

ASG-Recap™ User's Guide

Version: 6.0

Publication Number: RCX0200-60

Publication Date: February 2002

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

© 1993 - 2002 Allen Systems Group, Inc. All rights reserved.

All names and products contained herein are the trademarks or registered trademarks of their respective holders.



ASG Worldwide Headquarters Naples, Florida USA | asg.com

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

ASG Support Numbers

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

Please have this information ready:

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

If You Receive a Voice Mail Message:

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

Severity Codes and Expected Support Response Times

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

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

Business Hours Support

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

Non-Business Hours - Emergency Support

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

ASG Web Site

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

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

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

Contents

Preface	ix
About this Publication	ix
Related Publications	xi
ASG-Existing Systems Workbench (ASG-ESW)	xii
Invoking ESW Products	xv
ESW Product Integration	xvi
Examples	xvii
Publication Conventions	xix
1 Introduction	1
Introduction	1
Recap Overview	2
Application Definition and Analysis	2
Recap Reports	3
User Interface	4
Interfaces	4
Operating Systems	4
Printers	5
COBOL Compiler Support	5
Preprocessor Support	5
2 Concepts	7
Application Knowledge Repository (AKR)	7
Application Portfolio Analysis	8
Function Points	8
Function Point Counting	9

Information Needed for Function Point Analysis	9
Establishing the Boundaries of the Application	10
Counting Function Point Types	10
Determining Functional Complexity.	12
Computing the Unadjusted Function Point Value	14
Computing the Value Adjustment Factor (VAF)	15
Computing the Adjusted Function Point Value	16
Computing the Enhancement Function Point Count.	16
Metrics	17
Cyclomatic Complexity	18
Control Variable Complexity	18
Software Science Volume.	19
Essential Complexity	19
Knots Count	20
Statistical Terms	21
3 Building and Analyzing an Application.	23
FP Task Manager	27
Gathering Information	28
Initiating a Recap Session.	29
Verifying User Options.	31
Product Parameters	31
Product Allocations.	32
Log and List File Parameters	33
Script File Allocations	35
PF Key Settings.	36
Allocating an AKR	37
Creating an Application	40
Defining Application Components and Attributes.	42
Computing the Function Point Value Adjustment Factor	42
Analyzing the Application	44
Generating Recap Reports	44
4 Using Portfolio Analysis Data.	49
Portfolio Analysis Overview.	49
Uses of Function Points.	51
Scenario.	51
Solution.	52

Using the Portfolio Analysis Data	53
Continuous Improvement Scenario	53
Reengineering Scenario	57
Modifying the Executive Summary	69
Scenario	69
Exporting Portfolio Analysis Data	70
5 Viewing and Editing Function Point Data	71
View Function Point Information On-line	71
Preview Adjusted FP - Summary	72
Preview Enhancement FP - Summary	72
View Function Point Analysis Data in Reports	73
Editing Recap Function Points	74
Eliminating Work Files and Related Transactions	74
Removing Duplicate ILFs and EIFs	77
Reviewing and Reclassifying ILF and EIF Elements	78
Reclassifying EO/EI Pairs to EQ	79
Adding Function Point Types	82
Modifying Existing Function Point Types	82
What if the Application is Reanalyzed?	83
6 Report Descriptions	85
Report Headings	87
Report Help	88
Table of Contents	89
Application Summary	91
Executive Summary Report	93
Application Definition Report	94
Application Function Point Analysis Report	97
External Input Type Table	97
External Output Type Table	99
External Inquiry Type Table	100
Internal Logical File Type Table	101
External Interface File Type Table	102
Total Unadjusted Function Points by Element Types Table	103
General System Characteristics Table	105

Program Metrics History Report	107
Program Comparison Report	113
Application Metrics Report	117
Application Exceptions Report	120
Application Comparison Report	123
Application Progress Report	130
Enterprise Metrics Report	132
Enterprise Exceptions Report	135
Master Index	138
7 Report Generator Language.....	141
Description of Programming Language	143
Site Specific Configuration.....	143
Data Structures	144
Data Element Syntax.....	144
Primary and Subsidiary Tables	145
Accessing Elements in Subsidiary Tables.....	145
Language Constructs	145
Text Substitution	146
Data Manipulation.....	146
Control and Text Selection	147
Output Formatting.....	149
Data Elements - Table and Field Descriptions	151
Primary Table Elements	154
Subsidiary Tables	157
8 Import Facility.....	159
Importing an Application Definition.....	159
Import Language Reference.....	160
Import Language Syntax.....	160
Import File Structure.....	160
Sample Import File	174
Import Language Keywords	177
Application Information	177
Default Section	177
Library and Member Definition Section.....	178

9 Export Metrics Facility	179
Creating CDF Files for Export	179
Output from Export	181
Program and Application Metric Information	181
Detailed Function Point Information	182
Summarized Function Point Information	183
10 Exporting an Application Definition	185
Exporting the Application Definition	185
Export Procedure	186
Copying the Application Definition	187
Creating a Similar Application Definition	187
11 AKR Utilities	189
Introduction to AKR Management	189
AKR Structure	189
Online AKR Utilities	190
Batch AKR Utilities	190
Job Control Statements	191
Control Cards	192
Command Format	192
Command Syntax	193
ANLZSTAT Command	194
Function	194
Operands	194
Usage Notes	194
COMMENT Command	195
Function	195
Operands	195
Usage Notes	195
CONVERT Batch AKR Command	196
Function	196
Operand	196
Usage Notes	197
COPY Batch AKR Command	198
Function	198
Operands	198
Usage Notes	198
DELETE Batch AKR Command	199
Function	199
Operands	199

Usage Notes	199
EXPORT Batch AKR Command	200
Function	200
Operands	200
Usage Notes	200
HELP Batch AKR Command	201
Function	201
Operand	201
Usage Notes	201
INIT Batch AKR Command	202
Function	202
Operand	202
Usage Notes	202
MOVE Batch AKR Command	203
Function	203
Operands	203
Usage Notes	203
PRINT Batch AKR Command	204
Function	204
Operands	204
Usage Notes	204
PUNCH Batch AKR Command	205
Function	205
Operands	205
Usage Notes	206
Batch AKR Reports	207
AKR Utility Directory Report	207
AKR Utility Log	208
Analysis Status Report	209
Punch Directory File	214
Allocating and Expanding AKRs without ISPF	215
12 Application Conversion	221
Converting Recap Applications to Recap 3.2 and later	221
13 Report Options	223
14 Help Facility	227
Help Navigational Commands	229
Screen and Pop-up Help	230
Report Help	231

General Information	232
Specific Information	233
Help Abends	234
Help Messages	235
Printing Messages	236
15 Sample AKR	239
Glossary	241
Index	251

Preface

This *ASG-Recap User's Guide* describes how to use ASG-Recap (herein called Recap) for application definition and report generation. Recap offers automated inventory analysis capabilities to evaluate COBOL applications. Recap reports contain function point analysis and metrics data, program quality assessments, intra-application and inter-application comparisons and summaries, and historical reporting of function point and metrics data. The portfolio analysis information can also be viewed online or exported to PC databases, spreadsheets, or graphics packages.

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,"](#) gives an overview of Recap, its interfaces, and operating system.
- [Chapter 2, "Concepts,"](#) describes the concepts and terms related to the operation of Recap.
- [Chapter 3, "Building and Analyzing an Application,"](#) guides you through the procedures used to define an application and generate function point data.
- [Chapter 4, "Using Portfolio Analysis Data,"](#) presents several scenarios to illustrate potential uses of Recap's function point and inventory analysis data.
- [Chapter 5, "Viewing and Editing Function Point Data,"](#) discusses the various options for viewing and editing function point data.
- [Chapter 6, "Report Descriptions,"](#) describes and illustrates each report generated by Recap.

- [Chapter 7, "Report Generator Language."](#) provides ways to adjust or change report content and format to meet your organization's requirements using Recap's Report Generator Language.
- [Chapter 8, "Import Facility."](#) gives detailed information about the Import Facility and how it allows you to import an application definition into the AKR.
- [Chapter 9, "Export Metrics Facility."](#) guides you through the creation of files for export and for output from export.
- [Chapter 10, "Exporting an Application Definition."](#) provides information about copying the application definition, exporting it, and creating one similar to the original one.
- [Chapter 11, "AKR Utilities."](#) explains both online and batch AKR utilities.
- [Chapter 12, "Application Conversion."](#) states that to take advantage of the more accurate function point values of the more recent version of Recap, it will be necessary to do a full analyze of your applications.
- [Chapter 13, "Report Options."](#) summarizes the report options and product default for each option.
- [Chapter 14, "Help Facility."](#) describes the various online Help facilities available and methods to access them.
- [Chapter 15, "Sample AKR."](#) provides information concerning the AKR with sample applications provided on the Recap installation tape.

Related Publications

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

- *ASG-Application Definition and Analysis User's Guide (ALL0200-*nn*)* describes defining and analyzing an application in ASG-Alliance, ASG-Recap, and ASG-Estimate.
- *ASG-Center Installation Guide (CNX0300-*nn*)* provides installation and customization procedures for ASG-Center, the common platform for all ASG-Existing Systems Workbench components. ASG-Center must be installed before installing ASG-Recap.
- *ASG-Recap Installation Guide (RCX0300-*nn*)* contains information on the installation and maintenance of ASG-Recap.
- *ASG-Recap User's Guide (RCX0200-*nn*)* provides instructions for using ASG-Recap for application definition and reports generation.

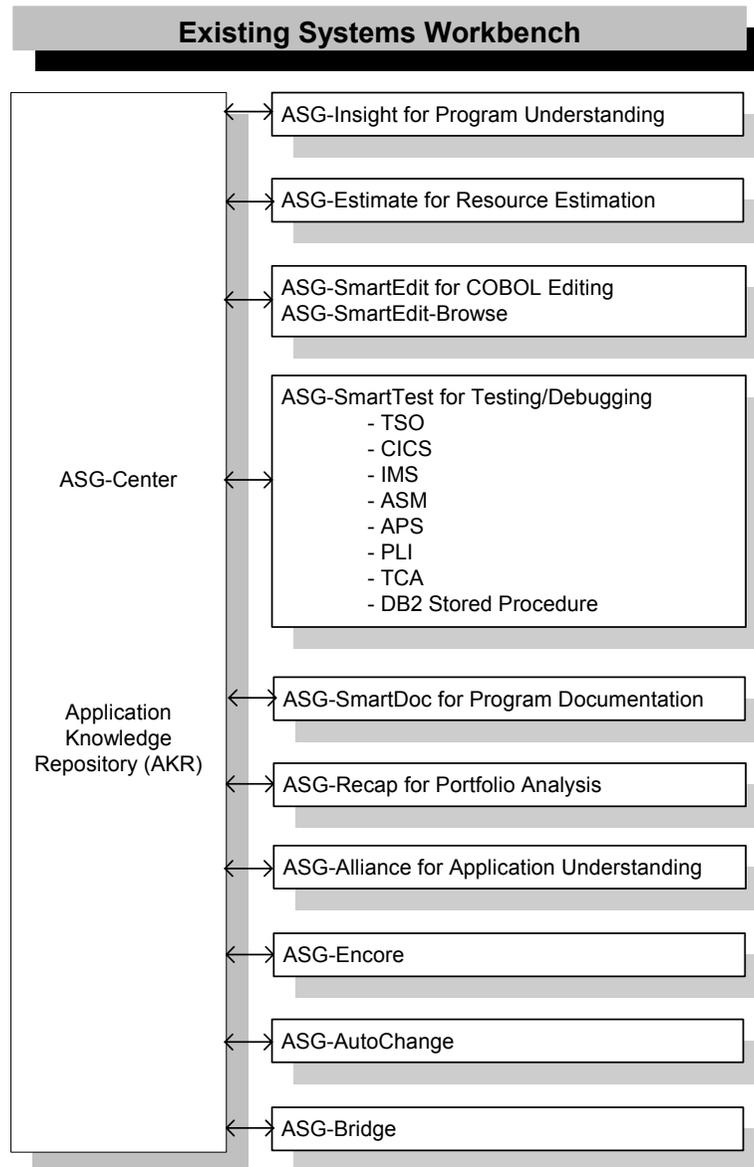
Note: _____

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

ASG-Existing Systems Workbench (ASG-ESW)

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

Figure 1 • ASG Existing Systems Workbench



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

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

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

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

Invoking ESW Products

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

The product names can also vary depending on whether you access a product directly or through ESW. See ["ESW Product Integration" on page xvi](#) for more information about using ESW.

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

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

ESW Product Integration

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

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

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

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

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

Figure 4 • File - Analyze Submit Screen

```

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

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

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

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

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

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

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

```

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

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

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

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

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

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

Publication Conventions

ASG uses these conventions in technical publications:

Convention	Represents
ALL CAPITALS	Directory, path, file, dataset, member, database, program, command, and parameter names.
Initial Capitals on Each Word	Window, field, field group, check box, button, 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 gives an overview of Recap, its interfaces, and operating system, and contains these sections:

Section	Page
Introduction	1
Recap Overview	2
Application Definition and Analysis	2
User Interface	4
Interfaces	4
Operating Systems	4
Printers	5
COBOL Compiler Support	5
Preprocessor Support	5

Introduction

Recap is the portfolio analysis component of ESW used to evaluate 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 be viewed interactively, or can also be exported to a CDF format and downloaded to your PC to be used in a database, spreadsheet, or graphics package.

Recap Overview

Recap provides IT professionals with an automated tool to conduct a portfolio analysis of every application in their environment. Using industry-accepted measures, Recap analyzes an application and provides:

- Metrics data
- Function point analysis
- Program quality assessments
- Intra-application and inter-application comparisons and summaries
- The ability to track the impact of process improvements and specific program changes and enhancements

Recap provides an Import Facility that allows you to import an application definition generated using another ESW product.

Review information generated from a portfolio analysis from easy to read reports produced directly from the mainframe component of Recap, or export the analysis information into a CDF format that can be downloaded to your PC for use in spreadsheet, database, and presentation packages. In these various formats, Recap analysis data can support these management activities:

- Feasibility, selection, and cost benefit of redevelopment/reengineering projects
- Allocation and justification of resources
- Scheduling of new software development projects
- Scheduling of software maintenance and enhancement

Application Definition and Analysis

Prepare an application for analysis by defining the COBOL, Load, Module, JCL, CICS, and IMS libraries and members and the library and member attributes in Recap. Attributes include the copy libraries, procedure libraries, maclibs, and compile parameters needed to successfully compile the members in the application. The application definition is stored in the Application Knowledge Repository (AKR). After the application has been defined, it is ready to be submitted for analysis.

In the analysis process, the ESW Application Analytical Engine (AAE) gathers information about each program in the application, including program relationships, logic, data, and execution paths. This information is then used to calculate the metrics and perform the progressive, comparative, and statistical analysis calculations.

Output from the analyze job is stored in the AKR where it can be retrieved for online viewing, report generation, export to a database manager, spreadsheet, or presentation graphics package.

Recap Reports

Recap provides this set of reports used to organize and present portfolio analysis data:

Program Reports	Application Reports	Enterprise Reports
Metrics History	Function Points	Executive Summary
Comparison	Definition	Metrics
Progress	Metrics	Exceptions
	Exceptions	
	Comparison	
	Progress	

- Program reports present portfolio analysis data for specific programs and help track the impact of specific program changes.
- Application reports present portfolio analysis data for specific applications and compare programs within an application to identify candidates for restructuring or re-engineering.
- Enterprise reports present portfolio analysis data for all applications in your AKR. These reports provide an overview of all applications in the organization and assist in the development and implementation of a comprehensive process improvement program.

See ["Report Descriptions" on page 85](#) and ["Using Portfolio Analysis Data" on page 49](#) for more information about report content.

User Interface

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

A screen is a full-width display of information, usually containing a full or shortened action bar. Recap screens are modeled after TSO/ISPF screens.

An action bar is the line of keywords that displays at the top of a screen. Each keyword represents actions that may be performed on that screen. To display a pull-down showing the available actions, move the cursor to the desired keyword and press Enter.

A pull-down is the list of actions that displays when a keyword is selected from the action bar. On a pull-down, actions followed by an ellipsis (...) display a pop-up when selected. Actions not followed by an ellipsis activate the associated function when selected.

A pop-up is a window that appears as the result of selecting an action on a pull-down or pop-up, or as the result of entering certain commands. It is superimposed on your screen to allow entry of information for the requested action. Enter the desired data or option, and follow instructions on the pop-up to process the information.

Note: _____

See [online help](#) for more information on the action bar, screens, pull-downs, and pop-ups.

Interfaces

Recap is primarily an interactive online product that allows you to define and submit applications for analysis, display application definitions and analysis data, select and generate portfolio analysis reports, and to set up and maintain the AKR. Reports can be selected and generated at the same time as the analyze is submitted or at a later date. Recap also contains an export facility to export portfolio analysis data to database managers, spreadsheets, or presentation graphics packages.

Operating Systems

Recap runs under MVS and uses ISPF (Version 3.5 through 4.8) as its standard online user interface.

Printers

Recap supports most printers using a standard character set. Program parameters are available to allow the use of alternate characters where required. See ["Report Options" on page 223](#) for more information.

COBOL Compiler Support

Recap supports these versions of COBOL compilers:

Product	COBOL Compiler
Compiler	<ul style="list-style-type: none"> • COBOL/370 • COBOL II (including releases 3 and 4) • ANSI COBOL • CASE Generated COBOL • COBOL for MVS and VM • COBOL for OS/390

Preprocessor Support

Recap supports these preprocessor languages directly:

Product	Language
Processor	<ul style="list-style-type: none"> • Command-Level CICS • Command-Level DL/I • IDMS • SQL

Note:

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

2

Concepts

This chapter discusses the concepts and terms related to the operation of Recap and contains these sections:

Section	Page
Application Knowledge Repository (AKR)	7
Application Portfolio Analysis	8
Function Points	8
Metrics	17
Statistical Terms	21

Application Knowledge Repository (AKR)

The Application Knowledge Repository (AKR) is a BDAM or VSAM file organization. Various utilities are provided to allocate and maintain the AKR. Multiple AKRs may be allocated.

When an application is analyzed by the Application Analyzer, the resulting portfolio analysis information is stored in the AKR. Recap uses that information to generate the portfolio analysis reports.

Note: _____

For more information about the structure of the AKR, or maintaining the AKR, see ["Batch AKR Utilities" on page 190](#). For more information about the Application Analyzer, see the *ASG-Application Definition and Analysis User's Guide*.

Application Portfolio Analysis

Application portfolio analysis information is used to assess the quantity and quality of software development products. This information can then be used to develop process improvement initiatives or to track the impact of program changes and enhancements. Recap automates the portfolio analysis process and provides this information:

- Function point analysis
- Program exceptions
- Software quality and complexity metrics
- Metrics history
- Comparative analysis statistics
- Progress analysis statistics

This information is presented in enterprise, application, and program level reports. For example, at the Enterprise level the Enterprise Metrics report provides a summary of metrics information for the applications within the enterprise.

Function Points

A function point measures the external economic value of a software application. Function points measure the size of an application from your point of view; that is, from the logical view, not the physical. The more functionality an application provides, the higher its function point count, and therefore, the greater its value to the organization.

The advantage of using function points is that they are not dependent on the language or technical environment for an application; they rely solely on the functionality provided to you and the complexity of the application or program being measured. Function points can also be used as the basis for other metrics, including productivity (e.g., Function Points / Work Effort), quality (e.g., Number of Defects / Function Points), and financial metrics (e.g., Project Cost / Function Points).

Note: _____

The function point analysis process used in Recap is based on the standards established by the International Function Point User's Group (IFPUG). These standards are described in the *IFPUG Function Point Counting Practices Manual*.

Function Point Counting

Function Point Counting is the process used to calculate the function point value for an application. This section describes the information required for Recap analysis and the IFPUG Function Point Counting process that is the basis for the function point analysis performed by Recap.

Note: _____

See ["Building and Analyzing an Application" on page 23](#) and ["Viewing and Editing Function Point Data" on page 71](#) for details on using Recap to count function points.

Information Needed for Function Point Analysis

To populate the AKR with accurate data certain application components must be included in the application definition and analyzed. For correct analysis, the code exceptions and metrics need COBOL source. The function point and enhanced function point metric needs COBOL source and these components:

Component	Purpose
Job Execution Load Modules	Relates the PROGRAM-ID, from the COBOL source, to a CSECT in a load module. The load module name is related to the program executed in the JCL or the transaction executed in CICS or IMS.
JCL	Determines EIF/ILF, EI, and EOs. It also allows Recap to recognize temporary work files and eliminate possible duplicate ILFs and EIFs.
CICS FCT tables	Relates the FCT DD name to the COBOL program EXEC CICS I/O statement dataset. The FCT helps Recap determine whether the FD in the program should be an EIF, ILF, or transaction. If your site retains FCT information in a CSD file rather than the FCT tables, the CSD file must be included.
CICS BMS Map	Counts DETs based on how many fields are inputs/outputs to the map.
IMS PSBGEN	Determines whether a segment is an EIF or ILF. The PSBGEN PSB is related to the PSB referenced in the IMS statement found in COBOL source. The segment is also used to count DETs. The IMS Stage 1 definition contains transactions and their associated load module names. The IMS/DC load module name is usually the same as the PSB name. Therefore, this name also relates to the PSB referenced in the IMS statement.

Establishing the Boundaries of the Application

Before function point counting can begin, the scope of the application must be clearly defined to identify components that are internal to the application and those that are external.

Using an external specification or another appropriate document (such as a system flow chart), a boundary can be drawn that defines the application.

Counting Function Point Types

After establishing the application boundaries, the application components are assigned one of these five function point element types:

Function Point Element Type	Description
Internal Logical File (ILF)	Logical group of data maintained and utilized by the application. For example, application data (such as personnel information in a payroll application) usually corresponds to at least one ILF.
External Interface File (EIF)	Logical group of data that is utilized by the application, but maintained by another application.
External Input (EI)	Components that maintain (add, change, or delete) data on an ILF. For example, if a screen within the application contains actions that add, change, and delete information, the screen counts for three External Inputs.
External Output (EO)	Process that sends data or controls information outside the application boundary. For example, reports generated by the application and data formatted and processed for use by another application.
External Inquiry (EQ)	Functionality provided for queries of ILFs and EIFs.

For a more thorough description of the five function point element types, see the *IFPUG Function Point Counting Practices Manual*. Included in that manual are definitions, methods of identifying, and examples of each type.

How Recap Classifies Function Point Element Types

This table shows the entities that are classified as function point types and indicates which function point type each entity could be.

Entity	EI	EO	EQ	ILF	EIF
Files (batch)	X	X	X	X	X
CICS files	X	X	X	X	X
IMS segments				X	X
DB2 tables				X	X
IDMS records				X	X
CICS maps	X	X	X		
IDMS maps	X	X	X		
IMS formats	X	X	X		

IMS segments, DB2 tables, and IDMS records are classified according to their CRUD information. If there is Read only information, they are classified as EIFs. If there is at least Create, Update, or Delete information, they are classified as ILFs.

Batch files and CICS files with Read only information could be either EIs or EIFs.

Batch files and CICS files with at least Create with the Read, Update, or Delete become candidate ILFs. These candidate ILFs become either iteration files, extract files, or real ILFs. Iteration files are temporary files that gather information before they are saved in real ILFs. Extract files are temporary files used to generate reports.

Backward and forward tracing of all fields movements to the closest entities determines the final function point type classification of the entities. Backward tracing is useful for Create (Write) only entities that are likely to be EOs or EQs. Backward trace results of a field show the source fields and whether the field was involved in a calculation. Backward trace results also indicate what entities move to the EOs and EQs.

Forward tracing is useful for Read only entities that could be EIs or EIFs. Forward trace results of a field show destination fields and comparison fields and indicate the entities that the EIs and EIFs move to and what entities they are compared to.

The most dense candidate ILFs become real ILFs. The most dense ILFs contain more unique fields than iteration or extract files. Other candidate ILFs that move to real ILFs become iteration files. In the same manner, candidate ILFs that are from real ILFs become extract files.

Closure of iteration/extract files is done. The closure links the input of the iteration/extract files to the corresponding output.

Any read only files, maps, or formats with fields in real ILFs become EIs.

Any create (Write) only files, maps, or formats containing fields from real ILFs or EIFs become EOs (field contains a calculation) or EQs (field does not contain a calculation). An EIF is then identified if its fields are referred to by an EO or EQ. The constraint that EIFs are Read only is maintained.

More EIFs can be identified based on EIs. An EIF might be compared with an EI to do a validation process of the EI before it updates ILFs. Consequently, read only files that are compared with EIs are EIFs.

Determining Functional Complexity

When components are classified as function point element types, each component is assigned a functional complexity of Low, Average, or High. For ILFs and EIFs, functional complexity is based upon the number of associated Record Element Types (RETs) and Data Element Types (DETs). The functional complexity of an EI, EO, and EQ is based on the number of File Types Referenced (FTRs) and DETs. This is a description of the three factors influencing functional complexity:

- RETs are logical subsets (based on your point of view) of ILFs and EIFs. For example, if the application being measured contains an ILF that is comprised of customer and product information, that ILF would have two RETs. ILFs and EIFs that cannot be broken down are considered to have one RET.
- DETs are user recognizable, non-recurring fields residing in an ILFs or EIFs and used by an EI, EO, or EQ.
- An FTR is counted for each ILF maintained (for an EI only) or each EIF or ILF referenced during the processing of an EI, EO, or EQ.

For more detailed information on RETs, DETs, and FTRs, see the *IFPUG Function Point Counting Practices Manual*.

To determine the functional complexity of each component, use the appropriate complexity matrix. Each function point element type has a distinct complexity matrix. [Figure 1](#) shows an example of the ILF Complexity Matrix.

Figure 1 • Internal Logical File Complexity Matrix

Internal Logical File (ILF) Complexity Matrix			
	1 to 19 DET	20 to 50 DET	51 or more DET
1 RET	L	L	A
2 to 5 RET	L	A	H
6 or more RET	A	H	H

Legend: RET = Record Element Type
DET = Data Element Type

Functional Complexity: L = Low
A = Average
H = High

For example, assume that in the application being counted, eight ILFs are identified. This chart displays the complexity ratings for each of these ILFs, given the number of RET and DETs.

Complexity Ratings for Sample ILFs								
ILF #	1	2	3	4	5	6	7	8
# of RET	1	3	1	6	4	1	2	2
# of DET	15	30	58	37	60	24	18	28
Complexity Rating	L	A	A	H	H	L	L	A

The functional complexity for each identified occurrence of the other four function point element types is determined in a similar manner.

For more information on the Complexity Matrices and determining functional complexity, see the *IFPUG Function Point Counting Practices Manual*.

Computing the Unadjusted Function Point Value

Each complexity rating for a function point element type has an associated unadjusted function point value. For example, this table shows the unadjusted function point values for the Internal Logical File function type.

Functional Complexity Rating	Unadjusted Function Points
Low	7
Average	10
High	15

Note:

Unadjusted function point values are different for each function point element type. For example, the unadjusted function point values for the EIF element type are 5 (for Low), 7 (for Average), and 10 (for High). Likewise, different unadjusted function point values exist for EIs, EOs, and EQs.

To compute the unadjusted function point value for a function point element type, multiply the complexity rating's unadjusted function point value by its number of occurrences.

In the earlier example, the application being measured contained 3 Low ILFs, 3 Average ILFs, and 2 High ILFs. Therefore, the ILF contribution towards the application's unadjusted function point value is calculated as:

```
# of Low ILFs      * Unadjusted Function Point value for Low) +  
# of Avg ILFs     * Unadjusted Function Point value for Average) +  
# of High ILFs    * Unadjusted Function Point value for High)
```

Or

$$(3 * 7) + (3 * 10) + (2 * 15) = 81$$

The unadjusted function point total for each of the remaining function point element types (EIF, EI, EO, and EQ) is calculated in the same manner. After calculating the unadjusted function points for the five element types, the numbers are added to obtain the Unadjusted Function Points for the application.

For more information on calculating Unadjusted Function Points, see the *IFPUG Function Point Counting Practices Manual*.

Computing the Value Adjustment Factor (VAF)

The Unadjusted Function Points reflect the functionality and value of the application from your perspective. However, other factors that are often transparent to users also need to be considered to accurately assess the value, quantity, and size of an application. For example, the complexity of the application can be dependent on environmental, physical, or logistical problems that may be encountered while implementing, operating, or maintaining the application. IFPUG has established a set of 14 General System Characteristics (GSC) to further assess the general functionality and complexity of an application.

These are the General System Characteristics, defined by IFPUG:

- | | |
|-------------------------------|-----------------------|
| 1. Data communication | 8. Online update |
| 2. Distributed processing | 9. Complex processing |
| 3. Performance | 10. Reusability |
| 4. Heavily used configuration | 11. Installation ease |
| 5. Transaction rates | 12. Operational ease |
| 6. Online data entry | 13. Multiple sites |
| 7. End user efficiency | 14. Facilitate change |

Note: _____

See the ["Glossary" on page 241](#) for a definition of each General System Characteristic. For more information on specifying the General Systems Characteristics in Recap, see ["Computing the Function Point Value Adjustment Factor" on page 42](#).

Each GSC is evaluated on a scale of 0 to 5 in terms of its degree of influence (DI) on the application. The degree of influence assigned to each characteristic is based on this scale:

- 0 = Not present or no influence
- 1 = Incidental influence
- 2 = Moderate influence
- 3 = Average influence
- 4 = Significant influence
- 5 = Strong influence throughout

For example, if the application is entirely batch, a degree of influence of 0 is assigned to Online data entry. However, if over half of the transactions are interactive data entry, a degree of influence of 5 is appropriate.

IFPUG has guidelines for scoring the degrees of influence for each GSC. See the *IFPUG Function Point Counting Practices Manual* for details.

To compute the Value Adjustment Factor

Add the 14 degrees of influence together to arrive at the Total Degrees of Influence (TDI). Use this industry accepted algorithm to compute the Value Adjustment Factor (VAF) to determine the influence of the GSCs on the application:

$$\text{VAF} = (\text{TDI} * .01) + .65$$

In this equation, these two constants are used: .01 and .65. The first constant, .01, is used to express the VAF in terms of percentages; the second, .65, is a preliminary adjusting factor.

Computing the Adjusted Function Point Value

The Adjusted Function Point value is calculated as:

$$\text{VAF} * \text{Unadjusted Function Point Value (for the application)}$$

When applied, the VAF adjusts the Function Point total any where between a positive 35% or a negative 35%. If all 14 GSCs are given a value of 5, the function point total is increased by 35%; if all are given a value of 0, the function point total is reduced by 35%.

In Recap, the Adjusted Function Point Value is calculated when the application is submitted for analysis.

Computing the Enhancement Function Point Count

The Enhancement Function Point count measures the modifications to an existing application that add, change, or delete user function within the scope of the project. This measurement can be used to calculate the effort required to enhance an application. In Recap, this value is calculated during application analysis.

In contrast to the Adjusted Function Point value that quantifies the functionality of the application in its present state, the Enhancement Function Point value quantifies the net change in functionality since the application was last analyzed.

The Enhancement Function Point value is calculated using this formula:

$$EFP = (ADD + CHGA) * VAFA + DEL * VAFB$$

where:

EFP is the Enhancement Function Point count.

ADD is the Unadjusted Function Point count of functions added in the enhancement project.

CHGA is the Unadjusted Function Point count of functions modified by the enhancement project.

DEL is the Unadjusted Function Point count of those function that were deleted by the enhancement project.

VAFA is the Value Adjustment Factor of the application after the enhancement project.

VAFB is the Value Adjustment Factor of the application before the enhancement project.

Metrics

Software metrics provide information about complexity, program architecture, and software quality that indicate the condition or state of applications. This information can then be used to perform these tasks:

- Predicting the amount of effort necessary to maintain an application.
- Identifying in-house applications in need of improvement.
- Measuring performance degradation/improvement.
- Helping to establish quality standards, with particular emphasis on continuous improvement.
- Assisting in evaluating vendor software applications.

Recap calculates software metrics that measure the level of complexity, size, and structure in a program and application. These metrics are stored in the AKR containing the application. Recap reports that present the metrics results at the enterprise, application, and program level are also available.

Cyclomatic Complexity

Cyclomatic Complexity is a measure of the complexity and the ability to understand and maintain an application or a program. The Cyclomatic Complexity metric measures the number of logic paths in a program. This metric is based on the premise that the number of paths in a program determines how difficult it is to understand. As a general rule, a value of 10 is desirable for a module.

For example, the sample code block shown in [Figure 2](#) contains three logic paths; therefore, the Cyclomatic Complexity for the code block is 3. (This is a simplification of the Cyclomatic Complexity calculation.)

Figure 2 • Sample Code for Cyclomatic Complexity

```
1      START
2      IF VAR-A < CONSTANT1 THEN...
3          IF VAR-B < CONSTANT2
4              THEN...
5                  DO...
6          ELSE
7              DO...
8          ENDIF
9      ENDIF
      STOP
```

Control Variable Complexity

Like the Cyclomatic Complexity, the Control Variable Complexity metric is also a measure of the complexity and the ability to understand and maintain a program. This metric measures the number of logic paths and the number of control variables in a program. This metric is based on the premise that programs with equal flow paths, but more variables controlling the flow, are more difficult to understand and maintain than programs with fewer control variables.

For example, the code block in 3 contains three logic paths. In addition, there are two control variables: VAR-A and VAR-B. Adding the logic paths (3) to the number of control variables (2) yields the Control Variable Complexity for the sample code (5). (This is a simplification of the Control Variable Complexity calculation.)

Software Science Volume

The Software Science Volume metric measures the size of a program. This metric factors in data and variables (i.e., operands) and COBOL verbs (i.e., operators). This metric is based on the premise that the larger the application or program, the more difficult it is to understand and maintain.

Use this formula to calculate the Software Science Volume metric:

$$V = (N1 + N2) * \log_2 (n1 + n2)$$

where:

V equals the Software Science Volume.

$N1$ equals the total number of operators in the program.

$N2$ equals the total number of operands in the program.

$n1$ equals the number of distinct operators in the program.

$n2$ equals the number of distinct operands in the program.

Essential Complexity

Essential Complexity measures how well a program is logically structured. The Essential Complexity metric measures the number of GO TOs that are not GO TO exits. This metric is based on the premise that the higher the number of GO TOs that are not GO TO exits, the more unstructured the program. A value of 1 indicates that a program is perfectly structured. (This is a simplification of the Essential Complexity calculation.)

For example, the sample code block shown in [Figure 3 on page 20](#) contains four GO TOs, none of which represent a GO TO exit; therefore, the Essential Complexity for the code block is 4.

Figure 3 • Sample Code for Essential Complexity

```

PARAGRAPH-A.
    IF VAR-A < CONSTANT1 THEN
        GO TO PARAGRAPH-C
    ELSE
        GO TO PARAGRAPH-E
    ENDIF.
PARAGRAPH-B.
    .
    GO TO PARAGRAPH-D.
PARAGRAPH-C.
    IF VAR-B > CONSTANT2
    THEN
        GO TO PARAGRAPH-B
    ELSE
        NEXT SENTENCE
    ENDIF.
PARAGRAPH-D.
    .
PARAGRAPH-E.
    
```

Knots Count

The Knots Count metric measures how well a program is physically structured. Each control path is marked with a line (e.g., from an IF to its ENDIF, or from a GOTO to its destination); the intersections of the control paths are the knots. For example, in [Figure 4](#), there are seven intersections, or knots, of the control paths.

Figure 4 • Illustration of Knots Count

```

PARAGRAPH-A.
    +- IF VAR-A < CONSTANT1 THEN
    +--+----- GO TO PARAGRAPH-C
    | | ELSE
+-----+----- GO TO PARAGRAPH-E
    | +- ENDIF.
    |+ PARAGRAPH-B.
    +--+----- GO TO PARAGRAPH-D.
    | ++ PARAGRAPH-C.
    | | +- IF VAR-B > CONSTANT2 THEN
    | +--+----- GO TO PARAGRAPH-B
    | | ELSE
    | | NEXT SENTENCE
    | +- ENDIF.
+---- PARAGRAPH-D.
    .
    .
    .
+----- PARAGRAPH-E.
    
```

A knot count of 0 is desirable. A high knot count indicates a program control flow path that is too complex, which may be difficult to understand and to maintain.

Statistical Terms

Recap uses these statistical terms in reporting portfolio analysis information.

Average

Recap calculates the average for function point and metrics values. The average is calculated using this formula:

$$\bar{x} = \frac{\sum x_n}{n}$$

where:

\bar{x} is the average.

x is a metric value.

n is the number of applications or programs for which the average is being calculated. (Function point values are not calculated at the program level.)

For example, this would be the calculation for the average number of cyclomatic complexities for an application containing three programs with values of 38, 45, and 50:

$$\frac{38 + 45 + 50}{3} = 44.3$$

Standard Deviation

Recap calculates the standard deviation for function point and metrics values to provide a measurement that shows how these values are clustered or scattered relative to the average.

The Standard Deviation is calculated using this formula:

$$S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

where:

S is the Standard Deviation.

x is a function point or metric value.

n is the number of programs or applications for which the standard deviation is being calculated.

For example, this is the calculation for the standard deviation for function points in the preceding example:

$$4.9 = \sqrt{\frac{((38-44.3)^2 + (45-44.3)^2 + (50-44.3)^2)}{3}}$$

3

Building and Analyzing an Application

This chapter guides you through the procedures used to define an application and generate function point data, and contains these sections:

Section	Page
FP Task Manager	27
Gathering Information	28
Initiating a Recap Session	29
Verifying User Options	31
Allocating an AKR	37
Creating an Application	40
Defining Application Components and Attributes	42
Computing the Function Point Value Adjustment Factor	42
Analyzing the Application	44
Generating Recap Reports	44

This table summarizes the procedures described in this chapter. Page references are provided to access a subject quickly. However, if you are using Recap for the first time, ASG recommends that you perform the procedures in sequence. The FP Task Manager screen described in this chapter is used to provide an online view of the status of Recap tasks.

Task	Procedure (Subject Reference)
Gather information for application definition	Review operational job streams. Interview application experts and review documentation to locate: <ul style="list-style-type: none">• JCL• Libraries• Programs• IMS and CICS Tables (See "Gathering Information" on page 28.)
Start Recap	Method is site dependent. You usually access the product from a TSO or ISPF menu. (See "Initiating a Recap Session" on page 29.)
Verify User Options if this is the first time you are using the product.	On the Options pull-down select: <ul style="list-style-type: none">• Product Parameters (see "Product Parameters" on page 31.)• Product Allocations (see "Product Allocations" on page 32.)• Log/list (see "Log and List File Parameters" on page 33.)• Script File Allocations (see "Script File Allocations" on page 35.)• PF keys (see "PF Key Settings" on page 36.)
Allocate AKR	To allocate an AKR: <ol style="list-style-type: none">1 Select File ► AKR Utility.2 Type A to allocate the AKR. (See "Allocating an AKR" on page 37.)
Create the application	Select File ► New application. (See "Creating an Application" on page 40.)

Task	Procedure (Subject Reference)
Define application components and attributes	<p>On the Edit pull-down select:</p> <ul style="list-style-type: none">• COBOL definition• Load module definition• JCL definition• CICS definition• IMS definition <p>For detailed information about defining application components and attributes, see the <i>ASG-Application Definition and Analysis User's Guide</i>.</p>
Review the Application Definition	<p>Select View ► Current application definition.</p> <p>For detailed information about application definition, see the <i>ASG-Application Definition and Analysis User's Guide</i>.</p>
Modify the application definition as needed	<p>Use the action on the Edit pull-down to modify the application definition.</p>
Compute Value Adjustment Factor for Function Point counting	<p>On the Edit pull-down:</p> <ol style="list-style-type: none">1 Select Function points2 Select Edit GSC <p>(See "Computing the Function Point Value Adjustment Factor" on page 42.)</p>
Analyze the application	<p>Select File ► Analyze.</p> <p>For detailed information about application analysis, see the <i>ASG-Application Definition and Analysis User's Guide</i>.</p>

Task	Procedure (Subject Reference)
Review the analyze results and resubmit members for analysis if necessary.	<ul style="list-style-type: none">• Review the Analyze Summary Report in output DD VIAARPT.• Correct problems that caused a return code of 8 or higher during the analyze.• Identify discovered components:<ol style="list-style-type: none">1 Select File ► Open application.2 Select Edit ► Discovered Components.• For resolvable discovered components, add to the application definition if needed.• For unresolvable discovered components, update the application definition as needed.• Review Missing and Dead Components:<ol style="list-style-type: none">1 Select View ► Missing Components.2 Select View ► Dead Components.• Use the actions on the Edit pull-down to modify the application definition. <p>For detailed information about reviewing analysis results, see the <i>ASG-Application Definition and Analysis User's Guide</i>.</p>
Repeat the analyze, review, and refine as needed.	Use the File pull-down to analyze, and use the Edit and View pull-downs to review and refine as needed.

FP Task Manager

The FP Task Manager screen, shown in [Figure 5](#), serves as an online checklist to keep you informed of the status of the application definition, analyze, function point editing, and report tasks. Select View ► FP Task Manager to display this screen.

Figure 5 • Function Point Task Manager Screen

```

File  Options  Help
-----
Command ==> FP Task Manager JCLTEST
Scroll ==> CSR
AKR : 'VIARAP.TEST.AKR' 1 OF 13

Cmd : B(Bypass); S(Select Task); E(Explain/Help);

Task  Description  Status
-----
- 1 Edit Application Definition
- 2 Analyze Application
- 3 Apply Previous Function Type Edit
- 4 Preview Previous Function Type Edit that cannot be applied
- 5 Edit External Input
- 6 Edit External Output
- 7 Edit External Inquiry
- 8 Edit Internal Logical File
- 9 Edit External Interface File
- 10 Edit General System Characteristics
- 11 Preview Adjusted FP
- 12 Preview Enhancement FP
- 13 Generate Reports

```

You can also access the screens and pop-ups related to these tasks from this screen. Type the S (Select) command in the line command field next to the task you want to perform. When the task is completed, the status field for that task displays Completed. To bypass a task, type the B (Bypass) command in the line command field for that task. If a task does not need to be completed, N/A displays in the Status field.

Some tasks may require the completion of previous tasks. If the previous tasks have not been completed, a warning message displays alerting you to complete the previous tasks.

Gathering Information

Start the application definition process by gathering the information discussed in this section. You may be able to obtain much of this information by reviewing the compile JCL that is used most frequently in the application. Record this information on the Application Inventory Worksheets provided for you in the *ASG-Application Definition and Analysis User's Guide*. This information is required:

- The Application Knowledge Repository (AKR) where the application resides. You may need the AKR's volume serial number (if the AKR is not cataloged) or its password (if the AKR is password protected).
- The name of the application and a description. All applications defined to Recap must be named; you also have the option to further describe the application by providing a 40 character application description.
- The COBOL, Load Module, JCL, CICS, and IMS source libraries that contain programs or members to be defined to the application. For each source library, you need to know:
 - Its source manager (for example, PDS, CA-Panvalet, or CA-Librarian).
 - Members that are to be defined to the application (referred to as selected members). If you need to include many members, you can select members from the member list available in Recap while you are defining the application.
 - For each COBOL library:
 - The copy libraries used by the majority of members included in the application definition.
 - Compile JCL dataset and member used for compiling the library's programs.
 - Whether DB2 or IDMS calls are used when executing the library's selected programs.
 - The procedure libraries used by the majority of members to be included in the application definition for each JCL library.
 - The maclibs used by the majority of the members to be included in the definition for each CICS or IMS library that uses included source or macros.
- Gather the attribute information for each member within a library that requires attributes (analysis parameters, copy libraries, procedure libraries, maclibs, or IDMS information) different from those specified for the library.

After the necessary information has been gathered, you are ready to initiate your Recap session.

Verifying User Options

During Recap installation, options that define the operating environment for Recap are set to default values. The first time you use Recap, you should verify that these default options reflect the appropriate settings for your environment. This section describes these user options:

- [PF Key Settings](#)
- ["Product Allocations" on page 32](#)
- ["Log and List File Parameters" on page 33](#)
- ["Script File Allocations" on page 35](#)
- ["PF Key Settings" on page 36](#)

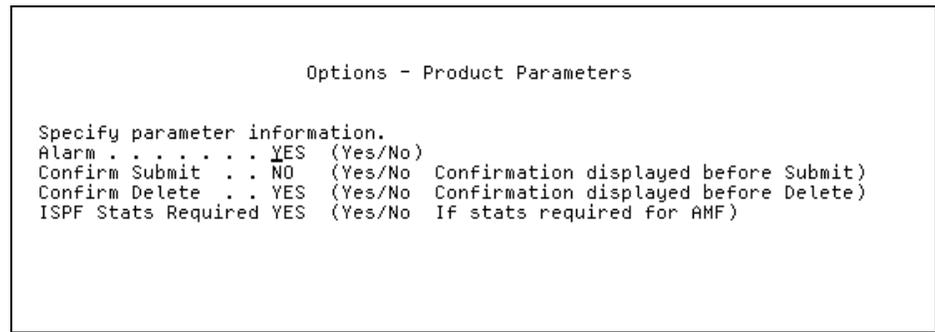
Product Parameters

Use the Options - Product Parameters pop-up, shown in [Figure 8](#), to specify alarm and confirmation settings.

To verify or change product parameter information

- 1 Select Options ► Product parameters to display the Options - Product Parameters pop-up.

Figure 8 • Options - Product Parameters Pop-up



- 2 Verify that the information on the Options - Product Parameters pop-up conforms to your needs. Make any desired changes. Field definitions for this pop-up are provided in online help.
- 3 Press PF3 to save your changes and exit the pop-up.

Product Allocations

Use the Options - Product Allocations pop-up, shown in [Figure 9](#), to verify or change the product allocations used to specify DASD information for the Log, List, and Work files.

To verify Recap product allocation settings

- 1 Select Options ► Product allocations to display the Options - Production Allocations pop-up.

Figure 9 • Options - Product Allocations Pop-up

```
Options - Product Allocations

Specify allocation information.

Log File:
Generic unit . . . SRTDA      (Generic group name or unit address)
Volume serial . . SRT801    (Blank for authorized default volume)

List File:
Generic unit . . . _____ (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 . . 050       (Space units)
Secondary space 010        (Space units)
Alternate node . . _____ (Alternate high level node)
```

- 2 Verify that the allocation information is correct for your site. Make any desired changes. Field definitions for this pop-up are provided in online help.
- 3 Press PF3 to save your changes and exit the pop-up.

Log and List File Parameters

Use the Options - Log/List Definition pop-up, shown in [Figure 10](#), to verify or change the settings for processing the Log and List files. Also use this pop-up to customize the dataset names and to specify print or delete options.

- 1 Select Options ► Log/list to display the Options - Log/List Definition pop-up.

Figure 10 • Options - Log/List Definition Pop-up

```

Options - Log/List Definition
Command ===> _____

1 - Process log file  2 - Process list file  3 - Customized data set name

Options              Log              List
-----              ---              ----
Process option      . . . . . PK              PK
Primary tracks      . . . . . 1              1
Secondary tracks    . . . . . 2              5
Lines per page      . . . . . 56             56
Sysout class        . . . . . *              *

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

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

```

- 2 Verify that the allocation and disposition choices conform to your site standards. Make any necessary changes. Field definitions for this pop-up are provided in online help.
- 3 If required for printing the Log or List file, enter the appropriate information in the Job statement information field.
- 4 Press PF3 to save your changes and exit the pop-up.

Note: _____

The Log file is only allocated if an internal error is encountered, such as an abend. It contains ESW error messages that can facilitate the debugging process. The List file contains printed output and is allocated when a request to print is issued.

To customize the Log or List dataset name

If you specify the K or PK process option, you can customize the dataset where the Log or List file is allocated. By default, Recap allocates the Log and List files as:

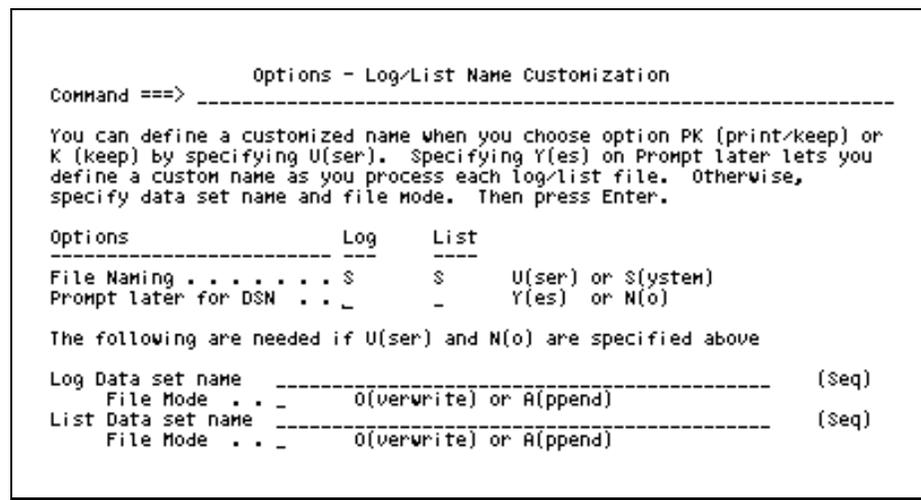
USERID.XEMnnnnn.VIAxxxxx

where *nnnnn* is a sequential number from 00001 to 99999 and *xxxxx* is LOG for Log and LIST for List files.

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

- 1 From the Options - Log/List Definition pop-up, type 3 and press Enter to display the Options - Log/List/Punch Name Customization pop-up, as shown in [Figure 11](#).

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



- 2 Type U in the File Naming field for Log or List to indicate a user-defined dataset name. If you specify N in the Prompt later for DSN field, you must enter a dataset name in the corresponding Data set name field, and specify Overwrite or Append in the File Mode field.

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

Figure 12 • 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

When Recap is installed, default script files are specified for the site. Use the Options -Script File Allocations pop-up, shown in [Figure 13](#), to display or change the default dataset names or the concatenation sequence for your session.

- 1 Select Options ► Script file allocations to display the Options -Script File Allocations pop-up.

Figure 13 • Options - Script File Allocations Pop-up

```

                                Options - Script File Allocations
Command ==> _____

R - Restore default Script allocations
Enter desired Script file concatenation.

Script file data set names:

_____
_____
_____
_____

```

- 2 Review the current default script file dataset names (when initializing for defaults, the lines are filled from the bottom to allow concatenation). Field definitions for this pop-up are provided in online help.
- 3 To make a change, enter the dataset names for the new script files in the Script file dataset names field. The files are concatenated in the order entered.
- 4 To restore the default, type R at the Command prompt.

- 5 Press PF3 to save your changes and exit the pop-up.

PF Key Settings

Use the Options - PF Key Definition pop-up, shown in [Figure 14](#), to verify or change current PF key assignments.

- 1 Select Options ► PF keys to display the Options - PF Key Definition pop-up. The current PF Key settings display.

Figure 14 • 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: 12      Terminal type: 3278

PF01 HELP
PF02 SPLIT
PF03 END
PF04 RETURN
PF05 RFIND
PF06 RCHANGE
PF07 UP
PF08 DOWN
PF09 SWAP
PF10 LEFT
PF11 RIGHT
PF12 CANCEL
```

- 2 To change the function of a PF Key, type over its current setting. If the PF Keys are not set to their default values, as shown in [Figure 14](#), change them now. Field definitions for this pop-up are provided in online help.
- 3 Press PF3 to save your changes and exit the pop-up.

Allocating an AKR

A Recap AKR should be established for each of the major business units (e.g., Accounting) in your company. Each AKR can contain multiple applications. If an AKR has not been allocated, you must use the File - AKR Utility pop-up, shown in [Figure 15](#), to allocate one. You can allocate as many AKRs as desired.

Note: _____

For information on the expected size of the AKR and analyze work files, see the *ASG-Application Definition and Analysis User's Guide*.

To allocate an AKR

- 1 Select File ► AKR utility to display the File - AKR Utility pop-up.

Figure 15 • File - AKR Utility Pop-up

```

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

Application Knowledge Repository (AKR):
Data set name . . 'USER.TEST_AKR'
Member . . . . . _____ (if "R" or "D" selected)
New name . . . . . _____ (if "R" selected)

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

```

- 2 Enter this information:
 - a Enter the name of the AKR to be allocated in the Data set name field. If the high level qualifier is different than your user ID, enter the name in single quotes (').
 - b Specify the allocate command by typing A in the command input area.
 - c If the AKR is not cataloged, enter the serial number of the volume where the AKR will reside in the Volume serial field.
 - d To password protect the AKR, enter the password in the Password field.

- 3 Press Enter to display the File - AKR Allocate/Expand pop-up, shown in [Figure 16](#).

Note:

The pop-up that displays depends on whether you use BDAM or VSAM AKRs and if SMS is turned on for your site. [Figure 16](#) assumes SMS with a VSAM AKR.

Figure 16 • File - AKR Allocate/Expand Pop-up with SMS and a VSAM AKR

```

File Edit View Facility Options Help
-----
File - AKR Allocate/Expand
-----
Command ==> _____
      S - Submit JCL      E - Edit JCL      C - Specify Catalog
Expand existing AKR . . . NO          (Yes or No)
AKR data set name . . . . 'USER.TEST.AKR'
Volume . . . . . _____
Unit . . . . . _____          (Generic unit name)
Space units . . . . . RECORDS      (Records, Tracks or Cylinders)
Primary space . . . . . 4000        (Primary amount in above units)
Secondary space . . . . . YES_      (Secondary amount in above units)

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

```

- 4 Enter this information:
 - a Specify NO in the Expand existing AKR field.
 - b If necessary, specify a management class for the new AKR in the Management class field. The default is MGMTCLAS.
 - c If necessary, specify a storage class for the new AKR in the Storage class field. The default is STORCLAS.
 - d In the Volume field, enter the volume serial number for the AKR. This field is required if Storage Class is not specified.
 - e If necessary, specify a data class for the new AKR in the Data Class field. Data Class is only valid if SMS is active at your site. The default is DATACLAS.
 - f In the Space Units field, enter the type of units to allocate. The units must be records, tracks, or cylinders. After specifying the unit type, enter the amount of space required in the Space amount field.
 - g In the Unique field, enter NO if the AKR is a VSAM data space.
 - h Specify a valid Jobcard in the Job statement information field.

- i** If you need to specify a private catalog or a password, type **C** in the command input area to display the AKR Catalog Information pop-up and enter the catalog dataset name and password.

Note: _____

For more information on the fields in the File - AKR Allocate/Expand pop-up, see online help.

- 5** Submit the JCL to allocate the AKR using one of these methods:

Method 1:

- a** Type **S** in the command input area.
- b** Press Enter.

Method 2:

- a** Edit the JCL by typing **E** in the command input area and pressing Enter.
- b** Make the appropriate modifications and issue the standard TSO ISPF SUBMIT command.
- c** Return to the File - AKR Allocate/Expand pop-up by issuing the END command or pressing PF3.

- 6** After the job completes, verify the allocation job by examining the job output. If the allocation was successful, condition codes of 0 should appear in the output.

Creating an Application

Note:

If you have created an application using a previous release of Recap, you need to convert the application before using it in the current release of the product. For information on converting applications, see ["Application Conversion" on page 221](#).

After allocating the AKR, you need to create the application and then specify the libraries and members to be included in the application. This section describes the steps used to create the application. See the *ASG-Application Definition and Analysis User's Guide* for information on defining the application.

In Recap, an application is a group of programs that perform a related function. Multiple applications are grouped together in an AKR that represents a single business unit such as Accounting. Typical applications within an Accounting AKR would be Accounts Payable, Accounts Receivable, General Ledger, Fixed Assets, Inventory, and Purchasing.

To create an application

- 1 Select File ► New application to display the File - New Application pop-up, as shown in [Figure 17](#).

Figure 17 • File - New Application Pop-up

```
File - New Application
Command ==> _____
Type AKR information, application name and analyze features.
Then press enter.

Application Knowledge Repository (AKR):
  Data set name . . . 'USER,TEST,AKR'
  Application name . . ABBEY-TM

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

Analyze features (Y/N):
ASG-Recap      : Y
ASG-Alliance   : Y
ASG-Estimate   : Y
```

- 2 Enter this information:
 - a Enter the AKR name in the Data set name field. If the high level qualifier of the name is different than your user ID, enclose the AKR name in single quotes (').
 - b Enter the Application name. The name can be up to ten (10) characters in length. For example, if you are defining a General Ledger application within an Accounting AKR, an appropriate application name is GEN-LEDG.
 - c If appropriate, enter a Volume serial or Password. The serial number of the volume where the AKR resides is required if the dataset is not cataloged; a password is required if the dataset is password protected.

Note: _____

If Alliance is installed, enter Y or N in the Features field Alliance option to specify whether you analyze the application using Alliance. ASG recommends that Recap and Alliance analysis be performed at the same time to optimize CPU cycles and DASD usage and avoid duplication of data.

- 3 Press Enter to redisplay the primary screen. You should see a message indicating the status of the application (e.g., *APPLICATION WAS SUCCESSFULLY CREATED*).

Note: _____

See the online help for more information on the File - New Application pop-up.

- 4 To further specify the application, enter a long description (up to 40 characters) to describe the application's business function. The long description displays on the Table of Contents.

To enter a long description

- 1 Select Edit ▶ Application description to display the Edit - Application Description pop-up.
- 2 On the Edit - Application Description pop-up, the user ID of the person that created the application displays in the Description field as the default description. To change the default, type over the existing text.
- 3 Press Enter to save the description and return to the primary screen.

After you successfully create your application, you can define the components and attributes that the application requires.

Defining Application Components and Attributes

Once an application is created, you must specify the application components. These include the COBOL, Load Module, JCL, CICS, and IMS libraries and members that you want to group for impact analysis.

For each library and member included in the definition, you may also need to specify attributes such as analysis parameters, copy libraries, procedure libraries, macro libraries, and IDMS information.

For complete information about defining application components and attributes, see the *ASG-Application Definition and Analysis User's Guide*.

Recap also provides an Import Facility to import application definition information from a file. For information on importing an application definition, see ["Import Facility" on page 159](#).

Computing the Function Point Value Adjustment Factor

When the application definition is correct, you should compute the function point Value Adjustment Factor (VAF) before you submit the application for a Recap analysis. The VAF is calculated from degrees of influence (DIs) assigned to the 14 General System Characteristics.

To assign DIs and calculate the VAF for an application

- 1 Select Edit ► Function Points to display the Edit - Function Points pop-up.
- 2 From the Edit - Function Points pop-up, select General System Characteristics to initiate the Function Points - General System Characteristics pop-up, as shown in [Figure 18](#).

Figure 18 • Function Points - General Systems Characteristics Pop-up

```
Function Points - General System Characteristics

Specify the degree of influence (0 to 5) for each of the system
characteristics. To accept the input press enter.

Data communication . . . . 0      On-line update . . . . . 0
Distributed processing . . 0      Complex processing . . . . 0
Performance . . . . . 0          Reusability . . . . . 0
Heavily used configuration 0      Installation ease . . . . . 0
Transaction rates . . . . . 0     Operational ease . . . . . 0
On-line data entry . . . . . 0    Multiple sites . . . . . 0
End user efficiency . . . . . 0   Facilitate change . . . . . 0
```

- 3 On the Function Points - General System Characteristics pop-up, assign the DIs for each of the 14 GSCs.
- 4 Press Enter to calculate the Total Degrees of Influence (TDI) and VAF. The values are placed in the appropriate fields.

If you analyze an application without first assigning DIs to the General System Characteristics, the Recap job uses default values of zero (0) for each GSC. This yields a VAF of .65; the result is an Adjusted Function Point value that is 35% less than the unadjusted amount. This may not be an accurate assessment of the application's functionality and complexity.

By assigning the DIs before the analyze process, the VAF more accurately reflects those additional environmental, physical, and logistical problems that may be encountered while implementing, operating, or maintaining your application. The baseline data that is generated from the Recap analysis more accurately reflects the application's true business value.

Note: _____

An overview of function points, including a discussion of the General Systems Characteristics, is provided in ["Function Points" on page 8](#).

On the Function Points - General System Characteristics pop-up, if the application has been analyzed, an asterisk displays next to the system characteristic when Recap determines that the application may contain that characteristic.

This table contains the characteristics found by Recap:

Characteristic	Description
Data Communication	Communication Section exists in the COBOL source.
Performance	CICS, DB2, DL/I, IDMS environments found in the COBOL source.
Transaction Rate	CICS, DB2, DL/I, IDMS environments found in the COBOL source.
Data Entry	CICS environment found in the COBOL source.

Characteristic	Description
Online Update	CICS environment found in the COBOL source.
Complex processing	Cyclomatic complexity/Adjusted Function Point > 100 *.

Note: _____
* The installation option Complex-Process-Threshold can be used to change the default value of 100. See your Systems Administrator for details specific to your site.

Analyzing the Application

After defining the application's libraries and members, specifying library and member attributes, and calculating an accurate VAF, the application can be analyzed to generate the inventory analysis data, including function points and software metrics. The full suite of Recap reports cannot be produced until the application is analyzed.

For complete information about analyzing applications, see the *ASG-Application Definition and Analysis User's Guide*.

Generating Recap Reports

After an application is analyzed, you can produce Recap reports. Recap reports provide progressive, comparative, and statistical analysis data related to function points, software metrics, and code exceptions at the program, application, and AKR (enterprise) level.

Note: _____

If you select the Run reports option on the Analyze - Submit Application pop-up, reports may also be produced during the application analysis.

If you are running Recap reports for the first time, see ["Helpful Hints" on page 47](#) for some suggestions that may assist you.

An application must be open for the Report process to be initiated.

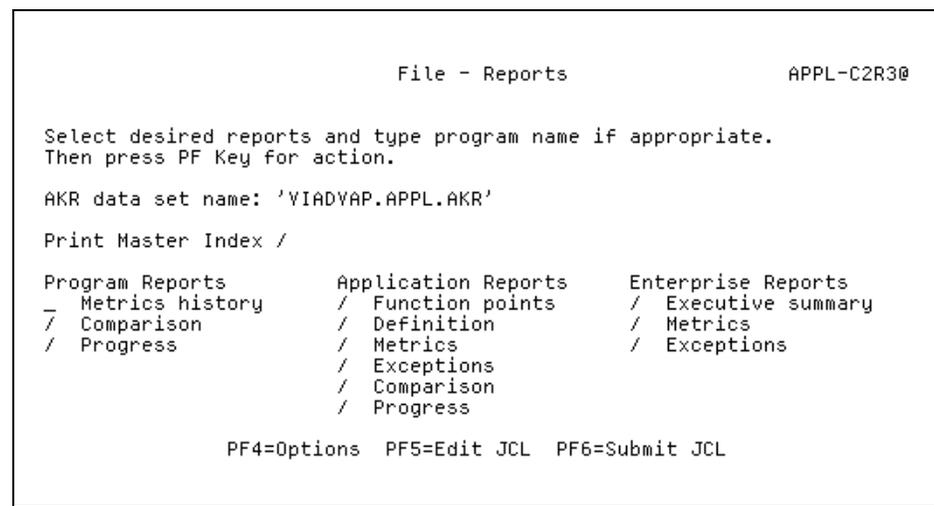
To open a Recap application

- 1 Select File ► Open application to display the File - Open application pop-up.
- 2 Enter this information on the File - Open Application pop-up:
 - a Enter the AKR name in the Data set name field. Enclose the AKR name in single quotes (') if the high level identifier is different from your user ID.
 - b Enter the name of the application in the Application name field.
 - c If appropriate, enter a Volume serial or Password. The serial number of the volume where the AKR resides is required if the dataset is not cataloged; a password is required if the dataset is protected. Ask your Systems Administrator if this information is required for the specified AKR.
- 3 Press Enter to open the application and redisplay the primary Recap screen. If the application was opened, this message displays: *APPLICATION [name of application] SUCCESSFULLY OPENED.*

After your application is open, perform these tasks to produce Recap reports:

- 4 Select File ► Reports to display the File - Reports pop-up, shown in [Figure 19](#).

Figure 19 • File - Reports Pop-up



- 5** Select the reports you want to generate by entering a non-blank character in the line command area to the left of each report name. In the previous figure, all reports except the Program Metrics history have been selected.

Note: _____

A detailed description of each report is provided in "[Report Descriptions](#)" on [page 85](#). Additional information on the File - Reports pop-up is also provided in online help.

- 6** If one of the selected reports is the Executive summary, press PF4 to display the Report Options pop-up and perform these actions.
- a** Select Executive Summary from the Report Options pop-up to display the Reports - Executive Summary pop-up.
 - b** On the Reports - Executive Summary pop-up, indicate the location of the input data file containing the report generator language for the Executive Summary Report; enter the dataset Name and Member name.
 - c** Press Enter. The Report Options pop-up redisplay.

Note: _____

The Executive Summary is printed in mixed case by default (member name is VIAXESUM). If you want to print the Executive Summary in upper case, select Executive Summary from the Report Options pop-up. The Reports - Executive Summary pop-up displays and you can specify the upper case data file (VIAXESUU) in the Member name field.

- 7** If necessary, choose Jobcard Information on the Report Options pop-up to display the Report - Job card information pop-up and perform these actions.
- a** Customize the jobcard information as appropriate.
 - b** Press Enter. The Report Options pop-up redisplay.
 - c** Select Exit to return to the File - Reports pop-up.

- 8 Press PF6 to submit the job. A message displays prompting you to close the application (reports cannot be run if the application remains open).

Note: _____

You can select PF5 to edit the JCL. Selecting the PF5 key also prompts you to close the application.

For an explanation of the PARM values in the REPORT step of the generated JCL, see ["Report Options" on page 223](#).

After editing the JCL, issue a TSO ISPF SUBMIT command to submit the report job. You may use the TSO ISPF END command to end the TSO edit session.

- 9 Press Enter. The application is closed and the report job is submitted.

Helpful Hints

Some helpful hints for running reports for the first time are listed below:

- After defining an application, it is a good idea to run the Application Definition report. This report shows the application's hierarchical organization, including its source libraries, programs, and any analyze attributes. By viewing this report, you can verify that you have properly defined the application. The Application Definition is the only Recap report that can be produced before the application has been analyzed.
- If you are running reports for an application that has just been analyzed for the first time, do not produce any progress or history reports (Program Metrics History, Program Progress, and Application Progress). Because the application has been analyzed only once, there is no progress data or history data available.
- If no other applications have been analyzed in the AKR, it is recommended that you not to produce the Enterprise Metrics, Enterprise Exceptions, and Application Comparison reports. These reports compare metric values and code exceptions among all the applications in the AKR. If there is only one application, these reports lose their effectiveness.

4

Using Portfolio Analysis Data

This chapter presents several scenarios to illustrate potential uses of Recap's function point and inventory analysis data, and contains these sections:

Section	Page
Portfolio Analysis Overview	49
Uses of Function Points	51
Using the Portfolio Analysis Data	53
Modifying the Executive Summary	69
Exporting Portfolio Analysis Data	70

Portfolio Analysis Overview

Today, many IT shops have implemented Quality and Productivity Improvement Programs to add value to their company by improving processes within their departments. These programs focus on improving the software applications used by the organization. But, how does an IT organization measure the effectiveness of software improvements?

To evaluate improvements to software applications, an organization should measure its applications at the beginning of any Quality and Productivity Improvement Program and at regularly scheduled intervals thereafter (e.g., monthly, quarterly, or semi-annually). The initial measurement establishes baseline data that provides a reference or starting point. The baseline data can help identify areas within an application or a development process that could be improved. Measurements taken at scheduled intervals provide progressive and statistical data concerning the applications. These measurements can help evaluate the success of the Improvement program. Measurements can also be taken at any point during an application's (or project's) life cycle (e.g., at the start of the project, mid-way through, or at the completion of the project).

In IT, two types of quality measurement are important: functional quality and technical quality. Functional quality is based on your point of view and focuses on how well a software application supports a business function. By assessing functional quality, an IT department can more effectively assess customer satisfaction. This enables IT to confirm that they are building the systems that meet their user requirements. Functional quality is typically measured through customer satisfaction surveys.

Note: _____

Functional quality is a subjective value and is not measured by Recap.

Technical quality focuses on how well an application is designed and implemented from a technical perspective. This includes the degree to which an application can be understood, maintained, and tested. Technical quality is measured through software metrics that provide objective, quantitative, and practical ways to measure the quality of software applications.

Because Recap provides MIS professionals with an automated inventory analysis tool to measure the technical quality of their software applications, Recap can be used to support a Quality and Productivity Improvement program. As part of its measurement process, Recap generates comparative, progressive, and statistical data related to function points and software metrics (at an enterprise, application, and program level). The inventory analysis data can be viewed by running the desired Recap reports. Management can then use this data to assist them in the decision making process.

Note: _____

An introduction to the software metrics that Recap computes is provided in ["Metrics" on page 17](#).

In this chapter, several scenarios are presented to illustrate potential uses of Recap's function point and inventory analysis data.

Note: _____

Recap's inventory analysis data can be used in more ways than those described in this chapter. To a large extent, how an organization uses the data depends on the goals of its Quality and Productivity Improvement Program.

Uses of Function Points

Function Point Analysis is a useful tool at many levels of management. At the topmost level, Function Point Analysis can provide Chief Executive Officers (CEO) with the current value of an organization's software and the savings realized by investing in new technology. At the Chief Information Officer (CIO) level, information such as trends in productivity and maintenance costs can be determined by using Function Point Analysis. A project manager can use Function Point Analysis for such activities as determining the feasibility of software projects, allocating resources, and scheduling maintenance and development projects.

This section presents a scenario that focuses on using Function Points at the Project Management level. As noted above, Function Points can be used by all levels of management. For a more comprehensive discussion on the uses of Function Points at different managerial levels, see the IFPUG publication, *Function Points as Assets -- Reporting to Management*.

Scenario

Your organization has just completed the requirements definition and external specifications for a Purchasing/Payables application. From the external specification, you estimate that the application's size is 1,800 Function Points. You also estimate that the project requires 12 months to complete. As project manager, you separate the project into three equal phases, each lasting four months and containing 600 Function Points. Six full-time information systems staff members comprise your project team.

The project's first phase is completed in the estimated four months. Recap is used to measure the application's size after this phase. After making the necessary adjustments, you arrive at an adjusted Function Point value of 640.

The second phase, however, is not completed until the tenth month. Again, you measure the application's size with Recap. After the second phase, the Function Point count now stands at 1,800. You want to know if the project can be completed within the next two months to remain on schedule.

Solution

- 1 Estimate the number of Function Points that remain. From the original estimate, 600 Function Points remain. In addition, the original estimate is one-third less than the actual amount. To obtain a more realistic estimate of the number of Function Points remaining, add one-third to the phase estimate of 600. This yields a value of 800 Function Points that remain.
- 2 Calculate the Function Points per man month:

Function Points/man month = Function Points completed / (# of persons * # of months)

Or

1,800 / (6 persons * 10 months) = 30 Function Points/man month

Based on the value of 30 Function Points per man month, you estimate that approximately 26.5 man months are required to complete the project (800/30). Given this fact, at a staffing level of six persons, the project will be completed approximately 2.5 months behind schedule:

26.5 man months / 6 persons = 4.4 months;
10 months + 4.4 months = 14.4 months;
14.4 months - 12 months = 2.4 months.

To complete the project on time (in two months), you need to increase staffing. To calculate the staffing level required, perform this calculation:

Man months remaining / Months remaining in original 12 month estimate

26.5 / 2 = 13.25.

The staffing level would have to be increased to 14 to complete the project within the next two months.

Using the Portfolio Analysis Data

This section presents several scenarios that illustrate how the portfolio analysis data generated by Recap could be used to assist IT management in decision making. The first scenario focuses on using the information to support a continuous improvement initiative. The second scenario illustrates how Recap can support decision making regarding re-engineering.

Continuous Improvement Scenario

Your IT department's goal is to improve the IT process through continuous improvement. Continuous improvement (in this case) means that the IT processes are gradually refined (improved) so that deliverables (systems and service) are of higher quality (from both a technical and functional perspective).

The first step in accomplishing this goal is to measure. By measuring, you establish a baseline of information. Using Recap, you build definitions and analyze these applications: Appl-A, Appl-B, Appl-C, and Appl-D.

Once the analyze procedure is run, any Recap report is available. Three levels of reporting exist in Recap:

- Program
- Application
- Enterprise

At this point, you want to get a feel for the overall status of the enterprise; that is, you want to see how the four applications within the AKR compare to each other. This should give you insight on which applications are in good shape and those that are not. For this global look at the enterprise, you run the Enterprise Metrics and Enterprise Exceptions reports.

The Enterprise Metrics report summarizes the metrics for the application in the enterprise (AKR). [Figure 20](#) shows an example of this report.

Figure 20 • Sample Enterprise Metrics Report

ASG-RECAP-OS (ESA) Rx.x LVLxxx		ENTERPRISE METRICS								DD-MMM-YYYY HH:MM:SS PAGE 216			
		AKR NAME: ASG.RCPSAMP.AKR											
		NON-CONFORMANT											
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES CODE TOTAL	STRUCT	CMPLX	BUSNS VALUE	
APPL-C	60	791	0	1315573 HI	17646 HI	7445 HI	23067	19279 HI	34731 HI	LOW	HIGH	HIGH	
CONFORMANT													
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES CODE TOTAL	STRUCT	CMPLX	BUSNS VALUE	
APPL-A	60	667	0	999458	1250 LO	115 LO	2623 LO	70 LO	22985	HIGH	LOW	LOW	
APPL-B	48	625	0	977421	2790	453	6675	1481	42950	LOW	HIGH	LOW	
APPL-D	6	171	0	146795 LO	474	63	1241	28	6579 LO	HIGH	LOW	LOW	
ENTERPRISE TOTAL/SUMMARY				(NUMBER OF APPLICATIONS = 10, NUMBER OF PROGRAMS = 333)									
METRICS	TOTAL	AVERAGE	STD. DEV.	LOW BOUND	HIGH BOUND								
ADJUSTED FUNCTION POINTS	2928	293	96	197	389								
ENHANCEMENT FUNCTION POINTS	0	0	0	0	0								
SOFTWARE SCIENCE VOLUME	6304566	630457	283961	346496	914418								
CYCLOMATIC COMPLEXITY	35554	3555	4804	0	8359								
ESSENTIAL COMPLEXITY	12626	1263	2146	0	3409								
CONTROL VARIABLE	59677	5968	5954	14	11922								
KNOTS	35030	3503	5587	0	9090								
NUMBER OF LINES OF CODE	226518	22652	7116	15536	29768								
LEGEND: LO = VALUE < AVG. - STD. DEV. LOW = VALUE <= AVG. LOW BOUND = AVG. - STD. DEV. STD. DEV. = STANDARD DEVIATION HI = VALUE > AVG. + STD. DEV. HIGH = VALUE > AVG. HIGH BOUND = AVG. + STD. DEV.													

Appl-A stands out as being much better than the others. Its complexity (Cyclomatic Complexity and Control Variable metrics) and structure (Essential Complexity and Knots) metrics are flagged with a LO tag. A LO tag indicates the metric value is greater than one standard deviation less than the enterprise average for that metric. This implies that Appl-A is much less complex and much more structured than the other applications in the enterprise. Yet, Appl-A is one of the larger applications (as indicated by its Volume metric) and it provides the organization with a reasonable level of business value (even though its business value is rated as low).

Another application, Appl-C, stands out as being much worse than the others. Appl-C is listed in the Non-Conformant section of the report, which means that it has at least one metric (excluding Function Points) that is greater than one standard deviation from the enterprise average for that metric. Metrics that fall into this category are flagged with a HI tag. These HI tags could represent potential problem areas. Appl-C's size (Volume), complexity (Cyclomatic Complexity and Control Variable Complexity metrics), and structure (Essential Complexity and Knots metrics) metrics are all flagged with a HI tag. This implies that Appl-C is much more complex and less structured than the other applications.

The Enterprise Exceptions report summarizes code exceptions affecting the quality of the applications in the enterprise. The existence of these code exceptions in the application's programs may force the program to perform differently than expected. [Figure 21](#) shows an example of the Enterprise Exceptions report.

Figure 21 • Sample Enterprise Exceptions Report

ASG-RECAP-OS (ESA) Rx.x LVLxxx				ENTERPRISE EXCEPTIONS				DD-MMM-YYYY HH:MM:SS PAGE 217			
				AKR NAME: ASG.RCPSAMP.AKR							
APPLICATION NAME	NUMBER OF PROGRAMS	NUMBER OF GO TOS	NUMBER OF ALTERS	NUMBER OF A-ENTRIES	NUMBER OF A-EXITS	NUMBER OF RECURSION	NUMBER OF JUMPS	NUMBER OF LIVE EXIT	LINES OF DEAD CODE	LINES OF DEAD DATA	
APPL-A	60	92	0	60	66	0	0	0	16	160	
APPL-B	43	1602	91	69	110	64	143	3	1050	313	
APPL-C	43	1621	210	73	2	120	496	5	1191	404	
APPL-D	33	1675	49	60	95	6	129	3	1398	217	
ENTERPRISE TOTAL/SUMMARY			(NUMBER OF APPLICATIONS = 4, NUMBER OF PROGRAMS = 179)								
EXCEPTIONS			TOTAL								
-----			-----								
GO TOS			17693								
ALTERS			275								
ANOMALOUS ENTRIES			5								
ANOMALOUS EXITS			58								
RECURSIONS			3809								
OUT OF PERFORM RANGE JUMPS			12235								
LIVE EXITS			582								
DEAD CODE			10595								
DEAD DATA			1091								
LEGEND: JUMPS = OUT OF PERFORM RANGE JUMPS A-ENTRIES = ANOMALOUS ENTRIES A-EXITS = ANOMALOUS EXITS											

Appl-A has far less code exceptions than the other three applications. Appl-C has a larger number of code exceptions.

Other metrics that you collected (outside of Recap) also make Appl-A appear favorably when compared to the other applications. These additional metrics also point out the fact that Appl-C is very poor in comparison to the other applications.

[Figure 22](#) shows a listing of some of these additional metrics.

Figure 22 • Other Important Metrics for Scenario

Other Important Metrics				
	Application			
	A	B	C	D
Cost per Function Point	\$210	\$971	\$995	\$510
Function Points per Man-month	110	14	15	38
Defects per Function Points	3	18	21	12
Code Exceptions per Function Points	.007	.01	.11	.06
Customer Satisfaction *	9.1	6.2	4.8	7.1
* Evaluated on a scale from 10 (Extremely Satisfied) to 1 (Extremely Unsatisfied)				

After analyzing the data, you want more of the applications to be as structured and as easy to understand and maintain as Appl-A. After investigating why there is such a disparity between Appl-A and the other applications you find that:

- Appl-A is a new application. It served as a pilot project utilizing CASE generated COBOL and had a minor amount of RAD (Rapid Application Development); prior applications were developed with traditional manual COBOL lifecycles with few productivity aids.
- User involvement was extensive in the development of Appl-A. In prior applications, an emphasis was not placed on user involvement during design/prototyping.

Using this data, your department decides that all future development projects utilize the same technologies and approaches as Appl-A. It is also decided that Appl-C is in need of a major overhaul. Appl-C is very complex and very unstructured and costs a large amount of money to maintain. A team is assembled to evaluate which is the more reasonable course of action with regards to Appl-C: Redevelopment or replacement.

Reengineering Scenario

Your IT department is contemplating whether or not to undertake a re-engineering initiative (versus continuing in a maintenance mode). To a certain extent, this decision is based on the status of the programs in your organization's software portfolio. Applications or programs that provide high business value and that are high in complexity are ideal candidates for re-engineering.

You have responsibility for these IT applications: Accounts Receivable, Customer Billing, Customer Information, and Sales. The Director of IT has asked you to recommend programs in which re-engineering could be beneficial. Since this is your initial effort, the Director asked you to confine your search to those applications that are very complex and provide the organization with substantial value.

To gain an overview of the applications, you first review the Application Comparison report and the Enterprise Metrics report. These reports present comparison metric information at the application level.

The Application Comparison report consists of two sets of graphs (quadrant reports) and tables. One set compares the applications (in an enterprise) in terms of structure (Essential Complexity) and complexity (Cyclomatic Complexity); the other, in terms of business value (Function Points) and complexity (Cyclomatic Complexity).

The graphs give you a bird's eye view of the overall structure/complexity and business value/complexity of the applications. The asterisks (*) represent one or more applications that have metrics at the X-Y coordinate. The area enclosed by the box represents all the coordinates that fall within one standard deviation of each of the variables being measured (either structure and complexity or business value and complexity). The two horizontal lines of the box represent plus and minus one standard deviation for the Y-axis variable. The two vertical lines of the box represent plus and minus one standard deviation for the X-axis variable. Each variable's average is represented by the line stretching from the axis to the edge of the graph. The intersection of the two lines (averages) forms the four quadrants.

The tables provide the supporting detail (i.e., the name of the applications) for the graphs.

The next four figures illustrate the Application Comparison report. [Figure 23 on page 58](#) shows the structure/complexity graph with [Figure 24 on page 59](#) presenting the table associated with the graph; [Figure 25 on page 60](#) shows the business value/complexity graph with [Figure 26 on page 61](#) presenting the table associated with that graph.

Figure 23 • Comparison - Essential Complexity versus Cyclomatic Complexity

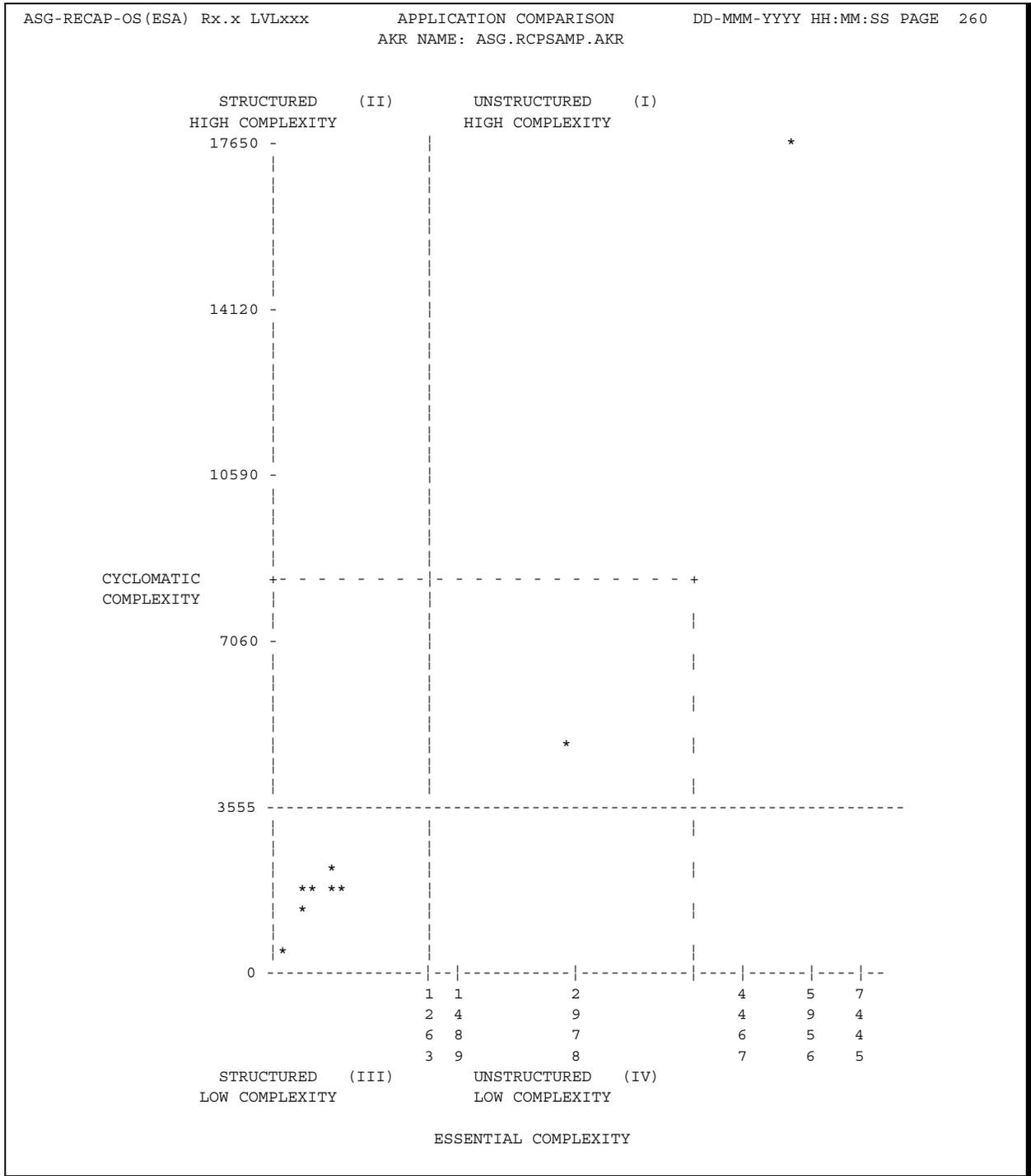
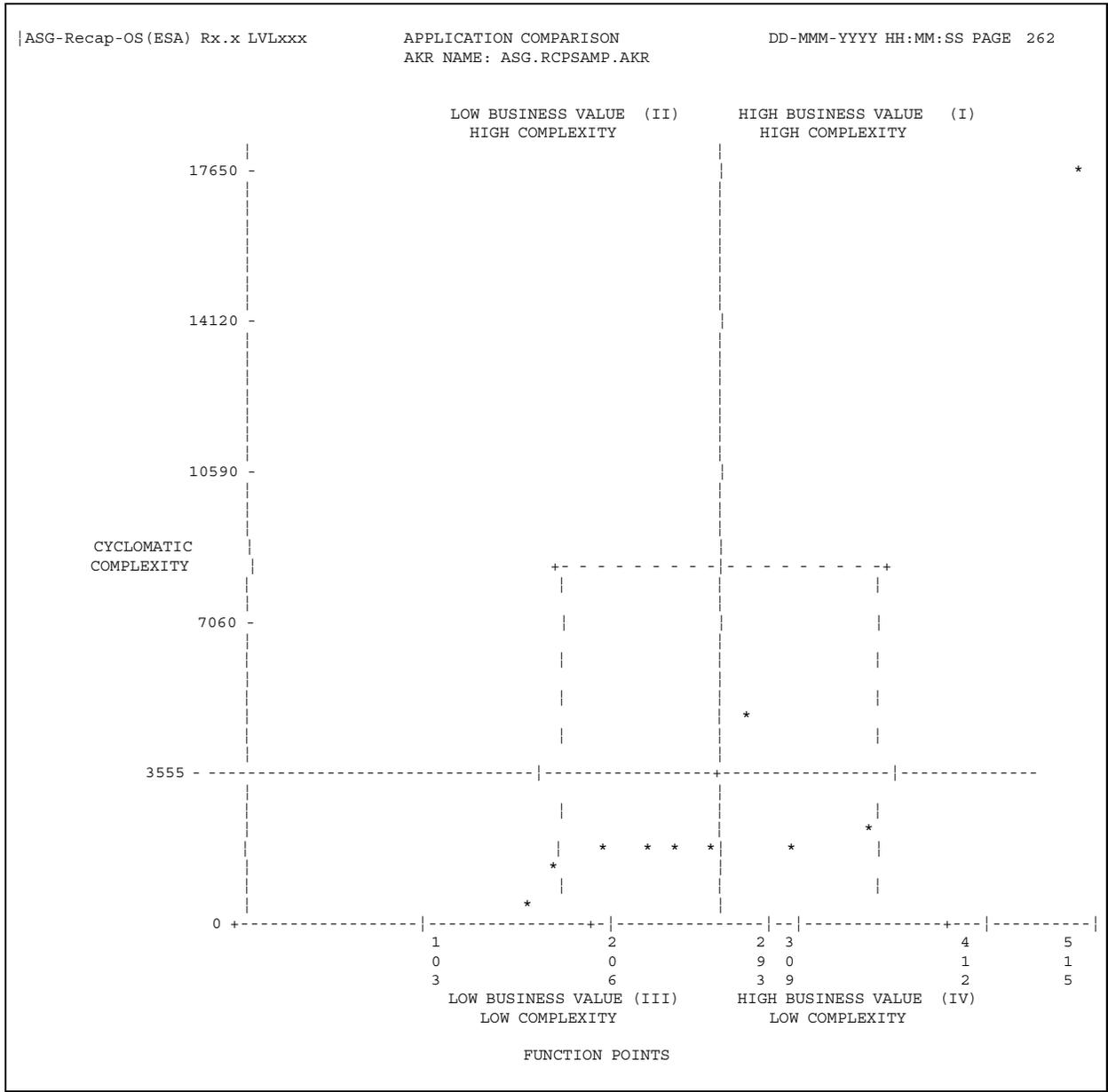


Figure 24 • Comparison - Essential Complexity versus Cyclomatic Complexity

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION COMPARISON		DD-MMM-YYYY HH:MM:SS PAGE 261	
AKR NAME: ASG.RCPSAMP.AKR					
QUADRANT I (UPPER RIGHT)	QUADRANT II (UPPER LEFT)	QUADRANT III (LOWER LEFT)	QUADRANT IV (LOWER RIGHT)		
UNSTRUCTURED-HI COMPLEX	STRUCTURED-HI COMPLEX	STRUCTURED-LO COMPLEX	UNSTRUCTURED-LO COMPLEX		
-----		-----			
MANF-CNTL (2317, 4729)		ACCT-PAYBL (225, 1542)			
SALES (7445, 17646)		ACCT-RCVBL (495, 1917)			
		CUST-BILL (474, 1853)			
		CUSTOMER (266, 1568)			
		EMPL-INFO (342, 1858)			
		INVENTORY (546, 1946)			
		PAYROLL (453, 2021)			
		TELEPHONE (63, 474)			
F					
FGRAPH COORDINATES = (ESSENTIAL COMPLEXITY, CYCLOMATIC COMPLEXITY)					

Figure 25 • Comparison - Function Points versus Cyclomatic Complexity



You are primarily interested in those applications that provide a high business value and are highly complex. Of the four applications you are responsible for, only one, Sales, falls into this category (see [Figure 25](#) or [Figure 26 on page 61](#), Quadrant I). Notice also that Sales is well outside the standard deviation box. This signals that Sales is much more complex and provides far greater value than any of the other applications in the enterprise.

The other three applications you are responsible for are of comparable business value and complexity. These three applications fall within the standard deviation box.

With this in mind, you see the Enterprise Metrics report. This report summarizes the metrics for the applications in the enterprise.

Figure 26 • Enterprise Metrics Report

ASG-RECAP-OS(ESA) Rx.x LVLxxx				ENTERPRISE METRICS						DD-MMM-YYYY HH:MM:SS PAGE 269					
AKR NAME: ASG.RCPSAMP.AKR															
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES CODE TOTAL	STRUCT	CMPLX	BUSNS VALUE			
MANF-CNTL	36	310	0	882297	4729	2317	8524	7272	31698	HI	LOW	HIGH	HIGH		
SALES	31	514	0	1315573	HI	17646	HI	7445	HI	23067	HI	19279	HI		
CONFORMANT															
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES CODE TOTAL	STRUCT	CMPLX	BUSNS VALUE			
ACCT-PAYBL	36	188	0	511021	1542	225	3245	1070	19048	HIGH	LOW	LOW	LOW		
ACCT-RCVBL	43	288	0	605906	1917	495	4035	1241	23493	HIGH	LOW	LOW	LOW		
CUST-BILL	43	337	0	601125	1853	474	3891	1226	23289	HIGH	LOW	HIGH	HIGH		
CUSTOMER	33	266	0	536930	1568	266	3320	1097	20566	HIGH	LOW	LOW	LOW		
EMPL-INFO	35	250	0	558657	1858	342	3622	1092	21325	HIGH	LOW	LOW	LOW		
INVENTORY	34	221	0	531911	1946	546	4623	1502	22985	HIGH	LOW	LOW	LOW		
PAYROLL	36	383	0	614351	2021	453	4109	1223	22804	HIGH	LOW	HIGH	HIGH		
TELEPHONE	6	171	0	146795	LO	474	63	1241	28	6579	LO	HIGH	LOW		
ENTERPRISE TOTAL/SUMMARY				(NUMBER OF APPLICATIONS = 10, NUMBER OF PROGRAMS = 333)											
METRICS	TOTAL	AVERAGE	STD. DEV.	LOW BOUND	HIGH BOUND										
ADJUSTED FUNCTION POINTS	2928	293	96	197	389										
ENHANCEMENT FUNCTION POINTS	0	0	0	0	0										
SOFTWARE SCIENCE VOLUME	6304566	630457	283961	346496	914418										
CYCLOMATIC COMPLEXITY	35554	3555	4804	0	8359										
ESSENTIAL COMPLEXITY	12626	1263	2146	0	3409										
CONTROL VARIABLE	59677	5968	5954	14	11922										
KNOTS	35030	3503	5587	0	9090										
NUMBER OF LINES OF CODE	226518	22652	7116	15536	29768										
F															
LEGEND: LO = VALUE < AVG. - STD. DEV.				LOW = VALUE <= AVG.				LOW BOUND = AVG. - STD. DEV.				STD. DEV. = STANDARD DEVIATION			
F HI = VALUE > AVG. + STD. DEV.				HIGH = VALUE > AVG.				HIGH BOUND = AVG. + STD. DEV.							

Sales is listed in the Non-Conformant section of the report, which means that it has at least one metric (excluding Function Points) greater than one standard deviation from the enterprise average for that metric. Metrics that fall into this category are flagged with a HI tag. These HI tags could represent potential problem areas. Sales's size (Volume), complexity (Cyclomatic Complexity and Control Variable Complexity metrics), and structure (Essential Complexity and Knots metrics) metrics are all flagged with a HI tag. This implies that Sales is much more complex and far less structured than the other applications.

Sales is rated as High in Complexity and Business Value. The High and Low ratings assigned to Structure, Complexity, and Business Value are relative assignments. For example, those applications rated as Low in complexity are not necessarily easy to understand (i.e., not complex). Instead, a Low complexity rating should be interpreted as meaning that in comparison to the other applications in the AKR, those applications rated as Low are not as complex as those applications rated as High.

The assignment of the Highs and Lows is dependent on whether the application's metric is above (High) or below (Low) the enterprise's average for that metric. The use of High and Low is consistent with the Application Comparison report. That is, if an application is in Quadrant I of the Application Comparison report, it is listed as High for business value (or structure, depending on which version of the graph you use) and complexity on the Enterprise Metrics report.

Based on the information from these two reports, it is worthwhile to investigate the Sales application further. Sales has a very high business value (almost two times greater than the next closest application in the AKR), and its Volume, Cyclomatic Complexity, Essential Complexity, Control Variable, and Knots metric values are at least one standard deviation greater than the enterprise average. The other three applications in your control fall within the standard deviation box, indicating that they have average values for their metrics. Although their Cyclomatic Complexity metrics are not ideal, they are over 10 times less than the Cyclomatic Complexity for Sales. Therefore, because you only want to look at applications that are the most complex, you decide to hold off on looking at these three applications in more detail.

To look at Sales more in-depth, the reports for the SALES application are run. This includes the Executive Summary, application level reports, and program level reports.

Review the Executive Summary first, shown in [Figure 27](#), which gives an overview of the application.

Figure 27 • Executive Summary (SALES Application)

ASG-RECAP-OS (ESA) Rx.x LVL:xxx	EXECUTIVE SUMMARY APPLICATION: SALES	DD-MMM-YYYY HH:MM:SS PAGE 1
GENERAL INFORMATION		
<p>The SALES (SALES INFORMATION) application inventory analysis was performed on DD-MMM-YYYY at HH:MM:SS. There is one historical version of the application. It consists of 37 analyzed programs. For additional details see the Application Definition Report.</p>		
APPLICATION COMPARATIVE ANALYSIS		
<p>The SALES application was compared against 9 other applications (see the Enterprise Metrics Report for the names of other applications).</p> <p>In comparison with the other 9 applications in this assessment, SALES has a high Adjusted Function Point value (687), an Enhancement Function Point value of 0, and at the same time it has many Lines Of Code (56128). This indicates a large application with good business value which may be hard to maintain. The programs within this application have a high number of lines of code per program (1517). SALES has a high Functional Complexity (27) (where Functional Complexity = Cyclomatic Complexity/Adjusted Function Points) and a high technical complexity as shown by the Cyclomatic Complexity (18537), indicating a conceptually difficult application. The Essential Complexity (7577) indicates that SALES has a poor logical structure. This correlates with the poor physical structure shown by the Knots value (19664). Of the 10 applications assessed, SALES has a relatively large number of Code Exception Conditions. See the Function Point Analysis, Enterprise Metrics and Enterprise Exceptions Reports for additional details.</p> <p>Due to the poor physical structure for an application with a high business value, we recommend that you restructure the application. In addition to the option of restructuring, the application could benefit from re-engineering because it has high technical complexity and poor physical structure, but provides good business value. The application contains a high number of exceptions that should be corrected to prevent flow problems (see PROGRAM COMPARATIVE ANALYSIS below).</p>		
PROGRAM COMPARATIVE ANALYSIS		
<p>Of the 37 programs in the application, 36 programs have code exceptions which may cause the programs to function differently than expected. We strongly recommend that you locate the program exceptions and correct them. See the Application Exceptions Report for additional information.</p> <p>Of the 37 programs, 2 programs are more than 1 standard deviation greater than the average for Software Science Volume, 2 programs are more than 1 standard deviation greater than the average for Cyclomatic Complexity, 2 programs are more than 1 standard deviation greater than the average for Control Variable Complexity, and 5 programs are more than 1 standard deviation greater than the average for Lines Of Code. If these programs provide important business functions, they may be candidates for re-engineering or decomposition. See the Application Metrics or Program Comparison Report for additional details.</p> <p>Of the 37 programs, 2 programs are more than 1 standard deviation greater than the average for Essential Complexity, and 1 program is more than 1 standard deviation greater than the average for Knots Count. If these programs show a history of difficult maintenance and they perform important business functions, they may be candidates for restructuring.</p> <p>No information is available to report how many programs have changed significantly.</p>		

The Application Comparative Analysis section discusses Sales in relation to the other applications in the AKR. This confirms that Sales may be a good candidate application for a restructuring/reengineering project.

The Program Comparative Analysis section discusses the programs within Sales. Indicated in this section are the number of programs within Sales that are greater than one standard deviation from the average (for the programs within Sales) for each of the software metrics. Recall that these programs could indicate potential problems.

Within the Program Comparative Analysis section, you are referred to additional reports that contain detailed information on the programs within Sales. The two of most interest to you are the Application Metrics and Program Comparison reports.

The Application Metrics report summarizes the metrics for the programs in the application. The format of this report is identical to that of the Enterprise Metrics report; the only difference is that this report concentrates on the programs within an application, while the Enterprise Metrics report focuses on the applications within the AKR.

The Application Metrics report can be used to identify candidate programs for redevelopment/reengineering projects. An ideal program is one that is high in business value and very complex.

HI tags are used in a similar manner as in the Enterprise Metrics Report. A HI value is one that is greater than one standard deviation from the application average. Those programs that contain a HI value metric are listed in the Non-conformant section of the report. The programs listed in the Non-conformant section are those that were briefly mentioned in the Executive Summary.

Two programs in Sales are very complex: KC601 and GENAMER1. One additional program has a Cyclomatic Complexity above 100, IX214. A value this large also indicates a program that is difficult to understand and maintain.

Now, you review the Program Comparison report. This report indicates each program's level of Structure/Complexity (Essential Complexity versus Cyclomatic Complexity). The format of this report is identical to the Application Comparison report; the only difference is that this report concentrates on the programs within an application, while the Application Comparison report focuses on the applications within an AKR.

Figure 30 • Comparison - Essential Complexity versus Cyclomatic Complexity

ASG-RECAP-OS (ESA) R.x.x LVLxxx		PROGRAM COMPARISON		DD-MMM-YYYY HH:MM:SS PAGE 240	
		APPLICATION: SALES		VERSION: 1	
QUADRANT I (UPPER RIGHT)	QUADRANT II (UPPER LEFT)	QUADRANT III (LOWER LEFT)	QUADRANT IV (LOWER RIGHT)		
UNSTRUCTURED - HI COMPLEX	STRUCTURED - HI COMPLEX	STRUCTURED - LO COMPLEX	UNSTRUCTURED - LO COMPLEX		
GENAMER1 (3107, 10865)		CM431 (1, 22)			
KC601 (4121, 5535)		DB204 (5, 31)			
		E02031 (8, 48)			
		IC110 (1, 1)			
		IC120 (1, 2)			
		IC207 (5, 33)			
		IC217 (1, 1)			
		IX203 (10, 52)			
		IX214 (45, 147)			
		NC110 (1, 2)			
		NC119 (5, 82)			
		NC154 (1, 43)			
		NC165 (1, 46)			
		NC209 (5, 54)			
		NC217 (5, 63)			
		NC431 (7, 70)			
		RL151 (3, 14)			
		RL431 (7, 31)			
		SG203 (15, 66)			
		SQ110 (25, 54)			
		SQ120 (13, 54)			
		SQ205 (13, 58)			
		SQ215 (5, 24)			
		ST110 (1, 1)			
		ST201 (9, 75)			
		ST211 (9, 49)			
		ST441 (5, 23)			
		TH205 (5, 50)			
		TH215 (5, 50)			

GRAPH COORDINATES = (FUNCTION POINTS, CYCLOMATIC COMPLEXITY)

The Program Comparison report confirms that KC601 and GENAMER1 are much more complex than the other programs in SALES. Because of their severe level of complexity and the fact that they are run often, you recommend KC601 and GENAMER1 as strong candidates for re-engineering.

You also recommend that program IX214 as a secondary candidate for re-engineering.

This scenario focused on the Sales application because it provided a high business value. However, while the other applications did not, on the whole, provide high business value (based on the Function Points), there may be programs within each application that could benefit from restructuring/reengineering (because the programs are highly complex). To find out if this is the case, each of the Application Metrics reports for the three other applications could be reviewed.

Modifying the Executive Summary

The Executive Summary presents high level information on the application, including a comparison to other applications and recommended actions. The Executive Summary can be changed to meet the specific needs of your organization by using the Report Generator Programming Language included with Recap. [Chapter 7, "Report Generator Language," on page 141](#) contains detailed information about this subject. This section presents a simple scenario that involves modifying the Executive Summary.

Scenario

In your MIS shop, an emphasis towards correcting code exceptions exists, with particular priority being placed on removing GOTOs and ALTERs. You want the Executive Summary to indicate if there is a high number of GOTOs or ALTERs.

To preserve the original Executive Summary code, make a copy of the Dataset Member that contains the code. You want to add a sentence in the first paragraph of the PROGRAM COMPARATIVE ANALYSIS section (immediately after the first sentence). The sentence added depends on if there is a high number of GOTOs, ALTERs, or both. The sentence should read *Of particular interest, there is a high number of . . .* The high number is determined by a Recap ranking. An explanation of this is included in [Chapter 7, "Report Generator Language," on page 141](#). If there is not a high number of GOTOs or ALTERs, no sentence should be added. This is the code added to accomplish:

```
&if &exception_rank(&gotos)>= 1 or &exception_rank(&alters) >= 1
Of particular interest,there is a high number of
&if &exception_rank(&gotos)>=1 and &exception_rank(&alters)>= 1
  GOTOs and ALTERs.
&else
  &if &exception_rank(&gotos) >= 1
    GOTOs.
  &else
    &if &exception_rank(&alters) >= 1
      ALTERs.
    &endif
  &endif
&endif
&endif
&endif
```

Exporting Portfolio Analysis Data

Recap provides a feature that creates a file used to export portfolio analysis data to PC database, presentation, and spreadsheet packages. A comma (,) and quote (" ") delimited file can be created, using the Export action on the File pull-down or the Export command from the Batch AKR utility, to export portfolio analysis data.

Detailed information on the Export Facility is described in ["Export Metrics Facility" on page 179](#).

5

Viewing and Editing Function Point Data

This chapter discusses the various options for viewing and editing function point data and contains these sections:

Section	Page
View Function Point Information On-line	71
View Function Point Analysis Data in Reports	73
Editing Recap Function Points	74
Adding Function Point Types	82
Modifying Existing Function Point Types	82
What if the Application is Reanalyzed?	83

View Function Point Information On-line

You can view adjusted and enhancement function point values online. From the FP Task Manager or the Edit - Function Points pop-up, select Preview Adjusted FP or Preview Enhancement FP to display the appropriate preview screen. These views can be printed by selecting File ► Print on each screen.

See online help for more detailed information on these screens.

Preview Adjusted FP - Summary

The Preview Adjusted FP - Summary browse-only screen presents a table weighing the complexity of the five function point types and shows the total Unadjusted Function Points value.

The Degree of Influence (DI) assigned to each general system characteristic and the resulting total is shown.

The calculations used to determine the Value Adjustment Factor (VAF) and Total Adjusted Function Points (FP) and their results are shown.

A detail view of the function points is available by selecting the View ► Detail on this summary screen. The detail view shows function point ranges for FTR, RET, DET, and complexity levels for each function point type.

Preview Enhancement FP - Summary

The Preview Enhancement FP - Summary browse-only screen presents tables weighing the complexity of function point types that have been added, changed, or deleted.

The DI assigned to each general system characteristic and the resulting total is shown for before and after function point enhancement.

The calculations used to determine the VAF and Total Enhancement Function Points (EFP) are shown. The VAF is shown before and after function point enhancement.

A detail view of the enhancement function points is available by selecting View ► Detail on this summary screen. The detail view shows function point ranges for FTR, RET, DET, and complexity levels for each function point type before and after enhancement.

View Function Point Analysis Data in Reports

The results of Recap's Function Point analysis can be viewed by running the Application Function Point Analysis report. If this report is not generated during the analyze procedure, you can produce it by using these steps (assuming that the application has been analyzed):

- 1 If the application is not already open, perform these steps (an application must be open for the Report process to be initiated):
 - a Select File ► Open application to display the File -Open Application pop-up.
 - b On the File - Open Application pop-up, type the name of the application in the Application name field and the AKR in which the application resides in the Data set name field.
 - c If appropriate, type the Volume serial or Password. The serial number of the volume where the AKR resides is required if the dataset is not cataloged; a password is required if the dataset is password protected. Ask your ESW Systems Administrator if this information is required for the specified AKR.
 - d Press Enter to open the application. You are returned to the primary Recap screen.
- 2 Select File ► Reports to display the File - Reports pop-up.
- 3 On the File - Reports pop-up, specify Function points in the Application Reports group.

Note: _____

If this is your first time running Recap reports, you may also have to customize your Jobcard information. This is accomplished by pressing PF4 and selecting Jobcard Information on the Report Options pop-up. See ["Generating Recap Reports" on page 44](#) for more information.

- 4 Press PF6 to submit the report job.

Note: _____

For descriptions and examples of the Application Function Point Analysis Report, see ["Report Descriptions" on page 85](#).

After viewing the results, you may want to change the method used to calculate function points. See ["Editing Recap Function Points" on page 74](#) for information on editing Recap function points.

Editing Recap Function Points

The method Recap uses to calculate function points is based on the *IFPUG Counting Practices Manual*. However, some variations exist between the Recap counting method and the IFPUG method. If you only plan to use the Recap function point data to make comparisons between applications analyzed using Recap, you probably do not need to adjust the function point data. Because Recap uses the same process to generate the function point count each time an application is analyzed, you can make a valid comparison between the function point data generated at different times or for different applications.

However, if you plan to compare the function point data generated by Recap to function point data generated by other means, such as a count calculated by a Certified Function Point Counter, you need to make these adjustments so that the Recap data conforms more closely with IFPUG standards:

- Eliminate work files and related transactions
- Remove duplicate EIFs and ILFs
- Review EIF and ILF classifications and reclassify if necessary

Procedures for making these adjustments are provided in the topics in this section.

Eliminating Work Files and Related Transactions

IFPUG standards do not include Work files in the function point count. If you have defined the application completely, including all JCL, load modules, CICS, and IMS components, Recap identifies work files and related transactions and does not include them in the function point count. However, if you have not defined the application completely, Recap cannot detect which files are work files. Consequently, the counts for these files and any related transactions (External Inputs, External Outputs, and External Inquiries associated with the file) need to be subtracted from the Recap function point count.

To remove a work file from the Internal Logical File classification (assuming the application is open)

- 1 Select Edit ► Function Points to display the Edit - Function Points pop-up.
- 2 Select Edit Internal Logical File to display the Adjust Function Points - Internal Logical File Type pop-up ([Figure 31 on page 75](#)).

Note:

You can also access the Adjust Function Points - Internal Logical File Type pop-up from the FP Task Manager screen by selecting Edit Internal Logical File.

Figure 31 • Adjust Function Points - Internal Logical File Pop-up

```

Adjust Function Points - Internal Logical File
Command ==>  Scroll ==> CSR
AKR : 'VIATTG.L3.AKR'
APPL : RC30-NEW                                1 OF 2
Select desired Element. Then press the PF key for action.
Data Element                                RET  DET  CPLX  MAN
-----
- VIAXE30.ILF1ST                            2-5  1-19  LOW   NO
- VIAXE30.ILF2ND                            2-5  1-19  LOW   NO
***** BOTTOM OF DATA *****

PF4=Move PF5=Modify PF6=Add PF10=Remove

```

- 3 Select entities that represent work files on this pop-up. These files are temporary or intermediate files used by a program to perform a technical task such as sorting a program. You can scroll the screen if you need to view all of the elements.
- 4 Press PF10 to remove the selected entities. If the Confirm Delete option on the Options - Product Parameters pop-up is set to YES, the Remove Function Points - Internal Logical File Type pop-up displays (see [Figure 32 on page 76](#)).

If the Options - Product Parameters pop-up is set to NO, the selected entities and its related transactions are deleted.

Figure 32 • Remove Function Points - Internal Logical File Type Pop-up

```
Remove Function Points - Internal Logical File Type

Press PF10 to remove the data element and its transactions, if desired.
Data Element: VIAXE30.ILF1ST
RET: 2-5  DET: 1-19  CMLPX: LOW  Options 2  1. Element & Transactions
                                           2. Element only

PF10=Remove  PF11=Browse Transactions
```

- 5 Select the Element & Transactions option and press PF10 to complete the removal process and return to the Adjust Function Points - Internal Logical File type pop-up when all entities are removed.

Note: _____

Press PF11 to display the Internal Logical File Type - Transaction List pop-up that shows a list of transactions related to the selected data element. This is a view-only pop-up.

- 6 Press PF3 to exit.

To remove a work file from the External Interface File classification (assuming the application is open)

- 1 Select Edit ▶ Function Points to display the Edit - Function Points pop-up.
- 2 Select Edit External Interface File to display the Adjust Function Points - External Interface File Type pop-up.

Note: _____

You can also access the Adjust Function Points - External Interface File Type pop-up from the FP Task Manager screen by selecting Edit External Interface File.

- 3 On the Adjust Function Points - External Interface File Type pop-up, select entities that represent work files. These files are temporary or intermediate files used by a program to perform a technical task such as sorting a program. You can scroll the screen if you need to view all of the elements.
- 4 Press PF10 to remove the selected entities. If the Confirm Delete option on the Options - Product Parameters pop-up is set to YES, the Remove Function Points - External Interface File Type pop-up displays.

If the Confirm Delete option on the Option - Product Parameters pop-up is set to NO, the selected entities and related transactions are deleted.

- 5 Select the Element and transactions option and press PF10 to complete the removal process and return to the Adjust Function Points External Interface File type pop-up when all entities are removed.

Note: _____

Press PF11 to display the External Interface File Type - Transaction List pop-up that shows a list of transactions related to the selected data element. This is a view-only pop-up.

- 6 Press PF3 to exit.

Removing Duplicate ILFs and EIFs

According to IFPUG guidelines, an ILF and EIF should only be counted once for the entire application. If you have defined the application completely, including all COBOL, JCL, Load Module, CICS, and IMS components, Recap identifies duplicate ILFs and EIFs and counts each unique ILF and EIF once for the entire application. However, if the application definition is not complete, Recap may not be able to identify duplicate files; consequently a file may be counted as an ILF and EIF for each program that uses the file. In this case, you need to review the EIFs and ILFs for each program to identify and eliminate any duplicates.

To remove duplicate ILFs

- 1 Access the Adjust Function Points - Internal Logical File Type pop-up from the Edit - Function Points pop-up or the FP Task Manager screen as described in ["Eliminating Work Files and Related Transactions" on page 74](#).
- 2 Select the entities representing duplicate files. You may need to scroll to view all the files. Press PF10 to display the Remove Function Points - Internal Logical File Type pop-up.
- 3 Select the Element only option and press PF10 to complete the remove process and return to the Adjust Function Points - Internal Logical File Type pop-up when all elements are removed.
- 4 Press PF3 to exit.

To remove duplicate EIFs

- 1** Access the Adjust Function Points - External Interface File Type pop-up from the Edit - Function Points pop-up or the FP Task Manager screen as described in ["Eliminating Work Files and Related Transactions" on page 74](#).
- 2** Select the entities representing duplicate files. You can scroll the screen if you need to view all the files. Press PF10 to display the Remove Function Points - External Interface File Type pop-up.
- 3** Select the Element only option and press PF10 to complete the remove process and return to the Adjust Function Points - External Interface File Type pop-up when all elements are removed.
- 4** Press PF3 to Exit.

Reviewing and Reclassifying ILF and EIF Elements

Recap classifies files as ILFs and EIFs using these rules:

- If the application contains WRITES to the file, the file is classified as an Internal Logical File.
- If the application contains only READs to the file, the file is classified as an External Interface File.
- If the same file is referenced multiple times within the application, Recap only counts the file one time for the application.

Recap relies on COBOL statements, JCL, Load Module, IMS, and CICS information in the application definition when applying the classification rules. However, in the context of an application, you may have files classified incorrectly because of other program or application information that Recap cannot interpret. Therefore, you need to review the file classifications and verify that they are correct. You can reclassify them if they are incorrect.

To reclassify an Internal Logical File as an External Interface File

- 1** Access the Adjust Function Points - Internal Logical File Type pop-up from the Edit - Function Points pop-up or the FP Task Manager screen as described in ["Eliminating Work Files and Related Transactions" on page 74](#).
- 2** Select the entities to reclassify as External Interface File Types.
- 3** Press PF4 to reclassify the entities and move them to External Interface File Types.
- 4** Press PF3 to exit.

To reclassify an External Interface File as an Internal Logical File, repeat these steps using the Adjust Function Points - External Interface File Type pop-up.

Reclassifying EO/EI Pairs to EQ

According to IFPUG standards, when a READ/WRITE combination functions as a query of an ILF or EIF, it should be counted and classified as an External Inquiry. Because Recap counts all READS as EOs and all WRITES as EIs, you may need to remove the EI/EO components of each query from the EI and EO list and add an External Inquiry to represent the READ/WRITE pair.

To record an element as an External Inquiry, you must delete the input portion of the query from the External Input type, delete the response portion of the query from the External Output type, then add the appropriate statement to the External Inquiry type.

To record an element as an External Inquiry

- 1** Review the Application Function Point Analysis Report to identify the EI/EO pair that comprises the query.
- 2** Determine which element has the higher function point count. The element with the highest function point count needs to be reclassified as an EQ and the other element needs to be removed from the function point count.
- 3** If the EI element has the higher function point count:
 - a** From the Edit - Function Point pop-up, select Edit External Input to display the Adjust Function Points - External Input Type pop-up (see [Figure 33 on page 80](#)).

Note: _____

You can also access the Adjust Function Points - External Input Type pop-up from the FP Task Manager screen by selecting Edit External Input.

Figure 33 • Adjust Function Points - External Input Type Pop-up

```

Adjust Function Points - External Input Type
Command ==> _____ Scroll ==> CSR
AKR : 'VIATTG.L3.AKR'
APPL : RC30-NEW                                1 OF 3

Select desired transaction. Then press the PF key for action.

Transaction ----- FTR DET CPLX MAN
- ADD VIAXE30.ILF1ST VIAXE30.ILF2ND          3+  1-4 AVG NO
- DELETE VIAXE30.ILF2ND                      0-1  1-4 LOW NO
- UPDATE VIAXE30.ILF1ST                      0-1  1-4 LOW NO
***** BOTTOM OF DATA *****

PF4=Move PF5=Modify PF6=Add PF10=Remove
    
```

- b** Select the transactions that represent the input portion of the query and press PF4 to display the Move - External Input pop-up.
- c** Select External Inquiry and press Enter to reclassify the EI as and EQ and return to the Adjust Function Points - External Input Type pop-up. The transaction and function point information is added to the EQ element list.
- d** Press PF3 to Exit.
- e** To remove the EO portion of the query:
 - From the Edit - Function Point pop-up, select Edit External Output to display the Adjust Function Points - External Output Type pop-up.

Note:

You can also access the Adjust Function Points - External Output Type pop-up from the FP Task Manager screen by selecting Edit External Output.

- Select the transactions that need to be removed and press PF10. If the Confirm Delete option on the Options - Product Parameters pop-up is set to Yes, the Remove Function Points - External Output Type pop-up displays. Press PF10 to complete the remove process and return to the Adjust Function Point - External Output Type pop-up when all transactions are removed.
- If the Confirm Delete option on the Options - Product Parameters pop-up is set to No, the transactions are removed from the Adjust Function Points - External Output Type pop-up.
- Press PF3 to Exit.

- 4 If the EO element has a higher function point count:
- a From the Edit - Function Point pop-up, select Edit External Output to display the Adjust Function Points - External Output Type pop-up.

Note: _____

You can also access the Adjust Function Points - External Output Type pop-up from the FP Task Manager screen by selecting Edit External Output.

- b Select the transactions that represent the output portion of the query and press PF4 to display the Move - External Output pop-up.
- c Select External Inquiry and press Enter to reclassify the EO as an EQ and return to the Adjust Function Points - External Input Type pop-up. The transaction and function point information is added to the EQ element list.
- d Press PF3 to Exit.
- e To remove the EI portion of the query:
 - From the Edit - Function Point pop-up, select Edit External Input to display the Adjust Function Points - External Input Type pop-up.

Note: _____

You can also access the Adjust Function Points - External Input Type pop-up from the FP Task Manager screen by selecting External Input.

- Select the transactions to be removed and press PF10. If the Confirm Delete option on the Options - Product Parameters pop-up is set to Yes, the Remove Function Points - External Input Type pop-up displays. Press PF10 to complete the remove process and return to the Adjust Function Point - Internal Output Type pop-up when all transactions are removed.
- If the Confirm Delete option on the Options - Product Parameters pop-up is set to No, the transactions are removed from the Adjust Function Points - External Input Type pop-up.
- Press PF3 to Exit.

Adding Function Point Types

To add a function point type

- 1 From the Edit Function Points pop-up or the FP Task Manager screen, select the Edit External Input, External Output, External Inquiry, Internal Logical File, or External Interface File task depending on the function point type you want to add.
- 2 The appropriate Adjust Function Point type pop-up displays. Press PF6 to display the appropriate Add Function Points pop-up.
- 3 Enter the transaction or data element you want to add.
- 4 Enter appropriate RET, DET, or FTR count. If you are adding an EI, specify whether it is counted as an Input or an Output.
- 5 Press PF6 to add the new function point type.
- 6 Press PF3 to return to the appropriate Adjust Function Point pop-up.

See online help for specific instructions related to each specific Adjust Function Point type and Add Function Point type pop-up.

Modifying Existing Function Point Types

To modify the complexity level of an existing Function Point type

- 1 From the Edit Function Points pop-up or the FP Task Manager screen, select the Edit External Input, External Output, External Inquiry, Internal Logical File, or External Interface File task, depending on the Function Point type you want to modify.
- 2 The appropriate Adjust Function Point type pop-up displays. Press PF5 to display the appropriate Modify Function Points pop-up.
- 3 Enter the appropriate new count for the RET, DET, or FTR. If you are modifying an EI, specify whether it is an Input or an Output.
- 4 Press PF5 to update the information and display the new complexity level.
- 5 Press PF3 to return to the appropriate Adjust Function Point pop-up.

See online help for specific instructions related to each specific Adjust Function Point type and Modify Function Points type pop-up.

What if the Application is Reanalyzed?

If you edit the Function Point data and later reanalyze the application, you do not have to perform the edits again. Recap stores the edits that you make; after you reanalyze the application, you can instruct Recap to automatically reapply any previous edits that you performed.

To apply previous Function Point edits to a reanalyzed application (assuming the application is open)

- 1 From the Edit - Function Points pop-up or the FP Task Manager, select Apply Previous Function Point Edits to display the Apply Previous Function Point edits screen.
- 2 Select File ► Apply Previous Edit to apply the previous function point edits to the new application analysis.

If any previously modified transactions are not found or a DET and RET (or FTR) changed in the reanalyzed programs, an exceptions list is produced. To view the exceptions list, select Preview Previous Function Type Edits that cannot be applied from the Edit - Function Points pop-up or the FP Task Manager screen to display the Preview Previous Function Type Edits that cannot be applied screen.

6

Report Descriptions

This chapter describes and illustrates each report generated by Recap and contains these sections:

Section	Page
Report Headings	87
Report Help	88
Table of Contents	89
Application Summary	91
Executive Summary Report	93
Application Definition Report	94
Application Function Point Analysis Report	97
Program Metrics History Report	107
Program Comparison Report	113
Application Metrics Report	117
Application Exceptions Report	120
Application Comparison Report	123
Application Progress Report	130
Enterprise Metrics Report	132
Enterprise Exceptions Report	135
Master Index	138

Application portfolio analysis information is used to assess the quantity and quality of an Information Service organization's software portfolio products. This information can then be used to: develop process improvement initiatives, track the impact of program changes and enhancements, evaluate re-development options, verify adherence to quality standards, evaluate vendor product quality, and cost justify development and maintenance staffing levels.

Recap automates the portfolio analysis process and generates a collection of reports that contain this information:

- Function point analysis
- Software quality and complexity metrics
- Enterprise/application/program exceptions
- Metrics history
- Comparative analysis statistics
- Progressive analysis statistics

All or selected reports can be generated simultaneously with an analyze job or at a later date. The report data can also be exported to a CDF [comma (,) and quote ("") delimited] file that can be used in database, spreadsheet, or graphic presentation packages.

See the *ASG-Application Definition and Analysis User's Guide* for more information about generating reports simultaneously with an analyze job.

See ["Export Metrics Facility" on page 179](#)) for more information on exporting information.

Report Headings

The Recap reports contain a standard heading unless otherwise specified, shown in [Figure 34](#).

Figure 34 • Report Heading Example

```

ASC-RECAP-OS(ESA) Rx. x LVLxxx (A)      PROGRAM COMPARISON (B)      DD-MMM-YYYY HH:MM:SS PAGE 1
      APPLICATION: SALES (C)      VERSION: 1 (G)      (D)      (E)
      PROGRAM: CH431 (F)

```

Report Field Descriptions

Field	Description
(A)	Product name, operating system, release number, and maintenance level.
(B)	Title of the report.
(C)	Name of the application for which the report was generated. On the application comparison, application progress, and enterprise reports, the name of the AKR containing the applications appears in this field.
(D)	Date and time the report was generated.
(E)	Page number of the report. The Table of Contents and the Application Summary are not page numbered.
(F)	On the Program Metrics History report, the name of the program for which the report was generated displays here.
(G)	Application version number assigned after the application was analyzed. This field displays in the headings only on these reports: <ul style="list-style-type: none"> • Program Comparison Report • Application Metrics Report • Application Exceptions Report

Report Help

All reports except for the Executive Summary begin with a help section providing definitions and usage notes. For example, the help section on the Program Metrics History report, shown in [Figure 35](#), explains the purpose of programs metrics and defines the metrics included on the report.

Figure 35 • Report Help Example

```
ASG-RECAP-OS (ESA) Rx.x LVLxxx          PROGRAM METRICS HISTORY          D-MMM-YYYY HH:MM:SS PAGE 210
                                APPLICATION: SALES

                                PROGRAM METRICS HISTORY HELP

SOFTWARE METRICS PROVIDE COMPLEXITY, PROGRAM ARCHITECTURE, AND
SOFTWARE QUALITY ASSESSMENTS THAT INDICATE THE CONDITION OR STATE
OR STATE OF EACH PROGRAM. TWENTY VERSIONS OF METRIC DATA ARE
RETAINED FOR EACH PROGRAM. THE PROGRAM METRICS HISTORY PROVIDES GRAPHS
FOR SOFTWARE SCIENCE VOLUME METRIC, CYCLOMATIC COMPLEXITY, ESSENTIAL
COMPLEXITY, CONTROL VARIABLE, AND KNOTS.

SOFTWARE SCIENCE VOLUME METRIC

SOFTWARE SCIENCE VOLUME METRIC IS PRIMARILY SIZE OR VOLUME METRICS BASED
ON THE PREMISE THAT THE LARGER THE PROGRAM, THE MORE DIFFICULT IT IS TO
UNDERSTAND AND MAINTAIN.

CYCLOMATIC COMPLEXITY METRIC

CYCLOMATIC COMPLEXITY METRIC MEASURES LOGICAL FLOW PATHS. THIS METRIC
IS BASED ON THE PREMISE THAT THE NUMBER OF PATHS IN A PROGRAM DETERMINE
HOW DIFFICULT IT IS TO UNDERSTAND.

ESSENTIAL COMPLEXITY METRIC

ESSENTIAL COMPLEXITY QUANTIFIES THE DEGREE OF LOGICAL STRUCTURE OF A
PROGRAM, I.E. THE DEGREE TO WHICH A PROGRAM HAS BEEN CONSTRUCTED USING
ONLY THE STANDARD STRUCTURED CONTROL FLOW CONSTRUCTS.

CONTROL VARIABLE METRIC

CONTROL VARIABLE METRIC MEASURES LOGICAL FLOW PATHS WITH THE NUMBER OF
PROGRAM CONTROL VARIABLES ALSO CONSIDERED. THIS METRIC IS BASED ON THE
PREMISE THAT PROGRAMS WITH EQUAL FLOW PATHS, BUT MORE VARIABLES
CONTROLLING THE FLOW, ARE MORE DIFFICULT TO UNDERSTAND AND MAINTAIN THAN
PROGRAMS WITH FEWER CONTROL VARIABLES.

CONTROL FLOW KNOTS

THE KNOTS METRIC IS A COUNT OF THE NUMBER OF INTERSECTIONS OF CONTROL
FLOW PATHS, HENCE IT QUANTIFIES THE DEGREE OF PHYSICAL STRUCTURE OF
A PROGRAM.
```

Table of Contents

The Table of Contents, generated for the reports, is divided into these sections if all reports are chosen. If only some of the reports are generated, the Table of Contents reflects the selection.

- I. Overview
 - 1. Executive Summary
 - 2. Application Definition
- II. Function Point Analysis
 - 1. Application Function Point Analysis
- III. Program Metric Information
 - 1. Program Metrics History
 - 2. Program Comparison
 - 3. Program Progress
- IV. Application Metric Information
 - 1. Application Metrics
 - 2. Application Exceptions
 - 3. Application Comparison
 - 4. Application Progress
- V. Enterprise Metric Information
 - 1. Enterprise Metrics
 - 2. Enterprise Exceptions
- VI. Master Index

Figure 36 shows the Table of Contents when all reports are generated.

Figure 36 • Table of Contents Report

```

ASG-RECAP-OS (ESA) Rx.x LVLxxx          TABLE OF CONTENTS          APPLICATION: SALES DD-MMM-YYYY HH:MM:SS

(A) *****
*
* APPLICATION DESCRIPTION: SALES INFORMATION
*
*****

          (B)                                (C)
I. OVERVIEW
  1. EXECUTIVE SUMMARY                      1
  2. APPLICATION DEFINITION                 2

II. FUNCTION POINT ANALYSIS
  1. APPLICATION FUNCTION POINT ANALYSIS    3

III. PROGRAM METRIC INFORMATION
  1. PROGRAM METRICS HISTORY               6
  2. PROGRAM COMPARISON                   8
  3. PROGRAM PROGRESS                     12

IV. APPLICATION METRIC INFORMATION
  1. APPLICATION METRICS                   20
  2. APPLICATION EXCEPTIONS               22
  3. APPLICATION COMPARISON               26
  4. APPLICATION PROGRESS                 30

V. ENTERPRISE METRIC INFORMATION
  1. ENTERPRISE METRICS                   33
  2. ENTERPRISE EXCEPTIONS               34

VI. MASTER INDEX                          35
    
```

Field Descriptions

Field	Description
(A)	Application Description specified on the Application - Description pop-up. The default is the TSO user ID where the application was defined.
(B)	List of reports generated for the application.
(C)	Page number on which the particular report begins.

Application Summary

Application Summary information ([Figure 37 on page 92](#)) is shown on the page following the Table of Contents and includes:

- Name of the AKR containing the application
- Name of the application
- Application version number
- Number of programs defined to the application
- Number of programs analyzed
- Number of Code Exceptions affecting application quality
- Metrics information for the application
- Parameters specified for the current execution of Recap
- Options in effect for the current execution of Recap

Figure 37 • Application Summary

```

ASG-RECAP-OS (ESA) Rx.x LVLxxx                A P P L I C A T I O N   S U M M A R Y                DD-MMM-YYYY HH:MM:SS

                                     (A)
*****
*
* AKR NAME . . . . . VIAXE20.PROD.SAMPAKR
* APPLICATION NAME . . . . . SALES
* VERSION . . . . . 1
* COBOL PGMS DEFINED . . . . . 37
* COBOL PGMS SUCCESSFULLY ANALYZED . . 31
*
* CODE EXCEPTIONS AFFECTING APPLICATION QUALITY:
* GO TOS . . . . . 3864
* ALTERS . . . . . 20
* ANOMALOUS ENTRIES . . . . . 1
* ANOMALOUS EXITS . . . . . 6
* RECURSIONS . . . . . 3794
* OUT OF PERFORM RANGE JUMPS . . . 10256
* LIVE EXITS . . . . . 571
* DEAD CODE . . . . . 1100
* DEAD DATA . . . . . 3
*
* METRIC INFORMATION:
*
* METRIC TYPE                -APPLICATION-  -----ENTERPRISE-----
*                               TOTAL          AVERAGE    STANDARD DEV
* -----
* ADJUSTED FUNCTION POINTS . . . . . 514          293          96
* ENHANCEMENT FUNCTION POINTS . . . . . 0           0            0
* SOFTWARE SCIENCE VOLUME . . . . . 1315573      630457      283961
* CYCLOMATIC COMPLEXITY . . . . . 17646       3555        4804
* ESSENTIAL COMPLEXITY . . . . . 7445        1263        2146
* CONTROL VARIABLE . . . . . 23067       5968        5954
* KNOTS . . . . . 19279       3503        5587
* NUMBER OF LINES OF CODE . . . . . 34731      22652       7116
*
*****

*****
*****
                                     (B)
ASG-RECAP WAS RUN WITH THE FOLLOWING PARAMETERS:
DS=SRENTTG,UN=SRTDA,APPL=SALES,PH,PF

                                     (C)
OPTIONS IN EFFECT: BANNER, COLON=., LINESPERPAGE=60, VCHAR=|, APLDEF, EXCSUM, PGMFPA, APLCMP, APLMTC,
APLPRG, APLFPA, APLEXC, PGMCMP, PGMHST, PGMPRG, ENTMTC, ENTEXC
    
```

Field Descriptions

Field	Description
(A)	Summary information, including the AKR and application name, application version number, number of programs defined to the application, number of programs analyzed, code exceptions, and metrics information showing the totals for the application metrics and the averages and standard deviations for the enterprise.
(B)	List of the parameters in effect for this execution of Recap.
(C)	Report options in effect for this execution of Recap.

Executive Summary Report

The Executive Summary Report provides a summary of the portfolio analysis information including general information about the analysis and a high level discussion of the results. This discussion includes recommended actions for the IT organization to take based on the reported results.

The contents and structure of the Executive Summary, shown in [Figure 38](#), can be changed to meet the specific needs of your organization by using the Report Generator Language included with the product. See [Chapter 7, "Report Generator Language," on page 141](#), for more information.

Figure 38 • Executive Summary Report

ASG-RECAP-OS (ESA) Rx.x LVLxxx	EXECUTIVE SUMMARY APPLICATION: SALES	DD-MMM-YYYY HH:MM:SS PAGE 1
(A) GENERAL INFORMATION		
The SALES (SALES INFORMATION) application inventory analysis was performed on DD-MMM-YYYY at HH:MM:SS. There is one historical version of the application. It consists of 37 analyzed programs. For additional details refer to the Application Definition Report.		
(B) APPLICATION COMPARATIVE ANALYSIS		
The SALES application was compared against 9 other applications (refer to the Enterprise Metrics Report for the names of other applications).		
In comparison with the other 9 applications in this assessment, SALES has a high Adjusted Function Point value (687), an Enhancement Function Point value of 0, and at the same time it has many Lines Of Code (56128). This indicates a large application with good business value which may be hard to maintain. The programs within this application have a high number of lines of code per program (1517). SALES has a high Functional Complexity (27) (where Functional Complexity = Cyclomatic Complexity/Adjusted Function Points) and a high technical complexity as shown by the Cyclomatic Complexity (18537), indicating a conceptually difficult application. The Essential Complexity (7577) indicates that SALES has a poor logical structure. This correlates with the poor physical structure shown by the Knots value (19664). Of the 10 applications assessed, SALES has a relatively large number of Code Exception Conditions. Refer to the Function Point Analysis, Enterprise Metrics and Enterprise Exceptions Reports for additional details.		
Due to the poor physical structure for an application with a high business value, we recommend that you restructure the application. In addition to the option of restructuring, the application could benefit from re-engineering because it has high technical complexity and poor physical structure, but provides good business value. The application contains a high number of exceptions that should be corrected to prevent flow problems (refer to PROGRAM COMPARATIVE ANALYSIS below).		
(C) PROGRAM COMPARATIVE ANALYSIS		
Of the 37 programs in the application, 36 programs have code exceptions which may cause the programs to function differently than expected. We strongly recommend that you locate the program exceptions and correct them. Refer to the Application Exceptions Report for additional information.		
Of the 37 programs, 2 programs are more than 1 standard deviation greater than the average for Software Science Volume, 2 programs are more than 1 standard deviation greater than the average for Cyclomatic Complexity, 2 programs are more than 1 standard deviation greater than the average for Control Variable Complexity, and 5 programs are more than 1 standard deviation greater than the average for Lines Of Code. If these programs provide important business functions, they may be candidates for re-engineering or decomposition. Refer to the Application Metrics or Program Comparison Report for additional details.		
Of the 37 programs, 2 programs are more than 1 standard deviation greater than the average for Essential Complexity, and 1 program is more than 1 standard deviation greater than the average for Knots Count. If these programs show a history of difficult maintenance and they perform important business functions, they may be candidates for restructuring.		
No information is available to report how many programs have changed significantly.		

Report Field Descriptions

Field	Description
(A)	General information about the application analysis including date and time of analysis, application version number, and the number of programs defined to the application.
(B)	Summary of the Application Comparative Analysis including: <ul style="list-style-type: none">• Number of applications compared• Interpretation of the portfolio analysis information in respect to the other applications• Recommended actions• References to other reports for additional information
(C)	Summary of the Program Comparative Analysis including: <ul style="list-style-type: none">• Number of programs compared• Interpretation of the portfolio analysis information in respect to other programs• Recommended actions• References to other reports for additional information• The amount programs have changed since the last analysis was performed

Application Definition Report

The Application Definition Report ([Figure 39 on page 95](#)) provides a hierarchical view of all libraries defined to the application. Each library is shown with its members/programs, copylibs, proclibs, compile parms, IDMS information, maclibs, and group names.

If a member/program has its own compiler information, the information is shown under that member/program and overrides the corresponding information specified for the library. See "[Building and Analyzing an Application](#)" on page 23 for more information about defining programs.

The Application Definition report also displays the user ID of the person who last modified the application definition and the date the modification was made.

Figure 39 • Application Definition Report

```

ASG-RECAP-OS (ESA) Rx.x LVLxxx                APPLICATION DEFINITION                DD-MMM-YYYY HH:MM:SS PAGE  2
                                           APPLICATION: SALES

(A)
LAST MODIFIED BY VIAMXM ON DD-MMM-YYYY AT HH:MM:SS
(B)
VIAXE21.PROD.SAMPAKR (SALES)
(C)
  LIBRARY (COBOL)
    ASG.COBOL.PANLIB - PAN
  (D)
    |--> ASM/COMPILE OPTION
    |      COBOL VERSION: COBOL74
    |--> MEMBER/PROGRAM-ID
  (E)
    |--> CM431/CM431
    |--> DB204/DB204
    |--> E02031/E02031
    |--> GENAMER1/GENAMER1
    |--> IC110/IC110
    |--> IC120/IC120
    |--> IC207/IC207
    |--> IC217/IC217
    |--> IX203/IX203
    |--> IX214/IX214
    |--> KC601/KC601
    |--> NC110/NC110
    |--> NC119/NC119
    |--> NC154/NC154
    |--> NC165/NC165
    |--> NC209/NC209
    |--> NC217/NC217
    |--> NC431/NC431
    |--> RL151/RL151
    |--> RL431/RL431
    |--> SG203/SG203
    |--> SQ110/SQ110
    |--> SQ120/SQ120
    |--> SQ205/SQ205
    |--> SQ215/SQ215
    |--> ST110/ST110
    |--> ST201/ST201
    |--> ST211/ST211
    |--> ST441/ST441
    |--> TH205/TH205
    |--> TH215/TH215

  LIBRARY (COBOL)
    ASG.CUST.COBOL - PDS
    |--> COPYLIB(S)
    |      ASG.COBOL.COPYLIB - PDS COBOL
    |      ASG.CUST.COBOL - PDS COBOL
    |      ASG.COBOL.COPYLIB2 - PDS COBOL
    |--> ASM/COMPILE OPTION
    |      COBOL VERSION: COBOLII R3
    |--> MEMBER/PROGRAM-ID
    |      |--> VS401/VS401
  (F)
    |      |--> COPYLIB(S)
    |      |      ASG.VS401.COPYLIB - PDS COBOL
    |      |      ASG.COBOL.COPYLIB - PDS COBOL
    |      |--> ASM/COMPILE OPTION
    |      |      COBOL VERSION: COBOL74
    |--> VS402/VS402
    |--> VS403/VS403
    |--> VS404/VS404
    |--> VS405/VS405
    |--> VS406/VS406

```

Report Field Descriptions

Field	Description
(A)	The user ID of the person who last modified the application definition and the date the modification was completed.
(B)	The name of the AKR and application.
(C)	Library name including source manager.
(D)	Copylibs, proclibs, compile parms, IDMS information, maclibs, or group names defined to the library.
(E)	Member name and Program ID as coded in the COBOL program.
(F)	Copylibs, proclibs, compile parms, IDMS information, or maclib defined to the member/program.

Application Function Point Analysis Report

The Application Function Point Analysis report shows an unadjusted function point value for the application. The report format consists of a table showing the total element type counts for the entire application, followed by tables showing the detail for each element type. The report also consists of the Total Unadjusted Function Point for the application and the General System Characteristics worksheet showing the general systems characteristics to calculate the adjusted function point value for the application.

Note: _____

See ["Concepts" on page 7](#) for more information about calculating unadjusted and adjusted function points for an application.

External Input Type Table

[Figure 40](#) shows an example of the External Input Type table.

Figure 40 • External Input Type Table

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION FUNCTION POINT ANALYSIS APPLICATION: SALES			DD-MMM-YYYY HH:MM:SS PAGE 120	
=====						
(A)						
EI - EXTERNAL INPUT TYPE						
(B)	(C)	(D)			(E)	
NUMBER	TRANSACTION	FTR	DET	CMPLX	MANUAL	

1	RECEIVE CM-INQUE SEGMENT INTO INCOMING-MSG	1	11	LOW	NO	
2	MERGE ST-FS4 ON ASCENDING KEY SORT-KEY USING SQ-FS1 SQ-FS2 *	2	2	LOW	NO	
3	WRITE REC-3 (FD ST-FS4)	2	1	LOW	NO	
4	WRITE IX-FS1R1-F-G-240 (FILE-RECORD-INFO)	1	3	LOW	NO	
5	WRITE NOTICE-TAPE-HEADER-RECORD FROM NOTICE-TAPE-RECORD	0	4	LOW	NO	
6	WRITE SAVED-NOTICE-TAPE-HDR-REC FROM NOTICE-TAPE-RECORD	0	4	LOW	NO	
7	WRITE FRA-INQUIRY-REC FROM FRA-INQUIRY-TN	0	6	LOW	NO	
8	WRITE 200-CHAR-COMP-S-O-REC FROM COMPLETED-S-O-FIELD	0	1	LOW	NO	
	.					
	.					
	.					
	.					
115	EXEC CICS RECEIVE MAP(MAPNAME) MAPSET(MAPSET) INTO(DCPREMOI*	1	67	AVG	NO	
116	EXEC CICS REWRITE DATASET('DCPRF000') FROM(SPO-FILE-AREA) L*	1	48	AVG	NO	
117	EXEC CICS WRITE DATASET('DCPRF000') FROM(SPO-FILE-AREA) LEN*	1	48	AVG	NO	
118	EXEC CICS RECEIVE MAP(MAPNAME) MAPSET(MAPSET) INTO(DCPREMI*	1	229	AVG	NO	
119	EXEC CICS RECEIVE MAP(MAPNAME) MAPSET(MAPSET) INTO(DCPREUI*	1	223	AVG	NO	
120	EXEC CICS REWRITE DATASET('DCPRF001') FROM(SP1-RECORD-TYPE-*	1	20	AVG	NO	
121	EXEC CICS REWRITE DATASET('DCPRF001') FROM(SP1-RECORD-TYPE-*	1	47	AVG	NO	
122	EXEC CICS WRITE DATASET('DCPRF001') FROM(SP1-FILE-AREA) LEN*	1	11	LOW	NO	
123	EXEC CICS RECEIVE MAP(MAPNAME) MAPSET(MAPSET) INTO(DCPREM3I*	1	29	AVG	NO	
124	EXEC CICS REWRITE DATASET('DCPRF000') FROM(SPO-FILE-AREA) L*	1	48	AVG	NO	
(F)	TOTALS	LOW = 61	AVERAGE = 59	HIGH = 4		
=====						

Report Field Descriptions

Field	Description
(A)	Name of element type.
(B)	The sequence number of external inputs in the application.
(C)	The transaction that generated the external input element.
(D)	The function point values for each transaction, including the number of file types referenced (FTR), the number of Data Element Types used (DET), and the level of complexity assigned to the element (CMPLX = high, average, or low).
(E)	This field indicates whether function point information for the transaction has been manually changed.
(F)	The function point totals for all external input elements counted for the application.

External Output Type Table

[Figure 41](#) shows an example of the External Output Type table.

Figure 41 • External Output Type Table

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION FUNCTION POINT ANALYSIS		DD-MMM-YYYY HH:MM:SS		PAGE 123	
		APPLICATION: SALES					
=====							
(A)							
EO - EXTERNAL OUTPUT TYPE							
(B)	(C)			(D)		(E)	
NUMBER	TRANSACTION	FTR	DET	CMPLX	MANUAL		

1	SEND CM-OUTQUE FROM SENT-WITH-ESI WITH ESI AFTER ADVANCING *	1	5	LOW	NO		
2	SEND CM-OUTQUE FROM SENT-WITH-EMI WITH EMI	1	5	LOW	NO		
3	SEND CM-OUTQUE FROM SENT-WITH-IDENTIFIER WITH END-IDENTIFIE*	1	5	LOW	NO		
4	SEND CM-OUTQUE WITH END-IDENTIFIER	1	5	LOW	NO		
5	SEND CM-OUTQUE FROM SENT-WITH-EGI WITH EGI AFTER ADVANCING *	1	5	LOW	NO		
6	READ SQ-FS1	1	1	LOW	NO		
7	READ SQ-FS2	1	1	LOW	NO		
8	READ SQ-FS3	1	1	LOW	NO		
9	READ JOB-SALARY-RANGE-FILE	1	7	LOW	NO		
10	READ PERSONNEL-MSTR-FILE	1	143	AVG	NO		
	.						
	.						
	.						
	.						
50	EXEC CICS READ DATASET('DCPRF001') INTO(SP1-RECORD-TYPE-P) *	1	20	AVG	NO		
51	EXEC CICS READ DATASET('DCPRF001') INTO(SP1-FILE-AREA) LENG*	1	11	LOW	NO		
52	EXEC CICS SEND MAP(MAPNAME) MAPSET(MAPSET) FROM(DCPREMI0) C*	1	77	AVG	NO		
53	EXEC CICS SEND MAP(MAPNAME) MAPSET(MAPSET) FROM(DCPREUI0) C*	1	75	AVG	NO		
54	EXEC CICS READ DATASET('DCPRF000') INTO(SP0-FILE-AREA) RIDF*	1	48	AVG	NO		
55	EXEC CICS READ DATASET('DCPRF001') INTO(SP1-FILE-AREA) RIDF*	1	11	LOW	NO		
56	EXEC CICS READ DATASET('DCPRF001') INTO(SP1-RECORD-TYPE-C) *	1	47	AVG	NO		
57	EXEC CICS SEND MAP(MAPNAME) MAPSET(MAPSET) FROM(DCPREM30) C*	1	12	LOW	NO		
(F)	TOTALS	LOW = 43	AVERAGE = 14	HIGH = 0			
=====							

Report Field Descriptions

Field	Description
(A)	Name of element type.
(B)	The sequence number of external inputs in the application.
(C)	The transaction that generated the external input element.
(D)	The function point values for each transaction, including the number of file types referenced (FTR), the number of Data Element Types used (DET), and the level of complexity assigned to the element (CMPLX = high, average, or low).
(E)	This field indicates whether function point information for the transaction has been manually changed.
(F)	The function point totals for all external input elements counted for the application.

External Inquiry Type Table

[Figure 42](#) shows an example of the External Inquiry Type table.

Figure 42 • External Inquiry Type Table

(B) NUMBER		(C) TRANSACTION	(A) EQ - EXTERNAL INQUIRY TYPE		(D) FTR	DET	CMPLX	(E) MANUAL
(F) TOTALS		LOW = 0	AVERAGE = 0		HIGH = 0			

Report Field Descriptions

Field	Description
(A)	Name of element type.
(B)	The sequence number of external inquiries in the application.
(C)	The transaction that generated the external inquiry element.
(D)	The function point values for the transaction including the number of file types referenced (FTR), the number of Data Element Types included (DET), and the level of complexity assigned to the element (CMPLX = high, average, or low).
(E)	This field indicates whether the function point information for the transaction has been manually changed.
(F)	The function point totals for all external inquiry elements counted for the application.

Internal Logical File Type Table

[Figure 43](#) shows an example of the Internal Logic File Type table.

Figure 43 • Internal Logic File Type Table

ASG-RECAP-OS (ESA) Rx.x LVL:xxx		APPLICATION FUNCTION POINT ANALYSIS		DD-MMM-YYYY HH:MM:SS PAGE 124	
		APPLICATION: SALES			
=====					
(A)					
ILF - INTERNAL LOGICAL FILE TYPE					
(B)	(C)	(D)		(E)	
NUMBER	DATA ELEMENT	RET	DET	CMPLX	MANUAL

1	FD SQ-FS1	1	1	LOW	NO
2	FD SQ-FS2	1	1	LOW	NO
3	FD SQ-FS3	2	2	LOW	NO
4	FD IX-FS1	3	45	AVG	NO
5	FD ALRU-SOR-INPUT-FILE	20	517	HIGH	NO
6	FD SAVED-ALRU-SOR-INPUT-FILE	20	517	HIGH	NO
7	FD FRA-INQUIRY-FILE	1	6	LOW	NO
8	FD COMPLETED-SERVICE-ORDER-FILE	19	19	AVG	NO
9	FD SAVED-COMPLETED-SER-ORD-FILE	19	19	AVG	NO
10	FD RL-FD1	2	2	LOW	NO

25	DCPRF000	1	48	LOW	NO
26	DCPRF001	3	78	HIGH	NO
27	DCPRF002	1	6	LOW	NO
28	FD MSG-FILE	1	42	LOW	NO
29	FD SPOOL-MASTER	8	83	HIGH	NO
(F)	TOTALS	LOW = 20	AVERAGE = 5	HIGH = 4	
=====					

Report Field Descriptions

Field	Description
(A)	Name of element type.
(B)	The sequence number of internal logical file types in the application.
(C)	The data element that generated the internal logical file element.
(D)	The function point values for each data element including the number of Record Element Types (RET), the number of Data Element Types used (DET), and the level of complexity assigned to the element (CMPLX = high, average, or low).
(E)	This field indicates whether function point information for the data element has been manually changed.
(F)	The totals for all internal logical file elements counted for the application.

External Interface File Type Table

Figure 44 shows an example of the External Interface File Type table.

Figure 44 • External Interface File Type Table

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION FUNCTION POINT ANALYSIS		DD-MMM-YYYY HH:MM:SS PAGE 125		
		APPLICATION: SALES				
(B) NUMBER	(C) DATA ELEMENT	(A) EIF - EXTERNAL INTERFACE FILE TYPE	RET	(D) DET	(E) CMLPX	MANUAL
1	FD JOB-SALARY-RANGE-FILE		1	7	LOW	NO
2	FD PERSONNEL-MSTR-FILE		2	144	HIGH	NO
3	FD IB-ISAM-ORG-FILE		4	66	HIGH	NO
4	FD IX-FD1		1	4	LOW	NO
5	FD CS-5-CHAR-USOC-FILE		1	1	LOW	NO
6	FD LINE-USOC-FILE		1	1	LOW	NO
7	FD DESIGN-LINE-CPE-USOC-FILE		1	1	LOW	NO
8	FD MULTI-LINE-ACCT-CS-FILE		1	1	LOW	NO
9	FD BUSINESS-OFFICE-CODE-FILE		1	1	LOW	NO
10	FD USOC-GRPCD-TABLE-FILE		1	1	LOW	NO

18	FD SPOOL-1-MASTER		1	6	LOW	NO
19	FD SPOOL-2-MASTER		1	6	LOW	NO
20	FD CONTROL-FILE		5	34	AVG	NO
21	FD DATASET-FILE-V		1	1	LOW	NO
22	FD DATASET-FILE		1	1	LOW	NO
(F)	TOTALS	LOW = 19	AVERAGE = 1	HIGH = 2		

Report Field Descriptions

Field	Description
(A)	Name of element type.
(B)	The sequence number of external interface file types in the application.
(C)	The data element that generated the external interface element.
(D)	The function point values for each data element including the number of Record Element Types (RET), the number of Data Element Types used (DET), and the level of complexity assigned to the element (CMLPX = high, average, or low).
(E)	This field indicates whether function point information for the data element has been manually changed.
(F)	The totals for all external interface file elements counted.

Usage Notes

If the definition is not complete, Internal Logical File elements cannot automatically be distinguished from External Interface File elements. In this case, the External Interface elements are counted as Internal Logical File elements and listed in the Internal Logical File Type table. If the definition is complete, the Internal Logical File elements are automatically distinguished from External Interface File elements.

To determine the External Interface Element count, use the Edit ILF option on the Edit - Function Points pop-up or the FP Task Manager screen to move the appropriate entities from the Internal Logical File element list to the External Interface File element list. To view the adjustments in the Application Function Point Analysis report, regenerate the report.

Total Unadjusted Function Points by Element Types Table

[Figure 45](#) shows an example of the Total Unadjusted FP by Element Types table.

Figure 45 • Total Unadjusted FP by Element Types

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION FUNCTION POINT ANALYSIS			DD-MMM-YYYY HH:MM:SS			PAGE 125
		APPLICATION: SALES						
=====								
(A)								
TOTAL UNADJUSTED FUNCTION POINTS BY ELEMENT TYPES								
(B)		LOW		(C) AVERAGE		HIGH		(D)
DESCRIPTION								TOTAL

EI - EXTERNAL INPUT TYPE		61 * 3 =	183	59 * 4 =	236	4 * 6 =	24	443
EO - EXTERNAL OUTPUT TYPE		43 * 4 =	172	14 * 5 =	70	0 * 7 =	0	242
EQ - EXTERNAL INQUIRY TYPE		0 * 3 =	0	0 * 4 =	0	0 * 6 =	0	0
ILF - INTERNAL LOGICAL FILE TYPE		20 * 7 =	140	5 * 10 =	50	4 * 15 =	60	250
EIF - EXTERNAL INTERFACE FILE TYPE		19 * 5 =	95	1 * 7 =	7	2 * 10 =	20	122

(E)								
UFP - TOTAL UNADJUSTED FUNCTION POINTS							---	1057
=====								

Report Field Descriptions

Field	Description
(A)	The name of the table.
(B)	The description of the element type.
(C)	<p>The function point value computed for the level of information processing function associated with each element type. For each information processing function category (Low, Average, High), the function point value is calculated using this formula:</p> $\text{Function point value} = \text{number of elements} * \text{information processing function}$ <p>The information processing function values are assigned in accordance with IFPUG standards.</p>
(D)	<p>The Total Unadjusted Function Points for each element type. The total for each element is calculated using this formula:</p> $\text{Total Unadjusted Function Points} = \text{Low function point value} + \text{Average function point value} + \text{High function point value}.$
(E)	The Total Unadjusted Function Points for the application.

General System Characteristics Table

Figure 46 shows the General System Characteristics Table used to calculate the Total Adjusted Function Points. The calculated values are placed on the Final Calculations Table.

Figure 46 • General System Characteristics and Final Calculations

(A)					
GENERAL SYSTEM CHARACTERISTICS					
#	DESCRIPTION	DI	#	DESCRIPTION	DI
C1	*DATA COMMUNICATION	0	C8	ON-LINE UPDATE	0
C2	DISTRIBUTED PROCESSING	0	C9	COMPLEX PROCESSING	0
C3	PERFORMANCE	0	C10	REUSABILITY	0
C4	HEAVILY USED CONFIGURATION	0	C11	INSTALLATION EASE	0
C5	TRANSACTION RATES	0	C12	OPERATIONAL EASE	0
C6	ON-LINE DATA ENTRY	0	C13	MULTIPLE SITES	0
C7	END USER EFFICIENCY	0	C14	FACILITATE CHANGE	0
TDI - TOTAL DEGREE OF INFLUENCE (C1 + C2 + ... + C14)					0
(B)					
FINAL CALCULATIONS					
	DESCRIPTION	EQUATION	RESULT		
VPF	VALUE ADJUSTMENT FACTOR	$0.65 + (0.01 * TDI)$	0.65		
FP	TOTAL ADJUSTED FUNCTION POINTS	$UFP * VAF$	687		

Report Field Descriptions

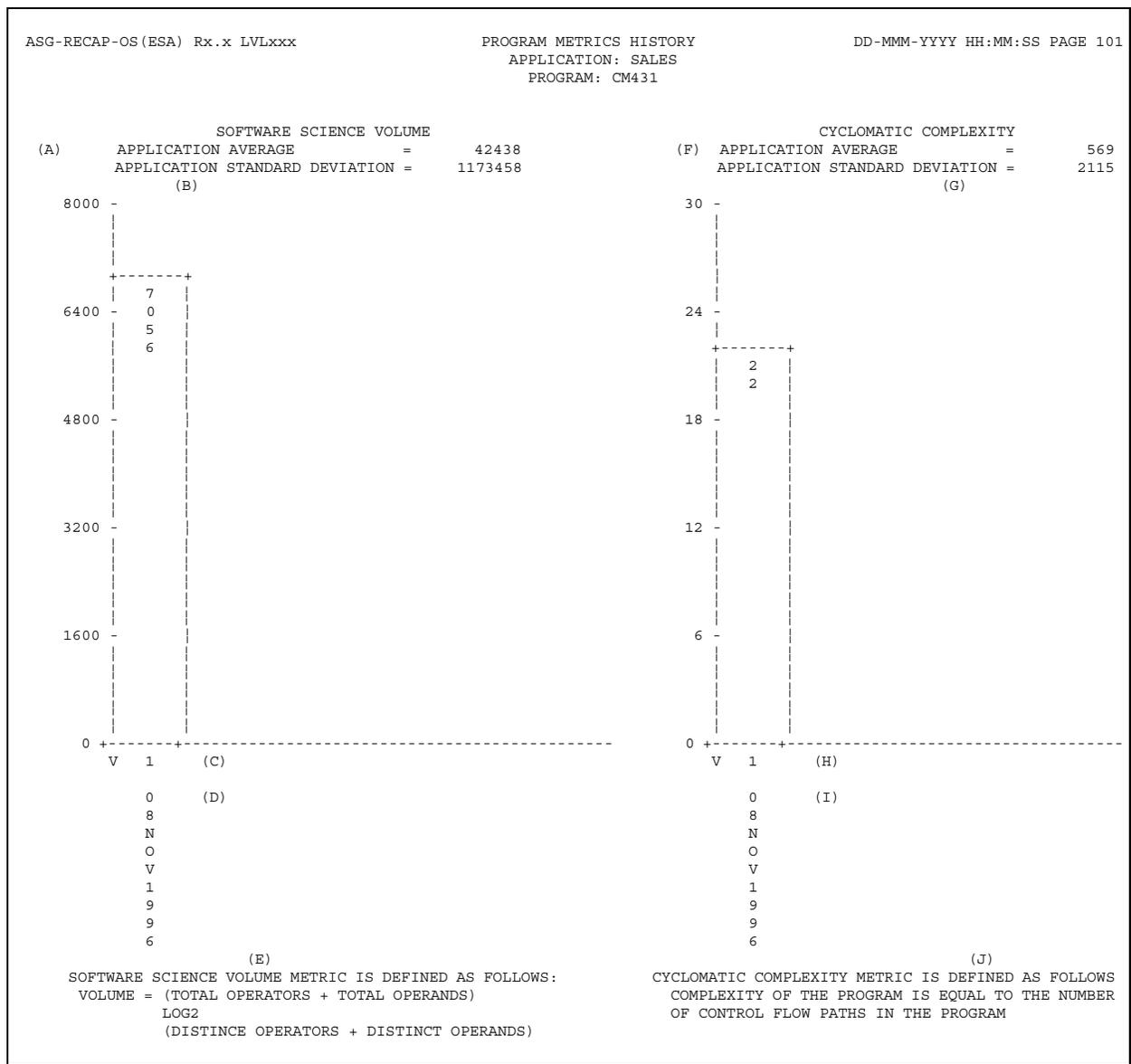
Field	Description
(A)	<p>This table provides the general systems characteristics values used to calculate the total degree of influence. The values are entered by selecting Edit GSC on the Edit - Function Points pop-up or the FP Task Manager screen. The total degree of influence equals the sum of these values.</p> <p>Note: _____ For more information on entering general systems characteristics information, see "Computing the Function Point Value Adjustment Factor" on page 42.</p>
(B)	<p>This table is used to record the Total Adjusted Function Points.</p>

Program Metrics History Report

The Program Metrics History report provides a metric value versus time graph for these metrics: Software Science Volume, Cyclomatic Complexity, Essential Complexity, Control Variable Complexity, and Control Flow Knots. The last 20 versions of each metric are retained for each program and included in the Programs Metrics History report.

[Figure 47](#) shows the graph for Software Science Volume Metrics and Cyclomatic Complexity.

Figure 47 • Program Metrics History Report (1 of 3)

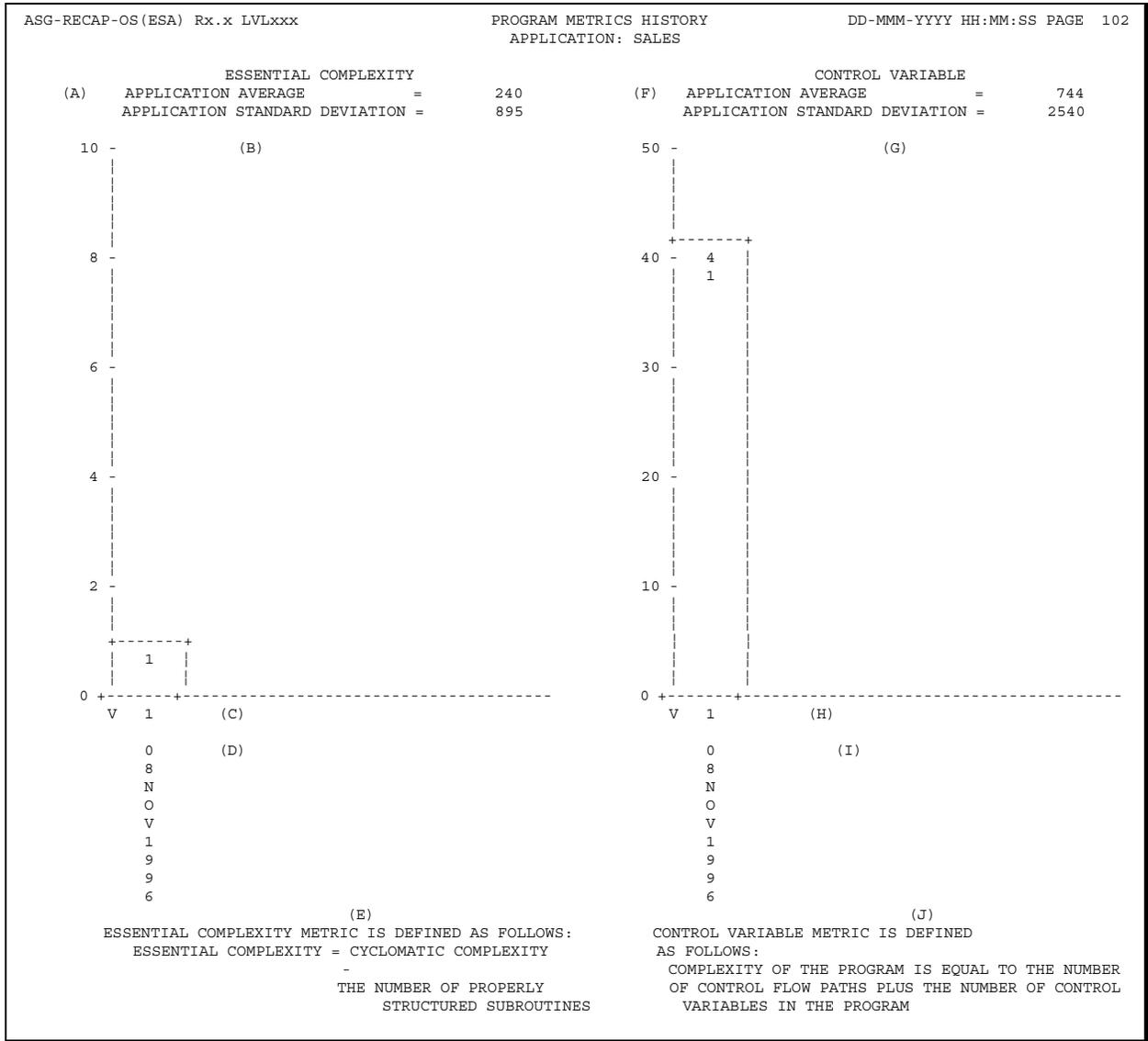


Program Metrics History Report Field Descriptions (1 of 3)

Field	Description
(A)	The average and standard deviation for Software Science Volume values, based on the last Software Science Volume values for all programs in the application.
(B)	A histogram of the Software Science Volume metric. Each column on the histogram represents one version of the program.
(C)	Application version number assigned after the program was analyzed.
(D)	Date that each version of the program was analyzed.
(E)	The Software Science Volume metric is briefly described.
(F)	The average and standard deviation for Cyclomatic Complexity values, based on the last cyclomatic complexity values for all programs in the application.
(G)	A histogram of Cyclomatic Complexity values. Each column on the histogram represents one version of the program.
(H)	Application version number assigned after the program was analyzed.
(I)	Date that each version of the program was analyzed.
(J)	The Cyclomatic Complexity Metric is briefly described.

Figure 48 shows the histograms for Essential Complexity and Volume Control.

Figure 48 • Program Metrics History Report (2 of 3)

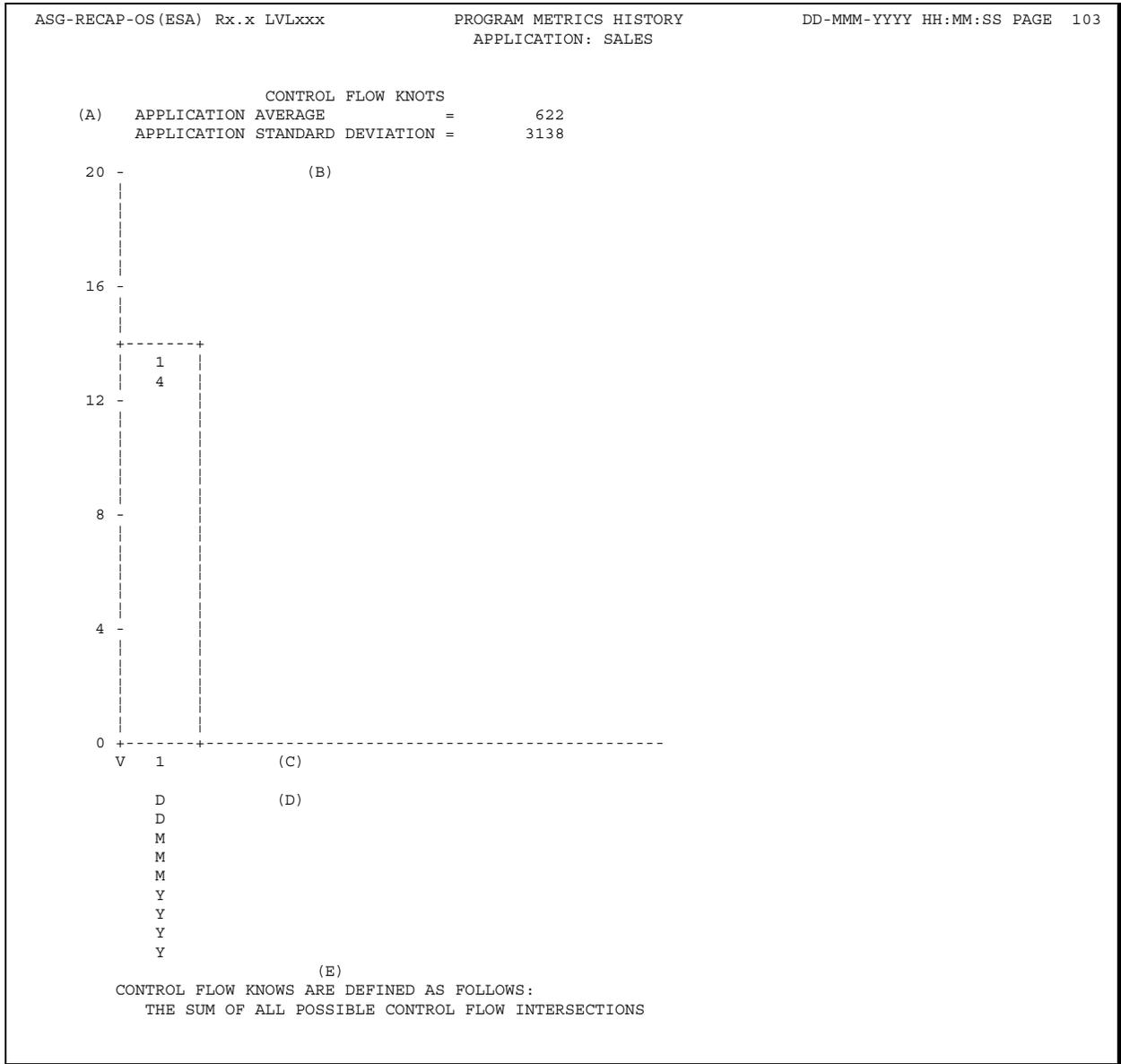


Program Metrics History Report Field Descriptions (2 of 3)

Field	Description
(A)	The average and standard deviation for Essential Complexity values, based on the last essential complexity values for all programs in the application.
(B)	A histogram of the Essential Complexity metric. Each column on the histogram represents one version of the program.
(C)	Application version number assigned after the program was analyzed.
(D)	Date that each version of the program was analyzed.
(E)	The Essential Complexity metric is briefly described.
(F)	The average and standard deviation for Control Variable Complexity metric values based on the last control variable complexity metric values for all programs in the application.
(G)	A histogram of Control Variable Complexity metric values. Each column on the histogram represents one version of the program.
(H)	Application version number assigned after the program was analyzed.
(I)	Date that each version of the program was analyzed.
(J)	The Control Variable Complexity metric is briefly described.

Figure 49 shows the histograms for Control Flow Knots metric.

Figure 49 • Program Metrics History Report (3 of 3)



Program Metrics History Report Field Descriptions (3 of 3)

Field	Description
(A)	The average and standard deviation for Control Flow Knots values, based on the last control flow knots values for all programs in the application.
(B)	A histogram of the Control Flow Knots values. Each column on the histogram represents one version of the program.
(C)	Application version number assigned after the program was analyzed.
(D)	Date that each version of the program was analyzed.
(E)	The Control Flow Knots metric is briefly described.

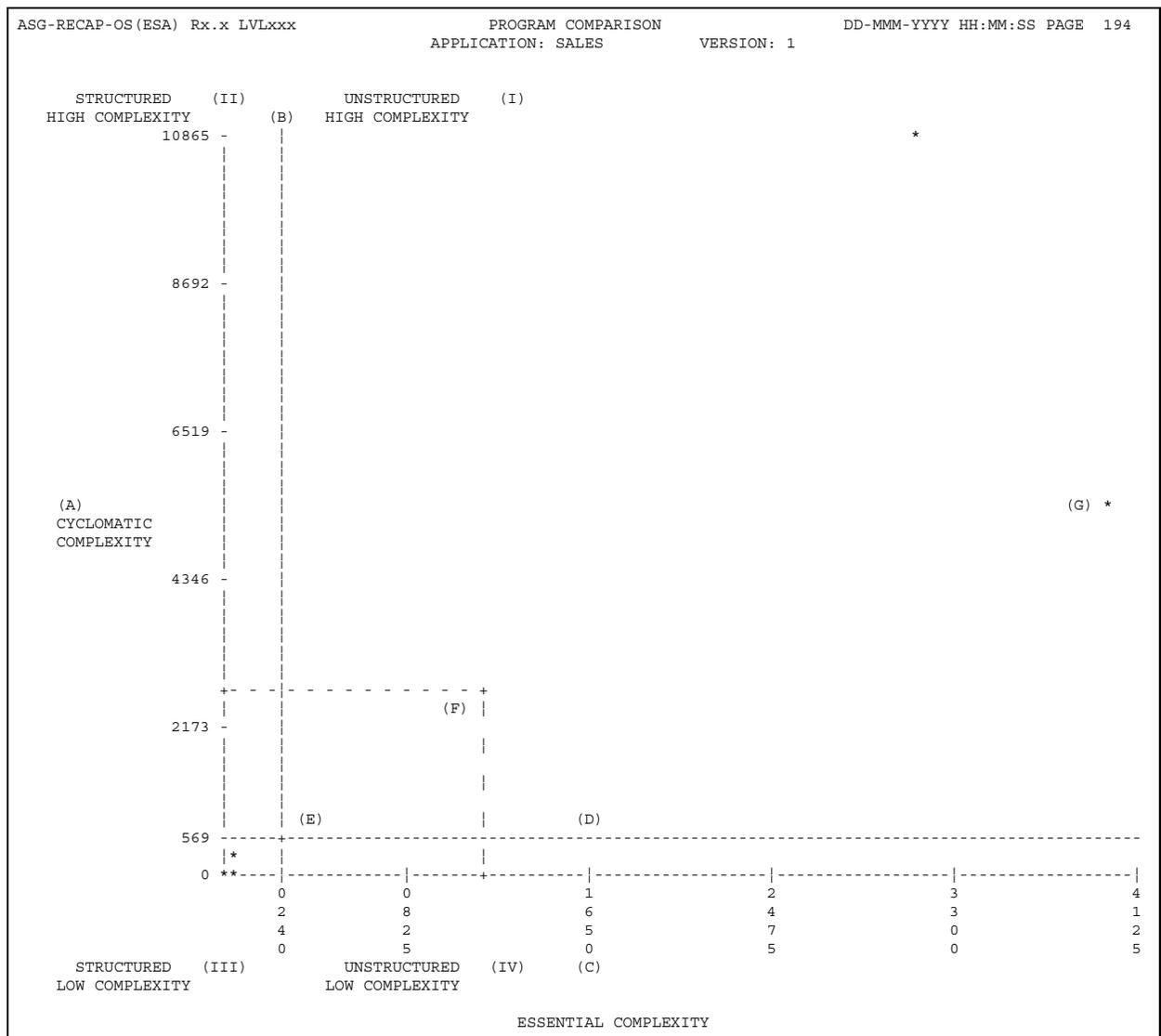
Program Comparison Report

The Program Comparison Report, shown in [Figure 50](#), compares specific metrics values for each program in the application to the average values for all programs in the application. The report consists of this self-scaling graph:

- The structure versus complexity graph represents essential complexity versus cyclomatic complexity.

The report also includes a table listing the programs found in each quadrant of the graph. On the structure versus complexity graph, the center for the quadrants is marked by the intersection between the average cyclomatic complexity value and the average essential complexity value.

Figure 50 • Program Comparison Report - Structure versus Complexity Graph (1 of 2)



Program Comparison Report Field Descriptions (1 of 2)

Field	Description
(A)	The cyclomatic complexity values are graphed on the vertical axis.
(B)	This solid line represents the average essential complexity for all programs compared.
(C)	The essential complexity values are graphed on the horizontal axis.
(D)	This dashed line represents the average cyclomatic complexity for all programs compared.
(E)	The intersection of the cyclomatic complexity average and the essential complexity average marks the center of the four quadrants.
(F)	The standard deviation box with the corners marked by a plus sign (+) represents the standard deviation from the average. The top and bottom of the box represent plus and minus one standard deviation for cyclomatic complexity. The right and left borders represent plus and minus one standard deviation for essential complexity.
(G)	A histogram of Control Variable Complexity metric values. Each column on the histogram represents one version of the program.

Figure 51 shows the Program Location table listing the programs found in each quadrant of the structure versus complexity graph.

Figure 51 • Program Comparison Report (2 of 2)

(A) QUADRANT I (UPPER RIGHT) UNSTRUCTURED - HI COMPLEX		(B) QUADRANT II (UPPER LEFT) STRUCTURED - HI COMPLEX		(C) QUADRANT III (LOWER LEFT) STRUCTURED - LO COMPLEX		(D) QUADRANT IV (LOWER RIGHT) UNSTRUCTURED - LO COMPLEX	
ASG-RECAP-OS(ESA) Rx.x LVLxxx		APPLICATION: SALES		VERSION: 1		DD-MMM-YYYY HH:MM:SS PAGE 195	
GENAMER1 (3107, 10865)				CM431 (1, 22)			
KC601 (4121, 5535)				DB204 (5, 31)			
				E02031 (8, 48)			
				IC110 (1, 1)			
				IC120 (1, 2)			
				IC207 (5, 33)			
				IC217 (1, 1)			
				IX203 (10, 52)			
				IX214 (45, 147)			
				NC110 (1, 2)			
				NC119 (5, 82)			
				NC154 (1, 43)			
				NC165 (1, 46)			
				NC209 (5, 54)			
				NC217 (5, 63)			
				NC431 (7, 70)			
				RL151 (3, 14)			
				RL431 (7, 31)			
				SG203 (15, 66)			
				SQ110 (25, 54)			
				SQ120 (13, 54)			
				SQ205 (13, 58)			
				SQ215 (5, 24)			
				ST110 (1, 1)			
				ST201 (9, 75)			
				ST211 (9, 49)			
				ST441 (5, 23)			
				TH205 (5, 50)			
				TH215 (5, 50)			

GRAPH COORDINATES = (ESSENTIAL COMPLEXITY, CYCLOMATIC COMPLEXITY)

Program Comparison Report Field Descriptions (2 of 2) Program Progress Report

Field	Description
(A)	Programs that are more unstructured and more complex than the average are listed in Quadrant 1.
(B)	Programs that are more structured and more complex than the average are listed in the Quadrant 2.
(C)	Programs that are more structured but less complex than the average are listed in Quadrant 3.
(D)	Programs that are less structured and less complex than the average are listed in Quadrant 4.

The Program Progress report shows how each program in the application has progressed by calculating the difference between the current and last values for each metric and each exception.

The Program Progress report, shown in [Figure 52](#), includes this information: Program version number, Software Science Volume, Cyclomatic Complexity, Essential Complexity, Control Variable Complexity and Control Flow Knots metrics, number of lines of code, and program exceptions. For each metric and exception, the initial (baseline) values, current values, and values from the previous version are provided. The report also lists the difference (D) between the current metric and exception values and the values from the previous version.

Figure 52 • Program Progress Report

ASG-RECAP-OS (ESA) Rxx.x LVLxxxx		PROGRAM PROGRESS APPLICATION: SALES					DD-MMM-YYYY HH:MM:SS PAGE 244				
(F)	(B)		(A)			(E)		PROGRAM: DB204			
	INITIAL	(C) CURRENT	(D) PREVIOUS	DELTA	%DELTA	INITIAL	CURRENT	PREVIOUS	DELTA	%DELTA	
VERSION	1	2	1			1	2	1			
SOFTWARE SCIENCE VOLUME	7056	7602	7311	291	4%	10089	14576	13731	845	6%	
CYCLOMATIC COMPLEXITY	22	27	23	4	17%	31	49	40	9	23%	
ESSENTIAL COMPLEXITY	1	1	1	0	0	5	3	4	1	25%	
CONTROL VARIABLE	41	39	43	(4)	(9)	61	89	72	16	24%	
KNOTS	14	0	13	13	-	10	3	13	10	77%	
NUMBER OF LINES OF CODE	377	487	408	79	19	470	737	659	78	12%	
GO TOS	19	0	5	5	-	20	5	8	3	38%	
ALTERS	0	0	6	6	-	0	0	0	0	0%	
ANOMOLOUS ENTRIES	0	0	0	0	0	0	0	0	0	0%	
ANOMOLOUS EXITS	0	0	0	0	0	0	0	0	0	0%	
RECURSIONS	0	0	0	0	0	0	0	0	0	0%	
OUT OF PERFORM RANGE JUMPS	0	0	0	0	0	0	0	0	0	0%	
LIVE EXITS	0	0	0	0	0	0	0	0	0	0%	
DEAD CODE	11	11	11	0	0	9	9	9	0	0%	
DEAD DATA	3	3	3	0	0	3	3	3	0	0%	
(G)	INITIAL : STRUCTURED, LO COMPLEXITY					INITIAL : STRUCTURED, LO COMPLEXITY					
	CURRENT : STRUCTURED, LO COMPLEXITY					CURRENT : STRUCTURED, LO COMPLEXITY					
	PREVIOUS: STRUCTURED, LO COMPLEXITY					PREVIOUS: STRUCTURED, LO COMPLEXITY					

Report Field Descriptions

Field	Description
(A)	The name of the program.
(B)	The initial (baseline) values for the program portfolio analysis data.
(C)	The current values for the program portfolio analysis data. If the program has only been analyzed once, these values are the same as the initial values.
(D)	The values from the previous version of the program.
(E)	DELTA represents the absolute value of the difference between the current and last values. The %DELTA is calculated using this formula: $\%DELTA = (DELTA/LAST) * 100$
(F)	The type of portfolio analysis data is listed in this column.
(G)	A description of the overall condition of the initial, last and current versions of the program. For example, the current version of program CM431 is structured and has a low complexity.

Note: _____

There may be programs that become obsolete but have history in prior versions. Those programs have an asterisk (*) next to the version numbers.

Application Metrics Report

The Application Metrics report shows metrics and statistical information for programs in the application. This information is included for each program:

- **Metrics information:** Values for Software Science Volume, Cyclomatic and Essential Complexity, Control Variable, Control Flow Knots, and lines of code are provided. If a metrics value is higher or lower than the high and low bounds calculated for all programs in the application, HI or LO is listed to the right of the value.
- **Overall program condition:** Structure and complexity classifications are assigned to each program based on the metrics information for the program and the averages for all programs. The classifications are low or high.
- **Comparative statistics:** The total, average, standard deviation, low bound and high bound for each metric in the application is also reported. The low bound value is obtained by subtracting the standard deviation from the average; if the result is a negative number, a zero displays. The high bound value is obtained by adding the standard deviation to the average. The high bound and low bound values determine whether a program's metric values are flagged HI or LO.

The Application Metrics report information ([Figure 53 on page 118](#)) is listed in table format. The first table shows non-conformant programs, if any. These programs have at least one metric value greater than one standard deviation from the application average. The second table displays conformant programs that have values within or lower than one standard deviation from the average for all metrics. The third table displays the comparative statistics.

Figure 53 • Application Metrics Report

ASG-RECAP-OS (ESA) Rxx.x LVLxxx		APPLICATION METRICS						DD-MMM-YYYY HH:MM:SS PAGE 206	
		APPLICATION: SALES			VERSION: 1				
		(A) NON-CONFORMANT							
(B) PROGRAM NAME	VOLUME VALUE	(C) CYCLOMATIC VALUE		ESSENTIAL VALUE	CNTL VAR VALUE	KNOTS VALUE	LINES CODE VALUE	(D) STRUCT CMLPX	
GENAMER1	262572 HI	10865 HI		3107 HI	12307 HI	661	7092 HI	LOW	HIGH
KC601	635414 HI (E)	5535 HI		4121 HI	8084 HI	17797 HI	10899 HI	LOW	HIGH
		(F) CONFORMANT							
PROGRAM NAME	VOLUME VALUE	CYCLOMATIC VALUE	ESSENTIAL VALUE	CNTL VAR VALUE	KNOTS VALUE	LINES CODE VALUE	STRUCT	CMLPX	
CM431	7056	22	1	41	14	379	HIGH	LOW	
DB204	10089	31	5	61	10	474	HIGH	LOW	
E02031	14949	48	8	122	18	882	HIGH	LOW	
IC110	200	1	1	1	0	63	HIGH	LOW	
IC120	245	2	1	3	0	100	HIGH	LOW	
IC207	9855	33	5	66	17	396	HIGH	LOW	
IC217	115	1	1	1	0	42	HIGH	LOW	
IX203	13644	52	10	106	15	610	HIGH	LOW	
IX214	54980	147	45	296	125	1702	HIGH	LOW	
NC110	648	2	1	2	1	61	HIGH	LOW	
NC119	33470	82	5	162	93	1034	HIGH	LOW	
NC154	15039	43	1	106	46	707	HIGH	LOW	
NC165	19314	46	1	98	46	799	HIGH	LOW	
NC209	20583	54	5	138	78	776	HIGH	LOW	
NC217	24633	63	5	131	43	875	HIGH	LOW	
NC431	35310	70	7	150	13	1137	HIGH	LOW	
RL151	4656	14	3	27	8	293	HIGH	LOW	
RL431	11376	31	7	58	8	567	HIGH	LOW	
SG203	17397	66	15	109	48	601	HIGH	LOW	
SQ110	12618	54	25	109	46	514	HIGH	LOW	
SQ120	11880	54	13	109	19	599	HIGH	LOW	
SQ205	15273	58	13	119	27	560	HIGH	LOW	
SQ215	5464	24	5	47	7	314	HIGH	LOW	
ST110	70	1	1	1	0	73	HIGH	LOW	
ST201	20313	75	9	152	49	806	HIGH	LOW	
ST211	17892	49	9	113	39	795	HIGH	LOW	
ST441	9918	23	5	49	8	525	HIGH	LOW	
TH205	16731	50	5	150	24	588	HIGH	LOW	
TH215	13869	50	5	149	19	468	HIGH	LOW	
		(G) LEGEND: LO = VALUE < AVG. - STD. DEV. LOW = VALUE <= AVG. LOW BOUND = AVG. - STD. DEV. STD. DEV. = STANDARD DEVIATION HI = VALUE > AVG. + STD. DEV. HIGH = VALUE > AVG. HIGH BOUND = AVG. + STD. DEV.							
(H) APPLICATION TOTAL/SUMMARY		(NUMBER OF PROGRAMS = 31)							
METRICS	TOTAL	AVERAGE	STD. DEV.	LOW BOUND	HIGH BOUND				
SOFTWARE SCIENCE VOLUME	1315573	42438	117345	0	159783				
CYCLOMATIC COMPLEXITY	17646	569	2115	0	2684				
ESSENTIAL COMPLEXITY	7445	240	895	0	1135				
CONTROL VARIABLE	23067	744	2540	0	3284				
KNOTS	19279	622	3138	0	3760				
NUMBER OF LINES OF CODE	34731	1120	2152	0	3272				
		(G) LEGEND: LO = VALUE < AVG. - STD. DEV. LOW = VALUE <= AVG. LOW BOUND = AVG. - STD. DEV. STD. DEV. = STANDARD DEVIATION HI = VALUE > AVG. + STD. DEV. HIGH = VALUE > AVG. HIGH BOUND = AVG. + STD. DEV.							

Report Field Descriptions

Field	Description
(A)	The first table presented in the Application Metrics report lists the metrics and program condition for non-conformant programs. Non-conformant programs are programs with one or more metrics values that are more than one standard deviation higher than the application average.
(B)	The program name is listed in this column.
(C)	The metrics values for each program are listed in columns two through seven.
(D)	The overall condition of the program in terms of structure [†] and complexity [‡] is listed in the last two columns. The values for each condition can be low or high.
(E)	If HI or LO appears to the right of a metrics value, that value is outside of the high bound or low bound for the application.
(F)	The metrics values and program condition for conformant programs are presented in this table. Conformant programs are defined as programs whose metrics values are within one standard deviation of the application average or whose values are lower than the application average which indicates good structure, and complexity.
(G)	<p>The legend that appears defines the formulas used to calculate the high and low bounds for metrics values. The low bound is calculated using this formula:</p> $\text{LOW BOUND} = \text{AVERAGE} - \text{STANDARD DEVIATION}$ <p>The AVERAGE and STANDARD DEVIATION values are the values calculated for the application. If a program metric has a value less than LOW BOUND, that metric is marked LO.</p> <p>The high bound is calculated using this formula:</p> $\text{HIGH BOUND} = \text{AVERAGE} + \text{STANDARD DEVIATION}$ <p>If a program metric has a value greater than HIGH BOUND, it is marked HI.</p>
(H)	The Application Total/Summary table provides a list of application totals, averages, standard deviations, low bounds, and high bounds calculated for each metric. The number of programs listed is the number of programs successfully analyzed.
†	Calculated using the Essential value column. The Program Essential Value is compared to the Application Average. If the program value is less than or equal to the application, the structure is marked high. Otherwise it is marked low. Therefore, the lower the Essential Value, the higher the structure.
‡	Calculated using the Cyclomatic value column. The Program Cyclomatic value is compared to the Application Average. If the program value is greater than the Application Average, the complexity is marked high. Otherwise it is marked low. Therefore, the higher the Cyclomatic value, the more complex the program.

Application Exceptions Report

The Application Exceptions report ([Figure 54 on page 122](#)) shows code exceptions affecting the quality of programs in the application.

Note: _____

If programs within an application have a high number of exceptions, run a SmartDoc analysis to locate and correct the exceptions. See the *ASG-SmartDoc User's Guide* for more information.

These items are reported on the Application Exceptions report:

- GO TOS

The number of GOTOs and ALTERS in the application.

- ALTERS

The number of ALTERS in the application.

- ANOMOLOUS ENTRIES

The number of ENTRY statements in the application.

- ANOMOLOUS EXITS

The number of STOP RUN, GOBACK, or EXIT PROGRAM verbs and CALL statements that are indicated as NORET (non-returning).

- RECURSION

The number of occurrences of recursion in the application. Recursion generally occurs when a paragraph or perform range performs itself. This type of programming technique can lead to endless loops.

- JUMPS (OUT OF PERFORM RANGE JUMPS)

The number of out of perform range jumps in the application. An exit is made from a perform range, due to the use of a GOTO, bypassing the normal perform range return to the line following the perform statement.

- LIVE EXITS

The number of live exits in the application. Live exits are exits from perform ranges that are left dangling by overlapping PERFORMs and GOTOs in the original performed paragraphs. Internally, the COBOL compiler creates a jump statement to return to the caller at the end of a perform range. When the jump is actually performed, the jump statement is cleared. If an exit is made from the perform range before the jump statement is cleared (the normal flow is changed), the internal jump statement is left live. If an overlapping perform range is subsequently executed, the jump statement set by the original perform range is still live and causes the program to jump back to the location of the first caller to that perform range. This results in the program not executing as expected.

- DEADCODE

The number of lines of dead code in the application. Dead code statements are PROCEDURE DIVISION statements that never execute.

- DEADDATA

The number of dead data occurrences in the application. Dead data occurrences are unreferenced data items or data items only referenced in DEADCODE.

Figure 54 • Application Exceptions Report

ASG-RECAP-OS (ESA) Rxx.x LVLxxx			APPLICATION EXCEPTIONS					DD-MMM-YYYY HH:MM:SS PAGE 208		
			APPLICATION: SALES		VERSION: 1					
(A)	(B)									
PROGRAM NAME	NUMBER OF GO TOS	NUMBER OF ALTERS	NUMBER OF A-ENTRIES	NUMBER OF A-EXITS	NUMBER OF RECURSION	NUMBER OF JUMPS	NUMBER OF LIVE EXIT	LINES OF DEAD CODE	LINES OF DEAD DATA	
CM431	19	0	0	0	0	0	0	11	0	
DB204	20	0	0	0	0	0	0	9	0	
E02031	14	0	0	1	0	0	0	1	0	
GENAMER1	497	0	1	2	3793	6077	565	104	0	
IC110	0	0	0	0	0	0	0	0	1	
IC120	0	0	0	0	0	0	0	8	0	
IC207	19	0	0	0	0	0	0	7	0	
IX203	24	0	0	0	0	0	0	7	0	
IX214	122	0	0	0	0	0	0	18	0	
KC601	2179	0	0	2	1	4135	3	44	0	
NC110	4	1	0	0	0	0	0	3	0	
NC119	158	0	0	0	0	0	0	156	0	
NC154	77	0	0	0	0	0	0	82	0	
NC165	85	0	0	0	0	0	0	109	0	
NC209	117	0	0	0	0	0	0	138	0	
NC217	60	0	0	0	0	0	0	100	0	
NC431	19	2	0	0	0	0	0	8	0	
RL151	13	0	0	0	0	0	0	12	0	
RL431	11	0	0	0	0	0	0	7	1	
SG203	103	17	0	0	0	1	0	53	0	
SQ110	47	0	0	0	0	0	0	5	0	
SQ120	28	0	0	0	0	15	0	5	0	
SQ205	49	0	0	0	0	0	0	11	0	
SQ215	8	0	0	0	0	0	0	7	0	
ST110	0	0	0	0	0	0	0	0	1	
ST201	73	0	0	0	0	28	3	99	0	
ST211	46	0	0	1	0	0	0	38	0	
ST441	9	0	0	0	0	0	0	44	0	
TH205	40	0	0	0	0	0	0	7	0	
TH215	23	0	0	0	0	0	0	7	0	

(C)	
APPLICATION TOTAL/SUMMARY	(NUMBER OF PROGRAMS = 30)
EXCEPTIONS	TOTAL
-----	-----
GO TOS	3864
ALTERS	20
ANOMALOUS ENTRIES	1
ANOMALOUS EXITS	6
RECURSIONS	3794
OUT OF PERFORM RANGE JUMPS	10256
LIVE EXITS	571
DEAD CODE	1100
DEAD DATA	3

LEGEND: JUMPS = OUT OF PERFORM RANGE JUMPS A-ENTRIES = ANOMALOUS ENTRIES A-EXITS = ANOMALOUS EXITS

Report Field Descriptions

Field	Description
(A)	The names of the programs in the application.
(B)	These columns list the number and type of exceptions found in the program.
(C)	The Application Total/Summary table lists the application totals for each type of exception. The number of programs is the number of successfully analyzed programs.

Application Comparison Report

The Application Comparison report compares specific metrics values for each application in the AKR to the average values for all applications in the AKR. The report consists of two self-scaling graphs:

- The structure versus complexity graph represents essential complexity versus cyclomatic complexity.
- The business value versus complexity graph represents the number of function points versus cyclomatic complexity.

The report also includes a table listing the applications found in each quadrant of the graph. On the structure versus complexity graph, the center for the quadrants is marked by the intersection between the average cyclomatic complexity value and the average essential complexity value. On the business value versus complexity graph, the center is marked by the intersection between the average function point value and the average cyclomatic complexity value.

Application Comparison Report Field Descriptions (1 of 4)

Field	Description
(A)	The cyclomatic complexity values are graphed on the vertical axis.
(B)	This solid line represents the average essential complexity for all applications compared.
(C)	The essential complexity values are graphed on the horizontal axis.
(D)	This dashed line represents the average cyclomatic complexity for all applications compared.
(E)	The intersection of the cyclomatic complexity average and the essential complexity average marks the center of the four quadrants.
(F)	This box, with the corners marked by a plus sign (+), represents one standard deviation from the average. The top and bottom of the box represent plus and minus one standard deviation for cyclomatic complexity. The right and left borders represent plus and minus one standard deviation for essential complexity.
(G)	Represents one or more applications that have metrics at the coordinate marked.

Figure 56 shows the Application Location table listing the applications found in each quadrant of the structure versus complexity graph.

Figure 56 • Application Comparison Report (2 of 4)

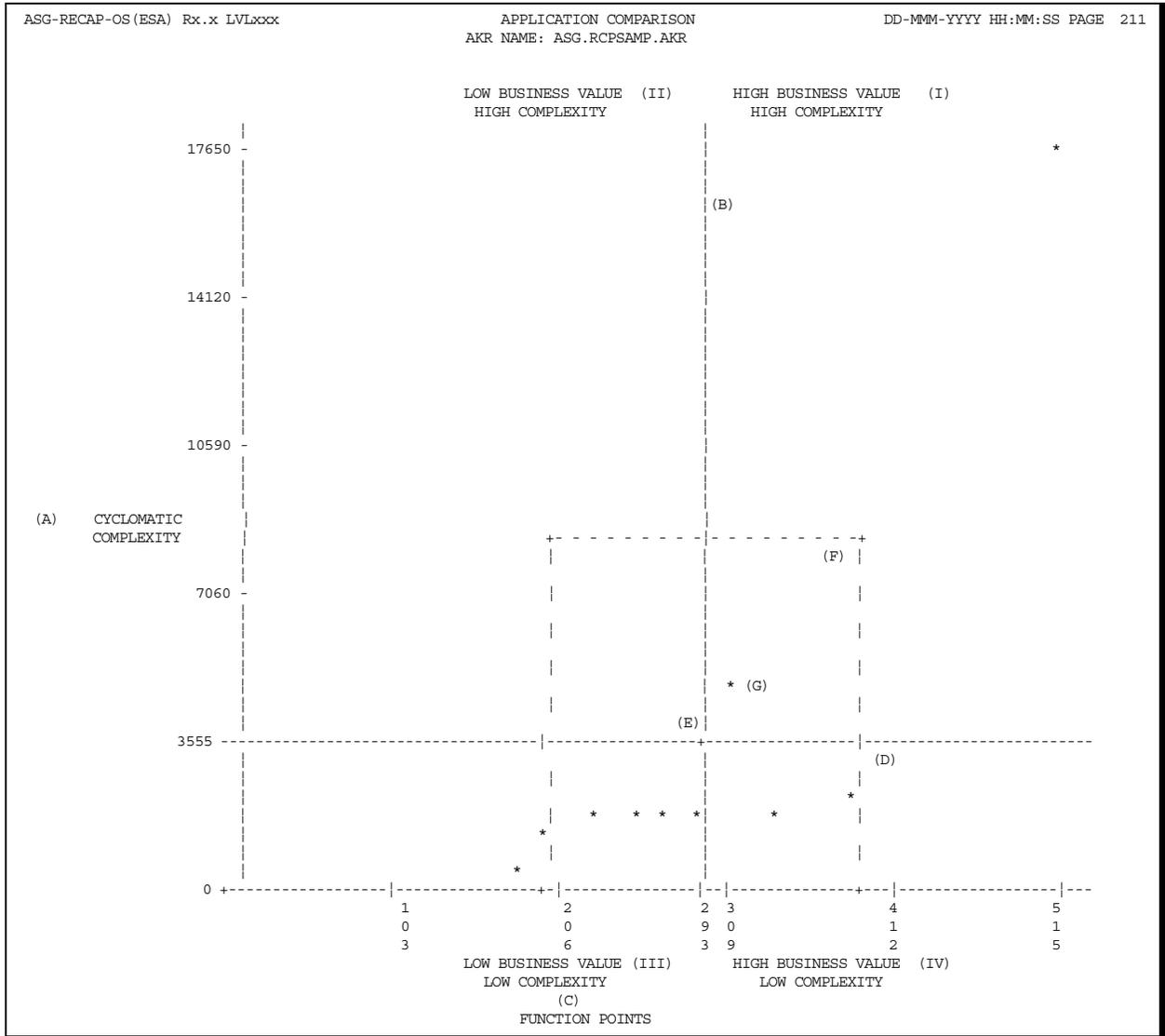
(A) QUADRANT I (UPPER RIGHT) UNSTRUCTURED - HI COMPLEX		(B) QUADRANT II (UPPER LEFT) STRUCTURED - HI COMPLEX		(C) QUADRANT III (LOWER LEFT) STRUCTURED - LO COMPLEX		(D) QUADRANT IV (LOWER RIGHT) UNSTRUCTURED - LO COMPLEX	
MANF-CNTL (2317, 4729) SALES (7445, 17646)				ACCT-PAYBL (225, 1542) ACCT-RCVBL (495, 1917) CUST-BILL (474, 1853) CUSTOMER (266, 1568) EMPL-INFO (342, 1858) INVENTORY (546, 1946) PAYROLL (453, 2021) TELEPHONE (63, 474)			
GRAPH COORDINATES = (ESSENTIAL COMPLEXITY, CYCLOMATIC COMPLEXITY)							

Application Comparison Report Field Descriptions (2 of 4)

Field	Description
(A)	Applications that are more unstructured and more complex than the average are listed in Quadrant 1.
(B)	Applications that are more structured and more complex than the average are listed in the Quadrant 2.
(C)	Applications that are more structured but less complex than the average are listed in Quadrant 3.
(D)	Applications that are more unstructured and less complex than the average are listed in Quadrant 4.

Figure 57 shows the business value versus complexity graph provided in the Application Comparison Report.

Figure 57 • Application Comparison Report (3 of 4)



Application Comparison Report Field Descriptions (3 of 4)

Field	Description
(A)	The cyclomatic complexity values are graphed on the vertical axis.
(B)	This solid line represents the average function point value for all applications compared.
(C)	The function point values (business value) are graphed on the horizontal axis.
(D)	This dashed line represents the average cyclomatic complexity value for all applications compared.
(E)	The intersection of the cyclomatic complexity average and the function points average marks the center of the four quadrants.
(F)	This box, with the corners marked by a plus (+), represents one standard deviation from the average. The top and bottom of the box represent plus and minus one
(G)	Represents one or more applications that have metrics at the coordinate marked.

Figure 58 shows the Application Location table listing the applications in each quadrant of the business value versus complexity graph.

Figure 58 • Application Comparison Report (4 of 4)

(A) QUADRANT I (UPPER RIGHT) HI BUSINESS VAL - HI COMPLEX		(B) QUADRANT II (UPPER LEFT) LO BUSINESS VAL - HI COMPLEX		(C) QUADRANT III (LOWER LEFT) LO BUSINESS VAL - LO COMPLEX		(D) QUADRANT IV (LOWER RIGHT) HI BUSINESS VAL - LO COMPLEX	
MANF-CNTL	(310, 4729)			ACCT-PAYBL	(188, 1542)	CUST-BILL	(337, 1853)
SALES	(514, 17646)			ACCT-RCVBL	(288, 1917)	PAYROLL	(383, 2021)
				CUSTOMER	(266, 1568)		
				EMPL-INFO	(250, 1858)		
				INVENTORY	(221, 1946)		
				TELEPHONE	(171, 474)		

GRAPH COORDINATES = (FUNCTION POINTS, CYCLOMATIC COMPLEXITY)

Field Descriptions (4 of 4)

Field	Description
(A)	Applications that have a higher than average business value and a higher than average complexity are listed in Quadrant 1.
(B)	Applications that have a business value equal to or lower than average and a higher than average complexity are listed in Quadrant 2.
(C)	Applications that have a business value equal to or lower than the average business value and a complexity equal to or lower than average complexity are listed in Quadrant 3.
(D)	Applications that have a higher than average business value and a complexity equal to or lower than the average complexity are listed in Quadrant 4.

Application Progress Report

The Application Progress report, shown in [Figure 59](#), shows how each application in the enterprise has progressed by calculating the difference between the current and last values for each metric and each exception.

The Application Progress report includes this information:

- Application version number
- Number of programs
- Adjusted Function Points
- Enhancement Function Points
- Software Science Volume
- Cyclomatic Complexity
- Essential Complexity
- Control Variable Complexity and Control Flow Knots metrics
- Number of lines of code, and application exceptions

For each metric and exception, the initial (baseline) values, current values, and values from the previous version are provided. The report also lists the difference (D) between the current metric and exception values and the values from the previous version.

Figure 59 • Application Progress Report

ASG-RECAP-OS (ESA) Rx.x LVLxxx		APPLICATION PROGRESS				DD-MMM-YYYY HH:MM:SS PAGE 213			
		AKR NAME: ASG.RCPSAMP.AKR							
		(A) APPLICATION: ACCT-PAYBL			(E)	APPLICATION: ACCT-RCVBL			
(B) INITIAL	(C) CURRENT	(D) PREVIOUS	DELTA	%DELTA	INITIAL	CURRENT	PREVIOUS	DELTA	%DELTA
VERSION	1	1			1	1			
NUMBER OF PROGRAMS	36	36			43	43			
ADJUSTED FUNCTION POINTS	188	188			288	288			
ENHANCEMENT FUNCTION POINTS	0	0			0	0			
SOFTWARE SCIENCE VOLUME	511021	511021			605906	605906			
CYCOMATIC COMPLEXITY	1542	1542			1917	1917			
ESSENTIAL COMPLEXITY	225	225			495	495			
CONTROL VARIABLE	3245	3245			4035	4035			
KNOTS	1070	1070			1241	1241			
NUMBER OF LINES OF CODE	19048	19048			23493	23493			
GO TOS	1510	1510			1602	1602			
ALTERS	19	19			29	29			
ANOMALOUS ENTRIES	0	0			0	0			
ANOMALOUS EXITS	2	2			3	3			
RECURSIONS	0	0			0	0			
OUT OF PERFORM RANGE JUMPS	5	5			88	88			
LIVE EXITS	0	0			3	3			
DEAD CODE	1235	1235			1050	1050			
DEAD DATA	2	2			3	3			
(F)	INITIAL : STRUCTURED, LO COMPLEXITY, LO BUS VALUE				INITIAL : STRUCTURED, LO COMPLEXITY, LO BUS VALUE				
	CURRENT : STRUCTURED, LO COMPLEXITY, LO BUS VALUE				CURRENT : STRUCTURED, LO COMPLEXITY, LO BUS VALUE				
	PREVIOUS: INFO NOT AVAILABLE				PREVIOUS: INFO NOT AVAILABLE				

Report Field Descriptions

Field	Description
(A)	The name of the application.
(B)	The baseline portfolio analysis data for the application.
(C)	The current values for the portfolio analysis data.
(D)	The portfolio analysis data from the previous version. If the application has only been analyzed one time, these values are not available.
(E)	<p>DELTA represents the absolute value of the difference between the current and last values. The %DELTA is calculated using this formula:</p> $\%DELTA = (DELTA/LAST) * 100$
(F)	A description of the overall condition of the initial, last and current versions of the application. For example, the current version of application ACCT-PAYBL is structured, has a low complexity and a low business value (low business value = low function point).

Enterprise Metrics Report

The Enterprise Metrics report ([Figure 60 on page 133](#)) shows metrics and statistical information for applications in the Enterprise (AKR). This information is included for each application:

- Metrics information: Values for program total, Adjusted Function Points, Enhancement Function Points, Software Science Volume, Cyclomatic and Essential Complexity, Control Variable, Control Flow Knots, and lines of code are provided. If a metrics value, other than Function Points, is higher or lower than the high or low bounds calculated for all applications in the enterprise, HI or LO is listed to the right of the value.
- Overall program condition: Structure, complexity, and business value classifications are assigned to each application based on the metrics information for the application and the averages of all applications in the AKR. The classifications are low or high.
- Comparative statistics: The total, average, standard deviation, low bound and high bound for each metric in the application is also reported. The low bound value is obtained by subtracting the standard deviation from the average; if the result is a negative number a zero displays. The high bound value is obtained by adding the standard deviation to the average. The high bound and low bound values determine whether a program's metric values are flagged HI or LO.

The Enterprise Metrics Report information is listed in table format. The first table shows non-conformant applications, if any exist. These applications have at least one metric value, other than Function Points, greater than one standard deviation from the enterprise average. The second table displays conformant applications that have values within or lower than one standard deviation from the average for all metrics. The third table displays the comparative statistics.

Figure 60 • Enterprise Metrics Report

ASG-RECAP-OS (ESA) Rx.x LVLxxx													ENTERPRISE METRICS		DD- <small>MM</small> - <small>YYYY</small> <small>HH</small> : <small>MM</small> : <small>SS</small> PAGE 216	
													AKR NAME: ASG.RCPSAMP.AKR			
													(A)			
													NON-CONFORMANT			
(B)	(C)	(D)	(E)										(F)		(G)	
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES TOTAL	CODE	STRUCT	CMPLX	BUSNS VALUE			
MANF-CNTL	36	310	0	882297	4729	2317	8524	7272	31698	HI	LOW	HIGH	HIGH			
SALES	31	514	0	1315573 HI	17646 HI (H)	7445 HI	23067 HI	19279 HI	34731 HI	HI	LOW	HIGH	HIGH			
													(I)			
													CONFORMANT			
APPLICATION NAME	PGM TOTAL	ADJ FP TOTAL	ENH FP TOTAL	VOLUME TOTAL	CYCLOMATIC TOTAL	ESSENTIAL TOTAL	CNTL VAR TOTAL	KNOTS TOTAL	LINES TOTAL	CODE	STRUCT	CMPLX	BUSNS VALUE			
ACCT-PAYBL	36	188	0	511021	1542	225	3245	1070	19048		HIGH	LOW	LOW			
ACCT-RCVBL	43	288	0	605906	1917	495	4035	1241	23493		HIGH	LOW	LOW			
CUST-BILL	43	337	0	601125	1853	474	3891	1226	23289		HIGH	LOW	HIGH			
CUSTOMER	33	266	0	536930	1568	266	3320	1097	20566		HIGH	LOW	LOW			
EMPL-INFO	35	250	0	558657	1858	342	3622	1092	21325		HIGH	LOW	LOW			
INVENTORY	34	221	0	531911	1946	546	4623	1502	22985		HIGH	LOW	LOW			
PAYROLL	36	383	0	614351	2021	453	4109	1223	22804		HIGH	LOW	HIGH			
TELEPHONE	6	171	0	146795 LO	474	63	1241	28	6579 LO		HIGH	LOW	LOW			
													(J)			
ENTERPRISE TOTAL/SUMMARY													(NUMBER OF APPLICATIONS = 10, NUMBER OF PROGRAMS = 333)			
METRICS				TOTAL	AVERAGE	STD. DEV.	LOW BOUND	HIGH BOUND								
ADJUSTED FUNCTION POINTS				2928	293	96	197	389								
ENHANCEMENT FUNCTION POINTS				0	0	0	0	0								
SOFTWARE SCIENCE VOLUME				6304566	630457	283961	346496	914418								
CYCLOMATIC COMPLEXITY				35554	3555	4804	0	8359								
ESSENTIAL COMPLEXITY				12626	1263	2146	0	3409								
CONTROL VARIABLE				59677	5968	5954	14	11922								
KNOTS				35030	3503	5587	0	9090								
NUMBER OF LINES OF CODE				226518	22652	7116	15536	29768								
													(K)			
LEGEND: LO = VALUE < AVG. - STD. DEV.				LOW = VALUE <= AVG.			LOW BOUND = AVG. - STD. DEV.		STD. DEV. = STANDARD DEVIATION							
HI = VALUE > AVG. + STD. DEV.				HIGH = VALUE > AVG.			HIGH BOUND = AVG. + STD. DEV.									

Report Field Descriptions

Field	Description
(A)	The first table presented in the Enterprise Metrics report lists the metrics and application condition for non-conformant applications. Non-conformant applications have one or more metrics values that are more than one standard deviation higher than the enterprise average.
(B)	The application name is listed in this column.
(C)	The total number of programs in the application is listed in this column.
(D)	The function point total for the application.
(E)	The enhancement function point total for the application.

Field	Description
(F)	The metrics values for each application are listed in columns five through ten.
(G)	The overall condition of the application in terms of structure, complexity, and business value is listed in the last three columns. The values for each condition can be low or high. (See the footnotes in "Report Field Descriptions" on page 119 for an explanation of these columns.)
(H)	If HI or LO appears to the right of a metrics value, that value is outside of the high bound or low bound for the enterprise.
(I)	The metrics values and application condition for conformant applications are presented in this table. Conformant applications have metrics values that are within one standard deviation of the enterprise average or values that are lower than the enterprise average which indicates good structure and complexity.
(J)	The Enterprise Total/Summary table provides a list of enterprise totals, averages, standard deviations, low bounds, and high bounds calculated for each metric.
(K)	<p>The legend that appears defines the formulas used to calculate the high and low bounds for metrics values. The low bound is calculated using this formula:</p> $\text{LOW BOUND} = \text{AVERAGE} - \text{STANDARD DEVIATION}$ <p>The AVERAGE and STANDARD DEVIATION values are the values calculated for the enterprise. If an application metric has a value less than LOW BOUND, that metric is marked LO.</p> <p>The high bound is calculated using this formula:</p> $\text{HIGH BOUND} = \text{AVERAGE} + \text{STANDARD DEVIATION}$ <p>If an application metric has a value greater than HIGH BOUND, it is marked HI.</p>
(A)	The first table presented in the Enterprise Metrics report lists the metrics and application condition for non-conformant applications. Non-conformant applications have one or more metrics values that are more than one standard deviation higher than the enterprise average.
(B)	The application name is listed in this column.

Enterprise Exceptions Report

The Enterprise Exceptions report ([Figure 61 on page 137](#)) summarizes code exceptions affecting the quality of the applications in the enterprise.

Note: _____

If an application within the enterprise has a high number of exceptions, run a SmartDoc analysis on each of the programs within the application to locate and correct the exceptions. See the *ASG-SmartDoc User's Guide* for more information.

These items are reported on the Enterprise Exceptions Report:

- NUMBER OF PROGRAMS
The number of programs in each application within the enterprise.
- GO TOS
The number of GOTOs and ALTER GO TOs in the enterprise.
- ALTERS
The number of ALTER GO TOs in the enterprise.
- ANOMOLOUS ENTRIES
The number of ENTRY statements in the enterprise.
- ANOMOLOUS EXITS
The number of statements containing a STOP RUN, GOBACK, or EXIT PROGRAM verb, and CALL statements that are indicated as NORET (non-returning).
- RECURSION
The number of occurrences of recursion in the enterprise. Recursion generally occurs when a paragraph or perform range performs itself. This type of programming technique can lead to endless loops.

- **JUMPS (OUT OF PERFORM RANGE JUMPS)**

The number of out of perform range jumps in the enterprise. An exit is made from a perform range, due to the use of a GOTO, bypassing the normal perform range return to the line following the perform statement.
- **LIVE EXITS**

The number of live exits in the enterprise. Live exits are exits from perform ranges that are left dangling by overlapping PERFORMs and GOTOs in the original Performed paragraphs. Internally, the COBOL compiler creates a jump statement to return to the caller at the end of a perform range. When the jump is actually performed, the jump statement is cleared. If an exit is made from the perform range before the jump statement is cleared (the normal flow is changed), the internal jump statement is left live. If an overlapping perform range is subsequently executed, the jump statement set by the original perform range is still live and causes the program to jump back to the location of the first caller to that perform range. This results in the program not executing as expected.
- **DEADCODE**

The number of dead code occurrences in the application within the enterprise. Dead code statements are PROCEDURE DIVISION statements that never execute under any conditions.
- **DEADDATA**

The number of dead data occurrences in the application within the enterprise. Dead data occurrences are unreferenced data items or data items only referenced in DEADCODE.

Figure 61 • Enterprise Exceptions Report

(A)		ENTERPRISE EXCEPTIONS							DD-MMM-YYYY HH:MM:SS PAGE 217	
ASG-RECAP-OS (ESA) Rx.x LVLxxx		AKR NAME: ASG.RCPSAMP.AKR								
APPLICATION NAME	NUMBER OF PROGRAMS	NUMBER OF GO TOS	NUMBER OF ALTERS	NUMBER OF A-ENTRIES	NUMBER OF A-EXITS	NUMBER OF RECURSION	NUMBER OF JUMPS	NUMBER OF LIVE EXIT	LINES OF DEAD CODE	LINES OF DEAD DATA
ACCT-PAYBL	36	1510	19	0	2	0	5	0	1235	2
ACCT-RCVBL	43	1602	29	0	3	0	88	3	1050	3
CUST-BILL	43	1621	5	0	2	0	12	0	1191	4
CUSTOMER	33	1675	49	0	1	0	54	3	1398	7
EMPL-INFO	35	1437	5	0	7	0	177	0	1244	2
INVENTORY	34	1551	3	1	2	0	0	0	897	7
MANF-CNTL	36	2665	145	1	2	14	1496	0	997	3
PAYROLL	36	1723	0	0	2	0	139	3	1460	0
SALES	31	3864	20	1	6	3794	10256	571	1100	3
TELEPHONE	6	45	0	2	31	1	8	2	23	0
(C)		ENTERPRISE TOTAL/SUMMARY (NUMBER OF APPLICATIONS = 10, NUMBER OF PROGRAMS = 333)								
EXCEPTIONS		TOTAL								
-----		-----								
GO TOS		17693								
ALTERS		275								
ANOMALOUS ENTRIES		5								
ANOMALOUS EXITS		58								
RECURSIONS		3809								
OUT OF PERFORM RANGE JUMPS		12235								
LIVE EXITS		582								
DEAD CODE		10595								
DEAD DATA		31								
LEGEND: JUMPS = OUT OF PERFORM RANGE JUMPS A-ENTRIES = ANOMALOUS ENTRIES A-EXITS = ANOMALOUS EXITS										

Report Field Descriptions

Field	Description
(A)	The names of the applications in the AKR.
(B)	These columns list the number and type of exceptions found in the application.
(C)	The Enterprise Total/Summary table lists the enterprise totals for each type of exception. The total number of applications and programs in the enterprise is also provided.

Master Index

The Master Index, shown in [Figure 62](#), is an alphabetical listing of all named entities in the Recap reports. This report provides a quick reference to determine where a particular entity is located within the Recap reports.

Figure 62 • Master Index

ASG-RECAP-OS (ESA) Rxx.x LVLxxx		MASTER INDEX		DD-MMM-YYYY HH:MM:SS PAGE 218
		APPLICATION: SALES		
(A) ENTITY NAME	(B) DEFINITION	(C) LOCATION (REPORT-PAGE NUMBER)		
ACCT-PAYBL	APPLICATION NAME	AC-210, AC-212, AP-213, EM-216, EE-217		
ACCT-RCVBL	APPLICATION NAME	AC-210, AC-212, AP-213, EM-216, EE-217		
CM431	PROGRAM NAME	AD-2, FP-3, FS-99, MH-101, PC-195, PC-197, PP-198, AM-206, AE-208		
CUST-BILL	APPLICATION NAME	AC-210, AC-212, AP-213, EM-216, EE-217		
CUSTOMER	APPLICATION NAME	AC-210, AC-212, AP-213, EM-216, EE-217		
DB204	PROGRAM NAME	AD-2, FP-6, FS-99, MH-104, PC-195, PC-197, PP-198, AM-206, AE-208		
EMPL-INFO	APPLICATION NAME	AC-210, AC-212, AP-214, EM-216, EE-217		
E02031	PROGRAM NAME	AD-2, FP-10, FS-99, MH-107, PC-195, PC-197, PP-198, AM-206, AE-208		
GENAMER1	PROGRAM NAME	AD-2, FP-13, FS-99, MH-110, PC-195, PC-197, PP-198, AM-206, AE-208		
IC110	PROGRAM NAME	AD-2, FP-16, FS-99, MH-113, PC-195, PC-197, PP-199, AM-206, AE-208		
IC120	PROGRAM NAME	AD-2, FP-19, FS-99, MH-116, PC-195, PC-197, PP-199, AM-206, AE-208		
IC207	PROGRAM NAME	AD-2, FP-22, FS-99, MH-119, PC-195, PC-197, PP-199, AM-206, AE-208		
IC217	PROGRAM NAME	AD-2, FP-25, FS-99, MH-122, PC-195, PC-197, PP-199, AM-206		
INVENTORY	APPLICATION NAME	AC-210, AC-212, AP-214, EM-216, EE-217		
IX203	PROGRAM NAME	AD-2, FP-28, FS-99, MH-125, PC-195, PC-197, PP-200, AM-206, AE-208		
IX214	PROGRAM NAME	AD-2, FP-31, FS-99, MH-128, PC-195, PC-197, PP-200, AM-206, AE-208		
KC601	PROGRAM NAME	AD-2, FP-34, FS-99, MH-131, PC-195, PC-197, PP-200, AM-206, AE-208		
MANF-CNTL	APPLICATION NAME	AC-210, AC-212, AP-214, EM-216, EE-217		
NC110	PROGRAM NAME	AD-2, FP-39, FS-99, MH-134, PC-195, PC-197, PP-200, AM-206, AE-208		
NC119	PROGRAM NAME	AD-2, FP-42, FS-99, MH-137, PC-195, PC-197, PP-201, AM-206, AE-208		
NC154	PROGRAM NAME	AD-2, FP-45, FS-99, MH-140, PC-195, PC-197, PP-201, AM-206, AE-208		
NC165	PROGRAM NAME	AD-2, FP-48, FS-99, MH-143, PC-195, PC-197, PP-201, AM-206, AE-208		
NC209	PROGRAM NAME	AD-2, FP-51, FS-99, MH-146, PC-195, PC-197, PP-201, AM-206, AE-208		
NC217	PROGRAM NAME	AD-2, FP-54, FS-99, MH-149, PC-195, PC-197, PP-202, AM-206, AE-208		
NC431	PROGRAM NAME	AD-2, FP-57, FS-99, MH-152, PC-195, PC-197, PP-202, AM-206, AE-208		
PAYROLL	APPLICATION NAME	AC-210, AC-212, AP-214, EM-216, EE-217		
RL151	PROGRAM NAME	AD-2, FP-60, FS-99, MH-155, PC-195, PC-197, PP-202, AM-206, AE-208		
RL431	PROGRAM NAME	AD-2, FP-63, FS-99, MH-158, PC-195, PC-197, PP-202, AM-206, AE-208		
SALES	APPLICATION NAME	AD-2, AC-210, AC-212, AP-215, EM-216, EE-217		
SG203	PROGRAM NAME	AD-2, FP-66, FS-99, MH-161, PC-195, PC-197, PP-203, AM-206, AE-208		
SQ110	PROGRAM NAME	AD-2, FP-69, FS-99, MH-164, PC-195, PC-197, PP-203, AM-206, AE-208		
SQ120	PROGRAM NAME	AD-2, FP-72, FS-99, MH-167, PC-195, PC-197, PP-203, AM-206, AE-208		
SQ205	PROGRAM NAME	AD-2, FP-75, FS-99, MH-170, PC-195, PC-197, PP-203, AM-206, AE-208		
SQ215	PROGRAM NAME	AD-2, FP-78, FS-99, MH-173, PC-195, PC-197, PP-204, AM-206, AE-208		
ST110	PROGRAM NAME	AD-2, FP-81, FS-99, MH-176, PC-195, PC-197, PP-204, AM-206, AE-208		
ST201	PROGRAM NAME	AD-2, FP-84, FS-99, MH-179, PC-195, PC-197, PP-204, AM-206, AE-208		
ST211	PROGRAM NAME	AD-2, FP-87, FS-99, MH-182, PC-195, PC-197, PP-204, AM-206, AE-208		
ST441	PROGRAM NAME	AD-2, FP-90, FS-99, MH-185, PC-195, PC-197, PP-205, AM-206, AE-208		
TELEPHONE	APPLICATION NAME	AC-210, AC-212, AP-215, EM-216, EE-217		
TH205	PROGRAM NAME	AD-2, FP-93, FS-99, MH-188, PC-195, PC-197, PP-205, AM-206, AE-208		
TH215	PROGRAM NAME	AD-2, FP-96, FS-99, MH-191, PC-195, PC-197, PP-205, AM-206, AE-208		

(D)		
LEGEND:	AC = APPLICATION COMPARISON	AP = APPLICATION PROGRESS
	AD = APPLICATION DEFINITION	AE = APPLICATION EXCEPTIONS
	AE = APPLICATION EXCEPTIONS	EM = ENTERPRISE METRICS
	AM = APPLICATION METRICS	FP = PROGRAM FUNCTION POINT ANALYSIS
		FS = APPLICATION FUNCTION POINT ANALYSIS
		MH = PROGRAM METRICS HISTORY
		PC = PROGRAM COMPARISON
		PP = PROGRAM PROGRESS

Report Field Descriptions

Field	Description
(A)	Each entity in the enterprise is listed.
(B)	Indicates whether the entity is a program or application.
(C)	Lists the report names and report page numbers where references to the entities can be found.
(D)	Legend of the symbols used to identify the report on which the entity is referenced.

7

Report Generator Language

This chapter provides ways to adjust or change report content and format to meet your organization's requirements using Recap's Report Generator Language, and contains these sections:

Section	Page
Description of Programming Language	143
Site Specific Configuration	143
Data Structures	144
Language Constructs	145
Data Elements - Table and Field Descriptions	151

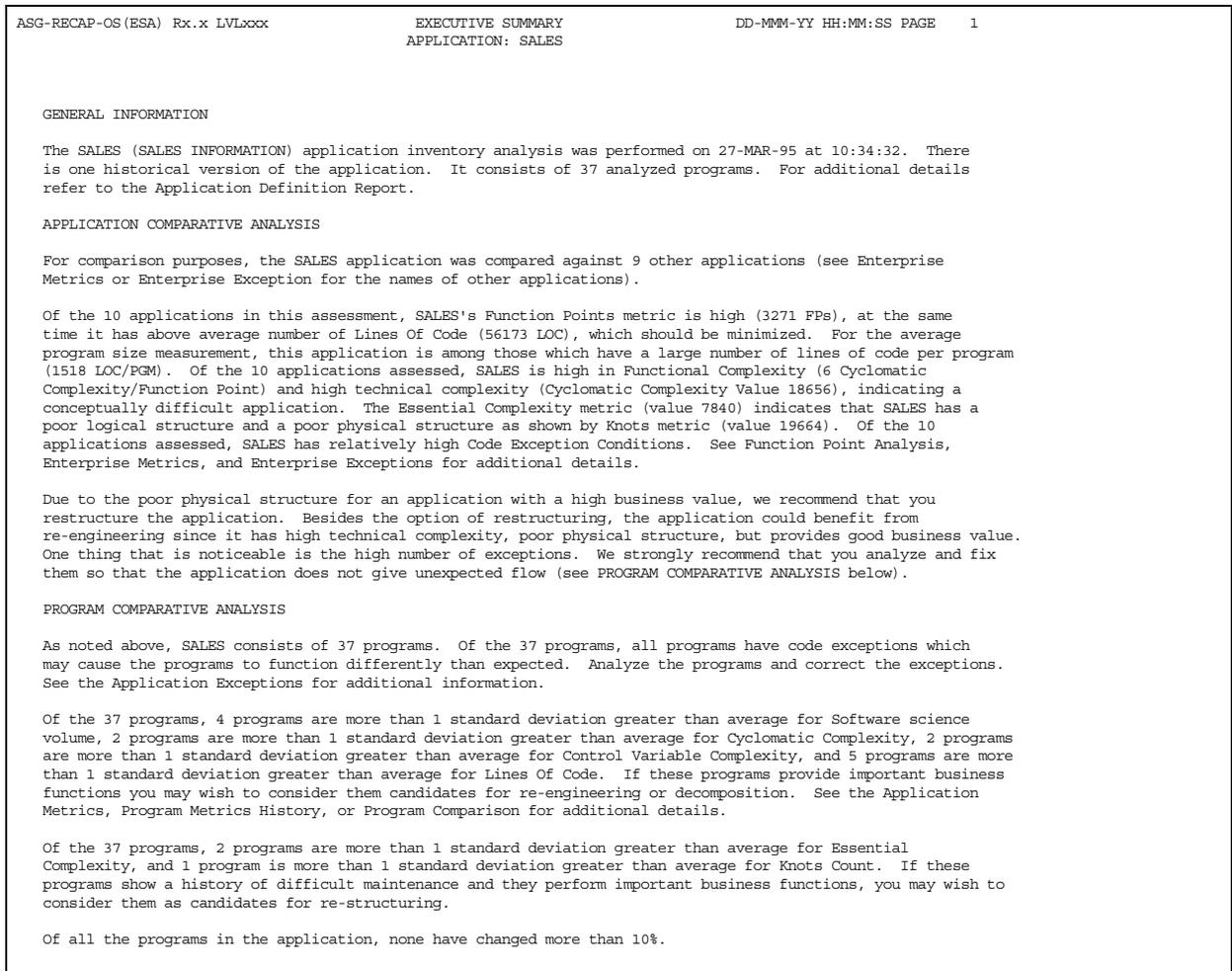
The Executive Summary report summarizes the portfolio analysis information provided in the Recap reports and provides recommendations based on this information. Initially, the Executive Summary report is generated from a preprogrammed file shipped with Recap. (See the *ASG-Recap Installation Guide* for more information about this file.) This file uses the Report Generator programming language that is also included with the product.

If necessary, the report contents and format can be adjusted or changed to meet the requirements of your organization by using the Report Generator programming language described in this chapter.

The Executive Summary report is printed in mixed case letters by default. If you want to print the report in upper case only, see "[Generating Recap Reports](#)" on page 44.

[Figure 63](#) shows the Executive Summary Report generated from the preprogrammed file.

Figure 63 • Executive Summary Report



Description of Programming Language

The Report Generator programming language has limited functionality, yet provides the root capabilities found in all high level languages. It is interpretive and intermingled with the input data. The key components are a tokenizer, instruction parser, and syntax checker driven from a single layer controller. The language parser distinguishes instruction tokens from text tokens. Input is considered as a constant stream of tokens, with comment lines denoted by an asterisk in the first column. The instructions are fixed format but positionally independent.

Site Specific Configuration

The data files VIAXESUM and VIAXESUU contain a section clearly labeled Site Specific Configuration. This section is used to set up site specific configuration for the data file. The Site Specific Configuration section is located at the beginning of the data file and may optionally be preceded by an unlimited number of comment records. The end of the Site Specific Configuration section is denoted by the words end Site Specific Configuration.

These instructions are located inside the Site Specific Configuration section:

Instruction	Description
Standard Deviation Difference	If any metrics or exception counts for a program or application differs from the average by more than the specified statistical standard deviation, then the data for that program or application is computed and ranked accordingly. The default value is 1 standard deviation from the average. You may want to increment this number if the standards requirements of your organization are less stringent. Setting the Standard Deviation Difference to a higher number, 2 for example, changes the ranking of a particular metric or exception to out of norm when the value differs by 2 or more standard deviations.

Instruction	Description
Percent Change	The Percentage Change is a threshold value used to determine which programs have changed significantly since the last analysis. If Percent Change is greater than the threshold Percent Change, the program is marked as changed significantly. Initially, the threshold is set at 10% and all programs with metrics or exception values that have changed more than 10% are counted as changed significantly. You may want to change this number to reflect the standards of your organization.
Identifier Character	The Identifier Character is a single character used to differentiate data fields and keywords from text. The identifier must be the first character of the data field or instruction. The character used as the identifier must not be used anywhere else in the data file. The default is an ampersand (&). If you need to use an ampersand (&) in the data file, you can change the default. Be sure that the new identifier you choose is not used for any other purpose in the data file.

Data Structures

Primary input to the Executive Summary Report Generator System is a free-format data file containing language text. The programming and formatting instructions are embedded in this file. Secondary input is all data collections that are used as input to the metrics report programs.

Data Element Syntax

Data elements are differentiated from text by the first character: all data elements must start with an identifier. Because data element names are differentiated by the identifier, the identifier (default = &) cannot be used as a printable word in the output file. See "Site Specific Configuration" on page 143 for information about the identifier.

Each data element has a variable data type. If the value assigned to an element is a string, the data type is a string. If the value assigned is numeric, the data type is numeric. The data type is automatically changed to correspond to the value assigned to the element. For example, if `&appl_total` has the value 8, the data type is numeric. If you reassign `&appl_total` a value of 'eight', the data type automatically changes to string. The data element names are limited to 40 characters in length; the values are limited to the width of a printed page (132 characters).

Primary and Subsidiary Tables

The data collections used as input to the programming language are available as individual elements in a primary table or as elements in subsidiary tables. Elements in subsidiary tables are accessed by indexing elements in the primary table. Each element of every table is available for inspection or modification at any time.

Each table contains 100 elements. The programmer may overwrite the default or pre-computed value of any element. Elements can be added to a table, but new tables cannot be created. The value of an element is retrieved from this table by specifying its name. For example, to retrieve the name of the application, you would enter the `&appl_name` data element as a token in the data file.

Accessing Elements in Subsidiary Tables

The elements of subsidiary tables are accessed through dereferencing. Table dereferencing or subscripting is achieved by specifying an index or offset, as is done in all generic programming languages. The index follows the data element name and is enclosed in brackets (`{}`). The index can be a numeric constant or another data item, either numeric or alphanumeric. For example, `&metric_history {&essential}` is a reference to the table named `&metric_history`, with an offset or index to the entry denoted by `&essential`. If you know that `&essential` is the tenth element in the table, the same results could be achieved by coding `&metric_history {10}`.

Embedded spaces between table names and the offset notational characters are optional and are used for improved readability only.

Note: _____

See “Data Elements - Table and Field Descriptions” on page 151 for descriptions of the individual tables and elements.

Language Constructs

The programming language intermixes the English text and the programming instructions. Therefore, data and programming elements must be differentiated from text elements using the identifier specified in the site specific configuration section. The default identifier is an ampersand (`&`).

The language consists of four distinct components:

- [Text Substitution](#).
- [Data Manipulation](#).
- “Control and Text Selection” on page 147.
- “Output Formatting” on page 149.

Text Substitution

When a data element is used as a token, the value of the element is automatically substituted. Therefore, to print the value of a data element you only have to specify the data element name. For example, assume the name of an application is Payroll, and that this application has 12 defined programs.

To output the text:

```
The Payroll application consists of 12 programs
```

The input file contains:

```
The &appl_name application consists of &pgm_total programs
```

Data Manipulation

The data manipulation instructions allow the programmer to set the values of elements in the data domain using these keywords:

&assign

The `&assign` instruction assigns a value to a data item. If the data item is defined, its value is overwritten. If it is undefined, an instance is created. An element can be created or stored in a subsidiary table; in that case the primary entry must be a reference to a pre-defined subsidiary table since tables cannot be created. The type of data can be changed from numeric to string or vice versa. This is the syntax for the `&assign` instruction:

```
&assign  item = value  &endassign
```

where `item` is the name of the data element to receive `value`. The `=` operator and `&endassign` keyword are mandatory. String values are represented as single (') or double quote (") delimited values, such as "this is a string".

These examples illustrate the format for `&assign` instructions:

```
&assign  &user_metric_index = 8                                &endassign  
&assign  &temp_string = "User defined metric"                &endassign  
&assign  &metric_name {&user_metric_index}=&temp_string      &endassign  
&assign  &metric_current {&user_metric_index} = 0            &endassign
```

&compute

The `&compute` instruction computes values of numeric instances. If the data item is undefined, an instance is created as with `&assign`. If it is defined, its value is overwritten. The type of data is numeric after the computation is made, regardless of its original type. This is the format of the instruction:

```
&compute  item = operand1 operator operand2  &endcompute
```

where:

item is the name of the data element to receive the value of the computation.

The = operator is mandatory.

operand1 and *operand2* are both existing data elements; either may be a numeric constant. Each of these operands may be a dereferenced data element (table lookup). The *item* field itself may be a dereferenced data element.

The *operator* must be one of the primitive numeric operators +, -, *, or /, for addition, subtraction, multiplication or division respectively. This example illustrates the format for an `&compute` instruction to compute the complexity density metric.

```
&compute &complexity_density =
           &metric_current {&cyclomatic}/&metric_current{&volume}
&endcompute

&compute &metric_current {&user_metric_index} =
           &complexity_density * &metric_current {&fpa}
&endcompute
```

Control and Text Selection

The control and text selection constructs allow the programmer to selectively output text. Programming instructions must also start with a character identifier. The format of these instructions is fixed, but they can appear anywhere in the text file. The tokens in the instruction may be specified on one line, but we recommend that you follow standard programming format guidelines to improve program readability.

&if

The `&if` instruction allows the programmer to test the values of two data elements and conditionally select text based upon the result. This example illustrates the format for `&if` instructions.

```
&if condition &then ... &else ... &endif
```

where *condition* is further defined as:

operand1 operator operand2

Operand1 and *operand2* are both existing data elements; either may be a numeric or alphanumeric constant. The ordinal type of the operands must be the same. The *operator* depicts the type of operation to perform on the two operands. Supported operators are:

Operator	Corresponding Definition
= or EQ	EQUAL
< or LT	LESS THAN
> or GT	GREATER THAN
<= or LE	LESS THAN OR EQUAL TO
>= or GE	GREATER THAN OR EQUAL TO
<>, ~=, !=, ¬, or NE	NOT EQUAL

The operator has the standard programming language interpretation. The keyword `&then` is optional. The `&else` portion of the instruction is optional as it is with all generic programming languages. The `&endif` keyword is mandatory. `&if` instructions can be nested to indefinite levels.

This example illustrates how the `&if` instruction can be used to control the report's output:

Assume you want to output the text "The complexity density is acceptable" if the programmatic complexity density value is less than some computed average value, otherwise you want to output the text "The complexity density is higher than the norm". The input data file may then contain instructions such as:

```
&compute    &complexity_density =
            &metric_current{&cyclomatic} / &metric_current{&volume}
&endcompute
```

The complexity density is

```
&if    &complexity_density < &average_complexity    &then
      acceptable
&else
      higher than the norm
&endif
```

&while

The `&while` instruction allows the programmer to test the values of two data elements and conditionally repeat a section of the input file based upon the result of the comparison. The `&while` instruction serves the same purpose as `while` or `loop` instructions in regular programming languages. This is the format for the `&while` instruction:

```
&while    condition ...    &endwhile
```

where *condition* has the same interpretation as for the `&if` instruction.

The `&endwhile` keyword is mandatory. If *condition* yields true, the text following the condition up to the corresponding `&endwhile` token is interpreted, otherwise parsing is resumed at the token following the `&endwhile`. `&while` instructions can be nested up to 30 levels deep, and may contain or be contained inside `&if` instructions.

This example illustrates use of the `&while` instruction:

Assume you want to output, in tabular format, the names and values of the software metrics. The metric name is 20 characters long and the metric value is 5 characters wide. You know that there are 7 different metrics. The programming code may then contain this code:

```
&assign    &loop_index = 1    &endassign

&while    &loop_index <= 7
    &metric_name {&loop_index}:20    &metric_current
{&loop_index}:5
    &compute    &loop_index = &loop_index + 1
    &endcompute
&endwhile
```

Output Formatting

The programming language also controls the formatting of the output to ensure readability. It handles word spacing, tabular output, line overflow and, page breaks. Periods, question marks, and exclamation marks are always followed by two spaces. Embedded spaces before periods, commas, question marks, exclamation marks, colons, and semi-colons are suppressed. In addition, there are several instructions available to assist in formatting the output.

Output Length Control and Occurrence Count (:n)

The length of output tokens is controllable. The length is specified by placing a colon and a numeric value representing the length after the data element name. Spaces are not allowed between the data element name and the colon. This numeric value may be a constant or a numeric data element. For example, to format the output of a data item to 10 characters, code it as `&data_name:10`. The value of the `&data_name` token is padded with as many spaces as required to make the total length 10 characters. Usage such as `&data_name:&length` is accepted.

Text strings are left justified and numeric values are right justified within the specified width. If the actual length of the substituted element is more than the specified length, then the length is ignored. This means that substituted values may exceed the specified length. These modes of operation are in conformance with standard programming practice.

The same functionality is available to some of the formatting instructions described below. Use the notation `:n` following the instruction or control name to indicate that a length or occurrence `n` can be specified. This length override is optional unless explicitly documented to the contrary.

&newline

The `&newline` instruction forces the next output token to appear on the next line in the output file or report.

¶graph

The `¶graph` instruction is the equivalent of two `&newline` instructions and is provided for ease of formatting only.

&space

The `&space:n` instruction is used to output a specified number of spaces. This instruction is required since multiple embedded spaces in the input file are collapsed to a single space. Moreover, this instruction accepts 0 as a length. This causes the prior token to be followed by 0 spaces, that is, the prior and following tokens are output without any spaces in between. The 0 length parameter is seldom used. An occurrence count is mandatory; a descriptive error message is produced if the `:n` parameter is omitted.

&lmargin and &rmargin

The left and right margins of the output are controlled by the `&lmargin:n` and `&rmargin:n` instructions respectively. The parameter `:n` is required and indicates the column at which the next line must start or may not exceed. You can use these instructions to force the margins to the specified columns. The default values are 3 for the left margin, and the width of a printed page for the right margin (132).

&col

The `&col : n` instruction causes a tab to the specified column number. The value of `: n` can range from a number larger than the left margin to a number less than the right margin. The next token is output starting at column number `n`. A `&newline` instruction is internally issued if the current line in the output buffer is already longer than the specified column number. The `&col` instruction is required to produce tabular output.

Data Elements - Table and Field Descriptions

The elements used as input to the programming language are available in a primary table and subsidiary tables. The elements in subsidiary tables are accessed by indexing elements in the primary table.

These tables show the relationship between elements in the primary table and elements in the subsidiary table. Descriptions of the elements in the tables are provided in the section that follows each table.

The elements in the primary table are grouped into these categories: General, Other Applications, Metrics Values and Names, Application Ranking, Non-Conformance, and Exceptions.

Primary Table	
Data Elements	Index
General:	
<code>&anzl_date</code>	none
<code>&anzl_time</code>	none
<code>&appl_desc</code>	none
<code>&appl_name</code>	none
<code>&appl_total</code>	none
<code>&changed_pgm_ct</code>	none
<code>&exception_pgm_ct</code>	none
<code>&exception_rank</code>	none
<code>&pgm_total</code>	none

Primary Table	
Data Elements	Index
&percent_change	none
&unchanged_pgm_ct	none
&std_dev_diff	none
&version	none
Other Applications:	
&other_appl_names	(A)
&other_appl_descs	(A)
Metrics Values and Names:	
&metric_current	(B1 - B10)
&metric_history	(B1 - B7, and B10)
&metric_name	(B1 - B7, and B10)
Application Ranking:	
&metric_rank	(B1 - B10)
Non-Conformance:	
&non_conform_ct	(B1 - B4, B6 - B7)
Exceptions Values and Names:	
&exception_current	(C)
&exception_name	(C)

The sections in the subsidiary tables are grouped into these categories: Other Application Names and Descriptions, Metrics, and Exceptions.

Subsidiary Tables

Table A - Other Application Names and Descriptions

Subsidiary Tables

There are (&appl_total - 1) names and (&appl_total - 1) descriptions.
Use numeric indexing to access individual elements.

Table B - Metrics

1	&volume
2	&cyclomatic
3	&essential
4	&control_var
5	&fpa
6	&knots
7	&appl_line_tot
8	&avg_lines-pgm
9	&cyc_per-fpa
10	&fpe

Table C - Exceptions

&gotos
&alters
&entriesl
&exits
&recursions
&perf_jumps
&live_exits
&dead_code
&dead_data

The index listed after each primary element represents the subsidiary table(s) used to index the primary element. For example, the `&metric_current` element has an index of {B}. To obtain current metric values, index `&metric_current` using elements from the Metrics Table. To print the current metrics value for cyclomatic complexity, the input file would contain this "The application has a cyclomatic complexity of `&metric_current{&cyclomatic}`." General elements are not indexed.

Primary Table Elements

General

To get the value of a general element specify the element name. If it appears as part of the text to output, its value is substituted. If it appears as part of a conditional expression its value is used during the comparison.

Data Element	Description
<code>&appl_name</code>	Application name
<code>&appl_desc</code>	Application description
<code>&anzl_date</code>	Date of last analysis
<code>&anzl_time</code>	Time of last analysis
<code>&pgm_total</code>	Number of programs in the application successfully analyzed
<code>&appl_total</code>	Number of applications in the enterprise
<code>&exception_pgm_ct</code>	Number of programs that have at least one exception
<code>&unchanged_pgm_ct</code>	Number of programs that differ less than n% from the last analysis (n is set in the site specific configuration section of the program.)
<code>&changed_pgm_ct</code>	Number of programs that differ more than n% from the last analysis (n is set in the site specific configuration section of the program.)
<code>&std_dev_diff</code>	Standard deviation difference read from the site specific configuration.
<code>&percent_change</code>	The percentage change read from the site specific configuration.

Data Element	Description
<code>&version</code>	Number of historical application versions.
<code>&exception_rank</code>	Application exception rank by the total number of exceptions compared with other applications within the enterprise.

Other Application Names and Descriptions

The names and descriptions of the other applications in the enterprise definition are available by numeric indexing. There are (`&appl_total - 1`) names and descriptions for other applications. The current application name and description are available in `&appl_name` and `&appl_desc` respectively, and the number of applications is available in `&appl_total`.

Data Element Description:

`&other_appl_names`

Table with names of other applications in the enterprise.

`&other_appl_descs`

Table with descriptions of other applications in the enterprise.

Metric Values and Names

These elements are indexed by the elements in the Metrics Table. To get the value, you must specify the name of the table along with the index offset into the table. For example, to print the name and current value of the Essential Complexity metric, you need this code:

Metric name: `&metric_name {&essential}`

Current value: `&metric_current {&essential}`.

Data Element Description:

`&metric_current`

Table of current metric values.

`&metric_history`

Table of previous metric values (last history).

`&metric_name`

Table of metrics names.

Application Ranking

The comparative ranking of applications in an enterprise returns a value that indicates how that application ranks when compared against the others in the enterprise. The values returned are 1 (high), 2 (average), and 3 (low). The `&metrics_rank` element ranks the metrics values and is indexed by the Metrics Table. The `&exception_rank` element ranks the count of all exceptions.

Data Element Description:

`&metric_rank`

Comparative ranking of all applications in the enterprise by individual metric values.

Non-Conformance

The number of programs that are non-conformant indicates how many programs exceed the standard deviation for their metric values. For example, to print the number of programs with Control Flow Knots metric value farther from the average than the standard deviation boundary specified in the site specific configuration section in its Knots metric, you may code:

```
"There are &non_conform_ct {&knots} programs with a Knots count  
greater than  
  
&std_dev_diff from the average."
```

Data Element Description:

`&non_conform_ct`

Number of programs that are non-conformant.

Exception Values and Names

These elements are indexed by elements in the exceptions table. To get the value, you must specify the name of the table along with the index offset into the table. For example, to print the name and current value of the `live_exits` exception, you need this code:

```
Exception name: &exception_name {&live_exits}  
Current value: &exception_current {&live_exits}.
```

Data Element Description:

`&exception_current`

Table that contains the current exception values.

&exception_name

Table that contains the names of exception conditions.

Subsidiary Tables

Metrics Table

The elements in this table are accessed by indexing an element in one of these primary tables: Metrics Values and Names, Application Ranking, and Non-Conformance.

Data Element	Description
<i>&volume</i>	Software Science value
<i>&cyclomatic</i>	Cyclomatic Complexity value
<i>&essential</i>	Essential Complexity value
<i>&control_var</i>	Control Variable Complexity value
<i>&fpa</i>	Function Points value
<i>&knots</i>	Control Flow Knots count
<i>&appl_line_tot</i>	Total number of lines of code in the application
<i>&avg_lines_pgm</i>	Average lines of code per program
<i>&cyc_per_fpa</i>	Cyclomatic Complexity value/Function Point value
<i>&fpe</i>	Function Point enhancement value

Exceptions Table

The elements in this table are accessed by indexing an element in the Exception Values and Names Table.

Data Element	Description
<i>&gotos</i>	The number of GO TO statements
<i>&alters</i>	The number of ALTER statements
<i>&entries</i>	The number of entry points

Data Element	Description
&exits	The number of program exits
&recursions	The number of recursive code occurrences
&perf_jumps	The number of perform range jumps (perform range violations)
&live_exits	The number of live exit occurrences
&dead_code	The number of lines of dead code
&dead_data	The number of unreferenced data definitions

Other Application Names and Other Application Descriptions Table

These elements are used to index the Other Application Names and Descriptions elements. The values for the other application names and other application descriptions are defined when the other applications have been analyzed successfully. The number of elements in the table is one less than the total number of applications (&appl_total).

8

Import Facility

This chapter gives detailed information about the Import Facility and how it allows you to import an application definition into the AKR, and contains these sections:

Section	Page
Importing an Application Definition	159
Import Language Reference	160
Import Language Keywords	177

The Import Facility allows you to import an Application definition into the AKR. To prepare the application definition to be imported, you must edit an existing file or create a new file, inserting keywords from the Import Language described in this chapter. These keywords identify the components and attributes for each library and member included in the file and to convert the application definition to the structure required by the AAE. To import the Application definition you create, select File ► Import/Export.

Importing an Application Definition

This procedure describes the steps required to import an application definition into the AKR.

- 1 Using the Import Language described in "[Import Language Reference](#)" on page 160, a file must exist containing the keywords to identify the application information, defaults, and the components and attributes of the application definition.
- 2 Save the file.
- 3 Start the product using the procedure defined for your site (see "[Initiating a Recap Session](#)" on page 29).
- 4 Select File ► Import/Export to display the File - Import/Export pop-up.
- 5 From the File - Import/Export pop-up, select Import Definition to display the File - Import Definition pop-up.

- 6 In the Data set name field, enter the name of the dataset that contains the import file you created in [step 1](#). If the high-level qualifier for the dataset name is different than the TSO user ID, enter the name in single quotes (' USER . TEST . APPLDEF ').
- 7 In the Member Name field, enter the name of the member containing the import information. Press Enter to display this message:

APPLICATION <application name> IMPORTED SUCCESSFULLY.

If the import does not complete successfully, verify the dataset and member name for the import file. If the names are correct, review the structure and syntax of the import file to locate any errors or invalid data input.

- 8 To open the application, select File ► Open application to display the File - Open Application pop-up. The name of the AKR and application specified in the import file automatically displays. Press Enter to open the application.

Import Language Reference

Import Language Syntax

Import statements may start anywhere on the line. Keywords are enclosed by the tags < >. Each keyword must be coded on a separate line. Each item associated with a keyword must also be coded on a separate line as shown in [Figure 64](#):

Figure 64 • Import Data Item Syntax

```
<AKR>
  USER . TEST . AKR
<APPLICATION NAME?
  MY-APPL
```

Import File Structure

The Import File for the application definition contains three major sections:

- Application Information (see [Application Information](#)).
- Default information (see ["Default Dataset and Attribute Information" on page 161](#)).
- Definition (see ["Definition" on page 166](#)).

Application Information

This section specifies the name of the AKR where the application is stored along with the application name and application description. Application information can be specified as:

AKR

Required. Identifies the dataset name of the AKR where the application is stored.

PRODUCT

Indicates whether the application definition is used in Recap, Alliance, or both. If both products are specified, separate the entries by a comma. Valid entries: ALLIANCE, RECAP.

APPLICATION NAME

Required. Specify a one to ten character name for the application you are defining.

APPLICATION DESCRIPTION

Optional. Specify a one to forty character description.

[Figure 65](#) shows the code for the Application Information section of an import file.

Figure 65 • Application Information Example

```
<AKR>
  USER.TEST.AKR
<PRODUCT>
  RECAP
<APPLICATION NAME>
  IMP-FULL
<APPLICATION DESCRIPTION>
  COMPLETE APPLICATION TEST
```

Default Dataset and Attribute Information

This section is used to specify the default information for the libraries defined to the application. All defaults are optional. The defaults specified in the import file override any corresponding site defaults that were specified during the installation of Recap or Alliance. The default section must begin with the keyword DEFAULT START and end with the keyword DEFAULT END.

This default information can be specified:

FILEORG

Identifies the default source manager that is used for all COBOL, JCL, CICS, and IMS datasets included in the application definition. Only one source manager can be specified. Valid Entries: PDS, PAN, LIB, VSAM, SEQ, or ENDV.

COBOL VERSION

Identifies the default COBOL version for COBOL libraries and members defined to the application. Only one version can be specified. If the COBOL version is not specified, the COBOL version specified during product installation is used by default. Valid Entries: COB68, COB74, COB2, COB2R3, COB370.

COMPILE EXECUTION PARM

Identifies the default compile batch execution parameters used for COBOL libraries defined to the application.

COBOL SOURCE EXTRACTION EXECUTION PARM

Identifies the default source library manager batch execution parameters used for COBOL libraries defined to the application.

JCL SOURCE EXTRACTION EXECUTION PARM

Identifies the default source library manager batch execution parameters used for JCL libraries defined to the application.

CICS SOURCE EXTRACTION EXECUTION PARM

Identifies the default source library manager batch execution parameters used for CICS libraries defined to the application.

IMS SOURCE EXTRACTION EXECUTION PARM

Identifies the default source library manager batch execution parameters used for IMS libraries defined to the application.

STDSQL(86)

Identifies the default SQL usage found in the JCL. If the DB2 STDSQL(86) parm is found in the compile/link JCL, the default is YES. Valid entries: YES, NO.

With STDSQL(86) or STDSQL(YES), the SQLCA is expanded where the definition of SQLCODE is found in the program and SQLCADE replaces the SQLCODE field in the SQLCA structure.

```
01 SQLCODE    PIC X9(9) COMP.
01 SQLCA.
    05 SQLCAID PIC X(8).
    05 SQLCABC PIC S9(9) COMPUTATIONAL.
    05 SQLCADE PIC S9(9) COMPUTATIONAL.
```

...

With STDSQL(NO) EXEC SQL SQLCA END-EXEC. Expands the SQLCA and SQLCODE is field in the SQLCA.

```
EXEC SQL INCLUDE END-EXEC.
01 SQLCA
    05 SQLCAID PIC X(8).
    05 SQLCABC PIC S9(9) COMPUTATIONAL.
    05 SQLCADE PIC S9(9) COMPUTATIONAL.
```

...

COPYLIB

Identifies the default copy library attributes for COBOL libraries and members defined to the application. Valid entries: Names of valid copy libraries; each library name must appear on a separate line.

These are attributes of the default copylib. Each attribute must appear on a separate line after the valid default copy library entry.

Attribute	Description
COPY FILEORG	Identifies the source manager that is used for the specified copy library. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB.
COPY SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO.

Attribute	Description
COPY PASSWORD	Identifies the password for the specified copy library. If the library is password protected, enter the password.
COPY VOLSER	Identifies the volume serial number for the specified copy. If the library or member is not cataloged, enter the volume serial number.

COPY USAGE

Identifies the location of the SYSLIB dataset in the compile JCL. Usage information is used to determine when the AAE needs to use the copy libraries defined to the application: during COBOL library processing, during DB2 include processing, or during both types of processing. Only one COPY USAGE can be specified. Valid entries: COB, DB2, BOTH.

CA-LIBRARIAN MEMBER COPYDD

Indicates whether the CA-Librarian master file for CA-Librarian members in the application definition requires the CopyDD feature. If this value is not specified, the default value is NO. Valid entries: YES, NO.

CA-LIBRARIAN MEMBER PSWD EXISTS

Indicates whether the CA-Librarian members in the application are password-protected. If this value is not specified, the default value is NO. Valid entries: YES, NO.

IMS MACLIB

Identifies the default maclibs for IMS libraries included in the application definition. Valid entries: Names of valid maclibs; each library name must appear on a separate line.

These are attributes of the default maclib for IMS members. Each attribute must appear on a separate line after the entry for the valid default maclib.

Attribute	Description
IMS MACLIB FILEORG	Identifies the source manager that is used for the specified maclib. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB.
IMS MACLIB SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO.
IMS MACLIB PASSWORD	Identifies the password for the specified macro library. If the library is password protected, enter the password.
IMS MACLIB VOLSER	Identifies the volume serial number for the specified macro library. If the library or member is not cataloged, enter the volume serial number.

CICS MACLIB

Identifies the default maclibs for CICS libraries included in the application definition. Valid entries: Names of valid maclibs; each library name must appear on a separate line.

These are attributes of the default maclib for CICS members. Each attribute must appear on a separate line after the entry for the valid default maclib.

Attribute	Description
CICS MACLIB FILEORG	Identifies the source manager that is used for the specified maclib. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB.
CICS MACLIB SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO.
CICS MACLIB PASSWORD	Identifies the password for the specified macro library. If the library is password protected, enter the password.
CICS MACLIB VOLSER	Identifies the volume serial number for the specified macro library. If the library or member is not cataloged, enter the volume serial number.

Note:

For more information on specifying maclibs in the application definition, see ["Analyzing the Application" on page 44](#).

PROCLIB

Identifies the default procedure libraries for JCL libraries and members included in the application definition. Valid entries: Names of valid procedure libraries; each library name must appear on a separate line.

[Figure 66](#) shows the code for the Default section of an Import file.

Figure 66 • Default Section Example:

```
<DEFAULT START>
  <FILEORG>
    PDS
  <COBOL VERSION>
    COB2
  <COPYLIB>
    SYS1.COBLIB
  <IMS MACLIB>
    IMSESA.MACLIB
  <CICS MACLIB>
    CICS330.MACLIB
  <PROCLIB>
    SYS1.PROCLIB
    SYS2.PROCLIB
  <COMPILE EXECUTION PARM>
    LIB,QUOTE
  <CA-LIBRARIAN MEMBER PSWD EXISTS>
    NO
  <CA-LIBRARIAN MEMBER COPYDD>
    NO
<DEFAULT END>
```

Definition

The Definition section is used to specify these types of application components:

- COBOL
- Load Module
- JCL
- CICS CSD files
- CICS PPT, PCT, and FCT tables
- CICS BMS maps
- IMS DBD, PDB, DFSMDA, MFS, and Stage 1 libraries

Each library must be defined separately and any members defined for that library must be specified immediately following the library definition. These keywords are used to define the various components of the library definitions:

LIBRARY

Identifies the dataset name for the COBOL, Load, JCL, CICS, IMS library or CICS CSD file being defined. This keyword must precede each library definition. Only one library can be defined at a time. Valid entries: a valid dataset name.

FILEORG

Identifies the source manager that is used for all COBOL, JCL, CICS, and IMS datasets included in the application definition. Only one source manager can be specified. If not specified, the FILEORG defined in the default section is used. Valid Entries: PDS, PAN, LIB, VSAM, SEQ, or ENDV.

OBJECT KIND (required)

Identifies the library type for the dataset specified. This entry is required for each library defined to the application. Only one library type can be specified at a time. Valid entries: CICS, COBOL, IMS, JCL, LOAD, or CSD.

COBOL VERSION

Identifies the COBOL version for the COBOL library and members defined to the application. Only one version can be specified. Valid Entries: COB68, COB74, COB2, COB2R3, COB370.

COMPILE EXECUTION PARM

Identifies the default compile batch execution parameters used for COBOL libraries defined to the application. Compile execution arms can only be specified within a COBOL library definition or in the MEMBER OVERRIDE information for COBOL member(s).

COBOL SOURCE EXTRACTION EXECUTION PARM

Identifies the source library manager batch execution parameters used for COBOL libraries defined to the application.

JCL SOURCE EXTRACTION EXECUTION PARM

Identifies the source library manager batch execution parameters used for JCL libraries defined to the application.

CICS SOURCE EXTRACTION EXECUTION PARM

Identifies the source library manager batch execution parameters used for CICS libraries defined to the application.

IMS SOURCE EXTRACTION EXECUTION PARM

Identifies the source library manager batch execution parameters used for IMS libraries defined to the application.

STDSQL(86)

Identifies the SQL usage found in the JCL. If the DB2 STDSQL(86) parm is found in the compile/link JCL, the default is YES. Valid entries: YES or NO.

With STDSQL(86) or STDSQL(YES), the SQLCA is expanded where the definition of SQLCODE is found in the program and SQLCADE replaces the SQLCODE field in the SQLCA structure.

```
01 SQLCODE      PIC X9(9) COMP.
01 SQLCA.
    05 SQLCAID   PIC X(8).
    05 SQLCABC   PIC S9(9) COMPUTATIONAL.
    05 SQLCADE   PIC S9(9) COMPUTATIONAL.
```

...

With STDSQL(NO) EXEC SQL SQLCA END-EXEC. Expands the SQLCA and SQLCODE is field in the SQLCA.

```
EXEC SQL INCLUDE END-EXEC.
```

```
01 SQLCA
    05 SQLCAID   PIC X(8).
    05 SQLCABC   PIC S9(9) COMPUTATIONAL.
    05 SQLCADE   PIC S9(9) COMPUTATIONAL.
```

...

PASSWORD

Identifies the password for the specified COBOL, JCL, CICS, or IMS library or member if the library or member(s) is password protected. Enter the password.

VOLSER

Identifies the volume serial number for the specified COBOL, JCL, CICS, or IMS library or member(s) if the library or member is not cataloged. Enter the volume serial number.

CA-LIBRARIAN MEMBER PASSWORD

Identifies the password for a CA-Librarian member if that member in the application is password protected. The member is password protected if the <CA-LIBRARIAN MEMBER PSWD EXISTS> is set to YES.

COPYLIB

Identifies the copy library for the specified COBOL library or member(s). Each copy library must appear on a separate line. Copy library information can only be specified within a library definition or in the MEMBER OVERRIDE information. For each copy library, this information can be specified.

Attribute	Description
COPY FILEORG	Identifies the source manager that is used for the specified copy library. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB
COPY SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO
COPY USAGE	Identifies the location of the SYSLIB dataset in the compile JCL. Usage information is used to determine when the AAE needs to use the copy libraries defined to the application: during COBOL library processing, during DB2 include processing, or during both types of processing. Only one COPY USAGE can be specified. Valid entries: COB, DB2, BOTH
COPY PASSWORD	Identifies the password for the specified copy library. If the library is password protected, enter the password.
COPY VOLSER	Identifies the volume serial number for the specified copy. If the library or member is not cataloged, enter the volume serial number.

[Figure 67](#) shows the code to define copy library information for MY.FIRST.COBOL.LIB:

Figure 67 • Copy Library Attribute Definition

```

<LIBRARY>
  MY.FIRST.COBOL.LIB
  <OBJECT KIND>
    COBOL
  <COBOL VERSION>
    COB68
  <PASSWORD>
    SRCPSWD
  <VOLSER>
    SRC001
  <COPYLIB>
    MY.FIRST.COPY.LIB
    <COPY FILEORG>
      PDS
    <COPY PASSWORD>
      CFPYPSWD
    <COPY VOLSER>
      SRC001
    <COPY USAGE>
      COB
  
```

IMS MACLIB

Identifies the maclib attributes for IMS libraries and members included in the application definition. Valid entries: Names of valid macro libraries; each library name must appear on a separate line.

Attribute	Description
IMS MACLIB FILEORG	Identifies the source manager that is used for the specified maclib. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB.
IMS MACLIB SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO.
IMS MACLIB PASSWORD	Identifies the password for the specified macro library. If the library is password protected, enter the password.
IMS MACLIB VOLSER	Identifies the volume serial number for the specified macro library. If the library or member is not cataloged, enter the volume serial number.

CICS MACLIB

Identifies the maclib attributes for CICS libraries and members included in the application definition. Valid entries: Names of valid macro libraries; each library name must appear on a separate line.

Attribute	Description
CICS MACLIB FILEORG	Identifies the source manager that is used for the specified maclib. Only one source manager can be specified. Valid Entries: PDS, PAN, or LIB.
CICS MACLIB SUBSYSTEM	Indicates whether the SYSLIB DD statement in the compile JCL makes use of the SUBSYS parameter. Valid Entries: YES or NO.
CICS MACLIB PASSWORD	Identifies the password for the specified macro library. If the library is password protected, enter the password.
CICS MACLIB VOLSER	Identifies the volume serial number for the specified macro library. If the library or member is not cataloged, enter the volume serial number.

PROCLIB

Identifies the procedure library attributes for the specified JCL library or member. Each procedure library must be entered on a separate line.

IDMS INFORMATION START and IDMS INFORMATION END

Marks the beginning and end of the IDMS attribute information. IDMS information can only be specified in a COBOL library definition, or in the MEMBER OVERRIDE information for COBOL members. For each IDMS attribute, the information listed below can be specified. If an item is not specified, the default specified in the Default section of the import file is used. If the default is not specified in the import file, the site default specified at installation is used.

Attribute	Description
IDMSDMLC EXECUTION PARM	If you are using IDMS release 10, enter the IDMSDMLC parameters for the IDMS preprocessor.
IDMS DDNAME/dataset NAME	Enter the IDMS DD name and the dataset name used by the library or member.

[Figure 68](#) shows the code used to define IDMS information for MY.FIRST.COBOL.LIB.

Figure 68 • Example IDMS Attribute Definition

```
<LIBRARY>
MY.FIRST.COBOL.LIB
<OBJECT KIND>
  COBOL
<COBOL VERSION>
  COB68
<IDMS INFORMATION START>
  <IDMSDMLC EXECUTION PARM>
    PARM GOES HERE
  <IDMS DDNAME/DATA SET NAME>
    STEPLIB IDMS.DEVL.STEPLIB
    STEPLIB IDMS.PROD.STEPLIB
    SYSCTL IDMS.PROD.SYSCTL
    DLODDB IDMS.PROD.DLODDB
    DICTDB IDMS.DEVL.DICTDB
    DICTDB IDMS.PROD.DICTDB

  <IDMS INFORMATION END>
```

MEMBER LIST START and MEMBER LIST END

After a library and its attributes have been defined, these tags identify the beginning and end of the list of members within the library that are to be included in the definition.

[Figure 69](#) shows the code required to include members in the library COBOL.SOURCE.LIBRARY.

Figure 69 • Member Definition

```
<LIBRARY>
COBOL.SOURCE.LIBRARY
<OBJECT KIND>
  COBOL
<MEMBER LIST START>
  Membername1
  Membername2
<MEMBER LIST END>
```

MEMBER OVERRIDE INFORMATION START and MEMBER OVERRIDE INFORMATION END

If you want to specify member attributes that are different than those defined at the library level, these keywords identify the beginning and end of the override information. The keyword should be followed by the list of members that use the specified attributes.

[Figure 70](#) shows the code to define the member override information for Membername1 and Membername2.

Figure 70 • Member Override Definition

```
<MEMBER OVERRIDE INFORMATION START>
  <MEMBER LIST START>
    Membername1
    Membername2
  <MEMBER LIST END>

  <FILEORG>
    LIB
  <COBOL VERSION>
    COB68
  <COPYLIB>
    SYS1.COPYLIB2

<MEMBER OVERRIDE INFORMATION END>
```

GROUP LIST START and GROUP LIST END

After a CSD file has been defined, these tags identify the beginning and end of the list of group names within the file that are to be included in the definition. Wildcard characters (* and ?) are accepted. Each group name should be listed on a separate line.

Sample Import File

[Figure 71](#) shows the Application Information section and Default Information section of an import file.

Figure 71 • Import File Example (1 of 3)

```

*****
*          APPLICATION INFORMATION          *
*          *                               *
<AKR>
USER.TEST.AKR
<PRODUCT>
RECAP
<APPLICATION NAME>
IMP-FULL
<APPLICATION DESCRIPTION>
COMPLETE APPLICATION TEST
*****
*          DEFAULT INFORMATION           *
*          *                               *
*****

<DEFAULT START>
<FILEORG>
PDS
<COBOL VERSION>
COB2
<COPY USAGE>
COB
<COPYLIB>
SYS1.COPYLIB
<IMS MACLIB>
IMSESA.MACLIB
<CICS MACLIB>
CICS330.MACLIB
<PROCLIB>
SYS1.PROCLIB
SYS2.PROCLIB
<COMPLETE EXECUTION PARM>
LIB,QUOTE
<CA-LIBRARIAN MEMBER PSWD EXISTS>
NO
<CA-LIBRARIAN MEMBER COPYDD>
NO
<DEFAULT END>

```

[Figure 72](#) and [Figure 73 on page 176](#) show the Application Definition section of an import file for Recap and Alliance.

Figure 72 • Import File Example (2 of 3)

```

*****
*
*   DEFINITION INFORMATION   *
*
*****

<LIBRARY>
MY.FIRST.COBOL.LIB
<OBJECT KIND>
  COBOL
<COBOL VERSION>
  COB68
<COPYLIB>
  MY.FIRST.COPY.LIB
  <COPY FILEORG>
    PDS
  <COPY USAGE>
    COE
<COPYLIB>
  MY.FIRST.DB2.INCLUDE.LIB
  <COPY USAGE>
    DB2
<COPYLIB>
  MY.FIRST.DB2.COPY.LIB
  <COPY USAGE>
    BOTH
<IDMS INFORMATION START>
<IDMSDMLC EXECUTION>
  P&M GOES HERE
<IDMS DDNAME/DAT& SET NAME>
  STEPLIB IDMS.DEVL.STEPLIB
  STEPLIB IDMS.PROD.STEPLIB
  SYSCTL IDMS.PROD.SYSCTL
  DLODDE IDMS.PROD.DLODDE
  DICTDE IDMS.DEVL.DICTDE
  DICTDE IDMS.PROD.DICTDE
<IDMS INFORMATION END>
<MEMBER LIST START>
  BATMAN
  ROBIN
  HARDY
<MEMBER LIST END>
<MEMBER OVERRIDE INFORMATION START>
<MEMBER LIST START>
  OZZIE
  HARRIET
<MEMBER LIST END>
<COBOL VERSION>
  COE370
<COPYLIB>
  MY.FIRST.COPY.LIB
<MEMBER OVERRIDE INFORMATION END>

```

Figure 73 • Import File Example (3 of 3)

```
<LIBRARY>
  MY.SECOND.COBOL.LIB
  <OBJECT KIND>
  COBOL
<LIBRARY>
  MY.FIRST.JCL.LIB
  <OBJECT KIND>
  JCL
  <PROCLIB>
  MY.FIRST.PROCLIB
  MY.SECOND.PROCLIB
  <MEMBER LIST START>
  MJCL001
  MJCL008
  MJCL009
  <MEMBER LIST END>
<LIBRARY>
  MY.FIRST.LOAD.LIB
  <OBJECT KIND>
  LOAD
  <MEMBER LIST START>
  MLOAD001
  MLOAD002
  <MEMBER LIST END>
<LIBRARY>
  MY.FIRST.CICS.LIB
  <OBJECT KIND>
  CICS
  <MEMBER LIST START>
  MFCT001
  MPPT001
  <MEMBER LIST END>
<LIBRARY>
  MY.FIRST.CSD.DSN
  <OBJECT KIND>
  CICS
  <FILEORG>
  VSAM
<LIBRARY>
  MY.FIRST.IMS.LIB
  <OBJECT KIND>
  IMS
  <IMS MACLIB>
  IMSESA.MACLIB
  <MEMBER LIST START>
  MDBDGEN1
  MPSBGEN1
  <MEMBER LIST END>
  <MEMBER OVERRIDE INFORMATION START>
  <MEMBER LIST START>
  MSTG1
  <MEMBER LIST END>
  <IMS MACLIB>
  IMSESA.SYSS.MACLIB
  <MEMBER OVERRIDE INFORMATION END>
<LIBRARY>
  MY.FIRST.SEQ.FILE
  <OBJECT KIND>
  COBOL
  <FILEORG>
  SEQ
<END DEFINITION>
```

Import Language Keywords

Application Information

- AKR
- APPLICATION NAME
- APPLICATION DESCRIPTION
- PRODUCT

Default Section

```

DEFAULT START
DEFAULT END
FILEORG
COBOL VERSION
COMPILE EXECUTION PARM
COBOL SOURCE EXTRACTION EXECUTION PARM
JCL SOURCE EXTRACTION EXECUTION PARM
CICS SOURCE EXTRACTION EXECUTION PARM
IMS SOURCE EXTRACTION EXECUTION PARM
STDSQL(86)
COPYLIB
    COPY FILEORG
    COPY SUBSYSTEM
    COPY PASSWORD
    COPY VOLSER
COPY USAGE
IMS MACLIB
    IMS MACLIB FILEORG
    IMS MACLIB SUBSYSTEM
    IMS MACLIB PASSWORD
    IMS MACLIB VOLSER
CICS MACLIB
    CICS MACLIB FILEORG
    CICS MACLIB SUBSYSTEM
    CICS MACLIB PASSWORD
    CICS MACLIB VOLSER
PROCLIB
CA-LIBRARIAN MEMBER COPYDD
CA-LIBRARIAN MEMBER PSWD EXISTS

```

Library and Member Definition Section

```
LIBRARY
  OBJECT KIND
  FILEORG
  COBOL VERSION
  COMPILE EXECUTION PARM
  COBOL SOURCE EXTRACTION EXECUTION PARM
  JCL SOURCE EXTRACTION EXECUTION PARM
  CICS SOURCE EXTRACTION EXECUTION PARM
  IMS SOURCE EXTRACTION EXECUTION PARM
  STDSQL(86)
  COPYLIB
    COPY FILEORG
    COPY USAGE
    COPY SUBSYSTEM
    COPY VOLSER
    COPY PASSWORD
  CA-LIBRARIAN MEMBER COPYDD
  CA-LIBRARIAN MEMBER PSWD EXISTS
  PASSWORD
  VOLSER
  IDMS INFORMATION START
    IDMSDMLC EXECUTION PARM
    IDMS DDNAME/DATA SET NAME
  IDMS INFORMATION END
  MEMBER LIST START (must appear after library attributes)
  MEMBER LIST END
    MEMBER OVERRIDE INFORMATION START
    MEMBER OVERRIDE INFORMATION END
  IMS MACLIB
    IMS MACLIB FILEORG
    IMS MACLIB SUBSYSTEM
    IMS MACLIB PASSWORD
    IMS MACLIB VOLSER
  CICS MACLIB
    CICS MACLIB FILEORG
    CICS MACLIB SUBSYSTEM
    CICS MACLIB PASSWORD
    CICS MACLIB VOLSER
  PROCLIB
  CA-LIBRARIAN MEMBER PASSWORD
  GROUP LIST START
  GROUP LIST END
```

9

Export Metrics Facility

This chapter guides you through the creation of files for export and for output from export and contains these sections:

Section	Page
Creating CDF Files for Export	179
Output from Export	181

Recap provides a feature that creates a file to export portfolio analysis data to spreadsheets, PC database managers, and graphics presentation packages. A comma (,) and quote (" ") delimited file can be created by selecting File ► Import/Export or the Export command from the Batch AKR Utility. The output can then be downloaded to your PC for input into spreadsheet, database manager, and graphics presentation software packages.

Creating CDF Files for Export

Assume that your organization uses Cyclomatic Complexity in determining if a program is a candidate for re-engineering. That is, in order for a program to be re-engineered, it must be very complex in comparison to other programs within the application.

In a spreadsheet package, you want to list all the applications within an enterprise (AKR) sorted by Function Points (primary sort) and Cyclomatic Complexity (secondary sort). In addition, you also want to list each program's Essential Complexity (to indicate the degree of structure in the program).

Because you want to list all the programs in an AKR, perform the steps outlined in this section for each application.

To create the appropriate CDF files

- 1 Select File ► Import/Export to display the File - Import/Export pop-up.

- 2 Select Export Metrics to initiate the File - Export Metrics pop-up, shown in [Figure 74](#).

Figure 74 • File - Export Metrics Pop-up

```
File - Export Metrics

Specify AKR data set name, application name, options, and DSNs to create
metrics and/or function points CDF files. Then press Enter.

AKR data set name 'USER.TEST.AKR'
Application name  APPL-C2R30

Options 1   1. Metrics and function points
           2. Metrics only
           3. Function points only

Metrics (Partitioned or Sequential data set)
Data set name _____

Function point summary (Partitioned or Sequential data set)
Data set name _____

Function point details (Partitioned or Sequential data set)
Data set name _____
```

- 3 Enter the appropriate information on the File - Export Metrics pop-up. You must enter the application name and the AKR dataset name where the application resides.
 - a Select the Metrics and function points option (type 1 in the Options field).
 - b If you want to have Recap generate the CDF files, press Enter. These CDF files are created:
 - USERID.XEMnnnnn.VIASCDFM contains program and application metrics data.
 - USERID.XEMnnnnn.VIASCDFP contains the application function point data
 - USERID.XEMnnnnn.VIASCDFS contains detail function point data for the application.
 - c If you want to specify a target dataset, other than the generated export target dataset, perform these steps:

To specify a partitioned dataset

- ▶ Type in the dataset and member name.
 - If the PDS does not exist, an error message appears indicating that the dataset is not found.
 - If the member already exists and contains data, a screen appears to confirm whether you want to overwrite the existing data. If the member is empty, the exported data is loaded into the empty member. If the member does not exist, it is created and the data loaded into the newly created member.

To specify a sequential dataset

- ▶ The dataset can exist but must be empty. If the existing dataset contains data, a screen appears to confirm whether you want to overwrite the existing data. If the dataset does not exist, it is created.

After these steps are performed for all the applications, use your communication software to download these files to your personal computer. Then, using a spreadsheet package, use its Import Facility to retrieve the CDF files. After formatting the spreadsheet appropriately, the programs are shown in comparison to each other in terms of complexity and structure.

Output from Export

When the data is exported up to three files can be created. One file contains application and program metric information (see [Program and Application Metric Information](#)). The other two files contain detailed (see [“Detailed Function Point Information” on page 182](#)) and summarized (see [“Summarized Function Point Information” on page 183](#)) function point information.

Program and Application Metric Information

If the default name is used for the file collecting the exported metric data, this is the format:

```
USERID.XEnnnnn.VIASCDFM
```

where *nnnnn* is a number assigned by Recap.

You can specify a file name if you do not want to use the default. This file provides one row of heading information followed by an application metric total row. Each program in the application follows with its respective metric information.

[Figure 75](#) shows the output for the program and application metric information.

Figure 75 • Application and Metric Information - Exported Data

```

"APPLICATION", "PROGRAM", "FPA", "FPE", "VOLUME", "CYCLOMATIC", "ESSENTIAL", "CONTROL", "KNOTS",
"GOTOS", "ALTERS", "ENTRIES", "EXITS", "RECURSIONS", "PERFORM JUMPS", "LIVE EXITS", "DEAD CODE",
"DEAD DATA", "LINES OF CODE", "DATE", "TIME", "PHYS LOC", "VERSION"
"TELEPHONE", "*TOTALS*", 171, 0, 146539, 474, 63, 1241, 28, 45, 0, 8, 37, 1, 8, 2, 23, 272, 6579,
"14-MAR-1995", "18:15:17", 9160, 1
"TELEPHONE", "VS101", ,, 12312, 45, 4, 116, 0, 1, 0, 1, 3, 0, 0, 0, 13, 36, 556, "14-MAR-1995",
"18:14:17", 779, 1
"TELEPHONE", "VS201", ,, 37830, 136, 10, 394, 18, 17, 0, 2, 17, 0, 0, 0, 0, 52, 2011, "14-MAR-1995",
"18:14:30", 3123, 1
"TELEPHONE", "VS202", ,, 2436, 15, 1, 43, 0, 0, 0, 1, 2, 0, 0, 0, 0, 205, "14-MAR-1995",
"18:14:38", 275, 1
"TELEPHONE", "VS203", ,, 15111, 42, 14, 110, 8, 9, 0, 2, 5, 0, 0, 0, 0, 89, 890, "14-MAR-1995",
"18:14:48", 1635, 1
"TELEPHONE", "VS204", ,, 39390, 117, 17, 287, 1, 9, 0, 1, 5, 0, 4, 0, 5, 46, 1459, "14-MAR-1995",
"18:15:00", 1676, 1
"TELEPHONE", "VS205", ,, 39460, 119, 17, 291, 1, 9, 0, 1, 5, 1, 4, 2, 5, 47, 1458, "14-MAR-1995",
"18:15:17", 1672, 1

```

Note:

The exported metric data heading information contains two references to lines of code. The LINES OF CODE heading refers to the logical lines of code. The PHYS LOC heading refers to the physical lines of code.

Detailed Function Point Information

If the default name is used for the file collecting the exported detailed function point information, this is the format:

```
USERID.XEnnnnn.VIASCDF5
```

where *nnnnn* is a number assigned by Recap.

You can specify a file name if you do not want to use the default. This file provides one row of heading information and a row of metric information for the application that provides the name of the application, including these names:

- DET
- FTR
- RET counts
- USE (level of complexity, HIGH, AVG, LOW)

[Figure 76](#) shows the output for the detail function point information.

Figure 76 • Detail Function Point Information - Exported Data

```

"APPLICATION", "TYPE", "FTR", "RET", "DET", "USE", "TEXT"
"SALES", "EI", "0-1", "0", "1-4", "LOW", "DELETE IX-FD1"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "2-3", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "2-3", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "2-3", "0", "1-5", "LOW", "REPORT MSG-FILE"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "2-3", "0", "1-5", "LOW", "REPORT FILE1"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT SQ-FS2"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"
"SALES", "EO", "0-1", "0", "6-19", "LOW", "REPORT SQ-FS2"
"SALES", "EO", "0-1", "0", "1-5", "LOW", "REPORT PRINT-FILE"

```

Summarized Function Point Information

If the default name is used for the file collecting the summarized function point information, this is the format:

```
USERID.XEnnnnn.VIASCDFP
```

where *nnnnn* is a number assigned by Recap.

You can specify a file name if you do not want to use the default. This file provides one row of heading information and a row for the application that contains the total for each function point type: External Input (EI), External Output (EO), External Inquiry (EQ), Internal Logical File (ILF), and External Interface File (EIF). A total for all function point types for the application is also calculated.

[Figure 77](#) shows the output for the summary function point information.

Figure 77 • Summary Function Point Information - Exported Data

```

"APPLICATION", "EI", "EO", "EQ", "ILF", "EIF", "TOTAL"
"SALES", 3, 40, 0, 24, 0, 67

```

10

Exporting an Application Definition

This chapter provides information about copying the application definition, exporting it, and creating one similar to the original one, and contains these sections:

Section	Page
Exporting the Application Definition	185
Copying the Application Definition	187

Recap provides a feature that gives the ability to export an application definition from Recap. The exported definition may be stored in a sequential file or a PDS member.

To export the application definition file, select File ► Import/Export.

Exporting the Application Definition

The exported application definition consists of a series of keyword control statements whose syntax is consistent with the requirements of the Import Facility. The keyword control cards identify the AKR, the application name and description, and the application components and their associated attributes.

See "[Import Facility](#)" on page 159 for a description of the keyword control statement syntax.

The exported application definition can be used to:

- Create a backup of the application definition for easy recovery.
- Copy an application definition from one AKR to another.
- Copy an application definition to use as a template in creating a similar definition.

Export Procedure

To export the application definition

- 1 Identify or create a sequential file or a PDS library to contain the exported definition. The file must have a record length of 80.
- 2 Select File ► Import/Export to display the File - Import/Export pop-up.
- 3 On the File - Import/Export pop-up, select Export Definition to display the File - Export Definition pop-up, shown in [Figure 78](#).

Figure 78 • File - Export Definition Pop-up

```
File - Export Definition
Command ==> _____
Type AKR and Export information, then press enter.
Application Knowledge Repository (AKR):
  Data set name . . . 'USER.TEST.AKR'
  Application name . . USERAPPL
Export:
  Data set name . . . _____
  Member . . . . . : USERAPPL _
```

- 4 On the File - Export Definition pop-up, specify this information:
 - a The AKR Data set name.
 - b The Application name.
 - c The Export Definition Data set name.
 - d If the Export Definition dataset is a PDS, specify the Export Definition Member name.
- 5 Press Enter to execute the export request.

When the export of the application definition is complete, the message "<application name>" EXPORTED displays in the short message area of the File - Export Definition pop-up.

Copying the Application Definition

If you intend to use the exported application definition as input to the Import Definition Facility to copy the application definition to another AKR, you must perform these edits on the exported application definition:

- 1 At the <AKR> keyword control statement, change the AKR dataset name to indicate the target dataset name.
- 2 At the <APPLICATION NAME> keyword statement, change the application name to the application name desired.

Note: _____

Within an AKR, application names must be unique.

Creating a Similar Application Definition

If you intend to use the exported application definition as input to the Import Facility to create a template for a new application definition, you must perform these edits on the exported application definition:

- 1 At the <AKR> keyword control statement, change the AKR dataset name to indicate the target dataset name.
- 2 At the <APPLICATION NAME> keyword statement, change the application name to the application name desired.

Note: _____

Within an AKR, application names must be unique.

- 3 Update the application definition using either of these methods:

Edit the exported application definition file. Modify the keyword control statements, and their associated attribute keyword control statements, to create the new application definition.

Or

Import the application definition into the target AKR, then use the online application definition function to modify the imported definition.

11

AKR Utilities

This chapter explains both online and batch AKR utilities and contains these sections:

Section	Page
Introduction to AKR Management	189
AKR Structure	189
Online AKR Utilities	190
Batch AKR Utilities	190
Batch AKR Reports	207
Allocating and Expanding AKRs without ISPF	215

Introduction to AKR Management

The AKR is the repository for all information used by the ESW family of components. Analyzed programs are stored in the AKR for use by ESW. A single AKR can be defined for use by all ESW users, or separate AKRs can be defined for use by departments, groups, or individual users. ESW provides both online and batch utilities for managing the AKR.

AKR Structure

The AKR is a BDAM or VSAM file organization. An AKR can be defined to be shared by all users, or multiple AKRs can be defined for use by departments, groups, or individuals.

Note: _____

See the *ASG-Center Installation Guide* for more information about the AKR.

Online AKR Utilities

The online AKR utilities includes these pop-ups:

- The File - AKR Utility pop-up used to rename or delete a program, or to display the AKR Directory.
- The File - AKR Directory pop-up used to view all programs in an AKR. This pop-up can be used to rename or delete a group of programs. Statistics about the AKR are also shown on this pop-up.
- The File - AKR Allocate/Expand pop-up used to allocate a new AKR or to expand an existing AKR.

Note: _____

When you use the Allocate/Expand utility, the default AKR organization type is applied. For example, if your site's default AKR type is sequential, any new AKR allocated is created as a sequential file, and any non-sequential AKR expanded is reorganized as a sequential file. See the online help for examples of each of these pop-ups.

Batch AKR Utilities

The Batch AKR Utility is used to maintain the AKR without using ISPF. This table lists the commands available in the batch utility.

Command	Function
ANLZSTAT (See page 194)	Produces the Analysis Status Report for one or more AKRs. This applies to Application level products only.
COMMENTS (See page 195)	Includes comments with the command.
CONVERT (See page 196)	Converts selected members from a previous product release level to the current release level.
COPY (See page 198)	Copies selected members from one AKR to another.
DELETE (See page 199)	Deletes selected members from the AKR.
EXPORT (See 200)	Creates metrics and function point CDF files. This applies to Recap only.

Command	Function
HELP (See page 201)	Prints the AKR Utility Help Report.
INIT (See page 202)	Formats a previously-defined dataset into an AKR format.
MOVE (See page 203)	Copies selected members from one AKR to another and deletes them from the original AKR.
PRINT (See page 204)	Prints AKR directory information, or COBOL source listings for selected AKR members.
PUNCH (See page 205)	Produces a file that contains the AKR directory information, or the COBOL source code for selected AKR members.

Job Control Statements

The Batch AKR Utility uses these JCL statements, shown in [Figure 79](#). The VIAAKRIN and VIAAKROT DD statements describe AKRs that are used for AKR Utility processing. The VIASYSIN DD control cards consist of the necessary Batch AKR commands described in [Command Format](#). See the description for each command to determine which DD statements are affected.

Figure 79 • Batch AKR JCL Statements

```
//UTILITY EXEC PGM=VIASAKRU,REGION=3000K,PARM=''
//STEPLIB DD DISP=SHR,DSN=(ASG load library)
//VIAAKRIN DD DISP=SHR,DSN=(Input AKR)
//VIAAKROT DD DISP=SHR,DSN=(Output AKR)
//VIAPRINT DD SYSOUT=A (Print file description)
//VIAPUNCH DD SYSOUT=B (Punch file description)
//VIALOG DD SYSOUT=A (Log file)
//VIASYSIN DD *
<control cards>
//
```

Control Cards

Commands are passed to the Batch AKR Utility with the control cards following the VIASYSIN DD statement. Control cards must conform to these standards:

- Command information must be contained in columns 1 through 72 of the control card.
- Only one command can be entered on each control card.
- Only one control card can be used per command.

All control cards with command disposition and command summaries are printed to the VIALOG AKR Utility Log file. Blank control cards are ignored.

Command Format

Commands that use member names accept special characters to signify generic names. An asterisk (*) represents zero or more characters. A question mark (?) represents one character. For example:

Command	Description
DBA*	Specifies all members that begin with DBA and end with any other characters.
D?A*	Specifies all members that begin with D followed by one character, followed by an A, then followed by any other characters.
DBA???	Specifies all members that begin with DBA and end with any three characters.

The LASTUSED parameter is used to provide the selection criteria for several commands. The specified number represents the number of days since the member was last referenced online, or the date the member was analyzed if it has not been referenced.

The REPLACE parameter is used to specify that members are to be replaced on the output AKR. The NOREPLACE parameter is used to prevent members from being replaced on the output AKR. NOREPLACE is the default value.

Command Syntax

Each Batch AKR Utility command is described in this chapter. The descriptions include the format and a brief explanation of the command parameters. These notational conventions and symbols are used to describe command syntax:

Item	Description
ABBREVIations	Command abbreviations are shown in uppercase letters; lowercase letters in the command are optional.
lowercase	Lowercase values indicate user-supplied variable information.
<u>Underline</u>	The default value of an operand is underlined.
Vertical Bar	Vertical Bar - A vertical bar separates synonymous commands or operands.
—————>	Right and left ending arrows indicate the end of the command syntax.
— required —	An operand or keyword appearing on the main command line is required.
— optional —	An operand or keyword appearing below the main command line is optional.
— choice1 — — choice2 —	Stacked operands below the main line show a choice of one optional item.

ANLZSTAT Command

ANLZSTAT 

Function

Produces the Analysis Status Report for one or more AKRs submitted with the job. The report provides: analysis composition and status information for each AKR; analysis composition and status information for applications within each AKR; and detailed analysis information on the components within each application.

Operands

None. However, the ANLZSTAT statement must be followed by one or more AKR dataset names, as shown in this example:

```
//VIAAKRU.VIASYSIN DD *  
ANLZSTAT  
VIAALXX.DEVL.SAMPAKR  
ASG.VIACENXX.AKR
```

Usage Notes

The ANLZSTAT command is specified in the input card for the Batch AKR job, VIASAKRU. The VIAAKRIN file is not referenced by this command. ANLZSTAT must be followed by one or more AKR dataset names. Each AKR dataset name must be on a separate line.

A report is written to the DD name, VIASARPT. In the default JCL, the DD name is defined as a SYSOUT file. You can override this DD statement and make it a permanent file. The permanent file must be deleted or a new file specified for each job run.

The ANLZSTAT command looks at each component in the each AKR submitted in the job. The Analysis Status Report is generated for application level components only (i.e., Alliance and Recap applications). An Analysis Status Report is not generated for program level components (e.g., Encore, Insight programs).

See "[Analysis Status Report](#)" on page 209 for examples of the report and additional information. Commands that use member names accept special characters to signify generic names.

Command	Description
DBA*	Specifies all members that begin with DBA and end with any other characters.
D?A*	Specifies all members that begin with D followed by one character, followed by an A, then followed by any other characters.
DBA???	Specifies all members that begin with DBA and end with any three characters.

COMMENT Command



Function

Includes a comment with the commands.

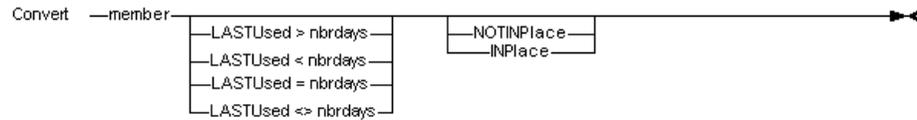
Operands

Operand	Description
Comment	User-supplied text.

Usage Notes

Blank control cards are ignored.

CONVERT Batch AKR Command



Function

Converts selected members that were analyzed by a prior release product to the current release level.

Note: _____

If you are converting from Recap 1.0 to 2.0 or a later release, use the procedure described in "[Application Conversion](#)" on page 221. These Operands and Usage Notes pertain to products other than Recap.

Operand

Operand	Description
member	Can be a specific or generic name as described in " Command Format " on page 192.

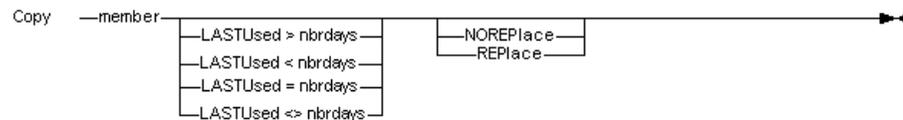
These operands are NOT valid for Recap

LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 192 .
NOTINPlace	Specifies that a member with the same name on the receiving AKR should be replaced with the member from the sending AKR.
INPlace	Specifies that a member is to be converted and kept within the AKR named in the VIAAKRIN DD statement. This option should be used with caution. Consult your systems programmer or the ASG Service Desk.

Usage Notes

Members are copied from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement, as described in ["Job Control Statements" on page 191](#).

COPY Batch AKR Command



Function

Copies selected members from one AKR to another.

Operands

Operand	Description
member	Can be a specific or generic name as described in "Command Format" on page 192 .
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 192 .
NOREPlace	Prevents existing members from being replaced by members with the same name. This is the default.
REPlace	Replaces members that have the same name on the receiving AKR.

Usage Notes

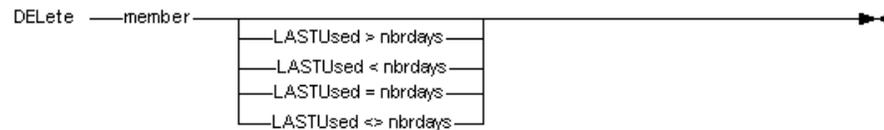
Members are copied from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement.

If the installation option AKR-DSORG-VSAM is specified as NO, expanding an existing VSAM AKR converts it to a BDAM AKR.

Note:

When you use the Allocate/Expand utility, the default AKR organization type is applied. For example, if your site's default AKR type is sequential, any new AKR allocated is created as a sequential file, and any non-sequential AKR expanded is reorganized as a sequential file.

DELETE Batch AKR Command



Function

Erases selected members from the AKR.

Operands

Operand	Description
member	Can be a specific or generic name as described in "Command Format" on page 192 .
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 192 .

Usage Notes

Note: _____
 Members are deleted from the AKR specified in the VIAAKRIN DD statement.

Members that begin with VIA cannot be deleted using this command. All ESW test members begin with VIA. If these members must be deleted, see ["Online AKR Utilities" on page 190](#).

EXPORT Batch AKR Command

EXPort —application— FPA —————▶

Function

Creates metrics and function point CDF files.

Operands

Operand	Description
application	Can be a specific or generic name as described in " Command Format " on page 192.
FPA	Generates only function point information. If FPA is not specified, both metrics and function point information are generated.

Usage Notes

Note: EXPORT is available only for Recap users.

HELP Batch AKR Command

Help | ? _____

Function

Prints a description of the Batch AKR Utility and the commands that can be used.

Operand

None

Usage Notes

- A question mark (?) can be used as an alternate command.
- The HELP command has no operands.

The Help Report is printed to the SYSOUT specified in the VIAPRINT DD statement.

INIT Batch AKR Command



Function

Initializes a new AKR. This internal command is used by the online AKR Utility Allocation function. See ["Online AKR Utilities" on page 190](#) for additional information.

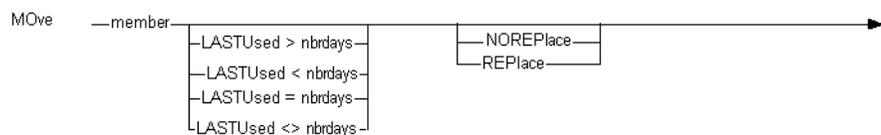
Operand

Operand	Description
DSname(dsname)	Used to specify the dataset name for the new AKR.

Usage Notes

The AKR dataset to be initialized must be created prior to the initialization. The AKR that is initialized can be described in the VIAAKRIN DD statement. The VIAAKRIN DD statement is ignored if the DSNNAME parameter is specified.

MOVE Batch AKR Command



Function

Moves selected members from one AKR to another. Specified members are copied to the receiving AKR and erased from the sending AKR.

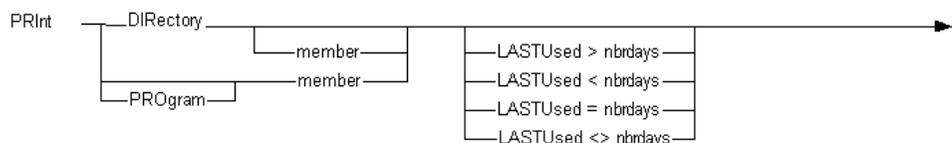
Operands

Operand	Description
member	Can be a specific or generic name as described in "Command Format" on page 192 .
LASTUsed	Used to select members based on the number of days since they were last used, as described in "Command Format" on page 192 .
NOREPlace	Prevents existing members from being replaced by members with the same name. This is the default.
REPlace	Replaces members that have the same name on the receiving AKR.

Usage Notes

Members are moved from the AKR specified in the VIAAKRIN DD statement to the AKR specified in the VIAAKROT DD statement.

PRINT Batch AKR Command



Function

Prints AKR directory information for the entire AKR or for a specified member, or the source code for a specified member.

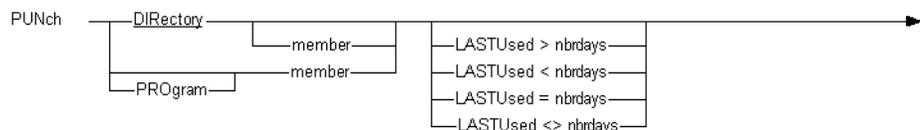
Operands

Operand	Description
blank	If the PRINT batch AKR command is entered with no operand, the AKR directory information is printed.
DIRectory	Prints AKR directory information. This is the default. If a member is specified, the AKR directory information for that member only is printed.
PROgram	Prints the COBOL source for the specified AKR member. The generated COBOL source listing contains expansions of all COPYBOOKs or INCLUDEs, as well as the results of any source preprocessors such as CICS macro expansion.
member	Can be a specific or generic name as described in "Command Format" on page 192 .
LASTUsed	Used to select a member based on the number of days since it was last used, as described in "Command Format" on page 192 .

Usage Notes

Directory information or COBOL source is extracted from the AKR specified in the VIAAKRIN DD statement and is printed to the SYSOUT specified in the VIAPRINT DD statement (see ["AKR Utility Directory Report" on page 207](#)).

PUNCH Batch AKR Command



Function

Produces a file that contains AKR directory information for the entire AKR or a specified member, or the source code for a specified member.

Operands

Operand	Description
blank	If the PUNCH batch AKR command is entered with no operand, a file is produced containing the AKR directory information.
DIRectory	Produces a file that contains directory information. This is the default. If a member is specified, the AKR directory information for that member only is printed.
PROgram	Produces a file that contains COBOL source code for the specified AKR member. The generated COBOL source contains expansions of all COPYBOOKs or INCLUDEs, as well as the results of any source preprocessors such as CICS macro expansion.
member	Can be a specific or generic name as described in "Command Format" on page 192 .
LASTUsed	Used to select a member based on the number of days since it was last used, as described in "Command Format" on page 192 .

Usage Notes

Directory information or COBOL source is extracted from the AKR specified in the VIAAKRIN DD statement and is written to the file specified in the VIAPUNCH DD statement. The file that is produced is in standard IBM IEBUPDTE Utility format. ADD control cards are produced for each logical entity. The NAME parameter contains the member name for COBOL source. The NAME parameter contains AKRDIR nn for directory information, where nn is a consecutively assigned number (see ["Punch Directory File" on page 214](#)).

This table shows the format of the file produced by the PUNCH DIRECTORY command:

AKR Punch Directory File Format		
Member name	10	Character
Number of source lines	6	Right justified
Days since last used	4	Right justified
Analyze date	9	DD MMM YY
Analyze job name	8	Character
Analyze CPU	4	Character
Analyze product level	8	Character
Last reference date	9	DD MMM YY
Last reference user ID	8	Character
Last reference CPU	4	Character

Batch AKR Reports

Examples of these reports are provided in this chapter:

- AKR Utility Directory Report, shown in [Figure 80](#).
- AKR Utility Log (see [Figure 81 on page 208](#)).
- Analysis Status Report (see [Figure 82 on page 210](#), [Figure 83 on page 211](#), [Figure 84 on page 212](#), and [Figure 85 on page 213](#)).
- File produced by the PUNCH DIRECTORY command (see [Figure 86 on page 214](#)).

AKR Utility Directory Report

The AKR Utility Directory report, shown in [Figure 80](#), lists the results of a PRINT DIRECTORY command and is written to the VIAPRINT DD file. The title line contains the ASG component level information, title, date and time the job was executed. The report lists the AKR dataset used, the command used to produce the report, and the directory information for the selected members.

Figure 80 • AKR Utility Directory Report

```

ASG-CENTER-OS Rxx LVLmmn          AKR UTILITY - DIRECTORY          DD MMM YY HH:MM:SS
Page 1

      AKR: ASG.ASGCENmmn.AKR
      Command: PRINT DIRECTORY * LASTUSE > 7

      Member          Last ----- Analyzed ----- Last
      Referenced ----
      Name    Lines  Use   Date    Time    Job   CPU   Level   Date    Time
      Job    CPU

      ACTG0018    40   16   DD MMM YY HH:MM:SS  SRENFKKA CPUA INO30000 DD MMM YY HH:MM:SS
      SRENTY CPUA
      PYRL0085    17   16   DD MMM YY HH:MM:SS  SRENFKKA CPUA INO30000 DD MMM YY HH:MM:SS
      SRENDEM CPUA
      PYRL0105    17    8   DD MMM YY HH:MM:SS  SRENDEMA CPUA INO30000
      SRO005A    493    8   DD MMM YY HH:MM:SS  SRENJDRA CPUC INO30000
      W550044     66   12   DD MMM YY HH:MM:SS  SRENFKKA CPUC INO30000 DD MMM YY HH:MM:SS
      SRENTY CPUA
      XRSCLO70   1714  12   DD MMM YY HH:MM:SS  SRENDEMA CPUA INO30000 DD MMM YY HH:MM:SS
      SRENTY CPUA
      XRSC1100    15    9   DD MMM YY HH:MM:SS  SRENFKKZ CPUC INO30000
      XRSC1200    41   10   DD MMM YY HH:MM:SS  SRENTY2Z CPUC INO30000 DD MMM YY HH:MM:SS
      SRENTY CPUA

      *** End of Directory Report ***

```

AKR Utility Log

The VIALOG AKR Utility Log, shown in [Figure 81](#), provides a summary of the commands issued to the Batch AKR Utility, and contains this information:

- Comments
- Commands
- Completion messages
- Short summary of commands processed

The heading includes the ESW component level information, and the date and time the job was executed. Comments are enclosed in a box comprised of asterisks. The second page contains the log summary.

Figure 81 • AKR Utility Log

```

ASG-CENTER-OS Rx.x LVLmnn          AKR UTILITY LOG          DD MMM YY  HH:MM:SS
Page 1

*****
* PRODUCE A REPORT CONTAINING DIRECTORY INFORMATION FOR ALL *
* MEMBERS OF ASG.RENAISSA.AKR (VIAAKRIN) THAT HAVE *
* NOT BEEN REFERENCED IN THE LAST 7 DAYS. *
*****
*
PRINT DIRECTORY * LASTUSE > 7
*****
VIA1289I 8 DIRECTORY ENTRIES SUCCESSFULLY PRINTED.
*****
*
*****
* PRODUCE A FILE CONTAINING DIRECTORY INFORMATION FOR ALL *
* MEMBERS OF ASG.RENAISSA.AKR (VIAAKRIN) THAT HAVE *
* NOT BEEN REFERENCED IN THE LAST 7 DAYS. *
*****
*
PUNCH DIRECTORY * LASTUSE > 7
*****
VIA1290I 8 DIRECTORY ENTRIES SUCCESSFULLY PUNCHED.
*****
VIA1314I *** END OF VIASYSIN ***

ASG-CENTER-OS Rx.x LVLmnn          AKR UTILITY LOG - SUMMARY  DD MMM YY  HH:MM:SS
Page 2

VIA1301I          8 DIRECTORY ENTRIES PRINTED      0 FAILED.
VIA1302I          8 DIRECTORY ENTRIES PUNCHED      0 FAILED.

VIA1315I *** END OF SUMMARY REPORT ***

```

Analysis Status Report

The Analysis Status Report provides a list of AKR names and statistics about the applications in each AKR, as the result of submitting an ANLZSTAT command. The report is written to the VIASARPT file and provides these levels of information:

- AKR summary and detail
- Application summary and detail
- Library summary and detail

At each reporting level, information is given on:

- The number of members.
- The number of members successfully analyzed, unsuccessfully analyzed, and not analyzed.
- The number of lines of source code in analyzed COBOL programs.

At the library level, additional detailed information is provided for each member:

- Application component type (i.e., COBOL, JCL, LOADMODULE, etc.).
- Detailed analysis information (e.g., time and date, return code).

Analysis Status Report - AKR Summary

The Analysis Status Report, shown in [Figure 82](#), begins with a cover page that lists the AKRs input for the report. The first section of the report provides AKR Summary information (i.e., the AKR(s) input for analysis and the number of applications and members in each AKR) and member analysis information.

Figure 82 • Analysis Status Report - AKR Summary

```

ANALYSIS STATUS REPORT                                PAGE: 1
                                                    DDDMMYYTYY HH:MM:SS

ANALYSIS REPORT SUMMARY
AKR COMPOSITION SUMMARY

                                                    COUNT (% OF TOTAL)
-----
# AKR                                                    APPLICATIONS    MEMBERS
-----
(A) 1 VIAALxx.DEVL.SAMPAKR                            1(100)          22(100)
-----
TOTAL                                                    1(100)          22(100)
=====

AKR ANALYSIS SUMMARY

                MEMBER COUNT (% OF TOTAL IN AKR)
-----
#    SUCCESSFUL    UNSUCCESSFUL    NOT ANALYZED    LINES OF CODE (% OF TOTAL)
-----
(B) 1             21 ( 95)         1( 5)           0( 0)           17,067(100)
-----
TOTAL             21 ( 95)         1( 5)           0( 0)           17,067(100)
=====
    
```

Field	Description
(A)	Each AKR is identified with a number and name.
(B)	The number used to reference that AKR.

Analysis Status Report - AKR Detail

The second section of the Analysis Status Report, shown in [Figure 83](#), provides AKR Detail information (i.e., a breakdown of the applications within each AKR) and member analysis information.

Figure 83 • Analysis Status Report - AKR Detail

```

ANALYSIS STATUS REPORT                                PAGE: 2
                                                    DMMMYYYY HH:MM:SS

AKR DETAIL INFORMATION

AKR:  VIAALxx.DEVL.SAMPAKR

NUMBER OF APPLICATIONS:                1
NUMBER OF LIBRARIES:                   3
NUMBER OF MEMBERS:                     22
LINES OF CODE:                         17,067

AKR COMPOSITION DETAIL

# APPLICATION DESCRIPTION                                COUNT (% OF TOTAL)
                                                    LIBRARIES    MEMBERS
-----
(A) 1 ACCTS-PYBL  VIAALxx SAMPLE ACCTS-PAYBL APPLI    3 (100)      22 (100)
TOTAL                                                    3 (100)      22 (100)
=====

AKR ANALYSIS DETAIL

# MEMBER COUNT(% OF TOTAL IN APPLICATION)
  SUCCESSFUL  UNSUCCESSFUL  NOT ANALYZED  LINES OF CODE(% OF TOTAL)
-----
(B) 1         21 ( 95)      1 (  5)      0 (  0)          17,067(100)
TOTAL        21 ( 95)      1 (  5)      0 (  0)          17,067(100)
=====

```

Field	Description
-------	-------------

(A)	Each application is identified by number and name.
-----	--

(B)	The number used to reference the application.
-----	---

Analysis Status Report - Application Detail

The last section of the Analysis Status Report provides:

- Application Detail information (i.e., a breakdown of the composition of each application) and members per library analysis information, shown in [Figure 84](#).
- Analysis/Composition Detail information for each member analyzed (see [Figure 85 on page 213](#)).

Figure 84 • Analysis Status Report - Application Detail

```

ANALYSIS STATUS REPORT                                PAGE: 3
                                                    DDDMMYYYY HH:MM:SS

APPLICATION DETAIL INFORMATION

APPLICATION:      ACCTS-PYBL

DESCRIPTION:     VIAALxx SAMPLE ACCTS-PAYBL APPLICATION
AKR:             VIAALxx.DEVL.SAMPAPR
TOTAL LIBRARIES:          3
TOTAL MEMBERS:          22
LINES OF CODE:         17,067

MEMBER ANALYSIS SUMMARY - MEMBERS(% OF TOTAL IN APPLICATION)
SUCCESSFUL:           21( 95)
UNSUCCESSFUL:         1(  5)
UNANALYZED:           0(  0)

APPLICATION COMPOSITION SUMMARY

# LIBRARY                                     COUNT(% OF TOTAL)
-----
# LIBRARY                                     MEMBERS      LINES OF CODE
-----
(A) 1 VIAALxx.SAMPAPPL.CNTL                   1(  5)         0(  0)
    2 VIAALxx.SAMPAPPL.COBOL                  15( 68)       17,067(100)
    3 VIAALxx.SAMPAPPL.LOADLIB                 6( 27)         0(  0)
-----
TOTAL                                         22(100)       17,067(100)
=====

```

Field	Description
(A)	Each library is identified by number and name.

Figure 85 • Analysis Status Report - Composition Detail

APPLICATION ANALYSIS/COMPOSITION DETAIL						
MEMBER	LIBRARY NUMBER	LINES OF CODE (% OF TOTAL)	TYPE	LAST ANALYSIS		
				TIME	DATE	RC
CD05	(B) 3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD05APGM	2	2,525(15)	COBOL	MM:SS	12FEB1995	4
CD05BPGM	2	563(3)	COBOL	MM:SS	12FEB1995	0
CD05CPGM	2	632(4)	COBOL	MM:SS	12FEB1995	0
CD10	3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD10APGM	2	1,661(10)	COBOL	MM:SS	12FEB1995	0
CD10BPGM	2	680(4)	COBOL	MM:SS	12FEB1995	0
CD11	3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD11OPGM	2	1,248(7)	COBOL	MM:SS	12FEB1995	0
CD15	3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD15APGM	2	2,597(15)	COBOL	MM:SS	12FEB1995	4
CD15BPGM	2	1,447(8)	COBOL	MM:SS	12FEB1995	0
CD18	3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD18OPGM	2	1,295(8)	COBOL	MM:SS	12FEB1995	0
CD20	3	0(0)	LOAD MOD	MM:SS	12FEB1995	0
CD20APGM	2	735(4)	COBOL	MM:SS	12FEB1995	0
CD20BPGM	2	934(5)	COBOL	MM:SS	12FEB1995	0
CD20CPGM	2	983(6)	COBOL	MM:SS	12FEB1995	4
CD20D PGM	2	863(5)	COBOL	MM:SS	12FEB1995	4
CD20E PGM	2	556(3)	COBOL	MM:SS	12FEB1995	0
CD20OPGM	2	348(2)	COBOL	MM:SS	12FEB1995	8
DAILY	1	0(0)	JCL	MM:SS	12FEB1995	0
TOTAL		17,067(100)				

Field	Description
-------	-------------

(B)	The number is then used to reference the library.
-----	---

Punch Directory File

The Punch Directory, shown in [Figure 86](#), is written to the VIAPUNCH DD file when the PUNCH DIRECTORY command is processed. The file is formatted in standard IBM IEBUPDTE Utility format. The first card, ./ADD . . . , is an IEBUPDTE control card that indicates the following cards are to be added to a PDS in the NAME parameter. The cards that follow are in the format described in the PUNCH DIRECTORY command description. The last card is an IEBUPDTE control card that indicates the end of the control cards. See ["PUNCH Batch AKR Command" on page 205](#) for the format of the AKR Punch Directory File.

Figure 86 • AKR Punch Directory File

```
./ ADD NAME=AKRDIR1,LIST=ALL
ACTG0018      40  16DD MMM YYSRENFRKACPUAINO30000DD MMM YYSRENCTY CPUA
PYRLO085      17  16DD MMM YYSRENFRKACPUAINO30000DD MMM YYSRENDEM CPUA
PYRLO105      17   8DD MMM YYSRENDEMCPUAINO30000
SR0005A       493  8DD MMM YYSRENJDRACPUAINO30000
W55004         66  12DD MMM YYSRENFRKACPUAINO30000DD MMM YYSRENCTY CPUA
XRSCLO70     1714  12DD MMM YYSRENDEMCPUAINO30000DD MMM YYSRENCTY CPUA
XRSCLO100      15   9DD MMM YYSRENFRKZCPUCINO30000
XRSCLO200      41  10DD MMM YYSRENCTYZCPUCINO30000DD MMM YYSRENCTY CPUA
./ ENDUP
```

Allocating and Expanding AKRs without ISPF

The Batch AKR Utility can allocate and expand VSAM AKRs without using ISPF. ESW provides the VIASAKRA JCL to allocate an AKR, and the VIASAKRX JCL to expand an AKR. [Figure 87](#) and [Figure 88 on page 216](#) show the VIASAKRA JCL. [Figure 89 on page 217](#), and [Figure 90 on page 218](#), and show the VIASAKRX JCL.

Figure 87 • VIASAKRA JCL for a VSAM AKR (1 of 2)

```

)CM ***** 00010000
)CM * ASG, INC.          ASG-CENTER Rx.x          MONTH YYYY          * 00020003
)CM * 00030000
)CM * SKELETON JCL FOR SUBMITTING A BATCH JOB TO ALLOCATE AND 00040000
)CM * INITIALIZE A ASG APPLICATION KNOWLEDGE REPOSITORY (AKR) . * 00050003
)CM ***** 00060000
)CM 00070000
)CM 00080000
)CM ***** 00090000
)CM * JOB STATEMENT INFORMATION * 00100000
)CM ***** 00110000
)CM 00120000
&VSV11JB1 00130000
&VSV11JB2 00140000
&VSV11JB3 00150000
&VSV11JB4 00160000
)CM 00170000
)CM ***** 00180000
)CM * IN-STREAM JCL PROCEDURE * 00190000
)CM ***** 00200000
)CM 00210000
)IM VIASAKAP OPT 00220000
//          PEND 00230000
//* 00240000
)CM 00250000
)CM ***** 00260000
)CM * EXECUTE JCL STATEMENTS * 00270000
)CM ***** 00280000
)CM 00290000
//VIASAKRA EXEC VIASAKAP 00300000
//* 00310000
)SEL &VSV32VAC = YES 00320000

```

To allocate an AKR, replace *XX* in the NAME (*XX*), NAME (*XX.DATA*), and the DSNNAME (*XX*) parameters with the name of the AKR to be allocated. Then replace *XX (XX)* with the allocation units values and quantities for your site.

Figure 88 • VIASAKRA JCL for a VSAM AKR (2 of 2)

```

//DEFAKR.SYSIN DD *                                00330000
  DEFINE CLUSTER -                                00340000
    (NAME(&VSVIDB) -                                00350000
  )SEL &VSV32SPU = CYLINDERS                        00360000
    CYLINDERS (&VSV32SPA) -                        00370000
  )ENDSEL                                          00380000
  )SEL &VSV32SPU = RECORDS                          00390000
    RECORDS (&VSV32SPA) -                          00400000
  )ENDSEL                                          00410000
  )SEL &VSV32SPU = TRACKS                          00420000
    TRACKS (&VSV32SPA) -                            00430000
  )ENDSEL                                          00440000
  )SEL &VSV32DCL *= &Z                              00450000
    DATACLAS (&VSV32DCL) -                        00460000
  )ENDSEL                                          00470000
  )SEL &VSV32MCL *= &Z                              00480000
    MGMTCLAS (&VSV32MCL) -                         00490000
  )ENDSEL                                          00500000
  )SEL &VSV32VOL *= &Z                              00510000
    VOLUME (&VSV32VOL) -                           00520000
  )ENDSEL                                          00530000
  )SEL &VSV32SCL *= &Z                              00540000
    STORCLAS (&VSV32SCL) -                         00550000
  )ENDSEL                                          00560000
    CONTROLINTERVALSIZE(4096) -                     00570000
    NUMBERED -                                       00580000
    RECORDSIZE(4089 4089) -                          00590000
    RECOVERY -                                       00600000
  )SEL &VSV32UNQ *= NO                              00610000
    UNIQUE -                                         00620000
  )ENDSEL                                          00630000
    SHAREOPTIONS(3 3) -                              00640000
  )SEL &VSV32CAT *= &Z && &VSV32PSW = &Z          00650000
    CATALOG (&VSV32CAT) -                           00660000
  )ENDSEL                                          00670000
  )SEL &VSV32CAT *= &Z && &VSV32PSW *= &Z          00680000
    CATALOG (&VSV32CAT/&VSV32PSW) -                 00690000
  )ENDSEL                                          00700000
    DATA -                                         00710000
    (NAME (&VSVIDB..DATA))                          00720000
  )ENDSEL                                          00730000
  /*                                              00740000
  //INITAKR.STEPLIB DD DSN=VIASH46.PROD.XALOAD,DISP=SHR ** IN-HOUSE** 00750002
  //INITAKR.VIASYSIN DD *                          00770000
    INIT DSNNAME (&VSVIDB)                          00780000
  /*                                              00790000
  )SEL &VSV32CAT *= &Z                              00800000
  //INITAKR.STEPCAT DD DSN=&VSV32CAT,              00810000
  //                                          00820000
    DISP=SHR                                         00830000
  /*                                              00840000
  )ENDSEL                                          00840000
  //DELETE.SYSIN DD *                              00850000
    DELETE &VSVIDB PURGE                             00860000
  /*                                              00870000
  )SEL &VSV32CAT *= &Z                              00880000
  //DELETE.STEPCAT DD DSN=&VSV32CAT,              00890000
  //                                          00900000
    DISP=SHR                                         00910000
  /*                                              00920000
  )ENDSEL                                          00920000
  )CM                                              00930000
  )CM *** END OF VIASAKRA ***                      00940000

```

Figure 89 • VIASAKRX JCL for a VSAM AKR (1 of 2)

```

)CM ***** 00010000
)CM * ASG, INC.          ASG-CENTER Rx.x          MONTH YYYY * 00020003
)CM *                   *                       * 00030000
)CM * SKELETON JCL FOR SUBMITTING A BATCH JOB TO EXPAND A * 00040000
)CM *   ASG APPLICATION KNOWLEDGE REPOSITORY (AKR) . * 00050003
)CM ***** 00060000
)CM 00070000
)CM 00080000
)CM ***** 00090000
)CM * JOB STATEMENT INFORMATION * 00100000
)CM ***** 00110000
)CM 00120000
&VSV11JB1 00130000
&VSV11JB2 00140000
&VSV11JB3 00150000
&VSV11JB4 00160000
)CM 00170000
)CM ***** 00180000
)CM * IN-STREAM JCL PROCEDURE * 00190000
)CM ***** 00200000
)CM 00210000
)IM VIASAKXP OPT 00220000
//          PEND 00230000
//* 00240000
)CM 00250000
)CM ***** 00260000
)CM * EXECUTE JCL STATEMENTS * 00270000
)CM ***** 00280000
)CM 00290000
//VIASAKRX EXEC VIASAKXP 00300000
//* 00310000
)SEL &VSV32VAC = YES 00320000
//DEFAKR.SYSIN DD * 00330000
  DEFINE CLUSTER - 00340000
    (NAME(&VSVIDB..EX) - 00350000
)SEL &VSV32SPU = CYLINDERS 00360000
  CYLINDERS(&VSV32SPA) - 00370000
)ENDSEL 00380000
)SEL &VSV32SPU = RECORDS 00390000
  RECORDS(&VSV32SPA) - 00400000
)ENDSEL 00410000
)SEL &VSV32SPU = TRACKS 00420000
  TRACKS(&VSV32SPA) - 00430000
)ENDSEL 00440000
)SEL &VSV32DCL ^= &Z 00450000
  DATACLAS(&VSV32DCL) - 00460000
)ENDSEL 00470000
)SEL &VSV32MCL ^= &Z 00480000
  MGMTCLAS(&VSV32MCL) - 00490000
)ENDSEL 00500000
)SEL &VSV32VOL ^= &Z 00510000
  VOLUME(&VSV32VOL) - 00520000
)ENDSEL 00530000
)SEL &VSV32SCL ^= &Z 00540000
  STORCLAS(&VSV32SCL) - 00550000
)ENDSEL 00560000
  CONTROLINTERVALSIZE(4096) - 00570000
  NUMBERED - 00580000
  RECORDSIZE(4089 4089) - 00590000
  RECOVERY - 00600000
)SEL &VSV32UNQ ^= NO 00610000
  UNIQUE - 00620000
)ENDSEL 00630000
  SHAREOPTIONS(3 3) - 00640000
)SEL &VSV32CAT ^= &Z && &VSV32PSW = &Z 00650000
  CATALOG(&VSV32CAT) - 00660000
)ENDSEL 00670000
)SEL &VSV32CAT ^= &Z && &VSV32PSW ^= &Z 00680000
  CATALOG(&VSV32CAT/&VSV32PSW) - 00690000

```

Figure 90 • VIASAKRX JCL for a VSAM AKR (2 of 2)

```

)ENDSEL                                00700000
      DATA -                            00710000
      (NAME(&VSVIDB..EX.DATA))           00720000
)ENDSEL                                00730000
//*                                     00740000
//INITAKR.STEPLIB DD DSN=VIASH46.PROD.XALOAD,DISP=SHR ** IN-HOUSE** 00750002
//INITAKR.VIASYSIN DD *                  00770000
INIT DSNAME(&VSVIDB..EX)                 00780000
//*                                     00790000
)SEL &VSV32CAT ^= &Z                    00800000
//INITAKR.STEPCAT DD DSN=&VSV32CAT,      00810000
//                                       00820000
      DISP=SHR
)ENDSEL                                00830000
//*                                     00840000
)SEL &VSV32EXD ^= MIX                    00850000
//REPRO.SYSIN DD *                      00860000
REPRO INDATASET(&VSVIDB) -               00870000
      OUTDATASET(&VSVIDB..EX) -         00880000
      REPLACE                            00890000
//*                                     00900000
)SEL &VSV32CAT ^= &Z                    00910000
//REPRO.STEPCAT DD DSN=&VSV32CAT,        00920000
//                                       00930000
      DISP=SHR
)ENDSEL                                00940000
//*                                     00950000
//RESIZE.STEPLIB DD DSN=VIASH46.PROD.XALOAD,DISP=SHR ** IN-HOUSE ** 00960002
//RESIZE.VIASYSIN DD *                  00980000
RESIZE DSNAME(&VSVIDB..EX)               00990000
//*                                     01000000
)SEL &VSV32CAT ^= &Z                    01010000
//RESIZE.STEPCAT DD DSN=&VSV32CAT,        01020000
//                                       01030000
      DISP=SHR
)ENDSEL                                01040000
)ENDSEL                                01050000
//*                                     01060000
)SEL &VSV32EXD = MIX                    01070000
//COPY.VIASYSIN DD *                   01080000
COPY * NOREPLACE                        01090000
)ENDSEL                                01100000
//RENAME.SYSIN DD *                    01110000
DELETE &VSVIDB PURGE                    01120000
                                           01130000
ALTER &VSVIDB..EX -                      01140000
      NEWNAME(&VSVIDB)                  01150000
)SEL &VSV32VAC = YES                     01160000
ALTER &VSVIDB..EX.DATA -                 01170000
      NEWNAME(&VSVIDB..DATA)           01180000
)ENDSEL                                01190000
//*                                     01200000
)SEL &VSV32CAT ^= &Z                    01210000
//RENAME.STEPCAT DD DSN=&VSV32CAT,        01220000
//                                       01230000
      DISP=SHR
)ENDSEL                                01240000
//*                                     01250000
//DELETE.SYSIN DD *                    01260000
DELETE &VSVIDB..EX PURGE                01270000
//*                                     01280000
)SEL &VSV32CAT ^= &Z                    01290000
//DELETE.STEPCAT DD DSN=&VSV32CAT,        01300000
//                                       01310000
      DISP=SHR
)ENDSEL                                01320000
//*                                     01330000
)CM                                     01340000
)CM *** END OF VIASAKRX ***              01350000
)CM                                     01360000

```

To expand an AKR, replace *XX* in the `NAME (XX . EX)`, `NAME (XX . EX . DATA)`, `DSNAME (XX . EX)`, `INDATASET (XX)`, and the `OUTDATASET (XX . EX)` parameters with the name of the AKR to be expanded. Then replace *XX (XX)* with the allocation units values and quantities for your site.

To rename an AKR, replace the *XX* in the `DELETE` statement with the AKR to be renamed, then enter the new AKR name in the `NEWNAME (XX)` parameter.

12

Application Conversion

Converting Recap Applications to Recap 3.2 and later

To take advantage of the more accurate function point values calculated in the 3.2 or higher release of Recap, ASG recommends that you do a full analyze of your applications from Recap.

Note: _____

You must reanalyze all COBOL programs from versions of Recap prior to 3.2 to convert them to a version of Recap 3.2 and later.

13

Report Options

This Chapter contains report options used to control report generation and to specify various report formats. Each report option is summarized in this section. The options may be set during installation but you may override the defaults by changing the JCL when you generate the reports. (See ["Generating Recap Reports" on page 44.](#))

In this table, the product default for each option is underlined and abbreviations are shown in uppercase (e.g., AplCmp abbreviates to AC, and NOAplCmp abbreviates to NOAC). These defaults reflect the information on the installation tape. See ["Report Descriptions" on page 85](#) for a list of the actual reports that were selected to be generated.

Report Options	Descriptions
<u>AplCmp</u> NOAplCmp	Generates the Application Comparison report. The default is APLCMP.
<u>AplDef</u> NOAplDef	Generates the Application Definition report. The default is APLDEF.
<u>AplExc</u> NOAplExc	Generates the Application Exceptions report. The default is APLEXC.
<u>AplFpa</u> NOAplFpa	Generates the Application Function Point Analysis report. The default is APLFPA.
<u>AplMtc</u> NOAplMtc	Generates the Application Metrics report. The default is APLMTC.
<u>AplPrg</u> NOAplPrg	Generates the Application Progress report. The default is APLPRG.
<u>BaNner</u> NOBaNner	Produces a banner page that precedes the Table of Contents for the generated reports. The default is BANNER.
CoLon=: (colon)	Specifies the substitution character to be used in place of colons on the reports. The default is : (colon).
<u>EntMtc</u> NOEntMtc	Generates the Enterprise Metrics report. The default is ENTMTC.

Report Options	Descriptions
<u>EntExc</u> NOEntExc	Generates the Enterprise Exceptions report. The default is ENTEXC.
<u>ExcSum</u> NOExcSum	Generates the Executive Summary report. The default is EXCSUM.
LiNesperpage= <u>60</u>	LINESPERPAGE is used to specify the number of lines to be printed on each page of the reports. The default is 60.
<u>PgmCmp</u> NOPgmCmp	Generates the Program Comparison report. The default is PGMCMP.
<u>PgmHst</u> NOPgmHst	Generates the Program History report. The default is NOPGMHST.
<u>PgmPrg</u> NOPgmPrg	Generates the Program Progress report. The default is PGMPRG.
VCHAR= <u> </u> - VE= <u> </u> -	Specifies the substitution character to use in place of vertical bars on the reports. The default is (vertical bar).
<u>MasterIndex</u> NOMasterIndex	Generates the report master index. The default is MASTERINDEX.

The Report Options Table lists the default report options as defined on the installation tape and their corresponding names.

Report Options	Execution JCL PARM Values
1. Report-Appl-Comparison=YES (Default) Report-Appl-Comparison=NO	APLCMP or AC NOAPLCMP or NOAC
2. Report-Appl-Definition=YES (Default) Report-Appl-Definition=NO	APLDEF or AD NOAPLDEF or NOAD
3. Report-Appl-Exception=YES (Default) Report-Appl-Exception=NO	APLEXC or AE NOAPLEXC or NOAE
4. Report-Appl-FunctionPoint=YES (Default) Report-Appl-FunctionPoint=NO	APLFPA or AF NOAPLFPA or NOAF
5. Report-Appl-Metrics=YES (Default) Report-Appl-Metrics=NO	APLMTC or AM NOAPLMTC or NOAM

Report Options		Execution JCL PARM Values
6. Report-AppI-Progress=YES	(Default)	APLPRG or AP
Report-AppI-Progress=NO		NOAPLPRG or NOAP
7. Report-Banner-Page=YES	(Default)	BANNER or BN
Report-Banner-Page=NO		NOBANNER or NOBN
8. Character-Colon=:		COLON or CL
9. Report-Enterprise-Metrics=YES	(Default)	ENTMTC or EM
Report-Enterprise-Metrics=NO		NOENTMTC or NOEM
10. Report-Enterprise-Exception=YES	(Default)	ENTEXC or EE
Report-Enterprise-Exception=NO		NOENTEXC or NOEE
11. Report-Executive-Summary=YES	(Default)	EXCSUM or ES
Report-Executive-Summary=NO		NOEXCSUM or NOES
12. Report-Lines-Per-Page=60		LINESPERPAGE or LN
13. Report-Pgm-Comparison=YES	(Default)	PGMCMP or PC
Report-Pgm-Comparison=NO		NOPGMCMP or NOPC
14. Report-Pgm-Metric-History=YES		PGMHST or PH
Report-Pgm-Metric-History=NO	(Default)	NOPGMHST or NOPH
15. Report-Program-Progress=YES	(Default)	PGMPRG or PP
Report-Program-Progress=NO		NOPGMPRG or NOPP
17. Character-Vertical-Bar=	(Default)	VE VCHAR
18. Report-Master-Index=YES	(Default)	MASTERINDEX OR MI
Report-Master-Index=NO		NOMASTERINDEX OR NOMI

14

Help Facility

This chapter describes the various online Help facilities available and methods to access them, and contains these sections:

Section	Page
Help Navigational Commands	229
Screen and Pop-up Help	230
Report Help	231
General Information	232
Specific Information	233
Help Abends	234
Help Messages	235
Printing Messages	236

Comprehensive and context sensitive Help facilities are provided that answer most questions online. The Help Tutorial contains help information for several subjects, such as screens, pop-ups, reports, messages, and abends. The Help Tutorial also includes a Table of Contents that describes each major Recap function, and a comprehensive Index for viewing specific information.

Recap online help facilities can be reached through several means. Help can be requested by selecting Help on the action bar, pressing PF1/13, or by typing HELP or ? (question mark) in the command input area on any screen or pop-up.

This table lists the various online help information provided by Recap, and the means of accessing them.

Help Topic	How to Access
Screen and pop-up help (See " Screen and Pop-up Help " on page 230.)	Help for the current screen or pop-up is requested by entering the HELP primary command, pressing PF1/13, or by selecting Help ► Current Screen from the Help pulldown. No messages can appear on the screen or pop-up at the time this help is requested.
Report help (See " Report Help " on page 231.)	Help for Recap reports is requested by entering HELP REPORTS, entering REPORTS and pressing PF1/13, or by selecting Help ► All reports from the Help pulldown. Help for specific reports may also be requested through the Help Tutorial Index.
General information (See " General Information " on page 232.)	General help information is requested by issuing the TOC command from within the Help Tutorial, or by selecting Help ► Table of contents.
Specific information (See " Specific Information " on page 233.)	Help for specific subjects is requested by entering the INDEX command from within the Help Tutorial, by selecting option 5 on the Help Table of Contents, or by selecting Help ► Index. Help for a specific topic can be viewed by selecting the appropriate index entry.
Abends (See " Help Abends " on page 234.)	Help for ESW user abends is requested by entering the HELP ABENDS command, or selecting Help ► Common Abends. The Abends screen displays. Select topic 2 on this screen to display the ESW Abend Codes screen, which lists all the ESW user abends, and explanations for each abend.
Messages (See " Help Messages " on page 235.)	Help for a short message, displayed in the upper right corner of the screen, is requested by entering the HELP command or pressing PF1/13. The corresponding long message displays. Help for a specific long message can be displayed by entering the HELP command with the msg# operand, by selecting Help ► Current Message or by selecting Help ► Specific Message.

Help Navigational Commands

All of the online help subjects listed in the previous table are contained in the Help Tutorial. Each online help topic can be reached from anywhere within the Help Tutorial by going through the Help Table of Contents or Index. Once the Help Tutorial is accessed, these are the commands available for navigating within the Help Tutorial:

Help Command	Purpose
BACK	Redisplays the previous Help Tutorial screen.
END	Exits the Help Tutorial.
ENTER	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	On an Index screen, entering an alphabetic character displays the Index screen corresponding to that character.

Screen and Pop-up Help

Help for the current screen or pop-up is requested by entering the HELP primary command; by pressing PF1/13 with no messages appearing on the screen; by entering the HELP SCREEN command; or by selecting Help ► Current Screen. The Help Tutorial for the current screen or pop-up displays.

The Help Tutorial for each screen or pop-up describes all the options available on that screen or pop-up, lists descriptions of all the fields, and notes any special processing considerations.

[Figure 91](#) shows the Help Tutorial for the Enterprise Metrics View screen.

Figure 91 • Screen Help Example

```
ASG-Recap      ----- View - Enterprise Metrics ----- HELP
==>
The View - Enterprise Metrics screen is used to view the most current metrics
data calculated for an enterprise.  This screen is displayed by selecting
Enterprise Metrics... from the View pull-down and entering the AKR data set
name on the View - Open Metrics Repository pop-up.

When this pop-up is first displayed, metrics are sorted alphabetically by
application name.  To change the sort order, select Sort... from the Options
pull-down to display the Options - Sort Metrics pop-up to specify the new sort
order.

NOTE:  The View - Enterprise Metrics screen has a shortened action bar.  The
following section describes the action bar choices on this screen.

Action Bar Descriptions

File          The File pull-down contains the following actions:
Print
Prints the metrics data for the enterprise.  To release
the print, select the Options pull-down and then choose
Process list file...
(More...press ENTER to continue.)
```

Report Help

Help for Recap reports is requested by entering HELP REPORTS; entering REPORTS and pressing PF1/13; or by selecting Help ► All reports. The ASG-Recap Reports help table of contents screen displays, as shown in [Figure 92](#). Any report can be selected for further information. Help for specific reports can also be requested through the Help Index.

Figure 92 • Recap Report Help Screen

```
ASG-Recap R6.0 ----- ASG-Recap Reports ----- HELP
===> _

The following will be presented in sequence, or may be selected by number:

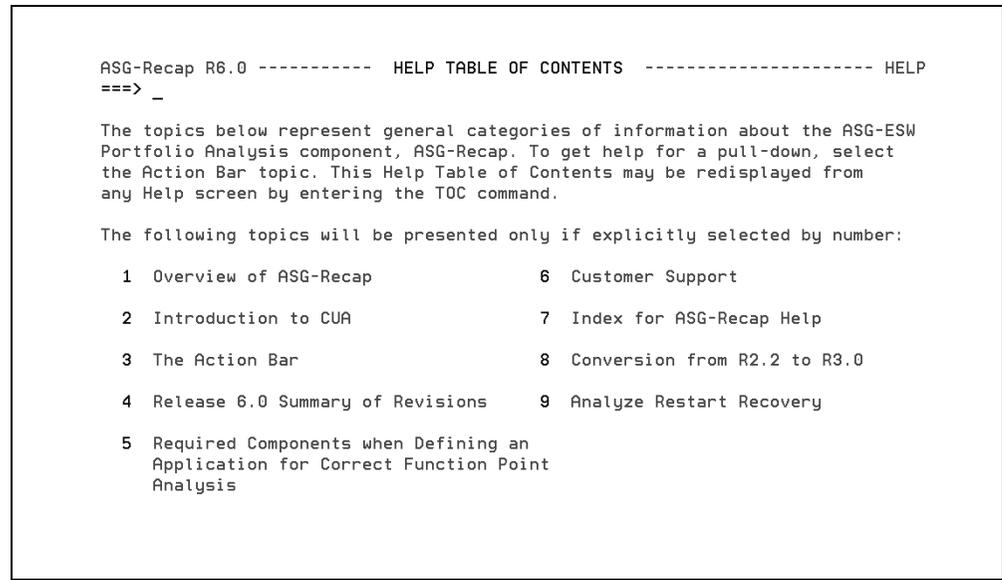
 1 - Application Comparison
 2 - Application Definition
 3 - Application Exceptions
 4 - Application Function Point Analysis
 5 - Application Metrics
 6 - Application Progress
 7 - Application Summary
 8 - Enterprise Exceptions
 9 - Enterprise Metrics
10 - Executive Summary
11 - Master Index
12 - Program Comparison
13 - Program Metrics History
14 - Program Progress
15 - Table of Contents for Reports
```

General Information

General help information is requested by entering the Help Tutorial and then entering the TOC command, or by selecting Help ► Table of contents.

[Figure 93](#) shows the Help Table of Contents.

Figure 93 • Help Table of Contents



Specific Information

Help for specific topics is requested by entering the Help Tutorial and then entering the INDEX command; by selecting option 6 on the Help Table of Contents; or by selecting Help ► Index. Help for a specific topic can then be viewed by selecting the appropriate Index entry.

On any Index screen, entering an alphabetic character displays the Index screen corresponding to that character.

[Figure 94](#) shows a Help Index screen.

Figure 94 • Help Index Example

```
ASG-Recap R6.0 ----- Index A ----- HELP
===> _

To select a topic, enter the two-character identifier.

A1 - Action Bar
A2 - Action Bar for View screens
A3 - Add Copylib pop-up
A4 - Add CICS CSD Dataset
A5 - Add CICS Library
A6 - Add Function Points - External Input Type pop-up
A7 - Add Function Points - External Interface File Type pop-up
A8 - Add Function Points - External Inquiry Type pop-up
A9 - Add Function Points - External Output Type pop-up
A10 - Add Function Points - Internal Logical File Type pop-up
A11 - Add IMS Library
A12 - Add JCL Library
A13 - Add Library - Librarian pop-up
A14 - Add Load Library pop-up
A15 - Add Source Library pop-up
A16 - Add Proclib pop-up

(More...press ENTER to continue.)
```

Help Abends

Help for ESW user abends is requested by entering the HELP ABENDS command; entering ABENDS in the command input area and pressing PF1/13; or selecting Help ► Common Abends. The ABENDS screen displays, as shown in [Figure 95](#). Selecting Topic 2 on this screen displays the ESW Abend Codes screen, which lists all the ESW user abend messages, and explanations for each message.

Figure 95 • ASG Abend Codes Screen

```
ASG-Recap 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

Recap messages are displayed in the long message area, which is the third line on a pop-up or the fifth line on a screen. The format for messages is:

VIAnnnnx text

where:

nnnn is the message number.

x is one of the severity levels listed below.

text is the long or short message text.

Level	Description
I	Informational - no required action.
W	Warning - an error condition exists that is not critical.
E	Error - a critical error condition exists.
D	Disaster - a serious error condition exists; the product is unable to continue.
T	Termination - the product terminated with the specified error.

Short messages are displayed when available. Long messages are displayed if a short message does not exist, or when help is requested immediately following a displayed short message.

Help for a specific message is displayed by entering the HELP primary command, followed by the message number, or by selecting Help ► Current Message or Help ► Specific Message. The Help Explanation and Action Panel for that message displays (see [Figure 96 on page 236](#)).

Figure 96 • Help Explanation and Action Panel

```
Command ==> _____ HELP Explanation and Action Panel _____ Scroll ==> CSR
Additional support may be found at our Web Site: support.asg.com
ASG6509 "#1" IS VALID ONLY AFTER AN APPLICATION HAS BEEN OPENED.
EXPLANATION:
  The specified command is valid only after an application has been
  opened from the Open Application action or selected from the AKR
  directory.
ACTION:
  Go to the Open Application action and select an application.
***** BOTTOM OF DATA *****
```

Printing Messages

All Recap 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.VIACENxx.CNTL(VIASMPRT). The entire message file prints 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; the default value for END is 5000. If only the START value is entered, messages print starting at the message number specified and ending with 5000. If only the END value is entered, messages print starting with 1 and ending with the message number specified.

The NOTES keyword specifies that any notes associated with a message print. The default is NONOTES. Typically, notes are provided to illustrate Center primary commands.

[Figure 97](#) shows the VIASMPRT JCL. [Figure 98 on page 238](#) shows the output from the job.

Figure 97 • VIASMPRT JCL

```
//ASG JOB ( ),'ASG-CENTER VIASMPRT'
//* INSERT '/*ROUTE PRINT NODE.USER' HERE IF NEEDED.
//*
//* *****
//* * ASG, INC.          ASG-CENTER  RX.X          MMM, YYYY *
//* *                                                           *
//* *          UTILITY TO PRINT ASG MESSAGES *
//* *                                                           *
//* *****
//VIASMPRT PROC VIASOFT='ASG', HIGH LEVEL NODE OF ASG DATA SETS
//          CENTER='VIACENXX', MIDDLE NODE OF ASG DATA SETS
//          SYSOUT='*',          PRINT OUTPUT MESSAGE CLASS
//          PRM='',             PARM FOR MESSAGES TO BE PRINTED
//
// *****
// *
// *          M E S S A G E    P R I N T    U T I L I T Y *
// *
// * THIS PROGRAM WILL PRINT ALL OF THE MESSAGES IN THE ASG *
// * MESSAGE FILE AND THE HELP TEXT ASSOCIATED WITH EACH *
// * MESSAGE IT WILL PRINT THE ENTIRE FILE BY DEFAULT. YOU MAY *
// * SELECT A GIVEN RANGE OF MESSAGES BY SPECIFYING THE OPTION- *
// * AL PARAMETER KEYWORDS: START AND END. FOR EXAMPLE: *
// *          PRM='START=300,END=499' *
// * WILL PRINT MESSAGES NUMBER 300 THROUGH 499, INCLUSIVE. *
// * THE DEFAULT VALUES FOR START AND END ARE 1 AND 99999 *
// * RESPECTIVELY. CONSEQUENTLY THE PRM VALUE 'END=300' WILL *
// * PRINT MESSAGES 1 THROUGH 300, AND THE PRM VALUE *
// * 'START=4000' WILL PRINT MESSAGES 4000 THROUGH 99999. *
// *
// * AN OPTIONAL KEYWORD, NOTES, WILL ALSO PRINT ANY NOTES *
// * ASSOCIATED WITH A MESSAGE. *
// *
// * ADDITIONALLY, THE KEYWORD 'ALL' WILL EXPLICITLY PRINT ALL *
// * MESSAGES. *
// * *****
//
//
//VIAMPRT EXEC PGM=VIASMPRT,REGION=4096K,
//          PARM='&PRM'
//STEPLIB DD DSN=&ASG..&CENTER..LOADLIB,DISP=SHR
//VIAMSGS DD DSN=&ASG..&CENTER..VIAMSGS,DISP=SHR
//SYSPRINT DD SYSOUT=&SYSOUT
//VIAPRINT DD SYSOUT=&SYSOUT
//VIALOG DD SYSOUT=&SYSOUT
//SYSUDUMP DD SYSOUT=&SYSOUT
//
//          PEND
//
//
//VIASMPRT EXEC VIASMPRT          PRINT MESSAGES
//
```

Figure 98 • VIASMPRT Output

```
PRINTING MESSAGES FROM 0771 TO 0772.  
2 MESSAGES PRINTED.  
END OF MESSAGE PRINT PROCESSING.  
VIA0771 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 re-enter the Edit screen with COBOLII  
selected; then you may enter commands for SUBSET COBOLII.  
  
VIA0772 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.
```

15

Sample AKR

An AKR with sample applications is provided with Recap. All applications contain metrics, function point analysis, and program quality data. The sample AKR should be used as a training tool to help the first-time user learn the online portion of the product as well as the reports generated by Recap. Because the original COBOL source is not available, an analyze cannot be executed on any of the sample applications.

The sample AKR is stored on the Recap installation tape. For installation information, see the *ASG-Recap Installation Guide*.

action bar

The line of keywords at the top of a screen. Each keyword represents a category of actions that may be performed on that screen. An action is selected by moving the cursor to the desired keyword and pressing Enter. See ["Introduction" on page 1](#) for more information.

AKR

See ["Application Knowledge Repository \(AKR\)" on page 242](#).

Alias Of

A field on a pop-up listing entries in the AKR. If the analyzed program contains an ENTRY point, Alias Of is the name of the program that contains the ENTRY point. If the name in the PROGRAM-ID statement was overridden at the time the analyze job was submitted, Alias Of is the name that was entered in the AKR program name field on the Analyze - Submit Application pop-up.

analyze

A batch process to gather portfolio analysis information, including organization, data relationships, and execution paths, Function Point analysis, and software metrics. This process stores the portfolio analysis information in the specified host Application Knowledge Repository (AKR). See the *ASG-Application Definition and Analysis User's Guide* for more information.

analyze options

Run-time options, similar to COBOL compiler options, used to control the Analyzer processing. Default values are established at installation time and can be overridden by editing the Analyzer JCL or by using the Analyze screens. ["Building and Analyzing an Application" on page 23](#) contains a complete description of each Analyze option.

analyze summary report

A summary of the run-time statistics and diagnostic messages that are produced when an Analyze job completes.

application

An application is any group of programs that you want to analyze/view as a whole. These programs would all be defined within the same application. See ["Building and Analyzing an Application" on page 23](#) for more information on defining an application.

Application Analytical Engine (AAE)

Analyzes all components of an application to allow portfolio analysis.

Application Definition

The COBOL, Load Module, JCL, CICS, and IMS components and attributes associated with an application.

Application Definition Facility

Provides the ability to define application attributes used to conduct an automated inventory analysis. See ["Building and Analyzing an Application" on page 23](#) for more information on using the application definition facility.

Application Knowledge Repository (AKR)

A BDAM or VSAM file organization that contains all analysis information produced by the Analyze job. Multiple AKRs can be defined. See ["AKR Utilities" on page 189](#) for more information.

attributes

The analysis parameters, copy libraries, IDMS information, maclibs, and procedure libraries used by a specific library or member.

command input area

The field on Recap screens where primary commands are entered, indicated by ==> on the fourth line of the screen.

complex processing

The general system characteristic used to assess the degree to which the processing logic influences the development of the application.

CUA (Common User Area)

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

data communication

The general system characteristic used to assess the degree to which an application communicates directly with the processor.

Data Element Type

User recognizable, non-recurring fields residing in an Internal Logical File or External Interface File and used by an External Input, External Output, or External Inquiry. The Data Element Type is used as one of the factors to determine functional complexity.

data name

A standard COBOL term for fields defined in the DATA DIVISION of a COBOL program. Variable names, files, groups, array elements, and fully qualified datanames.

DBCS

See [Double Byte Character Set \(DBCS\)](#).

Degree of Influence (DI)

A measurement, assigned to each General System Characteristic, that is used in a calculation to determine the Value Adjustment Factor.

DDL

DB2 SQL Data Definition Language, a subset of SQL.

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.

discovered component

Components discovered while processing JCL members during the analyze process that are not included in the application definition, but that appear to be needed by components that are defined. For example, the analyze may discover a source library in a COBOL compile JCL member that is not included in the application definition.

Discovered components may be resolvable or unresolvable. Resolvable components can be added to the application definition by Recap. Unresolvable components must be manually added to the application definition because Recap is unable to determine where they should be added.

distributed processing

The general system characteristic used to assess the degree to which the application transfers data among components of the application.

DL/I | DL/1

The database (DB) portion of the IMS system.

DML

DB2 SQL Data Manipulation Language, a subset of SQL.

Double Byte Character Set (DBCS)

A character set that uses two bytes to represent each character. Various Double Byte Character Sets are used with languages such as Chinese and Japanese that cannot be represented with single byte codes.

end-user efficiency

The general system characteristic used to assess degree of consideration for human factors and ease of use for the user of the application measured.

Enhancement Function Points

The Enhancement Function Point count for an application is a measure of the modifications to an existing application that add, change, or delete user function within the scope of the project. This measurement can be used to calculate the effort used to enhance an application.

enterprise

All applications defined in the same Application Knowledge Repository (AKR).

Export Facility

Provides the ability to export the application definition using keyword control statements. The export facility also provides the ability to export application analysis data from the AKR in CDF format. This data can then be used to build DB2 relational tables or to transfer to a workstation for loading into database applications.

external input

A component that maintains (add, changes, or deletes) data on an Internal Logical File. For example, if a screen within the application contains actions that add, change, and delete information, the screen counts for three External Inputs.

External Inquiry (EQ)

Represents the functionality provided for queries of Internal Logical Files and External Interface Files.

External Interface File (EIF)

Logical group of data that is utilized by the application but maintained by another application.

External Output (EO)

Processes that send data or control information outside the application's boundary. Examples of External Outputs include reports generated by the application and data that is formatted and processed for use by another application.

facilitate change

The general systems characteristic used to assess the degree to which an application has been developed for easy modification of processing logic or data structure.

File Types Referenced (FTR)

Counted for each Internal Logical File maintained (for an External Input only) or each External Interface File or Internal Logical File referenced during the processing of an External Input, External Output, or External Inquiry. The File Types Referenced is one of the factors used to determine the functional complexity of an External Input, External Output, and External Inquiry.

general systems characteristics

A set of 14 characteristics used to assess the influence of environmental, physical, or logistical problems that may be encountered while implementing, operating, or maintaining a system. These characteristics are used to calculate an adjusted function point number for a particular application.

heavily used configuration

The general system characteristic used to assess the degree to which computer resource restrictions influenced the development of the application.

help

Recap has three levels of Help: Long messages, notes, and tutorial screens. Specific command information is available by entering a command, then pressing PF1/13. The Help facility can also be accessed from the Help pull-down or any Recap screen. See the online help or ["Help Facility" on page 227](#) for more information.

Import Facility

Provides the ability to import an application definition that has been defined using keyword control statements. Usually, the input file is produced by the Export Facility. The input file must be a sequential file containing 80 byte records.

installation ease

The general systems characteristic used to assess the degree to which conversion from previous environments influenced the development of the application.

Label Name

Any PROCEDURE DIVISION paragraph or section name and the PROCEDURE and PROC literals.

live exit

An abnormality in program control caused by out of perform range GO TOs and overlapping perform ranges.

log file

A file that is allocated by Recap and used for error messages and log commands.

logical lines of code

The value for counting logical lines of code in a program is derived by adding the physical line counts of the Identification and Environment divisions, the number of statements in the Data Division (where a statement ends in a period), and the number of statements in the Procedure Division.

long message

A diagnostic or error message that displays on line five of Recap screens or line 3 of pop-ups. 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 can be the alias name found in the Application Knowledge Repository (AKR).

metrics

A measure of program quality or complexity.

missing component

A component that is expected but not found during the analyze. This could be a component that was found in the previous analyze and is now deleted or a component that is referred to by another component, but that is not in the application definition.

multiple sites

The general systems characteristic used to assess the degree to which an application has been developed for multiple locations and user organizations.

online data entry

The general systems characteristic used to assess the degree to which an application has been developed for interactive transactions.

online update

The general systems characteristic used to assess the degree to which data is entered through interactive transactions.

operational ease

The general systems characteristic used to assess the degree to which the application attends to operational aspects such as start-up, back-up, and recovery processes.

performance

The general systems characteristic used to assess the degree to which response time and throughput performance considerations influenced the application development.

perform range

A perform range consists of the source code contained in a PERFORM statement, and includes all code that is or could be executed as a result of GO TOs, PERFORMs, etc., within that PERFORM.

physical lines of code

The value for counting physical lines of code in a program is derived by adding all of the statements in the program, including comments, blank lines, and expanded copy/include statements.

pop-up

A window that appears as the result of selecting an item on a pull-down or pop-up, or as the result of entering certain commands. It is superimposed on the screen to allow entry of information for the requested action. See ["Introduction" on page 1](#) for more information.

portfolio analysis

A portfolio analysis consists of: inventory of all application components, functional priority of applications with respect to your business, complexity and quality measurements and degree/frequency of change.

primary command

An instruction entered in the command input area of the screen.

program

Program source member name, the name specified in the IDENTIFICATION DIVISION of a COBOL program, or the CSECT name of a program that is not COBOL.

pull-down

The list that appears when an action is selected on the action bar. On a pull-down, actions followed by ... display a pop-up when selected. Actions not followed by ... immediately activate internal commands. See ["Introduction" on page 1](#) for more information.

Record Element Type (RET)

Logical subsets of Internal Logical Files and External Interface Files. The Record Element Type is one of the factors used to determine the functional complexity of Internal Logical Files and External Interface Files.

recursion

A perform range or paragraph that performs itself.

reusability

The general systems characteristic used to assess the degree to which the creation or use of standardized software specifications influenced the development of the application.

reports

Hard copy reports of the results of the inventory analysis.

resolvable discovered components

Discovered components that Recap can add to the application definition. For resolvable discovered components, Recap can determine where to add the component within the application definition.

SBCS

See ["Single Byte Character Set \(SBCS\)" on page 248](#).

screen

A full-width display of information containing an action bar as the first line. Recap screens are modeled after TSO/ISPF screens. See ["Introduction" on page 1](#) for more information.

short message

A diagnostic or error message that displays in the upper right corner of Recap screens. Pressing PF1/PF13 after receiving a short message displays the corresponding long message.

Single Byte Character Set (SBCS)

(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.

software metrics

Measurements that provide an objective, quantitative, and practical way to determine the quality (in terms of size, structure, complexity) of software applications.

SQL

DB2 Structured Query Language, including DML and DDL.

transaction rates

The general systems characteristic used to assess the degree to which the rate of business transaction influenced the development of the application.

unresolvable discovered components

Discovered components whose location within the application definition cannot be determined by Recap. Unresolvable discovered components must be added manually to the application definition if they are needed.

version

A set of values for metrics and exceptions (i.e., missing and dead components) related to a specific time. These values are created by a full or updated during an incremental analysis.

VIAXESUM

A file included with Recap that contains the Report Generator Programming Language. This file provides information needed to generate the Executive Summary Report. If you want to print the Executive Summary Report in mixed case letters, specify this file as the member name in the Report - Executive Summary pop-up.

VIAXESUU

A file included with Recap that contains the Report Generator Programming Language identical to VIAXESUM. If you want to print the Executive Summary Report upper case letters, specify this file as the member name in the Report - Executive Summary pop-up.

work file

The work file is used for intermediate storage during an online session. The DD name assigned to the work file is VIAUT3. The file is allocated upon entry into Recap. If the analyze abends, the work file is retained and used during restart of the analyze job. The work file is deleted when you exit the product or when Recap completes a successful analyze. See the Analyze Resource Estimate table in the *ASG-Application Definition and Analysis User's Guide* for more information.

Symbols

&assign
 description 146
 example 146
&col 151
&col, description 151
&compute
 description 147
 example 147
&lmargin and &rmargin, description 150
&newline
 description 150
 example 150
¶graph
 example 150
¶graph, example 150
&space, description 150
&while
 description 149
 example 149

A

adjusting function points
 reclassifying EO/EI pairs as EQ
 element 79
 reclassifying ILF and EIF elements 78

AKR
 allocate without ISPF 215
 definition 7
 expand without ISPF 215
 management 189
 relationship to applications 40
 VIASAKRX JCL example 219

AKR batch utility
 DELETE Batch AKR command 199
 EXPORT batch AKR command 200
 HELP batch AKR command 201
 INIT batch AKR command 202
 MOVE batch AKR command 203
 PRINT batch AKR command 204
 PUNCH batch AKR command 205

AKR utility directory report 207

AKR utility log 208
AKR with sample applications 239
alarm settings, specifying in Recap 31
Alliance
 accessing from ESW screen xvi
 description xiii
 linking xvi
analysis status report
 AKR detail 211
 AKR summary 210
 application detail 212
analyzing an application 44
analyzing applications, manual 44
application
 analyze job 44
 creating 40
 exporting definition 185
 running reports 44
 steps for naming 41
application analysis, manual 44
application comparison report
 business value vs. complexity field
 descriptions 128
 report field descriptions 125
application components, listed 42
application definition
 checklist 24
 copying 187
 creating a template 187
 creating an application 40
 defining attributes 42
 defining components 42
 establishing the boundaries of the
 application 10
 exporting 185
 importing 159
 manual reference 42
 process 24
Application Definition and Analysis, User
 Guide 42
application definition, providing an
 application description 41

- application description
 - steps for entering 41
- application exceptions pop-up, report field descriptions 123
- application function point analysis report, final calculations table 105
- application metrics report 117
- application portfolio analysis, description 8
- application progress report
 - report field descriptions 131
- application summary, description 91
- applications, relationship to AKR 40
- AutoChange
 - accessing from ESW screen xvi
 - description xiii

B

- batch AKR utility
 - AKR utility log 208
 - analysis status report 209
 - command format 192
 - command syntax 193
 - control cards 192
 - CONVERT command 196
 - COPY command 198
 - DELETE command 199
 - EXPORT command 200
 - HELP command 201
 - INIT command 202
 - JCL 191
 - MOVE command 203
 - PRINT command 204
 - print directory report 207
 - PUNCH command 205
 - punch directory 214
 - reports 207
- Bridge
 - accessing from ESW screen xvi
 - description xiii

C

- Center, description xiii
- checklist, application definition 24
- COBOL, support 5
- command syntax diagrams 193
- compile JCL, importance of 28
- compilers
 - ANSI COBOL 5
 - CASE generated COBOL 5
 - COBOL for MVS and VM 5
 - COBOL for OS/390 and VM 5
 - COBOL II 5
 - COBOL/370 5

- complexity matrix, example 13
- computing the adjusted function point value, when counting function points 16
- computing the enhancement function point count 16
- computing the value adjustment factor, when counting function points 15
- computing unadjusted function points, when counting function points 14
- confirmation pop-ups, controlling operation of 31
- continuous improvement, scenario 53
- control variable complexity metric, described 18
- conventions page xix
- copying application definition 187
- counting function point types 10
- counting function points 10
 - see also function point counting 10
- cyclomatic complexity metric, example 18

D

- data element types 12
- data item
 - uninitialized 135
 - use 135
- DEADCODE 121, 136
- defining applications, manual 42
- degree of influence 15
- detailed function point information 182
- determining functional complexity, when counting function points 12

E

- editing function points
 - reclassifying EO/EI pairs as EQ elements 79
 - reclassifying ILF and EIF elements 78
- editing Recap function points, reasons to perform edits 74
- EI, reclassifying as external inquiry 79
- EIFs, reclassifying element type 78
- Encore
 - accessing from ESW screen xvi
 - description xiv
- enhancement function point count, computing 16
- enterprise exceptions report 135
- enterprise exceptions report, report field descriptions 137
- enterprise metrics report 132
- enterprise metrics report, example 133
- EO, reclassifying as external inquiry 79

- EQ
 - how Recap counts 79
 - reclassifying EO/EI pairs as EQ 79
- essential complexity metric, described 19
- establishing the boundaries of an application, when counting function points 10
- Estimate
 - accessing from ESW screen xvi
 - description xiv
- ESW
 - description xii
 - invoking products xv
 - product integration xvi
- executive summary report
 - report generator language 93
- Executive Summary Report, printing in mixed case 46
- Executive Summary Report, printing in upper case 46
- export application definition 185
- export metrics facility
 - creating CDF files for export 179
 - detailed function point information 182
 - output from export 181
 - program and application metric information 181
 - summarized function point information 183
- export procedure, application definition 186
- F**
- File - AKR Allocate/Expand pop-up 190
- File - AKR Directory pop-up 190
- File - AKR Utility pop-up 190
- file type referenced 12
- function point
 - adjusting 74
- function point analysis, description of process 9
- function point counting
 - adjusting 74
 - computing the adjusted function point value 16
 - computing the value adjustment factor 15
 - computing unadjusted function points 14
 - determining functional complexity 12
 - establishing the boundary 10
- function points
 - as a basis for other metrics 8
 - counting 10
 - defined 8
 - definition 8
 - scenario using 51
 - uses of 51
 - viewing Recap's calculations 73
- functional complexity
 - data element types 12
 - file type referenced 12
 - record element types 12
 - using a complexity matrix 13
- functional quality
 - description 50
 - how measured 50
- G**
- general system characteristics
 - assigning score to 15
 - influencing the value adjustment factor 15
 - list of 15
 - online worksheet 43
- GSC worksheet, how to access online 43
- H**
- help
 - for abends 234
 - for general information 232
 - for messages 235
 - for pop-ups 230
 - for reports 231
 - for screens 230
 - for specific information 233
 - how to obtain 229
 - index 233
 - navigational commands 230
 - printing messages 236
 - Recap reports 231
 - topics 229
 - tutorial 229
- I**
- ILFs, reclassifying element type 78
- import facility, described 159
- import language reference
 - application information 161
 - default section 161
 - definition section 166
 - definition section example 175, 177
 - file structure 160
 - keyword list 177
 - syntax 160
- importing an application definition 159

- Insight
 - accessing from ESW screen [xvi](#)
 - description [xiv](#)
 - using analysis functions [xvi](#)
- invoking Recap [29](#)
- ISPF
 - operating systems version [4](#)
 - when not installed [223](#)
- J**
- JUMP, out of PERFORM [120](#)
- K**
- knots count metric
 - described [20](#)
 - example [20](#)
- L**
- list file, allocating [32](#)
- live exit [121](#), [136](#)
- log file, allocating [32](#)
- M**
- master index [138](#)
- master index, report field descriptions [139](#)
- measurement, when performed [49](#)
- messages
 - long [235](#)
 - printing all [236](#)
 - short [235](#)
- metrics [17](#)
- MVS [4](#)
- O**
- Options - Product Allocations pop-up [32](#)
- Options - Script File Allocations pop-up
 - description [35](#)
 - example [35](#)
- output from export [181](#)
- P**
- pop-up
 - File - AKR Allocate/Expand [190](#)
 - File - AKR Directory [190](#)
 - File - AKR Utility [190](#)
 - Function Points - General System Characteristics [42](#)
 - options - product allocations [32](#)
 - Options - Script File Allocations [35](#)
 - Script File Allocations [35](#)
- portfolio analysis, description [8](#)
- preprocessor support
 - command level CICS [5](#)
 - command level DL/I [5](#)
 - IDMS [5](#)
 - SQL [5](#)
- printer support [5](#)
- product integration [xvi](#)
- product parameters, viewing current settings [31](#)
- program and application metric information [181](#)
- program metrics history report
 - report help example [88](#)
 - software science volume field descriptions [108](#)
- program progress report [115](#)
- Q**
- quality
 - functional [50](#)
 - technical [50](#)
- quality and productivity improvement program, as supported by Recap [50](#)
- R**
- reanalyzing application, effect on previous function point edits [83](#)
- Recap
 - accessing from ESW screen [xvi](#)
 - action bar - how to select actions [4](#)
 - action bar - selecting an item [4](#)
 - action bar description [4](#)
 - analyze process [2](#)
 - CUA features [4](#)
 - CUA interfaces [4](#)
 - description [xiv](#)
 - initiating a Recap session [29](#)
 - operating systems [4](#)
 - pop-ups - description [4](#)
 - pop-ups - how to process [4](#)
 - product overview [2](#)
 - pull-down description [4](#)
- Recap option
 - APLCMP [223](#)
 - APLDEF [223](#)
 - APLEXC [223](#)
 - APLFPA [223](#)
 - APLMTC [223](#)
 - APLPRG [223](#)
 - BANNER [223](#)
 - COLON [223](#)
 - ENTEXC [224](#)
 - ENTMTC [223](#)

- EXCSUM 224
 - LINESPERPAGE 224
 - NOAPLCMP 223
 - NOAPLDEF 223
 - NOAPLEXC 223
 - NOAPLFPA 223
 - NOAPLMTC 223
 - NOAPLPRG 223
 - NOBANNER 223
 - NOENTEXC 224
 - NOENTMTC 223
 - NOEXCSUM 224
 - NOPGMCMP 224
 - NOPGMHST 224
 - PGMCMP 224
 - PGMHST 224
 - Recap reports, help 231
 - record element types 12
 - reengineering decisions, scenario 57
 - report generator language
 - control & text selection - &if 147
 - control and text selection 147
 - control and text selection - &while 149
 - data manipulation - &assign 146
 - data manipulation - &compute 147
 - data structures 144
 - data structures - primary and subsidiary tables 145
 - data structures syntax 144
 - language constructs 145
 - language constructs - data manipulation 146
 - language constructs - text substitution 146
 - output formatting 149
 - output formatting - &col 151
 - output formatting - &lmargin and &rmargin 150
 - output formatting - &newline 150
 - output formatting - ¶graph 150
 - output formatting - &space 150
 - primary and secondary input 144
 - site specific configuration 143
 - report generator programming language
 - language constructs 145
 - language constructs - text substitution 146
 - report heading, program metrics history
 - report 87
 - report help 231
 - report help, example from program metrics
 - history report 88
 - report options
 - application comparison report 223
 - application definition 223
 - application exceptions report 223
 - application function point analysis report 223
 - application metrics report 223
 - application progress report 223
 - banner page 223
 - colon substitute 223
 - enterprise exceptions report 224
 - enterprise metrics report 223
 - executive summary report 224
 - lines per page 224
 - master index 224
 - program comparison report 224
 - program history report 224
 - program progress report 224
 - substitution character 224
 - reports
 - analysis status 209
 - application metrics 117
 - application progress 130
 - application summary 91
 - colon substitution character 223
 - enterprise exceptions 135
 - enterprise metrics 132
 - generate 223
 - generating 44
 - program progress 115
 - substitution character 224
 - suppress 223
- S**
- Sample AKR 239
 - Script File Allocations pop-up 35
 - SmartDoc
 - accessing from ESW screen xvi
 - description xiv
 - VCHAR option 224
 - SmartEdit
 - accessing from ESW screen xvi
 - description xv
 - SmartTest
 - accessing from ESW screen xvi
 - description xv
 - software metrics
 - control variable complexity 18
 - described 50
 - essential complexity 19
 - for complexity 18
 - for logical structure 19
 - for physical structure 20
 - for size 19

- knots count [20](#)
- software science volume [19](#)
- uses of [17](#)
- software science volume metric,
 described [19](#)
- starting Recap [29](#)
- summarized function point information [183](#)

T

- technical quality, how measured [50](#)
- total degrees of influence [16](#)

U

- unadjusted function point values
 - complexity rating association [14](#)
 - example [14](#)
- user interface [4](#)
- user options
 - allocating the log and list files [32](#)
 - listed [31](#)
 - Log/List Definition [33](#)
 - product parameters [31](#)
 - Script File Allocations [35](#)
- uses of software metrics [17](#)

V

- value adjustment factor
 - adjusted function point total effect [16](#)
 - computing [43](#)
 - constants used in equation [16](#)
- verifying user options, why important [31](#)
- version [248](#)
- VIAIN DD statement, DPARM
 - parameter [223](#)
- VIASAKRA JCL [215](#)
- VIASAKRX JCL [215](#)
- VIASMPRT program, JCL [236](#)
- VIASMPRTprogram [236](#)
- VIAXESUM [248](#)
- VIAXESUU [248](#)
- viewing current settings, for product
 - parameters [31](#)
- viewing Recap's function point analysis [73](#)

W

- work files
 - how Recap counts [74](#)
 - removing from function point
 count [74](#)

ASG Worldwide Headquarters Naples Florida USA | asg.com