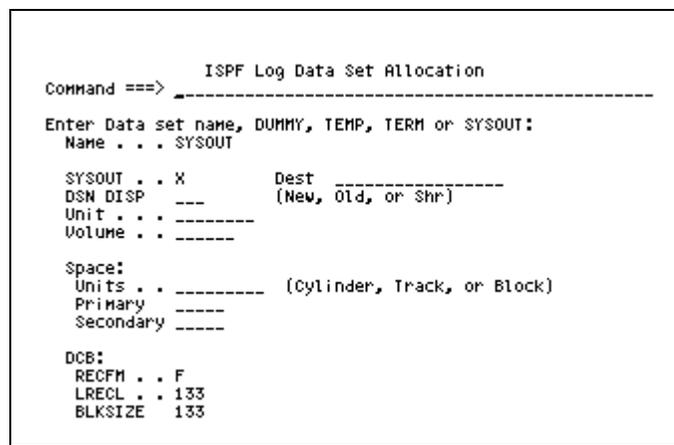
The SYSOUT and DEST fields are available only when you specify SYSOUT in the Name field.

### Allocating the ISPF Log Dataset

#### To specify an ISPF log dataset for the test

- 1 Type 5 in the primary command input area on the ISPF Allocation pop-up and press Enter. The ISPF Log Data Set Allocation pop-up, shown in [Figure 40](#), displays.

Figure 40 • ISPF Log Data Set Allocation Pop-up



- 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. You can specify TEMP to allocate a temporary dataset, TERM to allocate the dataset to a terminal, and SYSOUT 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. The disposition can be NEW, OLD, MOD, or SHR.
Dest	Specifies a JES destination of the SYSOUT output. This can be a remote ID or a 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 you specified TEMP in the Name field.
Volume	Specifies the volume serial number containing the allocated dataset.
Units	Specifies the type of space to be allocated for the dataset. You can specify space 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.
RECFM	Specifies the record format of the list dataset. You can specify record format 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**

### ***To initiate a test session***

- 1** Make sure you have completed these tasks before testing a program in ISPF:
  - a** Select the ISPF Dialog Manager environment.
  - b** Specify all appropriate testing options.
  - c** Specify the Load module and AKR.
  - d** Specify programs to be tested.
  - e** Specify ISPF file allocation information.
- 2** Type RUN or R in the command input area on the ISPF Session Setup screen and press Enter.

**Or**

Press PF4/PF16 (RUN) or PF6/PF18 (STEP).

The test session is active with the program source displayed in Program View.

## IMS/DB Programs in TSO Foreground

### Specifying IMS/DB Setup Information

*To specify the TSO test session parameters for IMS/DB programs and to initiate a test session*

- 1 Select IMS/DB on the Environment Selection pop-up. The IMS/DB Session Setup screen, shown in [Figure 41](#), displays.

Figure 41 • 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 the load module, the CLIST dataset name and optional member, and any appropriate options.

### 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 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
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.
Break on entry (Y/N)	Specifies whether the test session stops 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 is YES.
Break CSECT/ pgm id	Causes the test session to stop on entry to the specified CSECT in a statically linked module.
DLI/DBB/BMP	Specifies the mode of the IMS/DB test session. <b>DLI.</b> Uses private databases with database access through the TSO region; uses DBDLIB and PSBLIB. <b>DBB.</b> Uses privates databases with database access through the TSO region; uses ACBLIB. <b>BMP.</b> Uses public databases with database access through the IMS Control Region.
Plan name	Specifies the DB2 Plan that was generated for the program to be tested when the BIND was performed. If the database region 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 database region type specified is BMP, this field is not necessary.
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 in this field.
Member	Specifies the name of the allocation CLIST. If the Convert Batch JCL facility is used, the member name specified is shown in this field.

Field	Description
Deallocate after test	Specifies whether the CLIST processor will 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.
Execution parameters (APARM)	Specifies application parameter string. This field is for IMS/ESA user's only.

## 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. The IMS Allocation Selection pop-up, shown in [Figure 42](#), displays.

Figure 42 • 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 - IEFRDER

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.

```

- 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.

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

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

\_\_\_\_\_

- 3 Press PF3/PF15 to return to IMS/DB Session Setup screen.

## Options

Option	Description
1 - DFSRESLB/DFSESL	Displays the IMS DFSRESLB/DFSESL Allocation pop-up 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 - IEFORDER	Displays the IMS IEFORDER Allocation pop-up that is used to specify the dataset that invokes the IMS logging facility.
B - IMS Params (BMP)	Displays the IMS BMP Parameters pop-up that is used to specify IMS BMP execution parameters.
P - IMS Params (DLI or DBB)	Displays the IMS DLI/DBB Parameters 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. See the discussion on modifying installed CLIST libraries in the <i>ASG-SmartTest Installation Guide</i> for detailed information on the VIAPUIMS CLIST.

### 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
```

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 on the IMS Allocation Selection pop-up and press Enter. The IMS DFSRESLB/DFSESL Allocation pop-up, shown in [Figure 43](#), displays.

**Figure 43 • IMS DFSRESLB/DFSESL Allocation Pop-up**

```

IMS DFSRESLB/DFSESL ALLOCATION
Command ===> -----
Enter IMS Load Library Data set names:
'IMS.RESLIB'
-----
-----
Enter DB2 Load Library Data set names:
'DSN.DSNLOAD'
-----
-----
-----

```

- 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:

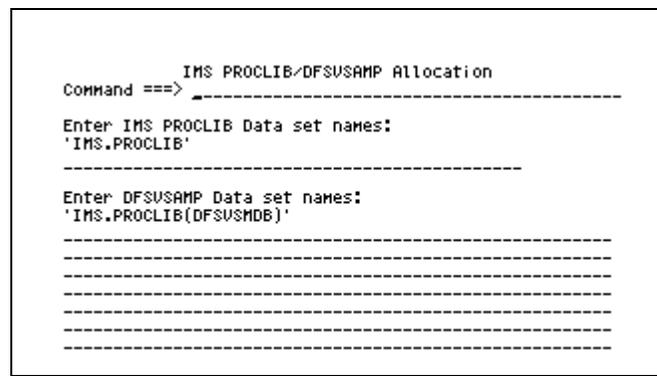
```
//PROCLIB DD DSN=IMS.PROCLIB,DISP=SHR
```

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. The IMS PROCLIB/DFSVSAMP Allocation pop-up, shown in [Figure 44](#), displays.

Figure 44 • IMS PROCLIB/DFSVSAMP Allocation Pop-up



- 2 Enter the appropriate information in the fields.
- 3 When all necessary information is specified, press PF3/PF15 to return to the IMS Allocation Selection pop-up.

## Fields

Field	Description
Enter IMS PROCLIB Data set names	The procedure library datasets to be allocated to the PROCLIB DDNAME.
Enter DFSVSAMP Data set names	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 Data base 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=USER.PSBLIB,DISP=SHR
// DD DSN=USER.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. The IMS PSB/DBD Allocation pop-up, shown in [Figure 45](#), displays.

**Figure 45 • IMS PSB/DBD Allocation Pop-up**

```

IMS PSB/DBD Allocation
Command ==> -----
Enter PSB/DBD Library Data set names:
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----
-----

```

- 2 Enter the PSB and DBD dataset names to be allocated for use by IMS. 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 Data base region type field on the IMS Session Setup screen contains DBB.

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

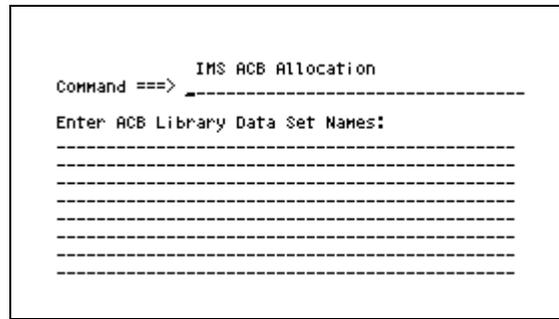
```
//IMSACB DD DSN=USER.ACBLIB, DISP=SHR
```

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

#### To specify the ACB library datasets

- 1 Type 4 in the primary command input area on the IMS Allocation Selection pop-up and press Enter. The IMS ACB Allocation pop-up, shown in [Figure 46](#), displays.

Figure 46 • IMS ACB Allocation Pop-up



- 2 Enter the ACB dataset names to be allocated to IMSACB for DBB testing.
- 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. The IMS IMSMON Allocation pop-up, shown in [Figure 47](#), displays.

Figure 47 • 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 you use the IMS monitor, allocate a temporary dataset or permanent dataset in the Name field. Additionally if the IMS monitor is to be invoked, 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. You can specify TEMP to allocate a temporary dataset. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. The 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.
Units	Specifies the type of space to be allocated for the dataset. Specify space 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.
RECFM	Specifies the record format of the IMSMON dataset. Specify record format 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 IEFORDER Allocation*

Use the IMS IEFORDER Allocation pop-up to specify an IEFORDER dataset to provide backout and recovery for your IMS databases. This is an example of a typical IEFORDER DD statement from batch execution JCL:

```
//IEFRDER DD DSN=IMSLOG, DISP=(NEW,CATLG),  
// UNIT=SYSDA, SPACE=(TRK,(3,2),RLSE),
```

See your batch execution JCL and PROCs for the IEFORDER 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. The IMS IEFRDER Allocation pop-up, shown in [Figure 48](#), displays.

**Figure 48 • 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:**

Entering DUMMY in the Name field causes the IMS backout process to fail, should you need to back out updates to your application databases.

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

**Fields**

Field	Description
Name	Specifies the name of the dataset. You can specify TEMP to allocate a temporary dataset. No allocation is performed if this field is left blank.
DSN DISP	Specifies the disposition of the dataset. The 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 IEFRDER dataset.
Units	Specifies the type of space to be allocated for the dataset. Specify space as CYLINDER, TRACK, or BLOCK.

Field	Description
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.
RECFM	Specifies the record format of the IEFORDER dataset. Specify record format 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:

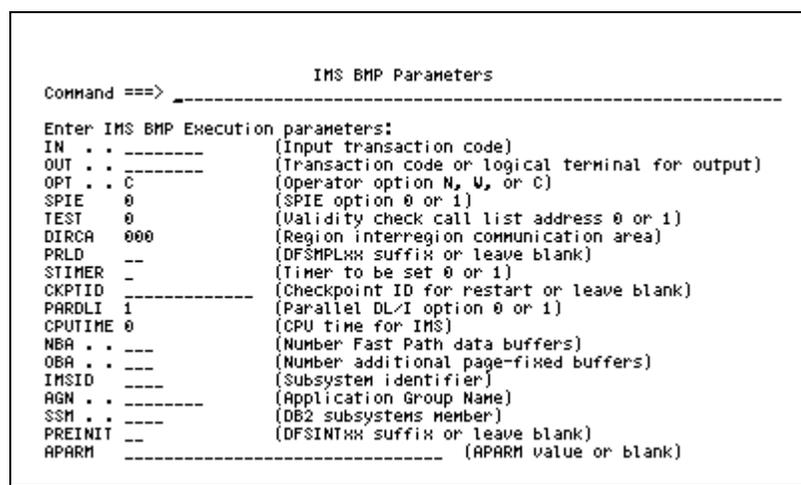
```
//STEPNAME EXEC PGM=DFSRR00, PARM=(BMP, &MBR, &PSB, &IN,
//                                &OUT, OPT&SPIE&TEST&DIRCA, &PRLD,
```

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. The IMS BMP Parameters pop-up, shown in [Figure 49](#), displays.

Figure 49 • IMS BMP Parameters Pop-up



- 2 Enter the appropriate information in the fields.

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	Indicates that the application program is accessing the message queues. The OUT parameter is ignored when this parameter is specified.
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. The valid options are N (notify), W (wait), or C (cancel). The default value is N.
SPIE	Indicates whether control is passed when a program exception (OC1-0CF) 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 and can be two alphabetical 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 you specify CPUTIME= <i>n</i> , the STIMER value must be 1. STIMER=1 results in performance degradation and should only be specified when gathering statistics. Zero (0) specifies that the timer is not to be set.
CKPTID	Specifies the checkpoint/restart ID used to restart a program.

Field	Description
PARDLI	Indicates where DL/I processing is to be performed. Zero (0) specifies that DL/I processing is to be performed within the BMP region. One (1) specifies that all DL/I processing is to be performed in the IMS control region.  <b>Note:</b> PARDLI has been forced to 1 for IMS system integrity issues.
CPUTIME	This parameter must be 0 (zero) for SmartTest-PLI.
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	Specifies 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	Specifies the application group name used for resource access security.
SSM	Specifies a site-specific value that is used to allow access to selected DB2 subsystems under IMS. You can enter 1 to 4 alphanumeric characters, or leave this field blank.
PREINIT	Specifies a suffix for DFSINT that can be two alphabetical characters. DFSINT <sub>xx</sub> 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.

### IMS DLI/DBB Parameters

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

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

```
//STEPNAME EXEC PGM=DFSRR00, PARM=(DLI, &MBR, &PSB, &BUF,
//                               &SPIE&TEST&EXCPVR&RST, &PRLD, &SRCH,
```

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. The IMS DLI/DBB Parameters pop-up, shown in [Figure 50](#), displays.

Figure 50 • IMS DLI/DBB Parameters Pop-up

```

                                IMS DLI/DBB Parameters
Command ==> -----
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.

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

- 3 When all necessary information is specified, press PF3/PF15 to return to the IMS Allocation Selection pop-up.

### Fields

Field	Description
BUF	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 Utility Control Facility (UCF) is to be used for restarts. The default value is 0 (zero), which indicates UCF is not to be used.
PRLD	A suffix for DFSMPL that can be two alphabetical characters. The specified suffix is used to pre-load 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	The checkpoint/restart ID used to restart a program.
MON	Indicates if the IMS monitoring option is active. The default 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 Formatted Dump Delete List (FDDL). 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 is Y (yes), which indicates the address space can be swapped.
DBRC	Indicates if Database Recovery Control is to 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.

Field	Description
IRLM	Indicates if the IMS Resource Lock Manager (IRLM) is to be used. Y specifies that the IRLM is to be used. The default is N, which specifies that the IRLM is not to be used.
IRLMNM	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 is N, which indicates the dynamic backout option is not active.

## BTS in TSO Foreground

### Specifying BTS Setup Information

*To specify the TSO test session parameters for BTS for testing*

- 1 Select BTS on the Environment Selection pop-up. The BTS Session Setup screen, shown in [Figure 51](#), displays.

**Figure 51 • 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 Complete these fields:
  - a Enter the load module in the Load module field.
  - b Enter the CLIST dataset name and optional member in the Data set name and member fields, respectively.
  - c Enter the appropriate information in the options. A description of each option is provided below.
- 3 Complete the remaining fields on the screen as necessary.

### Options

Option	Description
R	Invokes the CLIST to initiate the test session. For more information, see <a href="#">“SmartTest-PLI Test Session” on page 13</a> .
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. For more information, see <a href="#">“SmartTest-PLI Test Session” on page 13</a> .
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. For more information, see <a href="#">“SmartTest-PLI Test Session” on page 13</a> .
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. For more information, see <a href="#">“Specifying BTS File Allocation Information” on page 83</a> .

### 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.
Break on entry (Y/N)	Indicates whether the test session stops 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 is YES.
Break CSECT/ pgm id	Causes the test session to stop on entry to the specified CSECT in a statically linked module.

Field	Description
Data base region type: DLI/DBB/BMP	Specifies the mode of the IMS/DB test session.  <b>DLI:</b> Uses private databases with database access through the TSO region; uses DBDLIB and PSBLIB.  <b>DBB:</b> Uses privates databases with database access through the TSO region; uses ACBLIB.  <b>BMP:</b> Uses public databases with database access through the IMS Control Region.
Data set 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	Indicates that the CLIST processor will 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 is NO.
Data set 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-PLI. These transactions are saved from session to session.

The BTSIN data might 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. The Preview BTSIN Data Set screen, shown in [Figure 52](#), displays.

**Figure 52 • Preview BTSIN Data Set Screen**

```

Command ==> _____ Preview BTSIN Data Set _____ Scroll ==> CSR
Select the Transactions and Programs to be tested.
$ Trancode Program PSBNAME Lang Type PLC SPA Edit rtn DB2 PLAN
- ----
- ADDI TSTSAM04 TSTSAM04 CBL MSG 1
- ADDINU TSTSAM04 TSTSAM04 CBL MSG 1
- ADDPART TSTSAM04 TSTSAM04 CBL MSG 1
- ADDPN TSTSAM04 TSTSAM04 CBL MSG 1
- BR14 IEFBR14 DFSSAM02 ASM MSG 1
- CLOSE TSTSAM05 TSTSAM05 CBL MSG 1
- COBZD88 COBZD88 DFSSAM02 CBL MSG 1
- DISBURSE TSTSAM06 TSTSAM06 CBL MSG 1
- DLETI TSTSAM04 TSTSAM04 CBL MSG 1
- DLETIU TSTSAM04 TSTSAM04 CBL MSG 1
- DLETPART TSTSAM04 TSTSAM04 CBL MSG 1
- DLETPN TSTSAM04 TSTSAM04 CBL MSG 1
- DSPALLI TSTSAM07 TSTSAM07 CBL MSG 1
- LINKASM4 LINKASM4 DFSSAM02 ASM MSG 1
- LINKCOB LINKCOB DFSSAM02 CBL MSG 1
- LINKIMS LINKIMS DFSSAM02 CBL MSG 1
  
```

- 2 Select the transaction(s) to be tested by typing S in the line command area and pressing 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.
PLC	Specifies the processing limit count for this transaction. The default is 1.

Field	Description
SPA	Defines the size of the scratch pad area, in bytes, for the transaction named by the TC= operand.
EDIT <i>rtn</i>	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. The BTS Allocation Selection pop-up, shown in [Figure 53](#), displays.

Figure 53 • BTS Allocation Selection Pop-up

```

Command ==>      BTS Allocation Selection
-----
1 - BTS Load library
2 - FORMAT
3 - QIOPCB
4 - QALTPCB
5 - QALTRAM
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.

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

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

- 3 Press PF3/PF15 to return to the BTS Session Setup 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/ESA Message Format Services (MFS) datasets.
3 - QIOPCB	Displays the BTS QIOPCB Allocation pop-up that is used to specify the BTS work file dataset.
4 - QALPCB	Displays the BTS QALPCB 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. See <a href="#">“Specifying IMS/DB Setup Information” on page 63</a> for more information.
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. See the discussion on modifying installed CLIST libraries in the <i>ASG-SmartTest Installation Guide</i> for detailed information on the VIAPUBTS CLIST.

### Specifying the BTS Load Library

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

```
//STEPLIB DD DSN=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 for BTS

- 1 Type 1 in the primary command input area on the BTS Allocation Selection pop-up and press Enter. The BTS Load Library pop-up, shown in [Figure 54](#), displays.

Figure 54 • BTS Load Library Pop-up

```

BTS Load Library
Command ==> -----
Enter BTS Load Library Data Set Name:
'BTSLIB'
Enter BTS STAX indicator:  (Y,N, or blank)

```

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

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

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 message BTS0015A INVALID KEYWORD ON PARM STRING, 'DLI' ASSUMED 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.

### Specifying 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:

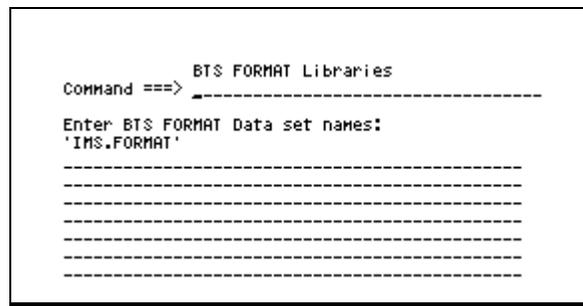
```
//FORMAT DD DSN=USER.FORMAT, DISP=SHR
//          DD DSN=IMS.FORMAT, DISP=SHR
```

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 library*

- 1 Type 2 in the primary command input area on the BTS Allocation Selection pop-up and press Enter. The BTS Format Libraries pop-up, shown in [Figure 55](#), displays.

**Figure 55 • BTS FORMAT Libraries Pop-up**



- 2 Enter the MFS datasets that are used to format the screen message.
- 3 Press PF3/PF15 to return to the BTS Allocation Selection pop-up.

### Allocating the BTS QIOPCB

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, SPCE=(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 for BTS**

- 1 Type 3 in the primary command input area on the BTS Allocation Selection pop-up and press Enter. The BTS QIOPCB Allocation pop-up, shown in [Figure 56](#), displays.

**Figure 56 • 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 . . VBS
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.
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 is 1.

Field	Description
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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 is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default is 3072.

### Allocating the *BTS QALTPCB*

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:

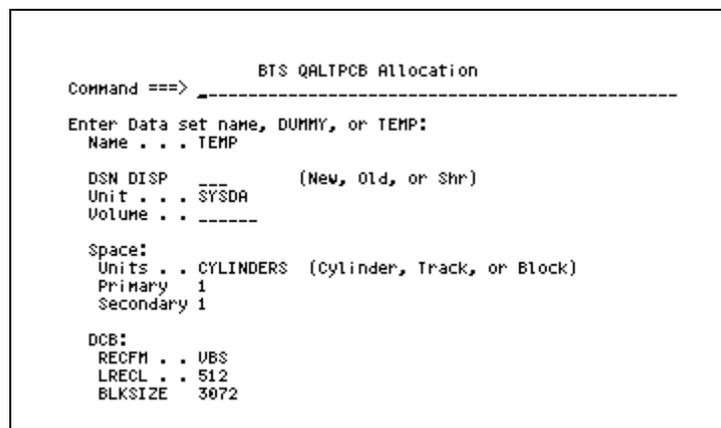
```
//QALTPCB DD UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=FB,LRECL=1024,BLKSIZE=3072)
```

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

### To allocate the alternate message queue dataset used by *BTS*

- 1 Type 4 in the primary command input area on the BTS Allocation Selection pop-up and press Enter. The BTS QALTPCB Allocation pop-up, shown in [Figure 57](#), displays.

**Figure 57 • BTS QALTPCB 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 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.
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 is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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 is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default is 3072.

## Allocating the BTS QALTRAN

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),
//          DBC=(RECFM=U,LRECL=1024,BLKSIZE=3072)
```

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. The BTS QALTRAN Allocation pop-up, shown in [Figure 58](#), displays.

**Figure 58 • 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 . . . CYLINDERS (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.

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.
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 is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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 is 1024.
BLKSIZE	Specifies the maximum length, in bytes, of a block for the allocated dataset. The default is 3072.

### Allocating the *BTS BTSOUT*

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:

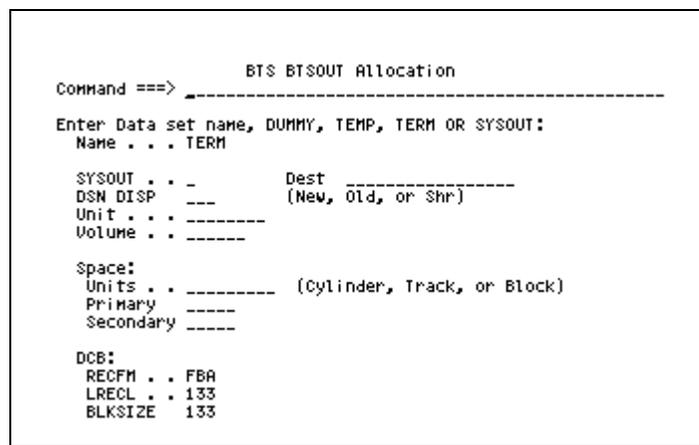
```
//BTSOUT DD SYSOUT=*
//          DCB=(RECFM=FBA, LRECL=133.B; LSOZE=133)
```

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. The BTS BTSOUT Allocation pop-up, shown in [Figure 59](#), displays.

**Figure 59 • 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.

Field	Description
Volume	Specifies the volume serial number containing the allocated dataset.
Units	Specifies the type of space to be allocated for the dataset. Space can be specified as CYLINDER, TRACK, or BLOCK. The default is CYLINDER.
Primary	Specifies the number of primary cylinders, tracks, or blocks allocated. The default is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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 is 133.
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.

### Allocating the *BTS BTSPUNCH*

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:

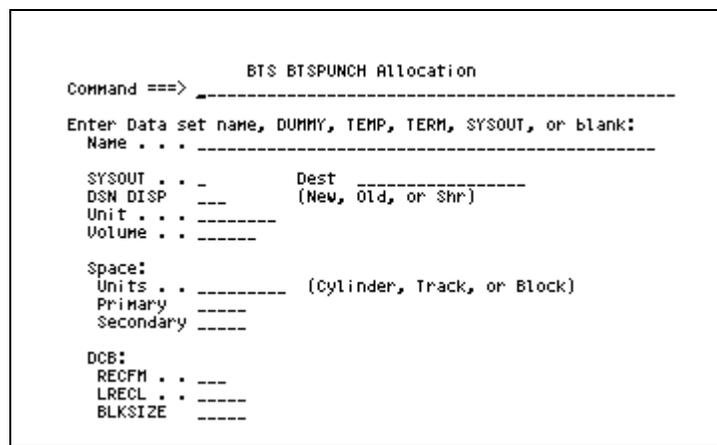
```
//BTSPUNCH DD DSN=USER.BTSPUNCH, DISP=(,CATLG),
//           UNIT=SYSDA, SPACE=(CYL,(10,2),RLSE),
//           DCB=(RECFM=FB,LRECL=80,BLKSIZE=6800)
```

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. The BTS BTSPUNCH Allocation pop-up, shown in [Figure 60](#), displays.

**Figure 60 • BTS BTSPUNCH Allocation Pop-up**



- 2 Enter 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.
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 is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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.

### Allocating the *BTS BTSDEBUG*

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:

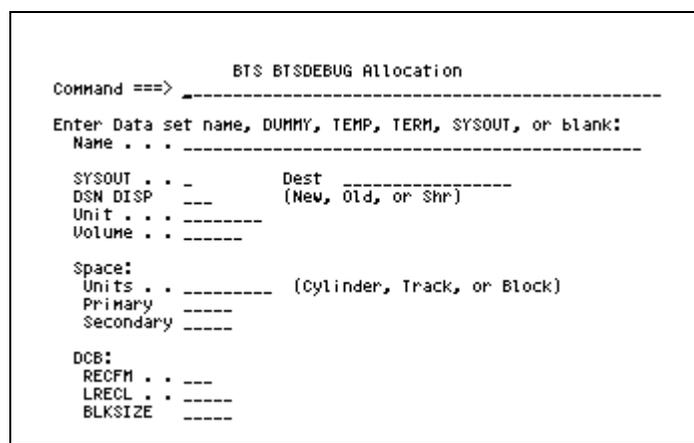
```
//BTSDEBUG DD DUMMY
```

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. The BTS BTSDEBUG Allocation pop-up, shown in [Figure 61](#), displays.

**Figure 61 • 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.
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 is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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.

### ***Allocating the BTS BTSSNAP***

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:

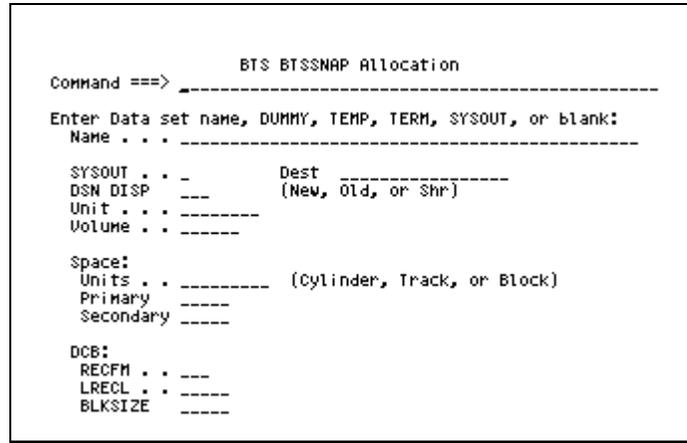
```
//BTSSNAP DD SYSOUT=*
```

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 taken by BTS during execution**

- 1 Type 9 in the primary command input area on the BTS Allocation Selection pop-up and press Enter. The BTS BTSSNAP Allocation pop-up, shown in [Figure 62](#), displays.

**Figure 62 • 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.
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 is 1.
Secondary	Specifies the number of secondary cylinders, tracks, or blocks allocated. The default is 1.
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 from BTS Allocation Selection**

Use the IMS Allocation Selection pop-up to select the IMS datasets, libraries, and parameters to be defined to IMS.

Select option I on the BTS Allocation Selection pop-up to display the IMS Allocation Selection pop-up.

See the *Batch Terminal Simulator: Program Reference/Operations Manual* (SH20-5523) for more information about BTS.

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

If your IMS file allocations were not entered previously or you need to change them, see [“Specifying IMS File Allocation Information” on page 65](#).

---

## DB2 Programs in TSO Foreground

### Specifying DB2 Setup Information

To specify the TSO test session parameters for DB2 programs

- 1 Select DB2 on the Environment Selection pop-up. The DB2 Session Setup screen, shown in [Figure 63](#), displays.

Figure 63 • 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 Complete these fields:
  - a 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 the appropriate information in the options.

### Options

Option	Description
R	Initiates the DB2 test session. This is the equivalent of entering the RUN command. For more information, see <a href="#">“SmartTest-PLI Test Session” on page 13</a> .
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
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.
Break on entry (Y/N)	Indicates whether the test session will 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 is YES.
Break CSECT/pgm id	Causes the test session to stop on entry to the specified CSECT in a statically linked module.
Execution parameters	Specifies required application parameters.
Data set 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	Determines whether the CLIST processor will 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 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 PL/I, COBOL, C/370, or Assembler. 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

- 1 Select DB2 Procedure on the Environment Selection pop-up. The DB2 Stored Procedures Setup screen, shown in [Figure 64](#), displays.

Figure 64 • DB2 Stored Procedures Setup Screen

```

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

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 RRSRF (Y/N) . . . NO
DB2 Schema name . VIR123
DB2 Procedure name _____

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

```

- 2 Under Execution, complete these steps:
  - 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, complete these steps:
  - 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). The default is YES.
    - 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 155](#) for more information.
  - b** To stop the test session on entry to a specified CSECT in a statically linked module, enter the program name in the Break on CSECT/pgm ID field.
- 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 and member name for the CLIST.
    - To deallocate the test files at the end of this test session or after a CANCEL command, type YES in the Deallocate after test field. To deallocate the test files at the end of the SmartTest session, type NO. (The default is NO.)
  - b** Type C to display the Covert Batch JCL pop-up. Follow the instructions in [“Converting Batch Execution JCL to a TSO CLIST” on page 40](#).
- 5** If you are expecting input parameters from the stored procedure, type D to display parameters  
**Or**  
If you want to review output, see [“Reviewing DB2 Stored Procedure Parameters” on page 105](#).
- 6** Type R to begin the test session. See [“Initiating the DB2 Stored Procedure Test” on page 107](#) for more information.

## Reviewing DB2 Stored Procedure Parameters

To review input and output parameters

- 1 Review the parameters on the DB2 Stored Procedures Parameters screen, shown in [Figure 65](#)). Each parameter specified in the PARMLIST column on the SYSIBM.SYSPROCEDURES table displays.

Figure 65 • DB2 Stored Procedures Parameters Screen

DB2 Stored Procedures Parameters						
Command ==>	Name	Type	Address	Structure	Value	Hex Value
	CUSTNO	IN	000D5148	CHARACTER(8)	A1234	C1F1F2F3 F4404040
	CUSTNM	OUT	000D5198	CHARACTER(20)	JONES	D1D6D5C2 C2C9D3
	CUSTBAL	OUT	000E0000	SMALLINT	50000	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 of the data at that storage location. The maximum is 8 characters.
Hex Value	Specifies the hexadecimal representation of the data at that storage location.

- 2 Type S in the line command area to the left of the parameter Name and press Enter.
  - If the data is defined as numeric, the DB2 Stored Procedures Numeric Display screen displays.
  - If the data is not defined as numeric, the Memory Display screen displays.

## 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 (see [Figure 66](#)).

**Figure 66 • DB2 Stored Procedures Numeric Display Screen**

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

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

Field	Description
Numeric Data	Displays the current numeric value for the parameter. You can 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).
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 changes.
- 4 Press PF3 twice to return to the DB2 Stored Procedures Setup screen.
- 5 Proceed to [“Initiating the DB2 Stored Procedure Test” on page 107](#).

**To review/change non-numeric values**

- 1 Review the selected parameter on the Memory Display screen (see [Figure 67](#)).

**Figure 67 • Memory Display Screen**

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

This table describes the fields on the Memory Display screen:

Field	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.

- 2 To make a change, type over the values displayed.
- 3 Press Enter to save your changes.
- 4 Press PF3 twice to return to the DB2 Stored Procedures Setup screen.
- 5 Proceed to [“Initiating the DB2 Stored Procedure Test” on page 107](#)

**Initiating the DB2 Stored Procedure Test**

You have set up your DB2 Stored Procedures test and 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) may be used during your DB2 test session.

### *To initiate the test session*

- 1 Type R on the DB2 Stored Procedures Setup screen and press Enter.
- 2 Watch the test session execute the stored procedure in Program View. If you did not set breaks during set up, the program executes until one of these conditions occurs:
  - An abend error condition
  - An address stop
  - Completion of the stored procedure

If you set Break on entry, the test session stops at the beginning of PROC (and at other breakpoints, as specified).

- 3 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 103](#)).
- 4 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 105](#)).

To use additional SmartTest functions, see [“Testing Techniques” on page 121](#) and [“The Intelligent Search Function” on page 163](#).

## Testing Programs in a Batch Region

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 in these circumstances:

- Program(s) to be tested requires too much region for execution in TSO foreground.
- Test requires media not accessible through TSO foreground (i.e., tape datasets).
- Test must consist of multiple interdependent steps.
- Test requires the use of special output forms.
- There is a limit of one program to test for each SmartTest batch execution.

- When 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.

The Batch Session Setup screens 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. Symbolic support may be unavailable for programs loaded from read protected libraries.

### Specifying Batch Connect Setup Information

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

#### To establish a batch test session in an MVS batch region

- 1 Select MVS Batch on the Environment Selection pop-up. The Batch Session Setup screen, shown in [Figure 68](#), displays.

Figure 68 • Batch Session Setup Screen

```

                                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 Complete these 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 the appropriate information in the options. A description of each option is provided below.

## 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	Display 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-PLI changes.
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
Load module	Specifies the initial load module to be tested.
Procedure Name	Specifies PROC name of the procedure containing the execute statement for the load module to be tested.
Step Name	Specifies step name in the batch job that executes the load module to be tested.
Break on entry (Y/N)	Indicates whether the test session will 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 is YES.

Field	Description
BREAK CSECT/pgm id	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 name of the 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.
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 used, specify the subsystem library dataset name.
Member	Specifies the member to be submitted or edited, or from which JCL is to be generated.
Data set name	Specifies the partitioned dataset that contains the generated JCL.
Member	Specifies the member that contains the generated JCL.
Session name for batch communications TSO Userid	Contains 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.

When you select option S or G on the Batch Session Setup screen, these changes are made to the specified batch execution JCL:

- A STEPLIB DD statement for the step to be tested is added or updated to concatenate the ESW load library to the user load library.
- The REGION size is changed to the value specified on this screen.
- A VIAQUEUE DD statement is added for the SmartTest-PLI 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-PLI to gain control at the appropriate step.

ASG recommends that the JOBNAME for the JCL you submit be unique. 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.

The REGION from the EXEC JCL card is changed to the specified REGION size on the screen. A DD card is added to the STEPLIB for the dataset containing the ESW library. A DD card is also added for the SmartTest VIAQUEUE dataset. This dataset is used to transfer data to and from the TSO address space when the test job is executing. PROCs are automatically expanded. Enter the required user PROCLIBs on the Procedure Libraries pop-up (ENV.P).

ASG also recommends that you verify that the work file allocation information entered on the Options - Product Allocations pop-up is valid for the batch test session. The GENERIC UNIT and VOLUME SERIAL fields should contain the appropriate values for your installation. Inappropriate work file allocations can cause unpredictable results during the batch test session. See [“SmartTest-PLI Test Session” on page 13](#) for detailed information on the Options - Product Allocations 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.

## **Submitting and Connecting to a Batch Job**

To establish a test session in a batch region, submit a batch job and connect to it. After submitting the job, use the Connect to Job pop-up to establish the TSO interface to the job.

### **Submitting 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:** \_\_\_\_\_

For future reference, note the job name and assigned job number shown in the TSO message that displays (see [Figure 69](#)).

\_\_\_\_\_

**Figure 69 • Connect to Job Message Screen**

```
+ASG2076I Batch JOB JOBNAME is waiting for connection by ASG-SmartTest
***
```

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

## Connecting to a 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. The Connect to Job pop-up, shown in [Figure 70](#), displays.

Figure 70 • Connect to Job Pop-up

```

                                Connect to Job
Command ==> ----- Scroll ==> CSR
TSO Userid USER12      (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
-----
        -

```

- 2 Select a job by typing **S** in a Select field that has a status of **WAITING FOR CONNECTION** and pressing Enter.

## Options

Option	Description
S	Connects to an available batch session. The selected job becomes active in SmartTest (see <a href="#">“Initiating the Batch Test” on page 114</a> ).
P	Purges (delete) a batch session or submitted job.

## Status Descriptions

Status	Meaning
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.

Status	Meaning
TEMPORARILY SUSPENDED	Indicates that the job was previously connected to the online test session by the specified TSO user ID but was suspended using 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.

### Initiating the Batch Test

After specifying all information for testing in the batch environment on the appropriate Batch Session Setup screens, you can initiate the test by submitting the JCL and connecting to the batch job.

When the batch connection is complete, the program source or disassembled object code displays in Program View (shown in [Figure 71](#)) and the test session is active.

**Figure 71 • Batch Connect Program View Screen**

```

File View Test Search List Options Help
-----
Command ==> Program View (A) VIAPPLI.VIAPPLI1 -A Scroll ==> CSR
(B)296
>>>>>> CALL PROGRAM_INIT;
000298
000314 CALL PARM_EDIT; /* INITIAL_PARM_EDIT */
000315
000316 IF PARM_TEXT = 'EXIT' THEN
000317 RETURN;
000318 ELSE
000319 IF PARM_TEXT = 'INVALID' THEN
000320 DO;
000321 CALL INVALID_PARM;
000322 RETURN;
000323 END;
000324
000325 IF CURRENT_PARM_NUMBER = +0 THEN
000326 CALL INVALID_PARM;
000327
+ (C) -----+
|STATUS: BREAK AT START OF TEST SESSION PROGRAM: VIAPPLI1 DATE: DDMMYYYY |
| STMT: 000297 OFF: 00029E AMODE: 31 MODULE: VIAPPLI TIME: HH:MM:SS |
|SOURCE: CALL PROGRAM_INIT; |
+-----+

```

### Screen Description

Screen Area	Description
(A) VIAPPLI.VIAPPLI1 -A	Reflects the module and program ( <i>module.program</i> ) being viewed and indicates the program is in an active (-A) test session.
(B) >>>>>>	Indicates the next statement to be executed. The statement is highlighted and chevrons appear in the number column.

Screen Area	Description
(C) STATUS: BREAK	Displays the status information when a test session is active:
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.
AMODE	Specifies the addressing mode for the load module.
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.
DATE	Specifies the current date.
TIME	Specifies the time the last program statement was executed.

### Batch Region Considerations

When submitting JCL through SmartTest-PLI for testing in a batch region, you may need to modify your JCL before processing. Keep these considerations in mind:

- The TSO SUBMIT command is issued internally within SmartTest so the system standard exit receives control and can process the JCL for job submission. The JCL must have a JOB card with a job name that will not be changed by the TSO SUBMIT exit.
- When testing DL/I in the batch environment, if the program to be tested does not access DB2 databases or use BMP to access the DB2 database:
  - Select the IMS Batch environment on the Environment Selection pop-up.
  - Submit the JCL from the IMS Batch Session Setup screen.
- There is a limit of one program to test for each SmartTest batch execution.

## 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 72](#).

Figure 72 • 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 screen.

## Testing BTS in the Batch Environment

### To use the BTS Batch environment

- 1 Select the BTS Batch environment on the Environment Setup Menu. The BTS Batch Session Setup screen, shown in [Figure 73](#), displays.

Figure 73 • 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 screen.

## 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. The DB2 Batch Session Setup screen, shown in [Figure 74](#), displays.

Figure 74 • DB2 Batch Session Setup Screen

```

                                DB2 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 pop-up.

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 execute of the procedure. The DD for SYSTSIN in the procedure may use the DDNAME Parameter to allow the DD statement to be after the execute.

**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.

## Support for Changed DB2 Interface

DB2 Version 4.1 provides an alternative method in which IMS/DB can set up the DB2 environment. The execution parameter specifies the application program name and must specify the SSM field, for example:

```
EXEC PGM=DFSRRRC00, PARM="DLI,pgm,psb,,,,,,,,,,,,,ssm"
```

SmartTest supports this new interface, but requires that the JCL specify the DDITV01DD card. See the *IBM DB2 for MVS Application Programming and SQL Guide* for more information on this alternative method.

## Testing DFHDRP in the Batch Region

### To test DFHDRP in the batch environment

- 1 Select the MVS Batch environment on the Environment Selection pop-up.
- 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 by typing S or SG on the setup screen.
- 4 Connect to the batch session using the batch Connect to Job screen.

### DFHDRP Batch JCL

[Figure 75](#) shows the DFHDRP batch JCL before being modified by SmartTest.

**Figure 75 • 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
//DRPTTEST EXEC DFHDRPB
//SYSOUT DD SYSOUT=&SYSOUT
//REPORT DD SYSOUT=&SYSOUT
//D121PART DD DSN=PART.DB10V09, DISP=SHR
```

[Figure 76](#) shows the DFHDRP batch JCL after being automatically modified by SmartTest.

**Figure 76 • 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 chapter describes testing techniques for SmartTest-PLI and contains these sections.

Topic	Page
<a href="#">Learning the Commands</a>	<a href="#">122</a>
<a href="#">Testing with SmartTest</a>	<a href="#">122</a>
<a href="#">Controlling Program Execution</a>	<a href="#">124</a>
<a href="#">Viewing and Changing Test Session Data</a>	<a href="#">135</a>
<a href="#">Program Execution History (Backtrack)</a>	<a href="#">140</a>
<a href="#">Using Pseudo Code</a>	<a href="#">149</a>
<a href="#">Making Proposed Changes Permanent</a>	<a href="#">154</a>
<a href="#">Using Multiple Programs</a>	<a href="#">154</a>
<a href="#">Tailoring a Test Session by Program</a>	<a href="#">155</a>
<a href="#">Capturing and Replaying Test Sessions</a>	<a href="#">155</a>
<a href="#">Position the Display at the Next Executable Statement</a>	<a href="#">157</a>
<a href="#">Saving Keystrokes</a>	<a href="#">158</a>
<a href="#">Linking to Alliance</a>	<a href="#">160</a>

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

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

---

## Learning the Commands

SmartTest provides both a Common User Access (CUA) and a command interface to accomplish your testing activities. CUA presents an intuitive way of performing tasks. As you gain experience with SmartTest, you may find that using commands is faster than CUA pull-downs for frequently used functions. To assist you in making the transition from CUA to commands, SmartTest includes a Learn Mode. When Learn Mode is on, SmartTest inserts a pop-up, before executing a request, that shows the primary command equivalent to the selections made on the CUA pull-downs and pop-ups. This pop-up helps you learn the command syntax for frequently used functions. The full CUA functionality remains available also.

To turn the learn mode on, type `SET LEARN ON` in the primary command input area and press Enter. The short message area displays the message `LEARN ON`.

To turn the learn mode off, type `SET LEARN OFF` in the primary command input area and press Enter. The short message area displays the message `LEARN OFF`.

## Testing with SmartTest

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

SmartTest has two distinct methods for controlling the application program—`MONITOR` and `NOMONITOR`. Each method has a strength and a weakness and each 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 in the `RUN` function. The `SET MONITOR` command determines the default. Guidelines for choosing either `MONITOR` or `NOMONITOR` are included in this manual.

### Testing Using the *MONITOR* Method

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

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

Using the MONITOR method also allows temporary modification of the program using COBOL-compatible pseudo code. Additional advantages to the MONITOR method include minimal test setup requirements, detection of INVALID BRANCH situations, etc.

Since the monitoring method requires execution of several instructions by SmartTest to execute one application instruction, the CPU overhead of monitoring a large test is significant. 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.

The NOMONITOR method places intentional abend instructions (breakpoints) at various points in the object code, then gains control when the code abends (S0C1) during execution. This technique requires that each program analyzed must be linked with the REUS parameter. This parameter is automatically supplied by the Analyze process, but can be overridden on the SmartTest List - Analyze Submit pop-up.

### **Guideline for Using the MONITOR/NOMONITOR Methods**

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

For example, the Execution Tracking screen shows all of the statements that executed, in execution 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 label where a suspected problem exists and then execute the program using NOMONITOR to execute the test 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 internal program checks set by SmartTest before it can examine them, yielding unpredictable results.

To test such programs, disable the SPIE and restart the test using NOMONITOR. If there is no way to disable the SPIE, it is necessary to run through the SPIE/ESPIE using MONITOR. Then, you can run through the rest of the code using NOMONITOR.

---

## Testing Dynamically Called Load Modules

Prior to executing a CALL verb, many application programs use the FETCH verb to locate or dynamically load the module to be CALLED. When using the MONITOR method of testing, SmartTest can detect whether a module that is FETCHed has been statically linked.

When using the NOMONITOR method, SmartTest cannot determine when control is transferred from one load module to another. If you want to make SmartTest debugging facilities available in the CALLED module, it is necessary to define to SmartTest the names of the FETCHed load modules. This specification allows SmartTest to intercept a CALL to those modules.

To test dynamically CALLED load modules, use the QUALIFY command to activate the CALLED module. Then, you can either set breakpoints in the module, or set the Break on Entry column to YES on the Test Session Tailoring screen.

## Controlling Program Execution

SmartTest gives you the opportunity to step through a program statement by statement or execute the program to a specified breakpoint where you can verify the program's behavior. You may interrupt or end a test session at any time without waiting for end-of-job.

Program execution is controlled using actions on the Test pull-down, primary commands, or line commands. [Figure 77](#) shows the Test pull-down.

Figure 77 • Test Pull-down Menu

```

File View Test Search List Options Help
-----
Command ===
  1. Setup Wizards...
  2. Run...
  3. Step...
  4. Cancel
  5. Break...
  6. Stop...
  7. Go...
  8. Move...
  9. Add...
 10. Subtract...
 11. Where...
 12. Testpoints...
 13. ASG-Alliance Interface...
 14. CICS Newcopy...
 15. Dump

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

Cop                                     unpublished work.
A proprietary product of ASG, Inc. Use restricted to authorized licensees.
Visit the ASG support Web Site at www.asg.com

```

### 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

To begin test execution or to resume execution after an interrupt occurs, follow this step:

- ▶ Select Test ▶ Run or type RUN. The Test - Run Request pop-up, shown in [Figure 78](#), displays.

Figure 78 • Test - Run Request Pop-up with BackTrack Off

```

Test - Run Request

Enter "Run To" line number and Monitor option,
as needed. Then press Enter.

Run to Line Number _-----

Select Monitor option
-- 1. Monitor
   2. No Monitor

```

An interrupt can result from any of these conditions:

- Abend error condition
- Breakpoint or Stop Address
- Completion of program

You can use the RUN command any time program execution is suspended. This command is usually assigned to a PF key for convenience.

These are examples of the RUN command:

Command	Description
RUN MON	Allows all execution features to be available.
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.

See the *ASG-SmartTest Reference Guide* for more information about using the RUN command.

## Executing a Specified Number of Statements Using STEP

To execute a specific number of statements any time program execution is suspended, follow this step:

- ▶ Select Test ▶ Step or type STEP. The Test - Step Request pop-up, shown in [Figure 79](#), displays.

Figure 79 • Test - Step Request Pop-up

```
Test - Step Request

Select Step type, Step Options, or Monitor Option, as needed.
Then press Enter.

Step ...
-- 1. to next STATEMENT (Default)
   2. to next PARAGRAPH/LABEL
   3. OVER (COBOL PERFORM, Assembler Linkage, or
      PL/I CALL)

Step Options:
Step Count _____ Enter 1 to 65535 (to 999 for
Auto)
Step Auto _ Enter non-blank to select

Monitor Option: (If OVER specified)
-- 1. Monitor
   2. No Monitor
```

An interrupt can result from any of these conditions:

- Abend error condition
- Breakpoint or Stop Address
- Completion of program

When there are no abends or breakpoints, the program executes the specified number of statements or runs until the program completes.

You may use the STEP command any time program execution is suspended. This command is usually assigned to a PF key for convenience.

These are examples of the STEP command:

Command	Description
STEP	Executes the next statement and stops.
STEP 5	Executes the next 5 statements before stopping.
STEP AUTO	Executes one statement and pauses for the time period specified using the SET DELAY command.

See the *ASG-SmartTest Reference Guide* for more information about using the STEP command.

## Changing Program Execution Sequence Using GO

SmartTest-PLI provides the capability of changing the program execution sequence. This may be useful to bypass certain code or to test only certain code.

Select Test ► Go, the GO line command, or the GO primary command to change the next statement to be executed. The RUN or STEP command causes execution to resume at the selected statement.

The Test - Go Request pop-up is used to transfer control to the statement containing the specified PL/I label name, pseudo code label, or line.

### Note:

You must be in an active test session to use the GO command. You can only enter a GO command for executable source lines. It is recommended you use the GO command only to transfer control to a statement within the PROCEDURE or BEGIN block currently being executed.

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.

## **Interrupting Test Execution Using Keystroke**

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

In TSO foreground, 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 PL/I statement. When you press PA1/ATTN after a STEP *n* command is issued, the remaining steps are not executed and the program is suspended at the next PL/I statement.

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

The interrupt keys do not apply to CICS when executing with NOMONITOR.  
\_\_\_\_\_

In a Batch Connect test session, when you press PA1/ATTN when a program is running, the connection to the batch region is suspended. The batch version of SmartTest continues to run until an abend or break, or the completion of the test. When the batch region is ready for command input, SmartTest sets the job state to Waiting for Connection. The SmartTest job waits for the tester to reconnect to the job and resume testing, or times out if waiting too long.

## **Inserting Breakpoints to Interrupt Execution**

SmartTest provides the capability of inserting breakpoints into a program at desired locations. Program execution is suspended each time a breakpoint is encountered during test execution. While the program is suspended, you can view data values, insert pseudo code, trace execution paths, or perform other SmartTest tasks.

SmartTest provides three methods of inserting breakpoints into your program:

- Manually, at specific lines
- Automatically, based on program knowledge
- Automatically, with an impact dataset

After inserting breakpoints, they remain active until deactivated or deleted. You may save breakpoints in the AKR for any program when you conclude a test session. In that way, the breakpoints are in the program if the test needs to be resumed later.

### **Manually Inserting Breakpoints in the Program**

Use the BR line command to manually insert a breakpoint before the source line where the command is entered. When you type BR on a non-executable or continuation line, the BREAK is inserted before the next executable statement in the program.

### *Automatically Inserting Breakpoints in the Program*

To automatically insert a breakpoint, follow this step:

- ▶ Select Test ▶ Break or type BREAK.

The Break facility uses the Intelligent Search Function to insert multiple breakpoints with a single command.

You can insert breakpoints before or after the specified target. For example, typing `BREAK 1000_INITIALIZE` inserts a breakpoint before the label `1000_INITIALIZE`.

For more information about the search function and available operands, see ["The Intelligent Search Function" on page 163](#) and ["Searching for PL/I Program Information" on page 167](#).

### *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

See 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.

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, as shown in [Figure 80](#).

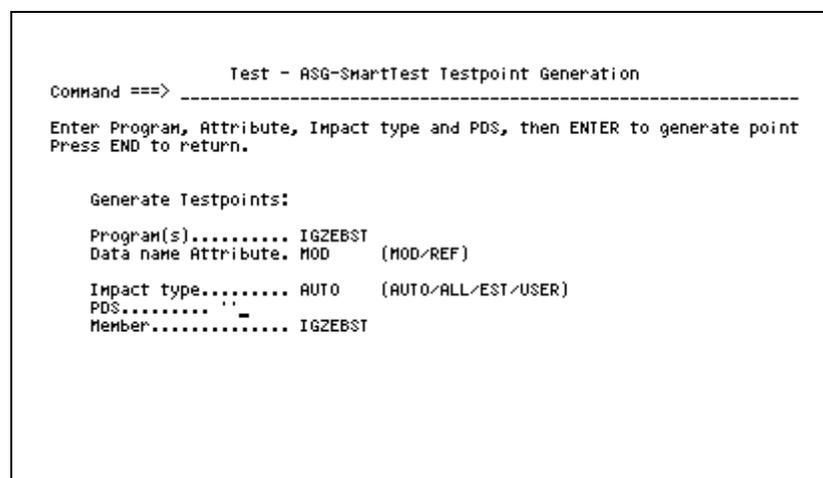
**Figure 80 • User-supplied Impact Dataset**

XXXX	TEST-DATE	00020000
XXXX	NUMBER1 OF TEST-NUM1	00030001
XXXX		00040000
XXXX		00050000

**To set breakpoints with impact datasets**

- 1 Type TESTPOINT on the command line, or choose Testpoints from the Test pull-down. The Test - ASG-SmartTest Testpoint Generation pop-up, shown in [Figure 81](#), displays.

**Figure 81 • Test - SmartTest Testpoint Generation Pop-up**



- 2 Complete the fields using the information this table:

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 apply the breakpoints.
Data Name Attribute	Specifies where to set breakpoints. One of two entries is allowed, either 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.

Field	Description
Impact Type	Indicates the source of the imported impact dataset. These are the allowable entries: <b>AUTO.</b> AutoChange <b>ALL.</b> Alliance <b>EST.</b> Estimate <b>USER.</b> User-supplied
PDS	Specifies the PDS containing the impact dataset to be imported.
Member	Lists the PDS member name containing the impact dataset for ALL, EST, or USR impact types.

- 3 Press Enter to display the Test - Testpoint Qualified Program List pop-up shown in [Figure 82](#).

**Figure 82 • 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..... 'ASG.VIACENXX.TESTPNT'
Program Specification: *

  Selected Program   Status
  -----
- TESTPLIA         READY
- TESTPLIB         READY
- TESTPLIC         READY
***** BOTTOM OF DATA *****

```

- 4 Type an S next to the names of those programs you wish to select and press Enter. The Breakpoints List screen, shown in [Figure 83](#), displays for the first program you selected.

**Figure 83 • Breakpoints List Screen**

```

Command ==> Breakpoints List TESTPLI.TESTPLI1
Scroll ==> CSR

Set pseudo . . ON Pseudo active: 13 Inactive: 0
Set breaks . . ON Breaks active: 13 Inactive: 0
Set whens . . . ON Whens active: 0 Inactive: 0

S Line Pseudo Code (breakpoints highlighted) A Count
-----
- 000046 *
- ''''1 BREAK Y 000000
- 000047 TESTNUM1=1;
-----
- 000048
- ''''1 BREAK Y 000000
- 000049 TESTNUM2=2;
-----
- 000050
- ''''1 BREAK Y 000000
- 000051 TESTDATE=1;
-----
- 000051 TESTDATE=1;
- ''''1 BREAK Y 000000
- 000052 TESTDATE=1;

```

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.
- 5 Exit the Breakpoints List screen. The Program View screen, shown in [Figure 84](#), displays breakpoints and the source code or disassembled object code and data.

**Figure 84 • Program View Screen**

```

File View Test Search List Options Help
-----
Command ==> Program View TESTPLI.TESTPLI1
Scroll ==> CSR

''''1 BREAK
000047 TESTNUM1=1;
000048
''''1 BREAK
000049 TESTNUM1=1;
000050
''''1 BREAK
000051 TESTDATE=1;
''''1 BREAK
000052 TESTDATE=1;
''''1 BREAK
000053 TESTDATE=1;
''''1 BREAK
000054 TESTDATE=2;
''''1 BREAK
000055 TESTDATE=2;
''''1 BREAK
000056 TESTDATE=2;
000057 *

```

- 6 Inspect the breakpoint settings in the Program View screen and exit the Program View screen. The Test - Testpoint Qualified Program List pop-up redisplay.

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 5](#) 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 performs these tasks:

- Finds all data items listed in the script
- Highlights the data items
- Issues a BREAK HI ALL command

Use the STFINDAT script to interrupt processing at key points so that you can examine or change date information. The STFINDAT script can be tailored to locate these items:

- Calls to user-written date routines
- Date-related comments
- Changes having a common pattern that are also date sensitive, such as in CALL routines and copybook areas

### ***Locating the Next Executable Statement***

When a test is suspended, you may want to scroll through the program examining other statements or data values. To reposition the screen to the next statement to be executed, type LOCATE \* in the primary command input area and press Enter.

## Displaying Breakpoints

To display breakpoints, follow this step:

- ▶ Select List ▶ Breakpoints or type `LIST BREAK`. The Breakpoints List screen displays listing all breakpoints for the program currently in Program View.

You can activate and deactivate individual breakpoints or all breakpoints in a program using the Breakpoints List screen.

## Removing Breakpoints Individually

Use the D line command on the Breakpoints List screen to delete individual breakpoints from the 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 the program. The Breakpoint included in blocks of pseudo code is only deleted if the `BREAK` statement is the first statement in the block. You can also use the `RESET PSEUDO` command to delete the breakpoints along with the rest of the pseudo code statements.

## Generating a Dump

To generate a dump, follow this step:

- ▶ Select Test ▶ Dump or type `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 19](#) for information about processing the log file.

## Canceling a Test Session

To cancel the current test session, follow this step:

- ▶ Select Test ▶ Cancel or type `CANCEL`.

The `CANCEL` command closes open test files and frees allocated storage. The program remains displayed in Program View. Use the `RUN` or `STEP` commands to restart the test if desired.

The short message reflects the cancellation of the test session, and the data window and status box are removed from the Program View screen.

### **Exiting a SmartTest-PLI 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. See "[SmartTest User Options](#)" on page 16 for more information on setting these parameters.

You can also exit SmartTest by continually pressing PF3 or by typing =x on the command line.

### **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 data in test files. You can view and change data by opening data windows within the program and typing over the values presented in the data window. You may open windows to view and change the data any time program execution is suspended.

Data windows may be displayed inline with the statement where the data is used, at any DECLARE statement, or at the top of the Program View screen. You may open windows on individual or group level data items. The values shown in data windows are always current and change as the program executes.

SmartTest determines the size of the data windows based on the amount of data to be displayed. You may override the default size for each data window on the Options - Modes screen. If a data window cannot contain all the data to be displayed, it becomes scrollable.

The message `UNABLE TO ACCESS DATA` in the value field indicates that storage for the data item has not been allocated.

### **Viewing Test Session Data Items Inline**

Use the ZD, ZH, ZG, ZGH, ZOOMDATA, K, and KEEP commands to perform these functions.

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 showing the data items referenced on the line where the command is entered. The data window is positioned after the statement where the command was entered. When the screen is scrolled, the data window moves with the display. If the data window cannot accommodate all the data items specified for it, you can scroll through the data window.

The data window contains the definition and 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 as shown in [Figure 85](#).

Figure 85 • Sample Inline Data Screen

```

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

000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. TESTCOBA.
000003 ENVIRONMENT DIVISION.
000004 CONFIGURATION SECTION.
000005 DATA DIVISION.
000006 WORKING-STORAGE SECTION.
000007 01 TESTER.
.....
..... 01 TESTER > PIC X(88) ADDR 00007990
..... VALUE > ..... <
..... +50 > ..... <
.....
000008 05 US-CHKP-COUNTER PIC 9999.
000009 05 US-CHKP-INTERVAL PIC 9(5). DEADDAT
000010 05 TEST3 PIC X(79).
000011 05 VAR-FIELD REDEFINES TEST3.
000012 10 NUMBER1 PIC S9(5) COMP-3.
000013 10 FILLER PIC X(76).
000014 SKIP2

```

When you enter the ZD or ZH line command on an array data item, the dimensions are shown to allow you to change the data item being displayed.

You may use the *n* (number) operand of the ZD command to display one data item in a statement containing multiple data items. Data items are assigned relative numbers from left to right on the line. The number entered represents the relative number of the data item to be displayed. For example, the line command ZD 2 opens a data window containing the second data item on the line.

When you enter a second 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 138](#)).

### Viewing Group Data Items Inline

Use the ZG line command to open a data window in display format showing the levels, value, and address of group data items and all of their subordinate data items.

Use the ZGH line command to open data windows in hexadecimal format showing 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. If the data window cannot accommodate all the data items specified for it, you can scroll through the data window.

When you enter the ZG or ZGH line command on a subscripted or indexed data item, the occurrence number is shown to allow you to change the entry being displayed (see [Figure 86](#)).

**Figure 86 • Relative Data Item in ZG**

```

File View Test Search List Options Help
-----
Command ==> _____ Program View          VIAPPLI.VIAPPLI1 -A
                                   Scroll ==> CSR
000240      05 SOC7_DATA,
*****
+-----+
***** | 02 SOC7_DATA          DATA(16)      ADDR 00142870
***** | 03 DATA_PACKED_DEC    DEC FIX (15,0)  ADDR 00142870
***** | VALUE > +123          <
***** | 03 DATA_DISPLAY_DEC   CHAR(8)         ADDR 00142878
***** | VALUE > +0000000 <
***** | 03 SOC4_SUB            CHAR(8)         ADDR 00142880
***** | VALUE > +0000000 <
***** | 03 SOC4_SUB_MAX        CHAR(8)         ADDR 00142888
***** | VALUE > +0000000 <
***** | 03 SOC4 CORE           CHAR(80)        ADDR 00142890
***** | DIMENSION ( 1 )
***** | VALUE > ..... <

```

As with ZD and ZH, you may use the *n* (number) operand to display one data item on a line containing multiple data items. Data items are assigned relative numbers from left to right on the line, and the number entered represents the relative number of the data item to be displayed.

The data window contains the definition and 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 a ZG or ZGH line command is entered for a data item with no subordinate data items.

### **Viewing Data Values Where Declared**

Use the ZOOMDATA, ZD, ZG, ZH, and ZHG primary commands to position the Program View screen at the definition of a specified data item and open an inline data window to display the data item. The ZD, ZG, ZH, and ZHG primary commands have the same meaning as the corresponding line commands.

The result of any of these primary commands is a data window positioned at the DCL. When the screen is scrolled, the data window moves with the display. If the data window cannot accommodate all the data items specified for it, the data window becomes scrollable.

The data window contains the definition and 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.

### **Removing Zoom Data Windows**

You may remove inline data windows from a program individually or globally.

#### **Removing Individual Zoom Data Windows**

Use the ZO or D line command to close a data window opened as a result of a ZOOM line or primary command.

#### **Removing All Zoom Data Windows**

Use the RESET ZOOM primary command 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**

To view data values at the top of the screen, follow this step:

- ▶ Select View ▶ Keep, type `KEEP`, or use the K line command to open a data window containing the value and address of data items in display format at the top of the screen.

Use the KH line or primary command to open a data window containing the value and address of data items in hexadecimal format at the top of the screen.

Use the KG line or primary command to open a data window containing the levels, value, and address of group data items in display format at the top of the screen. Use the KGH line or primary command to open a data window containing the levels, value, and address of group data items and all their subordinate data items in hexadecimal at the top of the screen.

The result of the Keep option; the Keep primary command; or K, KH, KG, or KGH line command 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. If the data window cannot accommodate all the data items specified for it, the data window becomes scrollable.

When you enter the K, KH, KG, or KGH command for an array data item, the dimensions are 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 on the line, and the number entered represents the relative number of the data item to be displayed. For example, the line command K 2 opens a data window containing the second data item on the line.

When you enter another keep command, the new data items are added at the end of the existing data window. When you enter another K, KH, KG, or KGH line command for the same statement, the information for those data items is added at the end of the data window.

The data window contains the definition and 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.

For example, the primary command KG COMPARISON\_KEYS displays the levels, values, and addresses of the data item COMPARISON\_KEYS and its subordinate data items.

When no longer needed, the data window may be removed or individual data items may be removed from the data window.

### ***Removing Individual Data Items From a Data Window***

Use the D line command to remove data items from the data window.

### ***Removing the Data Window***

Use View - Reset Request pop-up or the RESET KEEP primary command to close the data window.

## Changing Test Session Data Values

SmartTest allows you to change data values during a test session 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.

You may alter data values by typing over the value displayed in any data window. The new values take effect immediately. The value(s) shown in the data window is always current and changes as the program executes.

## Program Execution History (Backtrack)

SmartTest provides the Backtrack Facility to allow you to step backward and forward through the executed code and view data values as they existed when each statement was executed. You may view the execution history using the Backtrack Facility any time execution of the program is suspended.

The Backtrack Facility has two distinct modes—recording and reviewing.

### Recording Execution History

Enter the SET BACKTRACK primary command (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-PLI is entered.

The SET BACKTRACK primary command may also include the desired size of the Backtrack Recording memory buffer where data is collected (e.g., SET BACKTRACK 2M for a 2 MB buffer). This buffer is in memory with a default size of 1 MB (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.

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

You cannot run NOMONITOR when the Backtrack mode is ON.

---

### **Reviewing Execution 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 two methods:

- Enter the Backtrack Review mode using the Run or Step function. See "[Example Backtrack Review Session](#)" on page 141 for more information.
- Use the LIST BACKTRACK primary command (abbreviated LI BA) to display statements that modified a selected variable. See "[Using the BackTrack Variable History Function](#)" on page 147 for more information.

In Backtrack Review mode, the Program View status box shows the direction—forward or backward—of program execution history review. While in Backtrack Review mode, you can open data windows (Zoom or Keep) to view the contents of data fields at the time of execution of any statement.

### **Example Backtrack Review Session**

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

Before executing this example, type RESET PSEUDO to ensure all breakpoints and pseudo code statements are deleted.

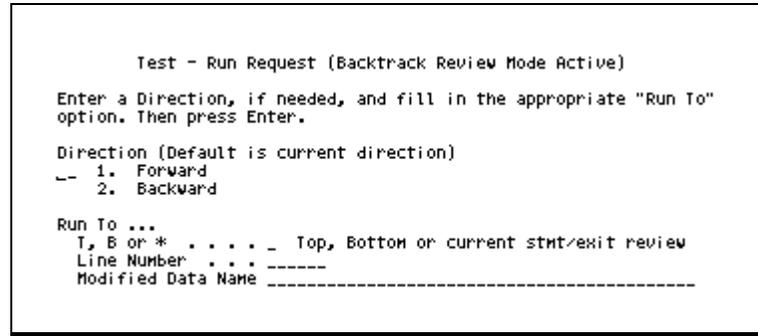
---

In this example from the VIAPPLI program provided with SmartTest-PLI, assume that the program being tested abends with a data exception (0C7) on the field DATA\_PACKED\_DEC. Backtrack Recording mode is ON. As the starting point of your investigation, you need to locate statements that have modified the value of DATA\_PACKED\_DEC.

**To locate the most recent statement that modified DATA\_PACKED\_DEC using Backtrack execution history**

- 1 Select Test ▶ Run and press Enter. The Test - Run Request (Backtrack Recording Mode Active) pop-up, shown in [Figure 87](#), displays.

**Figure 87 • Test - Run Request (Backtrack Recording Mode Active) Pop-up**



The Test - Run Request pop-up has changed because Backtrack Recording mode is active.

- 2 Enter BackTrack Review mode and initiate a search backward through BackTrack execution history for the last statement that modified DATA\_PACKED\_DEC using these steps:
  - a Select Backward in the Direction field.
  - b Type DATA\_PACKED\_DEC in the Run to Data Name field and press Enter. The BackTrack execution history on the Program View screen, shown in [Figure 88 on page 143](#), displays.

The Program View screen is positioned to the most recent statement in this test session that modified the field DATA\_PACKED\_DEC, indicated by the tag BKTR=>. This is also the statement that caused the 0C7 abend.

**Figure 88 • Reviewing Backtrack History of a Modified Data Item**

```

File View Test Search List Options Help
-----
Program View                                VIAPPLI.VIAPPLI1 -A
Command ==> _____ Scroll ==> CSR

(A) 481
BKTR=> DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
000483
000505 SOC7FIX1:
000506     RETURN;
000507
000508 END SOC7_DEMO;
000509
000510
000511 /* PAUSE HERE */
000512
000513 NOSOURCE_DEMO: PROC;
+-----+
+ (B) * REVIEWING BACKTRACK HISTORY * PROGRAM: VIAPPLI1 DIRECTION: BWD |
| STATUS: * REVIEWING BACKTRACK HISTORY * PROGRAM: VIAPPLI1 DIRECTION: BWD |
| STMT: 000482 OFF: 000C10 AMODE: 24 MODULE: VIAPPLI SEQ# -2682 |
| SOURCE: DATA_PACKED_DEC = DATA_PACKED_DEC + 1; |0|
+-----+

```

Screen Section	Description
(A) BKTR=>	Points to the most recent statement that modified the data field DATA_PACKED_DEC.
(B) * REVIEWING BACKTRACK HISTORY *	Indicates the test session is in Backtrack review mode.
(C) BWD	Indicates the direction of the Backtrack History review and -2682 indicates the program review is 2682 statements from the bottom of the Backtrack history.

- 3 Repeat [step 1](#) and [step 2](#) to find the next most recent statement that modified DATA\_PACKED\_DEC, a BASED variable.

As an option, you can type RUN BACK TO DATA\_PACKED\_DEC MOD in the primary command area.

- 4 To continue your investigation into the 0C7 abend, open a Zoom data window to display the value of the data field before it was modified by typing ZD over the tag BKTR=> in the line command area and pressing Enter. The Program View screen, shown in [Figure 89](#), displays.

**Figure 89 • Data Value Before Execution of the Statement**

```

File View Test Search List Options Help
-----
                                Program View          VIAPPLI.VIAPPLI1 -A
Command ==> _____ Scroll ==> CSR

000479
BKTR=>  SOC7_CHAR = ' ';
*****+-----+
*****| SOC7_CHAR          BASED CHAR(8)                |
*****| VALUE > ..... <                                |
*****| 03 DATA_PACKED_DEC      DEC FIX (15,0)         |
*****| VALUE > +0          <                            |
*****+-----+
000481
000482  DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
*****+-----+
*****| 03 DATA_PACKED_DEC      DEC FIX (15,0)         |
*****| VALUE > +0          <                            |
*****+-----+
+-----+
|STATUS: * REVIEWING BACKTRACK HISTORY *  PROGRAM: VIAPPLI1 DIRECTION: BWD  |
| STMT: 000480 OFF: 000C06 AMODE: 31      MODULE: VIAPPLI  SEQ# -2        |
|SOURCE: SOC7_CHAR = ' ';                                                    |0|
+-----+

```

[Figure 89](#) shows an opened Zoom data window displaying the value of the data field before it was modified by the assignment statement containing the BASED variable.

*To view the results of the execution of the assignment statement*

- 1 Select Test ▶ Step and press Enter. The Test - Step Request (Backtrack Review Mode) pop-up, shown in [Figure 90](#), displays.

**Figure 90 • Test - Step Request (Backtrack Review Mode Active) Pop-up**

```

          Test - Step Request (Backtrack Review Mode)

Select Step type, Direction, or Step Options, as needed.
Then press Enter.

Step ...
-- 1. to next STATEMENT (Default)
   2. to next Paragraph/Label

Direction (Default is current direction)
-- 1. Forward
   2. Backward

Step Options:
Step Count _____ Enter 1 to 65535 (999 if Auto)
Step Auto   _         Enter non-blank to select

```

- 2 On the Test - Step Request (Backtrack Review Mode) pop-up, select Forward in the Direction field and press Enter. The Program View screen, shown in [Figure 91](#), displays.

As an option, you can type STEP FORWARD in the primary command area.

**Figure 91 • Data Value after Execution of the Statement**

```

File View Test Search List Options Help
-----
                                Program View          VIAPPLI.VIAPPLI1 -A
Command ==>> _____ Scroll ==>> CSR

000479
000480  SOC7_CHAR = ' ';
'''''' +-----+
'''''' | SOC7_CHAR          BASED CHAR(8)          |
'''''' | VALUE >          <                          |
'''''' | 03 DATA_PACKED_DEC      DEC FIX (15,0)    |
'''''' | VALUE >          < * INVALID NUMERIC *    |
'''''' +-----+

000481
BKTR=>  DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
'''''' +-----+
'''''' | 03 DATA_PACKED_DEC      DEC FIX (15,0)    |
'''''' | VALUE >          < * INVALID NUMERIC *    |
'''''' +-----+

+-----+
|STATUS: * REVIEWING BACKTRACK HISTORY *  PROGRAM: VIAPPLI1 DIRECTION: FWD  |
| STMT: 000482 OFF: 000C10 AMODE: 31      MODULE: VIAPPLI  SEQ# -1        |
|SOURCE: DATA_PACKED_DEC = DATA_PACKED_DEC + 1;                          |0|
+-----+

```

[Figure 92](#) displays the value of the data field in the Zoom data window after the assignment statement containing the BASED variable was executed.

**To end the Backtrack review mode and resume program execution**

- 1 Select Test ▶ Run to display the Test - Run Request pop-up.
- 2 Select Forward in the Direction field and specify \* in the Run To field. As an option, you can enter the command RUN TO \*.

**Figure 92 • Resuming Program Execution**

```

File View Test Search List Options Help
-----
                                Program View
                                VIAPPLI.VIAPPLI1
Command ==> _____ Scroll ==> CSR
ASG2429W BOTTOM OF BACKTRACK HISTORY. BACKTRACK REVIEW EXITED. (A)
001228
001229 FIND_NEXT_PARAMETER: PROC;
001230
001231     SUB1 = SUB1 + 1;
001232     IF SUB1 > VALID_PARMS_COUNT THEN
001233         DO;
001234             IF VALID_PARM_TEXT (SUB1) = ' ' THEN
001235                 DO;
<<<<<<
001237                 CURRENT_PARM_NUMBER = VALID_PARM_NUMB (SUB1);
001238                 DAT1 = ' DEMONSTRATION OF: ' ;
001239                 DAT2 = VALID_PARM_TEXT (SUB1);
001240                 DISPLAY AREA LENGTH = 80;
001241                 PUT SKIP EDIT(DAT1,DAT2) (A);
001242                 SUB1 = VALID_PARMS_COUNT;
001243             END;
001244         END;
001245     END FIND_NEXT_PARAMETER;
-----+
|STATUS: BREAK ON ENTRY TO A PROGRAM   PROGRAM: IEFBR14  BACKTRACK:
| STMT: N/A  OFF: 000000 AMODE: 24      MODULE: IEFBR14  (B) RECORDING
|SOURCE: SR  R15,R15                    HEX: 1BFF
|-----+

```

Screen Section	Description
(A) ASG2429 BOTTOM OF BACKTRACK	The long message indicates Backtrack history review mode has ended.
(B) RECORDING	Indicates the Backtrack facility is in recording mode.
(C) Program View	Positioned at the last statement executed before Backtrack history was reviewed.

- To continue your test session, use the RUN or STEP command. A Program View screen similar to the one shown in [Figure 93](#) displays.

**Figure 93 • Recording Backtrack History to End of Test Session**

```

File View Test Search List Options Help
-----
Command ==> _____ Program View TEST ENDED, RC=0
                                Scroll ==> CSR

000344
>>>>> RETURN;      /* GOBACK TO OPERATING SYSTEM */
000346
000364 PROGRAM_INIT: PROC;
000365
000366 /* DEFAULT TO EXECUTION FAILURE */
000367 /* BUG */
000368 /* CALL GET_TIME_OF_DAY; */
000369 /* MOVE TIME_OF_DAY_WORK TO TIME_OF_DAY_START; */
000370
000371 RETURN_CODE = -1;
000372
000373 /* INITIALIZE WORKING STORAGE VALUES */
000374
000375 DATA_PACKED_DEC = 0;
000376 DATA_DISPLAY_DEC = 0;
-----+
|STATUS: BREAK AT END OF TEST SESSION  PROGRAM: VIAPPLI1 BACKTRACK:  |
| STMT: 000345 OFF: 000386 AMODE: 31   MODULE: VIAPPLI  RECORDING  |
|SOURCE: RETURN;      /* GOBACK TO OPERATING SYSTEM */           |
+-----

```

The program executes to completion and is suspended at the end of the test session. At this point in the program, you can either review the Backtrack history or end the test session.

- End the test session and discard the Backtrack history by typing RUN in the command input area and pressing Enter.

## Using the BackTrack Variable History Function

*To review BackTrack execution history using the LIST BACKTRACK primary command*

- Type LIST BACKTRACK to display the List - BackTrack Variable History pop-up shown in [Figure 94](#).

**Figure 94 • 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 _____

```

- 2 Enter a dataname to search the BackTrack execution history for statements that modified the specified data item. You can enter a pattern to select from a list of datanames.
- 3 Press Enter to display the List - BackTrack Variable History pop-up, shown in [Figure 95](#), displays showing the statements that modified the selected data item.

Figure 95 • List - Backtrack Variable History Screen

```

List - BackTrack Variable History      VIASUB37.VIASUB37 -A
Command ==> _____ Scroll ==> CSR
Variable: COMPARISON-CODE             Desc: PIC 9
Index: _____

$ RELMOD VALUE          LINE  STATEMENT CONTENTS
-----
***** TOP OF DATA *****
VIAMRG37.VIAMRG37
-3 .                    000187  MOVE ZEROS TO  COMPARISON-KEY-1, COM
- VIASUB37.VIASUB37
-2 0                    000038  MOVE 1 TO COMPARISON-CODE
- -1 1                  000038  MOVE 1 TO COMPARISON-CODE
-  0 1                  >>>>>> <CURRENT VALUE OF VARIABLE>
***** BOTTOM OF DATA *****

+-----+
| STATUS: STOPPED AFTER BREAK          PROGRAM: VIASUB37  BACKTRACK: |
| STMT: 000023  OFF: 000204  AMODE: 31  MODULE: VIASUB37  RECORDING |
| SOURCE: IF INFILEA-COMPARISON-KEY NOT EQUAL EOF-KEY                |
+-----+

```

The List - BackTrack Variable History screen lists the statements that modified the selected data item. You may view a statement in the program source by selecting the statement and pressing Enter.

You can review other statements in BackTrack execution history by typing LIST BACKTRACK and repeating [step 1 on page 147](#) through [step 3](#).

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 entering the RUN TO \* primary command on the Program View screen.

## Using Pseudo Code

SmartTest provides pseudo code to allow you to interactively modify a program during a test session to test proposed changes without recompiling the program.

### *Pseudo Code Concepts*

SmartTest-PLI provides COBOL-compatible statements called pseudo code. You can enter these statements inline with existing PL/I source code. Inserted code can change the execution sequence of the program or temporarily fix a data problem.

Enter pseudo code statements directly into the source code of a program displayed in Program View. Each pseudo code statement is associated with the executable source statement following it. Multiple pseudo code statements associated with an executable source statement form a pseudo code block. Pseudo code statements follow standard COBOL syntax rules.

The Pseudo Code List screen displays all pseudo code for the program currently in Program View. Pseudo code remains active until deactivated or deleted (see [“Viewing Pseudo Code in a Test Session” on page 153](#) and [“Removing Pseudo Code from a Program” on page 153](#)).

You can save pseudo code with any program stored on the AKR.

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

For setup options on automatically saving pseudo code, see ["SmartTest User Options" on page 16](#).

---

## Pseudo Code Statements Available

These are the available pseudo code statements and operands:

Statement	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 PL/I procedure 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.
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	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 instruction that modifies data.  <b>Note:</b> _____ Processing WHEN statements is resource intensive. Use sparingly. _____
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 label name. Pseudo code labels cannot be defined in WHEN statements.

Statement	Description
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 anywhere within the code. Pseudo code data items must be defined before being referenced and cannot be defined in WHEN commands.
&COUNT	Indicates 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 and incremented before the corresponding verb is executed. &COUNT can be referenced as required in any pseudo code statement, but cannot be modified (e.g., MOVE 2 TO &COUNT is invalid).

### 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. SmartTest supports these COBOL reserved words in pseudo code:

ALL	ELSE	PIC
ALPHABETIC	EQUAL	PICTURE
COMP	FROM	POSITIVE
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

These symbols can be used in conditional pseudo code statements:

=	equals
<	less than
>	greater than

## Using PL/I Datanames in Pseudo Code

These statements apply when using PL/I datanames in pseudo code:

- New datanames must be defined using a COBOL 77 statement using COBOL syntax.
- PL/I datanames defined in the program can be used in pseudo code statements.
- Non-unique elements of a PL/I structure can be coded using COBOL or PL/I syntax. For example:

```
DCL 1 REC_1,  
    05 A CHAR (1) INIT ('A' );  
DCL 1 REC_2,  
    05 A CHAR (1) INIT (' ');  
  
MOVE A OF REC_1 TO A OF REC_2.
```

**Or**

```
MOVE REC_1.A to REC_2.A.
```

## Using Constants in Pseudo Code

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.
```

## Entering and Editing Pseudo Code

You can enter pseudo code statements with existing PL/I source code on the Program View screen. During a test session, pseudo code statements can be entered whenever program execution is suspended. Pseudo code statements are written in standard COBOL format.

The C (copy), D (delete), I (insert), M (move), and R (repeat) line commands are available for entering and editing pseudo code statements in a program. These commands operate in the same manner as in TSO/ISPF, including block format.

## **Executing Pseudo Code in a Test Session**

Pseudo code is executed as if it is part of the program logic when a RUN or STEP command is issued.

Each pseudo code statement is associated with the source statement following it and is executed before that statement. For example, at a breakpoint, execution is suspended and the chevrons are positioned in the line number area of the next logic statement to be executed. If a block of pseudo code is inserted immediately prior to this statement, the RUN or STEP action starts executing with the first pseudo code statement in the inserted block.

SmartTest checks syntax of pseudo code statements when a command is entered referencing a pseudo code statement or variable and when saving the pseudo code to the AKR. The SmartTest-PLI commands that reference pseudo code are RUN, STEP, ZOOM commands, LIST BREAKS, LIST PSEUDO, LIST WHENs, LPRINT PSEUDO, and QUALIFY.

Pseudo code statements with correct syntax are executed as part of the program logic. When a pseudo code statement causing a program execution error is encountered, the current test is interrupted and an error message displays. Correct the pseudo code and resume the test session.

## **Viewing Pseudo Code in a Test Session**

To display the Pseudo Code List screen showing all pseudo code for the program currently in Program View, follow this step:

- ▶ Select List ▶ Pseudo code or type `LIST PSEUDO`.

You can activate and deactivate pseudo code statements individually or globally using the Pseudo Code List screen.

## **Removing Pseudo Code from a Program**

You can remove pseudo code from a program individually or globally.

### **Removing Individual Lines of Pseudo Code**

Use the D line command to delete individual lines of pseudo code from a program. This command operates in the same manner as the D (delete) line command in TSO/ISPF, including block format.

### **Removing All Pseudo Code**

To delete all pseudo code (including breakpoints) from a program, follow this step:

- ▶ Select View ▶ Reset or type `RESET PSEUDO`.

## Making Proposed Changes Permanent

Pseudo code provides a means of testing proposed changes to the program without recompiling. Permanent changes to the source code must be completed using the source code editor at your site. SmartTest does allow split screen sessions under ISPF so you can view pseudo code statements at the same time you are editing.

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

If you edit your program, you need to reanalyze the updated program so SmartTest-PLI includes the changes in the next test session.

---

## Using Multiple Programs

SmartTest-PLI provides the ability to view programs other than the one being tested without cancelling the active test. This process is called qualification.

To display a program (other than the program being tested) in Program View, follow this step:

- ▶ Select File ▶ Open or type QUALIFY.

Programs residing in the AKR are displayed in source code format. Programs not stored on the AKR are displayed in disassembled object code. Several programs may be qualified for viewing simultaneously.

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 CANCEL operand is used to remove the program being viewed from qualification for viewing.

These are some examples of the QUALIFY primary command:

QUALIFY <i>loadmod.program</i>	Brings a program from a different load module into Program View.
QUALIFY CANCEL ALL	Removes all programs from qualification and exits Program View.

## Tailoring a Test Session by Program

The Test Session Tailoring screen 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 List ▶ Test session tailoring

Session tailoring can reduce the execution time for a test by specifying options at the individual program level. For example, you can set Break on Entry as NO for programs that have already been verified, to avoid stopping at the entry point.

An example input line is provided on the first data line displayed, to indicate how to specify information for a program. The I (insert), R (repeat), and D (delete) line commands can be used when editing the list of programs. Block forms of these commands are not supported.

For details of the Test Session Tailoring screen shown in [Figure 96](#), see the *ASG-SmartTest Reference Guide*.

Figure 96 • Example Test Session Tailoring Screen

```

Command ==> _____ Test Session Tailoring _____ VIAPPLI.VIAPPLI1 -A
                      _____ Scroll ==> CSR

      Module.Program id  Monitor Track Count Break Break Break Pseudo Single
                      Act   Act   Act   Act   Entry Rtn   Act   Step
                      ---   ---   ---   ---   ---   ---   ---   ---
*** ***** TOP OF DATA *****
.... VIAPPLI.VIAPPLI1   YES   YES   YES   YES   YES   NO   YES   YES
.... VIAPPL2.VIAPPL21  YES   YES   NO   YES   YES   NO   YES   YES
*** ***** BOTTOM OF DATA *****

```

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

For PL/I programs, the program name specified on the Test Session Tailoring screen must be the PL/I compiler generated name.

\_\_\_\_\_

## Capturing and Replaying Test Sessions

SmartTest provides a Script facility for the capture and replay of primary command sequences. The captured commands are saved in a file that may be replayed to execute the recorded primary commands multiple times. Script files can be used to initialize a session, set default data values, insert breakpoints, open data windows, or perform other tasks that can be accomplished using primary commands.

SmartTest assigns a dataname for the script dataset and displays it in the long message area when the SET SCRIPT ON primary command is issued. The Script facility allocates and opens a dataset to record all SmartTest primary commands as they are entered. To stop recording the primary commands and close the script file, issue the SET SCRIPT OFF primary command. Record the generated dataset name for specification when the script file playback is requested.

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. The script file name is in this format:

`USERID.STTnnnnn.VIASCRIP`

where *nnnnn* is a number beginning with 00001 that is incremented by 1 for each new script file. Once the dataset is closed, it can be used 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.

\_\_\_\_\_

You can include comments in script files by typing an asterisk (\*) in the first column of the command by the comment text.

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

Actions executed using CUA and line commands are not recorded in the script file.

\_\_\_\_\_

For information on the Script related primary commands, see the *ASG-SmartTest Reference Guide*.

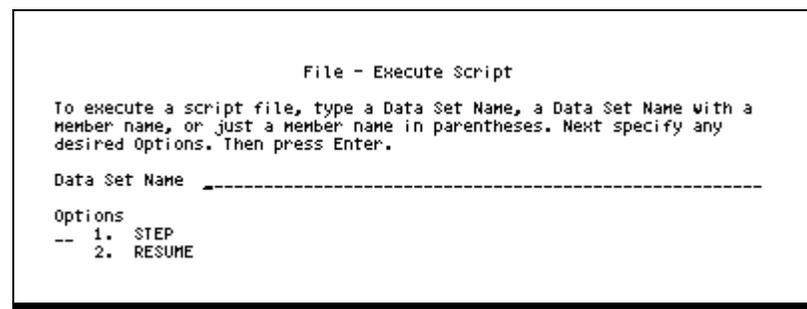
## **Replaying a Script File**

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 nested script files. Script files creating loops are recognized and an error message displays.

**To execute a script file**

- 1 Select File ► Execute and press Enter. The File - Execute Script pop-up, shown in [Figure 97](#), displays.

**Figure 97 • File - Execute Script Pop-up**

- 2 Enter the name of the script file in the Data Set Name field and press Enter.

**Note:**

Select the STEP option to invoke the debug option that allows stepping through each primary command in the script file. Each primary command in the script file displays in the primary command input area. The displayed primary command can be changed or erased from the primary command input area. Press Enter to execute the displayed primary command. Processing of the script file continues in this manner until all primary commands have been displayed and/or executed.

While stepping through the script file, select the RESUME option to execute the remaining script file primary commands without stepping through each primary command.

For more information on the Execute facility or the EXECUTE primary command, see the *ASG-SmartTest Reference Guide*.

## Position the Display at the Next Executable Statement

Use the LOCATE \* 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.

This command may only be used from the Program View screen during an active test session.

This command can be abbreviated as L \*.

## **Saving Keystrokes**

SmartTest provides several ways to reduce the number of keystrokes needed to enter commands.

### **Token/Cursor Substitution**

When entering commands, use the Cursor Substitution character while in the Program View screen to bypass typing datanames and label names as command targets.

Search commands allow you to type the Cursor Substitution character (default %) in place of a dataname or label name. After entering the command with the Cursor Substitution character, place the cursor on the name in the source to be used as the target. When the command is processed, the substitution character is replaced with the name from the cursor location.

For example, the primary command `FX %` finds the statements that contain the text located at the current cursor position. Place the cursor on the item before pressing Enter.

You may substitute multiple tokens (e.g., qualified datanames such as `MYDATA OR 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.

### **Substituting Short Names for Character Strings**

Use the `EQUATE` primary command to define a substitute name for any character string.

Equated names are used in the same manner any dataname is used. They can be saved in the `AKR` and may be viewed on the Equates List by issuing the `LIST EQUATES` primary command.

To delete an equate, type `EQUATE` using the equate name without any other operands. This command may be abbreviated as `EQ`.

For example, the primary command `EQUATE FILES HOW_MANY_FILES_READ` assigns a substitution short name (`FILES`) to a long field name (`HOW_MANY_FILES_READ`). The primary command `FX FILES` searches for occurrences of `HOW_MANY_FILES_READ`.

### **Retain Commands in the Command Input Area**

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, the primary command & FX END\_FILE\_COUNT executes a FINDXTND command targeting field references and leaves the command displayed in the command input area. You can type over the dataname field name to execute the FINDXTND command for another field.

### **Recall Commands, Messages, and Pop-ups**

Use the RECALL primary command to display previous primary commands, long messages, or pop-ups. The PF12/PF24 keys are assigned by default for the RECALL command.

The last 20 primary commands and the last 20 long messages are stacked in a recall buffer and may be displayed using the RECALL primary command. When the command or pop-up displays, it may be executed again with or without modification.

Use the MESSAGE operand to display previous messages. The NEXT operand can be used to move forward rather than backward through the stack.

Once the RECALL command has been entered with operands, subsequent RECALL commands with no operand use the last operand entered until a different operand is specified or another command is executed.

When the RECALL function has processed the entire command or message stack, further RECALL commands process the stack again.

For example, the primary command RECALL MSG displays the last message in the long message area. The primary command RECALL POPUP displays the last pop-up displayed.

These commands are not stacked for use by the RECALL command:

ANALYZE	RIGHT
RECALL	PLOG
PLIST	KEYS
UPDATE	RETURN
REDO	REPEAT
PRODLVL	DOWN
RSCROLL	RFIND
RHIGH	RESET
LEFT	END
HELP	UP

### **View Release and Level Numbers**

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.

### **Linking to Alliance**

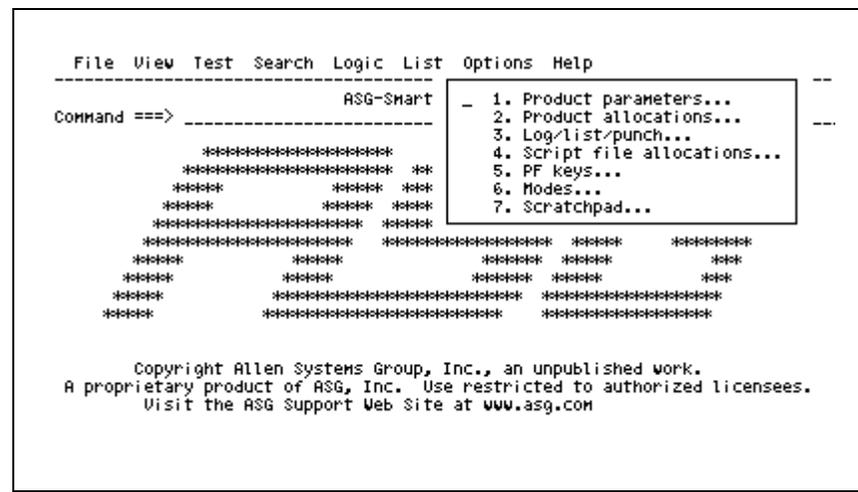
If Alliance is installed, SmartTest provides a dynamic link to it that can be used to display program analysis information. SmartTest automatically runs script queries appropriate for the current execution environment upon entry to Alliance.

Prior to establishing the link, a complete Alliance application analyze must be done for the program you wish to process in Alliance. The Alliance application analyze must include the load module libraries and execution JCL. Also, you must copy the VIAPALSC and VIAPQ\* members from the ESW CNTL library into a user-defined PDS and appropriately customize them for your needs.

**To link with Alliance**

- 1 Select Script file allocations from the Options menu on the SmartTest primary screen, as shown in [Figure 98](#).

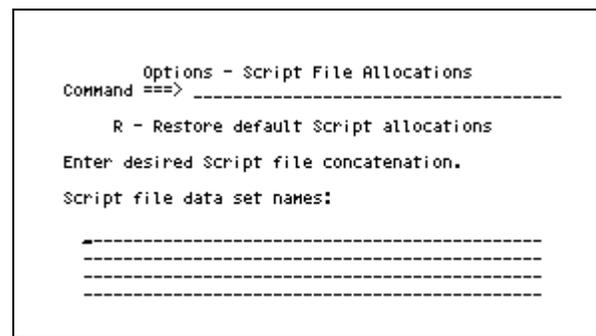
**Figure 98 • Selecting the Script File Allocation Menu Item**



The Options - Script File Allocations pop-up displays.

- 2 On 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, shown in [Figure 99](#), displays.

**Figure 99 • Options - Script File Allocations Pop-up**



- Exit to the SmartTest primary screen. On the command line, issue the ALLIANCE command. The ASG-SmartTest/ASG-Alliance Interface pop-up, shown in [Figure 100](#), displays.

**Figure 100 • SmartTest/Alliance Interface Pop-up**

```

ASG-SmartTest/ASG-Alliance Interface
Command ==> -----
Please enter the following :

Alliance AKR.... . . ASG-SmartTest/ASG-Alliance Interface.
Application Name . . -----
Program Name ..... -----
Query Name..... VIAPQIMS
    
```

- Complete these 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 the name of the query script that SmartTest selected from the PDS specified in the Options Script File Allocations pop-up. This field automatically contains with the name of a query script appropriate for the current environment; however, you can specify another.

- Press Enter, verify the information, and press Enter again. The VIAPALSC script executes and runs the query listed in the SmartTest/Alliance Interface screen. A series of screens flashes by, ending with an Alliance query display appropriate for the load module, JCL, transactions, and current SmartTest execution environment.
- Type END on the command line to exit the query display. The Alliance primary screen displays. Press PF3/PF15 to exit the Alliance primary screen and redisplay the SmartTest primary screen.

---

# 5

## Analysis Features

---

This chapter discusses analysis features of SmartTest-PLI and contains these sections:

Topic	Page
<a href="#">The Intelligent Search Function</a>	<a href="#">163</a>
<a href="#">Excluding Lines from the Display</a>	<a href="#">166</a>
<a href="#">Searching for PL/I Program Information</a>	<a href="#">167</a>
<a href="#">Printing Program Information</a>	<a href="#">171</a>

### The Intelligent Search Function

The SmartTest-PLI Intelligent Search Function uses built-in understanding of the PL/I language. This function can be used to search the program for information about PL/I statements, data variables, labels, and other program details. The search function recognizes and bypasses COMMENT statements, unless specifically requested.

There are six primary commands in the Intelligent Search Function: BREAK, EXCLUDE, FINDXTND, HIGH, LPRINT, and SCROLL.

### Searching for PL/I Language Subsets

SmartTest-PLI classifies statements into subsets by grouping together PL/I statements of a similar nature. This table describes each of the PL/I subsets and corresponding entities:

Subset	Description
Assignment	Contains statements that assign a value to a data item, such as A = B.
BLOCKs	Contains PROCEDURE, BEGIN block, and END statements.

Subset	Description
CALL	Contains statements related to subprogram calls such as CALL and FETCH.
CONditional	Contains statements or parts of statements that conditionally change the flow of control in a program such as IF, THEN, ELSE, and SELECT/WHEN.
DEFinition	Contains declaratives of data variables.
ENDProcedure	Contains statements containing an END PROC name.
ENtry	Contains internal entry points (PROCs) and all ENTRY statements for external entry points.
EXEcutable	Contains statements that are executable.
EXIt	Contains statements containing a RETURN that returns control to the calling PROC.
GOto	Contains statements containing a GOTO statement or a SELECT/WHEN structure.
INPUT IO Output	Contains IO statements (IO, Input, or Output) including CALL statements that are indicated as containing IO, Input, or Output.
LABel	Contains statements containing labels.
ONBlock	Contains BEGIN block statements that are executed when an ON condition is raised.
ONConditions	Contains statements establishing ON conditions error handling.
PROcedure	Contains PROC statements.
STORage	Contains statements controlling program storage allocation.
STructure	Contains a group of subsets, which together help show the general structure of the program.
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.

Many SmartTest-PLI commands can include one or more of these subsets. You can concatenate subsets by placing a plus sign (+) between them. For example, to highlight the subset of lines containing IO and PROC statements, type `HIGH IO + PROC`.

### **Screen Subsets**

These screen subsets generally are the result of an interactive command:

- Highlighted or HI
- NONHighlighted or NHI
- Excluded or X
- NONExcluded or NX

### **Tag Subset**

SmartTest-PLI uses a tag subset to provide information about the source code. The tag subset displays in columns 73 through 80 on the Program View screen and shows results of the program analyze as well as command execution.

TAGged or TAGS refers to all lines having informative tags on them. The tag subset can be used in commands that accept subsets as targets.

When a subset is specified as the target of these commands, the entire program is searched for all occurrences of statements associated with the subset. All information pertaining to a function can be presented with a single command.

### **Determining Impact of Change**

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.

The search function can be limited through the use of command operands.

These operands limit the search function to include only these specified data item access types: MOD, USE, and DEF. The default is REF and includes all dataname reference types.

The search function may be limited by using the direction operands NEXT and PREV to start the search from the current position and locate the closest occurrence. The FIRST and LAST operands present their respective reference to the dataname specified. ALL is the search function default, with the exception of the BREAK command, where NEXT is the default.

The scope of the search function can be limited to particular statement types through the use of the IN clause.

## **Excluding Lines from the Display**

To display only relevant information on the screen, you can remove extraneous statements. You can use the search function to suppress the display of statements meeting specified criteria or all lines in Program View.

To invoke the search function to remove all lines meeting the criteria from the display, follow this step:

- ▶ Select Test ▶ Exclude or type EXCLUDE.

Use the Exclude action to remove specific lines or all lines from the screen. Lines can be excluded from the display before or after other commands are issued. Excluded lines are represented by shadow lines stating *n* LINE(S) NOT DISPLAYED and a line of dashes. You can suppress the dashed line using the SET SHADOW command. For example, typing EXCLUDE NHI or X NHI removes all non-highlighted lines from Program View.

## **Displaying Individual Excluded Lines**

Use the F (First), H (Hidden), L (Last), and S (Show) line commands to display excluded lines.

The F and S line commands are used to display a specified number of lines in a block of excluded lines, starting from the top of the block. The H or L line command is used to display the last lines of a block of excluded lines.

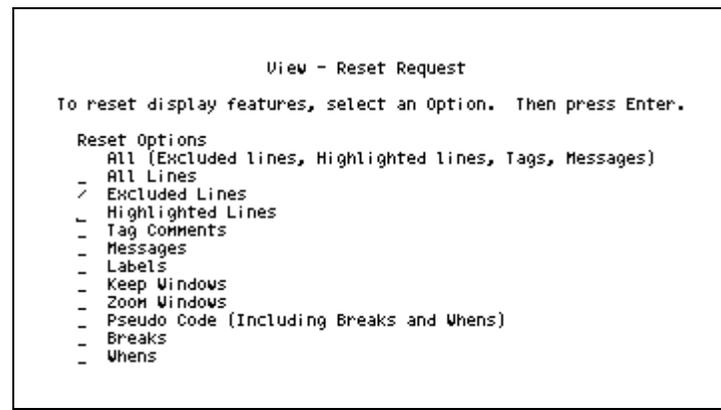
For example, the line command F2 displays the first two lines of an excluded block. The line command L5 displays the last five lines of an excluded block.

## **Displaying All Excluded Lines**

Select Excluded lines on the View - Reset Request pop-up or use the RESET EXCLUDED primary command to display all excluded lines without resetting any current highlighting and tags.

**To display all excluded lines**

- 1 Select View ► Reset and press Enter. The View - Reset Request pop-up, shown in [Figure 101](#), displays.

**Figure 101 • View - Reset Request Pop-up**

- 2 Type / on the Excluded lines field and press Enter.

## Searching for PL/I Program Information

The Intelligent Search function is used to locate statements that meet specified criteria. This function can be used to display and highlight specific program information, datanames, sets of lines, or patterns.

Use the Search pop-ups or the FINDXTND primary command to perform the search function. Statements that meet the criteria are displayed 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 a FINDXTND command results in all lines containing the specified target being highlighted with the cursor positioned on the first target. If lines containing targets are 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.

The FINDXTND primary command may be abbreviated as FINDX or FX.

Take care when examining PL/I storage, as usage is controlled by the programmer. SmartTest-PLI is unable to determine usage of a data item and can only display the contents based on the initial declaration. For example, a character string can be treated as a group of bit switches, but SmartTest-PLI attempts to treat the field as displayable characters. To display the hex equivalent, use the hex features within SmartTest.

Pointer variables and offset variables, which are types of locator data, contain an address of a storage location or an offset from an address. SmartTest displays both the actual address contained in the locator and the contents of the field at that address.

PL/I classifies variables as internal or external. The scope of the variable is determined by this classification; therefore, SmartTest-PLI indicates that a variable is not available if the test execution location is outside the scope of the variable.

### **Finding All IO Statements in a Program**

This example demonstrates how to display only those statements in the program containing Input and Output statements. The program must be displayed in Program View.

#### **To display only statements containing Input and Output statements**

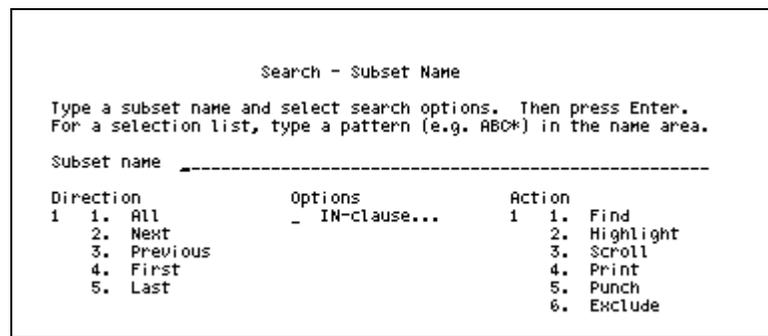
- 1 Omit all lines not in the search using the SET XMODE ON command.

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

XMODE is a toggle and remains on until set to OFF.  
\_\_\_\_\_

- 2 Select Search ▶ Subset and press Enter. The Search - Subset Name pop-up, shown in [Figure 102](#), displays.

**Figure 102 • Search - Subset Name Pop-up**



- 3 Complete these fields:
  - a Type IO in the Subset name field.
  - b Select All in the Direction field.
  - c Leave the Options field blank.
  - d Select Find in the Action field.
- 4 Press Enter to display the Program View screen shown in [Figure 103](#).

Figure 103 • Program View with Input and Output Search Results

```

File View Test Search List Options Help
-----
                          Program View                      13 STATEMENTS FOUND
Command ==> ----- Scroll ==> CSR

000476      PUT SKIP EDIT(' ') (A);
000477 /* BUG */
000478 /* PUT SKIP EDIT(DISPLAY_AREA) (A); */
000479      PUT SKIP EDIT(' ') (A);
000480      PUT SKIP EDIT(' ') (A);
000481      PUT SKIP EDIT(' ') (A);
000482      PUT SKIP
000483      EDIT('          VIA/SMARTTEST PL/I DEMONSTRATION COMPLETED') (A);
000484      PUT SKIP EDIT(' ') (A);
000485      PUT SKIP
000486      EDIT('          THANK YOU FOR YOUR PARTICIPATION') (A);
000487      PUT SKIP EDIT(' ') (A);
000488
000489      END SUMMARY_ROUTINE;
-----
| STATUS: * REVIEWING BACKTRACK HISTORY *   PROGRAM: VIAPPL1  DIRECTION: BWD
| STMT: 001498 OFF: 003172 AMODE: 31      MODULE: VIAPPLI   SEQ# -1
| SOURCE: BASED_VAR = '$0C4';
|-----|1

```

*To display the excluded lines*

- 1 Select View ► Reset and press Enter. The View - Reset Request pop-up, shown in [Figure 104](#), displays.

Figure 104 • 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

```

- 2 Type / in the Excluded Lines field and press Enter (see [Figure 105](#)).

**Figure 105 • Program View Screen after Displaying All Excluded Lines**

```

File View Test Search List Options Help
-----
                                Program View                                VIAPPLI.VIAPPLI1 -A
Command ==> _____ Scroll ==> CSR

>>>>> CALL PROGRAM_INIT;
000298
000314 CALL PARM_EDIT;                                /* INITIAL PARM_EDIT */
000315
000316 IF PARM_TEXT = 'EXIT' THEN
000317     RETURN;
000318 ELSE
000319     IF PARM_TEXT = 'INVALID' THEN
000320         DO;
000321             CALL INVALID_PARM;
000322             RETURN;
000323         END;
000324
000325 IF CURRENT_PARM_NUMBER = +0 THEN
000326     CALL INVALID_PARM;
000327
+-----+
|STATUS: BREAK AT START OF TEST SESSION   PROGRAM: VIAPPLI1  DATE: DDMMYYYY |
| STMT: 000297  OFF: 00029E  AMODE: 31     MODULE: VIAPPLI   TIME: HH:MM:SS |
|SOURCE: CALL PROGRAM_INIT;              |
+-----+

```

All lines that were excluded by the targeted search are displayed.

### Example of Finding Data Field References

This example demonstrates how to display, tag, and highlight all program references to a data field. The program must be in Program View. XMODE is assumed to be ON.

#### To find data field references

- 1 Select Search ▶ Data and press Enter. The Search - Data Name pop-up, shown in [Figure 106](#), displays.

**Figure 106 • 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 _____

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

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

```

- 2 Complete these fields:
  - a Type DATA\_PACKED\_DEC in the Data name field.
  - b Select All in the References field. This value is required.
  - c Select All in the Direction field.

**Note:**

The Indirect impact, Size change, Options, and Action fields are applicable for COBOL environments. Specifying values in these fields can result in an error.

- 3 Press Enter to display the Program View screen shown in [Figure 107](#).

**Figure 107 • Program View with Data Item Information Search Result**

```

File View Test Search List Options Help
-----
                          Program View                          VIAPPLI.VIAPPLI1 -A
Command ==> _____ Scroll ==> CSR
***** ***** TOP OF DATA *****
000277          10 DATA_PACKED_DEC      FIXED DEC (15) INIT(123),
          - - - - -                22 LINES NOT DISPLAYED
000300  DCL 1  $0C7_CHAR      CHAR(8)      BASED(ADDR(DATA_PACKED_DEC));
          - - - - -                126 LINES NOT DISPLAYED
000427          DATA_PACKED_DEC = 0;
          - - - - -                110 LINES NOT DISPLAYED
000538          DATA_PACKED_DEC = DATA_PACKED_DEC + 1;
          - - - - -                8 LINES NOT DISPLAYED
000547          *      1) OVERTYPE THE VALUE OF 'DATA_PACKED_DEC' IN THE ZOOMDATA*
          - - - - -                2 LINES NOT DISPLAYED
000550          *      'SET DATA_PACKED_DEC TO 0. ;RUN'
          - - - - -                948 LINES NOT DISPLAYED
+-----+
|STATUS: PROTECTION EXCEPTION (0C4)      PROGRAM: VIAPPLI1  DATE: 04OCT2000
|STMT: 001498  OFF: 003176  AMODE: 31    MODULE: VIAPPLI   TIME: 10:23:58
|SOURCE: BASED_VAR = 'S0C4';
|-----+

```

## Printing Program Information

To print program information, follow this step:

- ▶ Select Print on any appropriate Search pop-up or type LPRINT to invoke the search function to copy one or all occurrences of the specified target to the List File.

The List File can be printed at any time using the Option - Log/List/Punch Definition pop-up. For information on how to process the List File, see ["SmartTest User Options" on page 16](#).

A printed copy of program information can be obtained by using the search function to copy program source lines, meeting specified criteria, to the List File.

For example, the primary command LPRINT COND copies all conditional statements in the program to the List File for subsequent printing. The primary command LPRINT \* results in the entire virtual screen (all data viewed by scrolling forward and backward) being 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).

You can use the LPRINT \* command on these screens to copy them to the List file:

BackTrack Variable History Screen	List - Equates pop-up
Breakpoints List screen	List - Program/Subprogram Names pop-up
Execution Counts screen	List - User Marks pop-up
Execution Tracking screen	Pseudo Code List screen
List - CALL Statements pop-up	When Conditions List screen

## **Scrolling the Display**

The search function can be used to position the display to a statement meeting specified criteria.

Select Scroll on any appropriate Search pop-up or use the SCROLL command to invoke the search function to position the display to the first line containing the specified target. Highlighted lines remain unchanged.

The most common use of the SCROLL command is to move the display to the next occurrence of information already highlighted by a previous command.

For example, the primary command SC HI NEXT scrolls the display to the next occurrence of information already displayed by the FINDXTND command.

---

# 6

## Additional Languages

---

### SmartTest-PLI and Assembler

The SmartTest-ASM option is designed to alleviate the burden of testing and debugging 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-PLI. All testing and debugging functions are available with assembler including: Program View, execution control, breakpoints, monitoring and changing data, and abend processing. The only difference between using SmartTest-PLI and SmartTest-ASM is some limitation in certain analysis and debugging facilities.

SmartTest-ASM enables you to work with the source level display of your Assembler 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.

See the *ASG-SmartTest for COBOL and Assembler User's Guide* for more information about Assembler.

### SmartTest-PLI and COBOL

SmartTest for COBOL is a full-featured, source-level testing tool for COBOL programs.

For coverage of the testing and debugging features and additional information on SmartTest support for COBOL, see the *ASG-SmartTest for COBOL and Assembler User's Guide*.



---

# 7

## Help Facility

---

This chapter discusses the types of help available with SmartTest-PLI and contains these sections:

Topic	Page
<a href="#">Introduction</a>	<a href="#">175</a>
<a href="#">Screen Help</a>	<a href="#">177</a>
<a href="#">Command Help</a>	<a href="#">178</a>
<a href="#">General Information</a>	<a href="#">180</a>
<a href="#">Specific Information</a>	<a href="#">180</a>
<a href="#">Help Abends</a>	<a href="#">181</a>
<a href="#">Help Messages</a>	<a href="#">182</a>
<a href="#">Printing Messages</a>	<a href="#">184</a>

### 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.

You can access the SmartTest online help facility using any of the methods described in this table:

Help Topic	How to 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/PF13. No message 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/PF13. The <code>HELP COMMANDS</code> command displays a long message that lists most of the primary commands. Once this message displays, pressing PF1/PF13 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, selecting option 7 on the Help Table of Contents, or typing <code>INDEX</code> from within the Help Tutorial. Help for a specific topic can be viewed by selecting the appropriate index entry.
System, ASG, and IMS Abends	Help for abends is requested by selecting Help ► Common abends, or by typing <code>HELP ABENDS</code> . The Abends help screen displays, which lists and describes the abends. Select one of these options, depending on the type of abend: <ol style="list-style-type: none"> <li>1 displays the System Abends screen.</li> <li>2 displays the ASG Abend Codes screen.</li> <li>3, 4, 5 displays an IMS Abends screen.</li> </ol>
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/PF13. The corresponding long message displays at 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 the previous table are contained in the Help Tutorial. Each online help topic can be reached from anywhere within the Help Tutorial by going through the Help Table of Contents or Index. Once the Help Tutorial is accessed, there are several commands available for navigating within the Help Tutorial. These commands are listed in this table.

Help Command	Purpose
BACK	Redisplay the previous Help Tutorial screen.
END	Exits the Help Tutorial.
Enter key	Displays the next screen in a continuation series.
INDEX	Displays the first screen of the Help Index.
SKIP	Goes directly to the next subject.
TOC	Displays the Help Table of Contents.
UP	Displays the next higher-level subject.
Alpha character	Displays the Index screen corresponding to that character (valid only for index screens).

## Screen Help

Help for the current screen is requested by selecting Help ► Current screen. Help for the current screen or pop-up displays by typing `HELP`, or by pressing PF1/PF13. If any messages appear, help for the screen or pop-up displays by typing `HELP SCREEN`. The Help Tutorial for the current screen displays.

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 108](#) shows the Help Tutorial for the Options - Product Allocations pop-up.

**Figure 108 • 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

Help for a specific command is requested by selecting Help ► Specific command, typing the command in the command input area and pressing PF1/PF13, or typing HELP followed by the desired command name. A long message displays describing the command. Pressing PF1/PF13 again displays the Help Tutorial screen for that command. The Help Tutorial for each command displays the command syntax diagram, and gives a description of each operand in the command. Typing UP on a command help screen displays a complete list of all SmartTest commands.

For help on all SmartTest commands, select Help ► All commands or type HELP COMMANDS. The All commands action displays a complete list of all SmartTest commands. The HELP COMMANDS command displays a long message that lists most of the primary commands. Once this message displays, pressing PF1/PF13 displays the list of all SmartTest commands. From this list, information about a particular command can be displayed by selecting the appropriate number.

[Figure 109](#) shows the Help Tutorial screen for the RECALL command.

**Figure 109 • 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 some commands (on the Program View screen only), pressing PF1/PF13 after the long message, displays =NOTES= lines giving specific examples for using the command. Once the =NOTES= lines are displayed, pressing PF1/PF13 again displays the Help Tutorial for that command. The Help =NOTES= displays for those commands where specific examples are helpful. Use the RESET primary command to remove the =NOTES= lines.

[Figure 110](#) shows the Help =NOTES= for the RECALL command.

**Figure 110 • Help Notes Example**

```

File View Test Search List Options Help
-----
Command ==> Program View UIAPPLI
ASG4645I RECALL REDISPLAYS THE PREVIOUS COMMAND OR MESSAGE. Scroll ==> CSR
=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. |
=NOTE= +-----+
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

General help information is requested by selecting Help ► Table of contents, or by typing TOC on any Help Tutorial screen. The Help Table of Contents displays. Help information can be viewed by pressing Enter or by selecting a menu option.

[Figure 111](#) shows the Help Table of Contents.

**Figure 111 • Help Table of Contents**

```
ASG-SmartTest - R6.0 ----- HELP TABLE OF CONTENTS ----- HELI
===> _
The topics below represent general categories of information about the ASG-ESU
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 CUR
  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, selecting option 7 on the Help Table of Contents, or typing INDEX from within the Help Tutorial. Help for a specific topic can then be viewed by selecting the appropriate Index entry.

On any Index screen, typing an alphabetic character displays the Index screen corresponding to that character.

[Figure 112](#) shows a Help Index screen.

**Figure 112 • 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 abends is available by selecting Help ► Common abends, typing `HELP ABENDS`, or typing `ABENDS` in the command input area and pressing PF1/PF13. Each of these actions displays the ABENDS screen, as shown in [Figure 113](#).

**Figure 113 • 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 this screen displays the ASG Abend Codes screen, which lists and describes the ESW user abend messages.

[Figure 114](#) shows the ASG Abend Codes screen.

**Figure 114 • ASG Abend Codes Screen**

```
ASG-SmartTest - R6.0 ----- ASG ABEND CODES ----- HELP
===> _
Abend codes in the range 900 - 999 (X'384 - X'3E7') bypass ASG error
recovery, causing the abend to be handled by ISPF or by the system. If the
problem cannot be resolved, call Customer Support.

965 X'3C5'   Unable to intercept program.
967 X'3C7'   The ASG-Center AUTHORIZE password was not specified during
            installation.
968 X'3C8'   An internal error occurred during initialization.
970 X'3CA'   A package load module was called directly.
972 X'3CC'   The ASG Edit Monitor encountered a severe error.
974 X'3CE'   An invalid VIASBASE module was found. The current product
            expects a level of CE050 or greater. Enter HELP 4988 for more
            information.
                (continued)
```

## Help Messages

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 below.

*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.

Help for a specific message displays by selecting Help ► Specific message, or by typing HELP followed by the message number. The Help Explanation and Action Panel for that message displays.

[Figure 115](#) shows the Help Explanation and Action Panel.

**Figure 115 • 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<sub>xx</sub>.CNTL(VIASMPRT) where <sub>xx</sub> represents the version of Center installed at your installation. The entire message file is printed unless a specific range is specified in the PRM parameter. For example, PRM='START=300,END=499' would print 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 end with the message number specified.

The NOTES keyword specifies that any notes associated with a message will be printed. The default is NONOTES. Typically, notes are provided to show Center primary commands.

[Figure 116](#) and [Figure 117 on page 186](#) show the VIASMPRT JCL and the output from the job.

**Figure 116 • VIASMPRT JCL**

```

//ASG JOB ( ),'ASG-CENTER VIASMPRT' 00010000
//* INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED. 00020000
//* 00030000
//* ***** 00040000
//* * ASG, INC. ASG-CENTER * 00050000
//* * * 00060000
//* * UTILITY TO PRINT ASG MESSAGES * 00070000
//* * * 00080000
//* ***** 00090000
//* 00100000
//VIASMPRT PROC VIASOFT='ASG', ASG HI-LVL NODES 00110000
// CENTER='VIACENXX', ASG MIDDLE NODES 00120000
// SYSOUT='*', PRINT OUTPUT MESSAGE CLASS 00130000
// PRM='', PARM FOR MESSAGES TO BE PRINTED 00140000
//* 00150000
//* ***** 00160000
//* * 00170000
//* * MESSAGE PRINT UTILITY * 00180000
//* * 00190000
//* * THIS PROGRAM WILL PRINT ALL OF THE MESSAGES IN THE ASG * 00200000
//* * MESSAGE FILE AND THE HELP TEXT ASSOCIATED WITH EACH * 00210000
//* * MESSAGE IT WILL PRINT THE ENTIRE FILE BY DEFAULT. YOU MAY * 00220000
//* * SELECT A GIVEN RANGE OF MESSAGES BY SPECIFYING THE OPTION- * 00230000
//* * AL PARAMETER KEYWORDS: START AND END. FOR EXAMPLE: * 00240000
//* * PRM='START=300,END=499' * 00250000
//* * WILL PRINT MESSAGES NUMBER 300 THROUGH 499, INCLUSIVE. * 00260000
//* * THE DEFAULT VALUES FOR START AND END ARE 1 AND 99999 * 00270000
//* * RESPECTIVELY. CONSEQUENTLY THE PRM VALUE 'END=300' WILL * 00280000
//* * PRINT MESSAGES 1 THROUGH 300, AND THE PRM VALUE * 00290000
//* * 'START=4000' WILL PRINT MESSAGES 4000 THROUGH 99999. * 00300000
//* * * 00310000
//* * AN OPTIONAL KEYWORD, NOTES, WILL ALSO PRINT ANY NOTES * 00320000
//* * ASSOCIATED WITH A MESSAGE. * 00330000
//* * * 00340000
//* * ADDITIONALLY, THE KEYWORD 'ALL' WILL EXPLICITLY PRINT ALL * 00350000
//* * MESSAGES. * 00360000
//* ***** 00370000
//* 00380000
//* 00390000
//VIAMPRT EXEC PGM=VIASMPRT,REGION=4096K, 00400000
// PARM='&PRM' 00410000
//STEPLIB DD DSN=&ASG..&CENTER..LOADLIB,DISP=SHR 00420000
//VIAMSGS DD DSN=&ASG..&CENTER..VIAMSGS,DISP=SHR 00440000
//VIAPRINT DD SYSOUT=&SYSOUT 00450000
//VIALOG DD SYSOUT=&SYSOUT 00460000
//SYSUDUMP DD SYSOUT=&SYSOUT 00470000
//* 00480000
// 00490000
//* 00500000
//VIASMPRT EXEC VIASMPRT PRINT MESSAGES 00510000
//* 00520000

```

**Figure 117 • 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 Parms field of the  
  Edit Options panel.  
  
ACTION:  
  Refer to the Reference Manual or the Reference Card for a list of  
  valid editor parameters.
```

---

## Glossary

---

### **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 "[The CUA Interface](#)" on page 4 for more information.

### **active program**

A program that is being viewed and/or tested on the Program View screen. Also see [qualified program](#).

### **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 \(AKR\)](#).

### **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 that contains the ENTRY point. If the name in the PROC MAIN 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, PL/I, or Assembler program for testing. See [analyzer](#).

**analyzer**

The batch component of SmartTest that executes complex algorithmic formulas to analyze the complexities of a COBOL, PL/I, or Assembler program. It produces a detailed analysis of the elements and relationships for a program, then places this information in the AKR. There are three analyzers. The Program Analyzer analyzes COBOL programs. The Assembler Analyzer analyzes Assembler programs. The PL/I Analyzer analyzes PL/I programs.

**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 that contains the PSB and DBD, and is created by the ACBGEN process.

**Application Knowledge Repository (AKR)**

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 or Assembler Analyzer. Multiple AKRs can be defined for your installation. 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 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 \(BTS\)](#).

**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 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. The command and cursor character are typed in the command input area, the cursor is placed on the desired token, then Enter is pressed. 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, a DECLARE in PL/I, or a data area label in an Assembler program. Includes variable names, files, groups, array elements, and fully qualified datanames.

**data usage**

Defines how a data item is used: DEF indicates the statements 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. Note: This function is only available for COBOL.

**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 PL/I, COBOL, C/370, or Assembler. The SmartTest DB2 Stored Procedure option enables programmers/analysts to use SmartTest features to interactively test DB2 Stored Procedures.

**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 on line three when PF1/PF13 is pressed or HELP is entered for a short message. 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, Korean, 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 on SmartTest screens. Long messages are sometimes preceded by short messages that are displayed in the upper right corner of the screen. Pressing PF1/PF13 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 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 lengths 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.

**PCB**

See [Program Communication Block \(PCB\)](#).

**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](#).

**PL/I subset**

PL/I statements of a similar nature that have been grouped together. For example, input/output statements are grouped into the IO subset. The LIST SUBSETS command can be used to display all subsets online.

**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.

**primary command**

A series of keywords that is 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, the PROCEDURE name of a PL/I 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.

**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.

**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

**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 automates and centralizes the management of storage. To manage storage, SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

**Stored Procedure**

See [DB2 Stored Procedure](#).

**subset**

A language subset, screen subset, or tagged lines subset. See the *ASG-SmartTest Reference Guide* for detailed information on subsets.

**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 \(TCA\)](#).

**Test Coverage Analysis (TCA)**

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 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 (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.

**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 159  
&COUNT command 151  
./P cards 83  
./T command 81

## Numerics

3270 terminals supported 25  
77 pseudo code statement 151

## A

### ACB

allocation 70  
definition 188

action bar 4, 187

active program 5, 187

ADD command 150

adding pseudo code 150

### address

commands definition 187  
space 78  
stop 187  
swapped space 78

AGN execution parameter 76

### AKR 2

allocating 27  
catalog 28  
concatenating 36, 38  
definition 1  
directory 37  
password 28  
SMS classes 28  
utilities 27

alias 187

Alliance 129, 160

accessing from ESW screen *xiii*  
description *x*  
linking *xiii*

Alliance, linking to 160

ALLOC parameter 41

### allocating

ACB library datasets 70  
alternate message queue 88  
alternate PCB output dataset 90  
an AKR 27  
BTS files 83  
BTS input dataset 93  
BTS program output dataset 91  
DB monitoring datasets 71  
DFSRESLB and DFSESL 67  
execution parameters 74  
IEFRDER dataset 72  
IMS files 65  
IMS PROC libraries 68  
ISPF list file 59  
ISPF log dataset 60  
Log/List/Punch/Work files 18  
output message queue 86  
PSB and DBD library datasets 69  
QALTPCB 88  
QIOPCB 86  
script files 24  
SNAP dump dataset 95, 97  
VSAM buffer pools 68

allocation CLIST 41, 101

### alternate

message queue 88  
PCB 84

### analyze method

from a User screen 34  
from an Edit session 32  
through SmartTest 31

Analyze Submit Parameters pop-up 33

analyzing a program 30

application control block

*see ACB*

application group name 76

Application Knowledge Repository

*see AKR*

application parameters 101

ASG Abend Codes screen 182

ASG-SmartTest Primary screen 15

- Assembler
  - and SmartTest-PLI 173
  - integration 2
- AutoChange 129
  - accessing from ESW screen [xiii](#)
  - description [x](#)
- B**
- backout and recovery datasets 72
- backtrack
  - recording history 140
  - reviewing history 141
- Batch
  - connecting for test 112
  - DFHDRP testing 119
  - initiating test 114
  - region testing 108
  - session setup 109
  - setup considerations 115
  - submitting the job 112
  - test session 112, 188
  - testing BTS 117
  - testing DB2 118
  - testing DL/I 116
- Batch Connect facility 108
- Batch Session Setup screen
  - example 109
  - field descriptions 110
  - options 110
- Batch Terminal Simulator 188
- BIND 101
- BKO execution parameter 79
- BMP
  - execution parameter 66
  - parameters 74
  - region 76
- BREAK command 129, 150
- break on entry 80, 101
- breakpoint
  - definition 189
  - displaying 134
  - inserting 128, 133
  - removing 134
  - setting with impact datasets 129
- Breakpoints List screen 132
- Bridge
  - accessing from ESW screen [xiii](#)
  - description [x](#)
- BTS
  - allocating input dataset 93
  - allocating program output dataset 91
  - alternate message queue 88
  - alternate PCB output dataset 90
  - BTSDEBUG 95
  - BTSOUT 91
  - BTSPUNCH 93
  - BTSSNAP 97
    - definition 188
    - format libraries 86
    - in TSO session setup 79
    - input 84
    - output message queue 86
  - QALTPCB 88
  - QALTRAN 90
  - QIOPCB 86
    - setup information 79
  - SNAP dumps dataset 95
  - STAX indicator 85
  - system parameters 84
  - system variables 84
  - testing in Batch 117
  - transactions to monitor 81
  - work file 84
- BTS Allocation Selection pop-up
  - example 83
  - options 84
- BTS Batch Session Setup screen 117
- BTS BTSDEBUG Allocation pop-up
  - example 96
  - field descriptions 96
- BTS BTSOUT Allocation pop-up
  - example 92
  - field description 92
- BTS BTSPUNCH Allocation pop-up
  - example 94
  - field descriptions 94
- BTS BTSSNAP Allocation pop-up
  - example 98
  - field descriptions 98
- BTS FORMAT Libraries pop-up 86
- BTS Load Library pop-up 85
- BTS QALTPCB Allocation pop-up
  - example 88
  - field descriptions 89
- BTS QALTRAN Allocation pop-up
  - example 90
  - field descriptions 90
- BTS QIOPCB Allocation pop-up
  - example 87
  - field descriptions 87
- BTS Session Setup screen
  - example 79
  - options 80
- BTSDEBUG allocation 95
- BTSIN
  - dataset 81
  - patch cards 83
- BTSOUT allocation 91

- BTSPUNCH allocation 93
- BTSSNAP allocation 97
- BUF execution parameter 77
- buffers, page-fixed 76
- C**
- CALLed load modules, testing 124
- CANCEL command 134
- cancelling execution 134
- capturing scripts 155
- Center, description x
- changing data values 135, 140
- checkpoint/restart ID 75, 78
- CKPTID execution parameter 75, 78
- CLIST
  - converting JCL 40
  - VIAPUBTS 84
  - VIAPUIMS 66
  - VIAPUSPF 56
- COBOL
  - CASE generated COBOL support 3
  - COBOL II support 3
  - pseudo code 151
  - reserved words 151
  - subsets 163
- command scripts 155
- command syntax, learning 122
- commands
  - & (retain) 159
  - &COUNT 151
  - ./T 81
  - ADD 150
  - ALLIANCE 162
  - BR (Breakpoint) 128
  - BREAK 129, 150, 163
  - CANCEL 134
  - EQUATE 158
  - EXCLUDE 163, 166
  - EXECUTE 133, 156
  - F (first) 166
  - FINDXTND 163, 167
  - GO 127
  - GO TO 150
  - HIGH 163
  - IF/THEN/ELSE 150
  - KEEP 138
  - KG (keep group) 138
  - KGH (keep group hex) 138
  - KH (keep hex) 138
  - L (last) 166
  - line 6
  - LIST ADSTOP 122
  - LIST BREAKS 134
  - LIST COUNTS 122
  - LIST EQUATES 158
  - LIST PSEUDO 153
  - LIST TAILOR 155
  - LIST TRACKING 122
  - LOCATE \* 133, 157
  - LPRINT 163, 171
  - MOVE 150
    - primary 5
    - processing order 4
  - PRODLVL 160
  - QUALIFY 124, 154
  - RECALL 159
  - RESET 138
  - RESET EXCLUDED 166
  - RESET PSEUDO 153
  - RUN 122, 125
  - RUN PROGRAM 101
  - S (show) 166
  - SCROLL 163, 172
  - SET 26
  - SET BACKTRACK 140
  - SET SCRIPT 155
  - SET XMODE 168
  - simplify entering 158
  - STEP 126
  - SUBTRACT 150
  - TESTPOINT 130
  - WHEN 150
  - ZD (zoom display) 135
  - ZG (zoom group) 137
  - ZGH (zoom group hex) 137
  - ZH (zoom hex) 135
  - ZO (zoom out) 138
  - ZOOMDATA 138
- Common User Access (CUA) 4
- concatenating AKRs 36
- Connect to Job pop-up
  - example 113
  - options and status descriptions 113
- continuous execution 125
- controlling execution
  - locating next statement 133
  - specified number of statements 126
- conventions page xvi
- Convert Batch JCL pop-up
  - example 41
  - field descriptions 42
  - options 41
- converting JCL to CLIST 40
- CPUTIME execution parameter 76
- CUA
  - action bar 4
  - overview 2, 4
  - pop-up 4

- cursor character 189
- cursor substitution 158
- cursor token 158
- customizing DSN for Log, List, and Punch files 23
- D**
- data
  - classes, SMS 28
  - element 7
  - field references, finding 170
  - values, changing 140
- data windows
  - at top of screen 138
  - inline 135
  - Keep 138
  - removing from screen 139
  - Zoom 135
- database
  - DB2 support 4
  - dynamic backout 79
  - ISM/DB support 4
  - plan name 191
  - VSAM support 4
- Database Recovery Control 78
- dataname
  - alias 187
  - description 7
  - elementary dataname 7
  - fully qualified 7
  - group level 7
- DB monitoring datasets 71
- DB2
  - allocating load library 67
  - in TSO session setup 100
  - load library datasets 67
  - plan 101
  - plan name 191
  - subsystems 76
  - test session 100
  - testing in Batch 118
- DB2 Batch Session Setup screen 118
- DB2 Session Setup screen
  - example 100
  - field descriptions 101
  - options 100
- DB2 Stored Procedure
  - definition 102
  - displaying parameters 105
  - entering test data 105
  - initiating the test 107
  - requirements 102
  - session setup 102
  - setting up the test 103
  - Testing Option 102
- DBB
  - execution parameter 66
  - parameters 76
- DBCS (Double Byte Character Set) 190
- DBD
  - allocation 69
  - libraries 66
- DBRC execution parameter 78
- DD statements
  - BTS load library 85
  - BTSDEBUG 95
  - BTSOUT 91
  - BTSPUNCH 93
  - BTSSNAP 97
  - DFSRESLB and DFSESL 67
  - DFSVSAMP and PROCLIB 68
  - FORMAT 86
  - IEFRDER 72
  - IMSACB 70
  - IMSMON 71
  - PSB and DBD 69
  - QALTPCB 88
  - QALTRAN 90
  - QIOPCB 86
  - VIAQUEUE 111
- DEALC parameter 42
- default ISPF libraries 58
- defining PF key values 25
- DFHDRP
  - batch JCL 119
  - testing in Batch 119
- DFSESL allocation 67
- DFSRESLB allocation 67
- DFSVSAMP allocation 68
- DIRCA execution parameter 75
- directory, AKR members 37
- displaying memory
  - see address command*
- displaying, excluded lines 166
- DLI execution parameter 66
- DLI parameters 76
- Double Byte Character Set 190
- E**
- editing pseudo code 152
- Encore
  - accessing from ESW screen xiii
  - description xi
- ending a test session 49
- entering pseudo code 152

- entry point, alias name [187](#)
  - Environment Selection pop-up
    - example [36](#)
    - options [36](#)
  - equate [190](#)
  - EQUATE command [158](#)
  - Estimate [129](#)
    - accessing from ESW screen [xiii](#)
    - description [xi](#)
  - ESW
    - description [ix](#)
    - invoking products [xii](#)
    - product integration [xiii](#)
  - ESW integration [2](#)
  - EXCLUDE command [166](#)
  - excluding lines
    - in search function [166](#)
    - SET XMODE [168](#)
  - EXCPVR execution parameter [78](#)
  - executable statement, finding next [157](#)
  - EXECUTE command [133](#)
  - executing a program
    - by statement [126](#)
    - cancelling [134](#)
    - changing data [135](#)
    - changing sequence [127](#)
    - continuous [125](#)
    - displaying breakpoints [134](#)
    - interrupting [128](#)
  - executing a specified number of statements [126](#)
  - executing pseudo code [153](#)
  - execution parameter
    - BMP [66](#)
    - DLI/DBB [66](#)
  - Execution Tracking screen [123](#)
  - Existing Systems Workbench (ESW)
    - example of primary screen [14](#)
  - exiting SmartTest-PLI [135](#)
- F**
- F (first) command [166](#)
  - fast path
    - data buffers [76](#)
    - databases [76](#)
  - File - AKR Allocate/Expand pop-up [28](#)
  - File - AKR Utility pop-up [27](#)
  - File - Analyze Submit pop-up [31](#)
  - File - Execute Script pop-up [157](#)
  - File pull-down [154](#)
  - finding
    - data field references [170](#)
    - I/O statements [168](#)
  - FINDXTND command [167](#)
  - FMTO execution parameter [78](#)
  - format libraries [86](#)
  - Formatted Dump Delete List (FDDL) [78](#)
  - formatted dump output [78](#)
- G**
- getting started [13](#)
  - GO command [127](#)
  - GO TO command [150](#)
  - guidelines for using
    - MONITOR/NOMONITOR [123](#)
- H**
- help
    - for abends [181](#)
    - for commands [178](#)
    - for general information [180](#)
    - for messages [182](#)
    - for screens [177](#)
    - for specific information [180](#)
    - index [180](#)
    - navigational commands [177](#)
    - NOTES [179](#)
    - online [175](#)
    - printing messages [184](#)
    - requesting [177](#)
    - Table of Contents [180](#)
  - Help Explanation and Action Panel [183](#)
- I**
- I/O statements, finding [168](#)
  - IEFRDER allocation [72](#)
  - IF/THEN/ELSE statements [150](#)
  - impact datasets [129](#)
  - impact of change [165](#)
  - impact types [131](#)
  - IMS
    - AGN execution parameters [76](#)
    - allocating backout and recovery datasets [72](#)
    - BKO execution parameters [79](#)
    - BMP execution parameters [66](#)
    - BMP parameters [74](#)
    - BUF execution parameters [77](#)
    - CKPTID execution parameters [75, 78](#)
    - control region [76](#)
    - control region options [75](#)
    - CPUTIME execution parameters [76](#)
    - DBB execution parameters [66](#)
    - DBRC execution parameters [78](#)
    - DIRCA execution parameters [75](#)
    - DLI execution parameters [66](#)
    - EXCPVR execution parameters [78](#)

- file allocations 65
- FMTO execution parameters 78
- identifier 78
- IMSID execution parameters 76, 78
- IN execution parameters 75
- IRLM execution parameters 79
- IRLMNM execution parameters 79
- load library 66
- load library datasets 56, 67, 85–86
- LOGA execution parameters 78
- logging facility 66
- MON execution parameters 78
- monitoring 78
- NBA execution parameters 76
- OBA execution parameters 76
- OPT execution parameters 75
- OUT execution parameters 75
- PARDLI execution parameters 76
- PREINIT execution parameters 76
- PRLD execution parameters 75, 78
- RST execution parameters 78
- SPIE execution parameters 75, 77
- SRCH execution parameters 78
- SSM execution parameters 76
- STIMER execution parameters 75
- SWAP execution parameters 78
- system parameters 66
- system variables 66
- TEST execution parameters 75, 77
- IMS ACB Allocation pop-up 70
- IMS Allocation Selection pop-up
  - example 65
  - options 66
- IMS Batch Session Setup screen 116
- IMS BMP parameters
  - AGN 76
  - CKPTID 75
  - CPUTIME 76
  - DIRCA 75
  - IMSID 76
  - NBA 76
  - OBA 76
  - OPT 75
  - OUT 75
  - PARDLI 76
  - PREINIT 76
  - PRLD 75
  - SPIE 75
  - SSM 76
  - STIMER 75
  - TEST 75
- IMS BMP Parameters pop-up
  - example 74
  - field descriptions 75
- IMS DFRESLB/DFSESL Allocation pop-up
  - example 67
  - field descriptions 67
- IMS DLI/DBB parameters
  - BKO 79
  - BUF 77
  - CKPTID 78
  - DBRC 78
  - EXCPVR 78
  - FMTO 78
  - IMSID 78
  - IRLM 79
  - IRLMNM 79
  - LOGA 78
  - MON 78
  - PRLD 78
  - RST 78
  - SPIE 77
  - SRCH 78
  - SWAP 78
  - TEST 77
- IMS DLI/DBB Parameters pop-up
  - example 77
  - field descriptions 77
- IMS IEFORDER Allocation pop-up
  - example 73
  - field descriptions 73
- IMS IMSMON Allocation pop-up
  - example 71
  - field descriptions 72
- IMS PROCLIB/DFSVSAMP Allocation pop-up
  - example 68
  - field descriptions 69
- IMS Resource Lock Manager (IRLM) 79
- IMS/DB
  - setup 63
  - TSO foreground 63
- IMS/DB PSB/DBD Allocation pop-up 69
- IMS/DB Session Setup screen
  - example 63
  - field descriptions 64
  - options 63
- IMS/OSAM buffer pool 77
- IMSCTRL macro 79
- IMSID execution parameter 76, 78
- IMSMON allocation 71
- IN clause, in search function 166
- IN execution parameter 75
- Insight
  - accessing from ESW screen xiii
  - description xi
  - using analysis functions xiii
- Intelligent Search Function 163

- intercepting
    - load modules 54
    - programs 124
  - intercepts 124
  - interregion communication area 75
  - introduction for new users 13
  - IO statements 165
  - IRLM (IMS/VS Resource Lock Manager) 79
  - IRLM execution parameter 79
  - IRLMNM execution parameter 79
  - ISAM/OSAM buffer 78
  - ISPF
    - allocating log dataset 60
    - specifying list file 59
    - system variables 56
  - ISPF Allocation pop-up
    - example 55
    - options 55
  - ISPF List Data Set Allocation pop-up
    - example 59
    - field descriptions 59
  - ISPF Log Data Set Allocation screen
    - example 60
    - field descriptions 61
  - ISPF Panel/Link Libraries pop-up 57
  - ISPF Program Load Library pop-up 56
  - ISPF Session Setup screen
    - example 52
    - field descriptions 53
    - options 53
  - ISPF Table/Message/Skeleton Libraries
    - pop-up 58
  - ISPMAIN, specifying location 55
  - ISPMLIB 58
  - ISPSLIB 58
  - ISPTLIB 58
- J**
- JCL
    - convert to CLIST 40
    - VIASMPRT program 184
  - Job Pack Area (JPA) 78
  - job statuses 113
  - JOBLIB/STEPLIB 78
  - JPA (Job Pack Area) 78
- K**
- K (keep) command 138
  - KEEP command 138
  - Keep window
    - removing globally 139
    - removing individually 139
  - KG (keep group) command 138
  - KGH (keep group hex) command 138
  - KH (keep hex) command 138
- L**
- L (last) command 166
  - label name 7
  - language support
    - Assembler 3
    - COBOL 3
    - PL/I 3
  - LEARN mode 122
  - learning command syntax 122
  - libraries
    - default 58
    - panel and link 55
    - PSB and DBD 66
    - table/message/skeleton 55
  - line range 8
  - link libraries, specifying 55
  - Link Pack Area (LPA) 78
  - LINKLST 78
  - List - Analyze Submit pop-up 123
  - List - BackTrack Variable History
    - pop-up 148
  - LIST ADSTOP command 122
  - LIST BREAKS command 134
  - LIST COUNTS command 122
  - LIST dataset 56
  - LIST EQUATES command 158
  - LIST PSEUDO command 153
  - LIST TAILOR command 155
  - LIST TRACKING command 122
  - Load Module Intercept List pop-up 54
  - load modules, intercepting 54
  - LOCATE \* command 133
  - LOCATE command 157
  - locating next executable statement 133
  - log dataset, ISPF 56
  - Log Name Customization pop-up 23
  - Log/List/Punch processing options 19
  - LOGA execution parameter 78
  - logging
    - access method 78
    - facility 66
    - run-time activities 66
  - logical terminal name 75
  - long message area 6
  - LPA (Link Pack Area) 78
  - LPRINT command 171

## M

MAIN option, advisory modes 26  
management classes, SMS 28  
Message Format Service library 86  
message libraries, specifying 55  
message queues 75, 88  
messages  
    long 6, 183  
    printing 184  
    short 5, 183  
MFS (Message Format Services) 84  
modify program logic 149  
MON execution parameter 78  
MONITOR  
    guidelines 123  
    testing mode 122  
monitoring  
    BTS transactions 81  
    dataset 66  
MOVE command 150  
multiple program processing 154  
MVS Batch session 188

## N

NBA execution parameter 76  
new user introduction 13  
NOMONITOR  
    facilities not available 123  
    guidelines 123  
    testing mode 123  
NOTES in help 179  
NVC 75

## O

OBA execution parameter 76  
Open pull-down 154  
OPT execution parameter 75  
optimization considerations 9  
Options - Allocation Definition  
    pop-up 112  
Options - Log/List/Punch Definitions pop-up  
    example 20  
    field descriptions 20  
Options - Modes pop-up 26  
Options - PF Key Definition pop-up 25  
Options - Product Allocations pop-up  
    example 18  
    field descriptions 19  
Options - Product Parameters pop-up  
    example 17  
    field descriptions 18

Options - Script File Allocations pop-up  
    example 24  
    field descriptions 24  
    setting defaults 24  
Options Menu screen 16  
options, setting and verifying 16  
OUT execution parameter 75

## P

page-fixed buffers 76  
panel libraries, specifying 55  
parameter definition 17  
PARDLI execution parameter 76  
patch (./P) cards 83  
pattern string 8  
PCB alternate output 84, 90  
PF keys, defining 25  
PL/I  
    optimization considerations 9  
    statements 163  
    subsets 163  
plan name 191  
pop-up 4  
PREINIT execution parameter 76  
Preview BTSIN Data Set screen  
    example 82  
    field descriptions 82  
printing  
    messages 184  
    program information 171  
PRLD execution parameter 75, 78  
PROC statements 165  
PROCLIB allocation 68  
PRODLVL command 160  
product integration xiii  
profile  
    restoring 49  
    saving 47  
    sharing 48  
    test session setup 47  
Profile Data Set Member List screen 47  
program  
    analyzing 30  
    controlling execution 124  
    entry point alias name 187  
    intercepting 124  
    modifying logic 149  
    printing information 171  
    VIASMPRT 184  
Program View screen 5, 133  
    example 114  
    line numbers 8  
    screen descriptions 114

- PSB
  - allocation 69
  - libraries 66
- pseudo code
  - &COUNT 151
  - 77 151
  - adding 150
  - BREAK 150
  - COBOL reserved words 151
  - concepts 149
  - definition 192
  - editing 152
  - entering and editing 152
  - executing 153
  - GO TO 150
  - IF/THEN/ELSE 150
  - listing 153
  - making permanent 154
  - MOVE 150
  - pslabel 150
  - removing 153
  - SUBTRACT 150
  - viewing 153
  - WHEN 150
- pslabel statement 150
- pull-down, Test 125
- punch file 192
- Q**
  - QALTPCB allocation 88
  - QALTRAN allocation 90
  - QIOPCB allocation 86
  - qualified program 5, 192
  - QUALIFY command 124, 154
- R**
  - RECALL command 159
  - recalling
    - commands 159
    - messages 159
    - pop-ups 159
  - Recap
    - accessing from ESW screen xiii
    - description xi
  - recording backtrack execution history 140
  - region size 112
  - release and level numbers 160
  - removing
    - data windows 138
    - pseudo code 153
  - replaying scripts 156
  - repositioning the display 172
  - RESET command 138
  - RESET EXCLUDED command 166
  - RESET PSEUDO command 153
  - RESET ZOOM command 138
  - restoring test session setups 49
  - retaining commands 159
  - REUS parameter 123
  - RST execution parameter 78
  - RUN command 122, 125
  - RUN PROGRAM command 101
  - run-time activities 66
- S**
  - S (show) command 166
  - Sample Edit Session screen 32
  - sample SmartTest session 13
  - saving
    - scripts 155
    - test session setup 47
  - scratch pad area 83
  - screen
    - Analyze Submit Parameters 33
    - ASG Abend Codes 182
    - Batch Session Setup 109
    - Batch Submit 118
    - Breakpoints List 132
    - BTS Allocation Selection 83
    - BTS Batch Session Setup 117
    - BTS BTSDEBUG Allocation 96
    - BTS BTSOUT Allocation 92
    - BTS BTSPUNCH Allocation 94
    - BTS BTSSNAP Allocation 98
    - BTS FORMAT Libraries 86
    - BTS Load Library 85
    - BTS QALTPCB Allocation 88
    - BTS QALTRAN Allocation 90
    - BTS QIOPCB Allocation 87
    - BTS Session Setup 79
    - command input area 5
    - Connect to Job 113
    - Connect to Job Message 112
    - Convert Batch JCL 41
    - DB2 Batch Session Setup 118
    - DB2 Session Setup 100
    - Environment Selection 36
    - Execution Tracking 123
    - Existing Systems Workbench 14
    - File - AKR Allocate/Expand 28
    - File - AKR Utility 27
    - File - Analyze Submit 31
    - File - Execute Script 157
    - format example 5
    - general field descriptions 5
    - Help Explanation and Action Panel 183

- IMS ACB Allocation 70
- IMS Allocation Selection 65
- IMS Batch Session Setup 116
- IMS BMP Parameters 74
- IMS DFSRESLB/DFSESL Allocation 67
- IMS DLI/DBB Parameters 77
- IMS IEFORDER Allocation 73
- IMS IMSMON Allocation 71
- IMS PSB/DBD Allocation 69
- IMS/DB in TSO session setup 63
- ISPF Allocation 55
- ISPF List Data Set Allocation 59
- ISPF Log Data Set Allocation 60
- ISPF Panel/Link Libraries 57
- ISPF Program Load Library 56
- ISPF Session Setup 52
- ISPF Table/Message/Skeleton Libraries 58
- line command input area 6
- List - Analyze Submit 123
- List - BackTrack Variable History 147-148
- Load Module Intercept List 54
- long message area 6
- Options - Log/List/Punch Definition 20
- Options - Modes 26
- Options - PF Key Definition 25
- Options - Product Allocations 18, 112
- Options - Product Parameters 17
- Options - Script File Allocations 24
- Options Menu 16
- Preview BTSIN Data Set 82
- PROCLIB/DFSVSAMP Allocation 68
- Profile Data Set Member List 47
- Program View 5, 8, 114, 133
- Pseudo Code List 153
- Sample Edit Session 32
- Script File Allocations 161
- Search - Data Name 170
- Search - Subset Name 168
- short message area 5
- SmartTest Primary 15
- SmartTest/Alliance Interface 162
- subsets 165
- Test - ASG-SmartTest Testpoint Generation 130
- Test - Run Request 125
- Test - Run Request (Backtrack active) 142
- Test - Step Request 126
- Test - Step Request (Backtrack active) 144
- Test - Testpoint Qualified Program List 131
- Test Session Tailoring 155
- TSO Session Setup 38
- TSO/ISPF Edit 5
- View - Reset Request 167, 169
- screen subset 193
- script
  - definition 193
  - file 2, 193
  - recording 155
  - replaying 156
  - STFINDAT 133
- SCROLL command 172
- scrolling 6, 172
- Search - Data Name pop-up 170
- Search - Subset Name pop-up 168
- search function
  - excluding lines 166
  - impact of change 165
  - IN clause 166
  - intelligent 163
  - language subsets 163
  - limiting 165
  - primary commands 163
- session tailoring 155
- SET BACKTRACK command 140
- SET command 26
- SET SCRIPT command 155
- SET XMODE command 168
- setting up the DB2 Stored Procedure test 103
- setup
  - Batch connect 109
  - BTS in TSO 79
  - DB2 in TSO 100
  - DB2 stored procedure 102
  - IMS/DB in TSO 63
  - ISPF Dialog Manager 52
  - using wizards 125
- setup considerations
  - Batch region 115
  - TSO foreground 46
- sharing test session setups 48
- simplify entering commands 158
- skeleton libraries, specifying 55
- SmartDoc
  - accessing from ESW screen xiii
  - description xi
- SmartEdit
  - accessing from ESW screen xiii
  - description xii

- SmartTest
    - accessing from ESW screen [xiii](#)
    - description [xii](#)
    - method of operation [122](#)
    - terminals supported [25](#)
  - SmartTest-ASM [173](#)
  - SmartTest-PLI
    - and Assembler [173](#)
    - and COBOL [173](#)
    - exiting a test session [135](#)
    - getting started [13](#)
    - sample session [13](#)
    - setting up a test session [35](#)
  - SMS classes [28](#)
  - SNAP dump [84](#)
  - SNAP dumps dataset, allocating [95](#)
  - SPIE execution parameter [75, 77](#)
  - SRCH execution parameter [78](#)
  - SRM (System Resource Manager) [78](#)
  - SSM execution parameter [76](#)
  - statement execution [126](#)
  - status box [6](#)
  - status of job [113](#)
  - STAX indicator [85](#)
  - STEP command [126](#)
  - STFINDAT script [133](#)
  - STIMER execution parameter [75](#)
  - storage classes, SMS [28](#)
  - subset
    - screen [165](#)
    - search function [163](#)
    - tag [165](#)
  - subsystem identifier [76, 78](#)
  - SUBTRACT command [150](#)
  - SWAP execution parameter [78](#)
  - System Resource Manager (SRM) [78](#)
- T**
- table entry [7](#)
  - table libraries, specifying [55](#)
  - tag subset [165](#)
  - target
    - dataname [7](#)
    - description [7](#)
    - label name [7](#)
    - pattern string [8](#)
  - TCA definition [194](#)
  - terminals supported by SmartTest [25](#)
  - terminating a test session [49](#)
  - Test - ASG-SmartTest Testpoint Generation pop-up [130](#)
  - Test - Run Request pop-up [125](#)
  - Test - Run Request pop-up (with Backtrack active) [142](#)
  - Test - Step Request pop-up [126](#)
  - Test - Step Request pop-up (Backtrack active) [144](#)
  - Test - Testpoint Qualified Program List pop-up [131](#)
  - Test Coverage Analysis (TCA) option [194](#)
  - TEST execution parameter [75, 77](#)
  - test initiation
    - Batch [114](#)
  - Test pull-down menu [125](#)
  - test session
    - allocating an AKR [27](#)
    - analyzing a program [30](#)
    - beginning [14](#)
    - converting JCL to CLIST [40](#)
    - initiating [45](#)
    - restoring setups [49](#)
    - saving setup [47](#)
    - selecting the test environment [35](#)
    - set up [35](#)
    - setting and verifying options [16](#)
    - sharing setups [48](#)
    - terminating [49](#)
    - TSO setup [38](#)
    - using setup wizards [125](#)
  - Test Session Tailoring screen [155](#)
  - testing
    - Batch region [108](#)
    - DB2 [118](#)
    - DFHDRP [119](#)
    - dynamically CALLED load modules [124](#)
    - using MONITOR [122](#)
    - using NONMONITOR [123](#)
  - TESTPOINT command [130](#)
  - token [194](#)
  - token/cursor substitution [158, 189](#)
  - trace table [84](#)
  - transaction code name [82](#)
  - TSO
    - SYSPROC DD statement [42](#)
    - terminal attention exit [85](#)
  - TSO foreground
    - BTS [79](#)
    - DB2 programs [100](#)
    - IMS/DB programs [63](#)
    - setup [46](#)
  - TSO Session Setup screen
    - example [38](#)
    - field descriptions [39](#)
  - TSO/ISPF environment [1](#)

## U

UCF (Utility Control Facility) 78  
user call list 75, 77  
user options 16  
Utility Control Facility (UCF) 78

## V

VIAPALSC CNTL member 161  
VIAPDLI 119  
VIAPQ\* CNTL members 161  
VIAPUBTS CLIST 84  
VIAPUIMS CLIST 66  
VIAPUSPF CLIST 56  
VIAQUEUE 111  
VIASMPRT program  
    example JCL 185  
    JCL 184  
    output example 186  
    printing messages 184  
View - Reset Request pop-up 167, 169  
viewing data 135  
    at top of screen 138  
    declarations 138  
    inline 135  
    removing windows 138  
viewing pseudo code 153  
VS identifier 78  
VSAM buffer pool, allocating 66

## W

WHEN command 150  
wizard, setting up environment 125

## Z

ZD (zoom display) command 135  
ZG (zoom group) command 137  
ZGH (zoom group hex) command 137  
ZH (zoom hex) command 135  
ZO (zoomout) command 138  
ZOOMDATA command 138



ASG Worldwide Headquarters Naples Florida USA | [asg.com](http://asg.com)