

Advantage™ CA -Optimizer®

User Guide

7.0



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2002 Computer Associates International, Inc. (CA)

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introduction

Chapter 2: Using Optimizer

Optimizer Options	2-2
Compile-Time Options	2-2
Commonly Used Optimizer Options	2-3
JCL	2-4
Usage Considerations	2-4
Optimizer Reports	2-5
Options Page	2-6
Merged Data Map (MDMAP)	2-6
Merged Listing (MLIST)	2-9
Condensed Cross-Reference Dictionary (CXREF)	2-11
Program Global Table (PGT) Memory Map	2-12
Program Global Table (PGT)	2-13
Task Global Table Memory Map (TGT)	2-14
PERFORM Range Table and ALTERed GO TO Table	2-15
Special Usage Note for CICS Programs	2-18
CA-Optimizer Defined Recursion Examples	2-19
CA-Optimizer Non-Recursion	2-19
OS/VS COBOL Diagnostic Messages and Compiler Statistics Report	2-20
Optimization Results Report	2-21
Compiler Source Listing	2-22
Merged Data Map (MDMAP)	2-22
Compiler Data Map (MAP)	2-23
Condensed Listing of the Optimized Object Code (CLIST)	2-24
Procedure Map Object Listing (PMAP)	2-25
Selected Procedure Map (SELECT)	2-26

Chapter 3: Using the COBOL Postprocessor

COBOL Postprocessor Drivers	3-1
JCL	3-1
Sequential Listing Files	3-2
Partitioned Listing Data Sets	3-3
CA-Librarian	3-4
CA-Panvalet	3-5
CA-Endevor	3-6
COBOL Postprocessor Options	3-7
COBOL Postprocessor Reports	3-8
Options Page	3-8
Merged Data Map	3-9
Merged CLIST	3-10
Compiler Data Map (DMAP)	3-12
Compiler Condensed Procedure Map (CLIST)	3-13
Compiler Procedure Map Object Listing (PMAP)	3-14
Condensed Cross-Reference (CXREF)	3-15
Compiler Cross-Reference (XREF)	3-16
Usage Considerations	3-17

Chapter 4: Using Detector

Detector Options	4-1
JCL	4-5
Compile Time	4-5
Execution Time	4-5
Example	4-7
Detector Reports	4-7
Intercepts Report	4-9
Detector Abend Report	4-11
The Offending Instruction	4-13
Module Call Sequence	4-14
File Section Display	4-15
WORKING-STORAGE SECTION and LINKAGE SECTION Displays	4-16
Register Contents	4-17
Memory Map Display	4-18
PERFORM Return Address Analysis	4-20
Paragraph Flow Report	4-22

Open Files Report	4-22
Save Area Trace Report	4-24
IMS Report	4-26
DB2 Reports	4-32
CA-IDMS Report	4-36
Snap Report	4-42
CAIOPTS File Processing Report	4-43
Execution Monitor Summary	4-43
Detector Facilities	4-46
Merged vs. Dumped Data Displays	4-46
Merged Display	4-46
Dumped Display	4-49
Subscript Range Checking (SUBRNGCK)	4-50
Detection of Uninitialized Working-Storage (WSINIT)	4-51
Detection of Invalid Record Area References (FDCHECK)	4-51
Parameter List Validity Checking (PARMCHK)	4-51
Paragraph Flow Analysis (PFLOW)	4-52
SNAP	4-52
INTERCEPTS	4-53
Usage Considerations	4-53
Detector with IMS	4-53
Dynamic Allocation	4-53
Multimodule Reporting	4-55
Restrictions	4-55

Chapter 5: Using Extended Detector

Extended Detector Options	5-1
JCL Requirements At Execution Time	5-2
Extended Detector Reports	5-3
Hierarchy of Information	5-4
Usage Considerations	5-5
Dynamic Allocation	5-5
Restrictions	5-5
Extended Detector with Language Environment (LE)	5-6

Chapter 6: Using the CAIPRINT Repository Viewer

Starting the Viewer	6-1
The Repository Index	6-2
Line Commands	6-4
Select	6-4
Delete	6-5
Lock	6-5
Unlock	6-6
Print	6-6
Options	6-7
Adding Symbolic Information	6-8
Invoking the COBOL Postprocessor	6-9
Invoking the CSL Display Utility	6-11
Displaying Installation Options	6-12
Help	6-13

Chapter 7: Using Analyzer

Analyzer Options	7-4
Overview	7-4
Report-Time Considerations	7-5
JCL	7-6
Compile Time	7-6
Execution Time	7-7
Analyzer Reporter Job Step	7-8
Example	7-9
Analyzer Reports	7-10
CAIOPTS File Processing Report	7-10
Execution Monitor Report	7-10
Run Summary Report	7-13
Module Summary Report	7-14
Source Listing Count and Timing Report (SOURCEMAP)	7-16
Change Summary Report	7-19
Report of Unexecuted Paragraphs	7-20
Paragraph Execution-Time Summary	7-21
Execution Count Cross-Reference Report (CNTXREF)	7-22
Instruction Detail Report (INSTRMAP)	7-24

Usage Considerations	7-26
Analyzer with IMS	7-26
Analysis of Counts Results	7-26
Interpretation of Timings	7-27
For Multimodule Environments	7-27
For Multi-Run Reports	7-28
Automatic Invocation of Analyzer Reporter	7-28
Tagging a Source Statement for the Change Summary Report	7-29
Restrictions	7-30

Chapter 8: CMS, TSO, and CA-Roscoe Support Facilities

CMS Support	8-1
Option 1: OPTIMIZE	8-3
Option 9: EXECUTE OPTIMIZED PROGRAM	8-4
Option A: TRANSFER CSL TO OS	8-4
TSO Support	8-5
CA-Optimizer Primary Option Menu	8-5
Invocation Panel	8-6
Output File Information Panel	8-7
Defaulted Options Panel	8-7
Browse CA-Optimizer Output Panel	8-8
Browse CA-Optimizer Errors	8-9
Print CA-Optimizer Listing	8-9
Delete CA-Optimizer Listing Data Sets	8-10
Execute Optimized Program Panel	8-10
Invoke Management Reporting System Panel	8-11
CA-Roscoe Support	8-12
CA-Optimizer Primary Option Menu	8-12
Invocation Panel	8-13
Browse CA-Optimizer Output Panel	8-13
Browse CA-Optimizer Errors	8-14
Print CA-Optimizer Listing	8-15
Delete CA-Optimizer Listing Data Sets	8-15
Execute Optimized Program Panel	8-16
Invoke Management Reporting System Panel	8-17

Chapter 9: Utilities

Management Reporting System (MRS)	9-1
JCL	9-2
Reports	9-2
Detail Report	9-2
Summary Report	9-4
Usage Considerations	9-4
OS/VS COBOL Reporting	9-5
Statistics Report	9-5
JCL	9-5
Report	9-6
Initialization Summary	9-7
JCL	9-7
Report	9-8
Options Summary	9-9
JCL	9-9
Report	9-9
CSL Summary	9-10
JCL	9-10
Report	9-10
CAIPRINT Repository Utility	9-11
JCL	9-11
Commands	9-11
Usage Considerations	9-14

Chapter 10: Using Options

Compile-Time Overview	10-1
Coding Options	10-1
Overriding Defaults	10-2
Order of Precedence in Establishing Option Values	10-2
COBOL Postprocessor Time	10-2
Coding Options	10-2
Overriding Defaults	10-3
Execution, Report, and View Time	10-3
Coding Options	10-3
Overriding Defaults	10-4
Optimizer Options	10-4

Detector Options	10-5
Specifying Detector Compile-Time Options	10-6
Indicating Detector Execution-Time and View-Time Options	10-7
Standard Detector Execution-Time Options	10-7
Detector Formatting Options	10-7
Data Management Control Block Options	10-7
VSAM Control Block Options	10-8
Extended Detector Options	10-9
Specifying Extended Detector Compile-Time Options	10-10
Indicating Extended Detector Execution-Time and View-Time Options	10-10
Standard Extended Detector Execution-Time Options	10-10
Extended Detector Formatting Options	10-10
Data Management Control Block Options	10-11
VSAM Control Block Options	10-11
Analyzer Options	10-12
Compile-Time Options	10-12
Execution-Time Options	10-12
Report-Time Options	10-13
Guidelines for Using Report-Time Options	10-13
Relationship of PROGRAM-ID and CSECT Name	10-14
Alphabetized Option Listing	10-15

Appendix A: Data Sets and Block Sizes

Compile Step DD Statements	A-1
Execution Time DD Statement	A-2

Glossary

Index

Introduction

CA-Optimizer is a COBOL programmer productivity product for the MVS/XA, MVS/ESA, OS/390, and z/OS environments. CA-Optimizer automatically optimizes COBOL programs and assists programmers in creating, testing, debugging, fine-tuning, and maintaining COBOL programs.

CA-Optimizer consists of the following:

Optimizer	Increases the program's efficiency and greatly enhances the source listing
Detector/Extended Detector	Aids in debugging a program in the production and test environments
Analyzer	Allows programmers to fine-tune the program thoroughly and assure its quality
COBOL Postprocessor (CPP)	Creates a Condensed Source Listing (CSL) member from a COBOL compiler listing.
CAIPRINT Repository Viewer	Allows programmers to view, print, and maintain the contents of the central VSAM repository.
Utilities	Allows programmers to: ■ Provide load library reports that identify and describe COBOL CSECTs in detail ■ Generate reports detailing optimization results ■ Generate a report detailing the options set as installation defaults ■ List the contents of the CSL data sets ■ Generate a report describing which CA-Optimizer modules are currently loaded in the CSA area of LPA ■ List and maintain the contents of the central VSAM repository

The use of each of these facilities is discussed in the succeeding chapters of this guide.

To maximize programming productivity, CA-Optimizer:

- Expands programming capacity
CA-Optimizer aids in the development and maintenance of more efficient and reliable programs so that programmers can concentrate on programming.
- Reduces program development time
CA-Optimizer eliminates the guess work in programming by helping programmers create or modify programs, and by showing them exactly what their programs do (or do not do) right on the program source listing.
- Speeds program debugging
CA-Optimizer provides COBOL oriented diagnostic information that helps programmers resolve problems quickly.

To provide better programming techniques, CA-Optimizer:

- Teaches programmers better programming practices
CA-Optimizer helps programmers minimize time-consuming, nonprogramming procedures by providing a thorough debugging and testing facility enabling them to easily identify programming errors. Thus, the opportunity to concentrate on developing better techniques, such as structured and team programming, is provided.
- Improves program efficiency
CA-Optimizer checks subscript calculations, data conversion, register usage, redundant instructions, verb usage, data addresses, and PERFORM ranges.

To improve program reliability, CA-Optimizer:

- Ensures adequate testing
CA-Optimizer reports provide clear documentation of what has been tested and pinpoints each unexecuted verb to aid programmers in determining what additional test data is needed.
- Highlights hidden logic problems
CA-Optimizer reports show overall logic flow, help spot hidden performance problems and pinpoint untested logic.
- Reduces production errors and missed production schedules
With CA-Optimizer, programmers get more production runs with fewer failures, scheduling problems, and reruns. Thus, programmers can use system time, otherwise lost due to failures or reruns, more productively.

To enhance program performance, CA-Optimizer:

- Identifies performance bottlenecks

CA-Optimizer improves COBOL program performance by identifying time-consuming statements and eliminating bottlenecks before the program goes into production.

- Reduces test time

CA-Optimizer's analytical capabilities help programmers get the most from each compile and test run by allowing them to locate multiple logic and performance problems in a single compile or execution.

CA-Optimizer is compatible with OS/VS COBOL. No system or source program changes are required. CA-Optimizer is easily installed and immediately available for use.

Unsupported Features

The following are not supported by CA-Optimizer:

- The STERLING CURRENCY feature of COBOL
- The segmentation features of ANS COBOL
- The debugging features (STATE, FLOW, COUNT, SYMDMP, TEST)
- The OPT feature of the COBOL compiler
- Batch compiles
- L132 with LSTCOMP (L120 is supported)
- LSTONLY or SYNTAX
- Debug declaratives of VS 2.0 with level 74 standards
- Nonliteral subscripting or indexing to modify BLL cells (CICS only)
- The use of the DCB parameter BUFNO

Analyzer and DETECTOR are not supported under CICS or CMS. CA-InterTest supports interactive debugging of COBOL programs under CICS.

Summary of New Features for Version 7.0

- Detector and Extended Detector reports are now written to a central VSAM repository.
- An ISPF interface has been provided for viewing, printing, and maintaining the contents of the repository.
- A batch utility has been provided for listing and maintaining the contents of the repository.
- A COBOL postprocessor utility has been provided to allow CSL members to be created from COBOL compiler listings.
- COBOL postprocessor drivers have been provided to create CSL members from listings residing in CA-Endevor, CA-Librarian and CA-Panvalet libraries as well as from partitioned data sets.
- Detector and Extended Detector have been enhanced to allow symbolic support to be merged into CAIPRINT reports at view time, rather than dump time.
- Detector and Extended Detector have been enhanced to search multiple CSL libraries for symbolic information, automatically locating the correct version when multiple listings exist for a program.
- Detector and Extended Detector have been enhanced to produce additional reporting specifically for DB2, IMS, and CA-IDMS application environments.
- Detector and Extended Detector have been enhanced to provide symbolic support for items defined in a COBOL programs Data Division, allowing data item values to be displayed alongside their definitions.
- Miscellaneous enhancements have been made to Detector and Extended Detector reports, including:
 - Improved abend page
 - Previous record display for VSAM files
 - Improved register mapping for assembler programs
 - Floating-point register display
- The callable snap program, CAODSNAP, has been enhanced to support calls from any program, including non-COBOL or non-optimized programs.

Documentation Changes

The CA-Optimizer Systems Programmer Guide has been merged with *CA-Optimizer Getting Started*.

Command Notation

This guide uses the following command notation.

Enter the following exactly as they appear in command descriptions:

UPPERCASE	Identifies commands, keywords and keyword values which must be coded exactly as shown.
MIXed Cases	Identify command abbreviations. The uppercase letters are the minimum abbreviation; lowercase letters are optional.
symbols	All symbols, such as commas, equal signs and slashes, must be coded exactly as shown.

The following clarify command syntax; do not type these as they appear:

lowercase	Indicates that you must supply a substitution (a user-supplied value).
[]	Identify optional keywords or parameters.
{ }	Require choosing one of the keywords or parameters listed.
<u>underlining</u>	Shows default values that need not be specified.
	Separates alternative keywords or parameters, choose one.
...	Means the preceding items or group of items can be repeated more than once.

Sample Command	Explanation
REad field1 [field2] [A I R]	RE is the command. field1 is a required value you must supply. field2 is an optional value you may specify. A I R indicates optional parameters.
LOGon[,USERID={ <u>CA01</u> xxxxxxxx}]	LOG is the command. USERID is the optional keyword. CA01 is the default value for the USERID keyword. xxxxxxxx is an alternate value you may specify instead of CA01.

Using Optimizer

Optimizer works on the compiler-produced source listing and object code of a program. It greatly enhances the source listing while reducing the execution time. This is accomplished by optimizing the object code. Optimizer often eliminates 30 percent or more of the machine instructions. In addition, Optimizer optionally “instruments” the object code in a variety of ways. Thus, you are provided with, not only testing and quality assurance tools, but production time diagnostic facilities as well. A flowchart of the Optimizer Facility is provided for your reference after the following discussion.

Optimizer does not change the processing performed by the program or alter user-defined data areas. An optimized program does the same job, but does it faster. However, an optimized program may require additional CA-Optimizer modules at execution time, depending upon the options selected at compile/optimize time. The following is a partial list of the object code optimization techniques:

- Optimizing:
 - Register usage
 - Addressing for Data and Procedure Division references
 - Subscript and index calculations
- Eliminating:
 - Redundant instructions
 - Repetitive calculations
- Consolidating:
 - MOVEs whenever possible
 - Repeated logic with global analysis
- Removing unreferenced code
- Reducing PERFORM linkage to as little as one instruction
- Taking code out of loops
- Simplifying IF logic

The following is a partial list of the enhanced source listing report features:

- Merged Data Map (MDMAP)
- Merged condensed procedure listing (MLIST)
- Summary of unconditionally true or false code and unexecutable code
- PERFORM range summary with identification of PERFORM ranges violating structured programming techniques
- Complete Memory Map display (TGT/PGT) with one line for each possible area (for example, BL cells), even if the area is of zero length in the program

Optimizer Options

Using Optimizer together with the OS/VS COBOL compiler is similar to using the compiler alone. The compiler and Optimizer are executed in a single job step by a control routine, CAOMSMON, which links first to the compiler and then to the Optimizer component. The option defaults are set up so that your program is always optimized. Specify the NOCOPT option at compile time when you do not want optimization to take place.

Compile Time Options

The following chart is a summary of all CA-Optimizer options available to you at compile/optimize time.

To invoke Optimizer:	COPT		
Environmental Options:	CICS HOGAN	IDMSDC SHADOW	DCI TPMON
Output Producing Options:	CLIST CXREF DMAP MDMAP	MLIST PMAP RDIS SELECT	SOURCE XMDMAP XREF
Analyzer features:	XCOUNT		
Detector features:	DTECT FDCHECK	PFLOW SUBRNGCK	WSINIT
Other Options:	COBCC CSL DECK	LINECOUNT LOWER OBJECT	OPTRC STATS

You should be aware that some options, when included in your PARM statement, have the ability to turn ON or OFF CA-Optimizer components, as well as other options. Chapter 10, “Using Options” in this guide provides a comprehensive description of all CA-Optimizer options (including any other options or facilities that they affect), listed in alphabetical order. You should also familiarize yourself with the topic Order of Precedence in Establishing Option Values, which contains information concerning the sequence of option processing, and options that are either forced ON or OFF by other options that are encountered during processing. The compile time features listed in the previous chart for the Detector and Analyzer facilities of CA-Optimizer are also discussed in their respective chapters of this guide.

Commonly Used Optimizer Options

The following table illustrates usage of the most frequently specified Optimizer compile time options. These options allow you to control the Optimizer source listing and object code output.

To:	Specify:
Determine if any statements have been flagged as unexecutable or unconditional (provides the Optimizer return code)	OPTRC=rc
Produce	
■ Compiler data map	DMAP
■ Compiler data map merged with source	MDMAP
■ Relative decimal byte displacement (on the data map or Merged Data Map listing)	RDIS
■ Condensed procedure map	CLIST
■ Condensed procedure map merged with source	MLIST
■ Object code for entire program	PMAP
■ Object code for specified statements	SELECT
■ The optimized object module on SYSPUNCH	DECK
■ The object text written to SYSLIN	OBJECT
Produce a cross-reference	
■ Not condensed	XREF
■ Condensed	CXREF
Suppress	
■ Optimizer	NOCOPT
■ Source listings	NOSOURCE

JCL

To execute the Optimizer component, refer to a copy of your installation's cataloged procedures for a compile/optimize and link (or compile/optimize, link, and go).

The standard Computer Associates procedure is a compile/optimize and link and is called CAIOPT. Following is an example of how to invoke the procedure and specify Optimizer component options:

```
//STEP1 EXEC CAIOPT,  
//      COBOPT='MLIST,MDMAP'
```

where COBOPT is a symbolic option defined in the CAIOPT procedure.

Note: You can combine Optimizer options with compiler options in any order. If the procedure at your installation does not have a symbolic option for the PARM options, you can pass Optimizer options the same way you specify compiler options:

```
//STEP1 EXEC CAIOPT,  
//      PARM.COBOPT='MLIST,MDMAP'
```

Usage Considerations

Other than the features not supported, there are no other restrictions on COBOL source language use. Any valid COBOL program can be optimized. You do not have to make any changes, additions, or modifications to any COBOL program to use Optimizer.

There are times when CA-Optimizer may appear to produce incorrect paging on the output listing when a SKIP or EJECT statement has been coded. This is because CA-Optimizer does not read the source program, it reads only the compiled output listing. Therefore, it can only identify a SKIP or EJECT statement by the action taken by the compiler. Some compiler actions can be misleading to CA-Optimizer and cause incorrect paging. For example:

Failure to honor EJECT statements

An EJECT that falls on the first line of a page in the compiled listing is ignored by the compiler. Because it does not appear at all in the compiled listing, it is also ignored by CA-Optimizer.

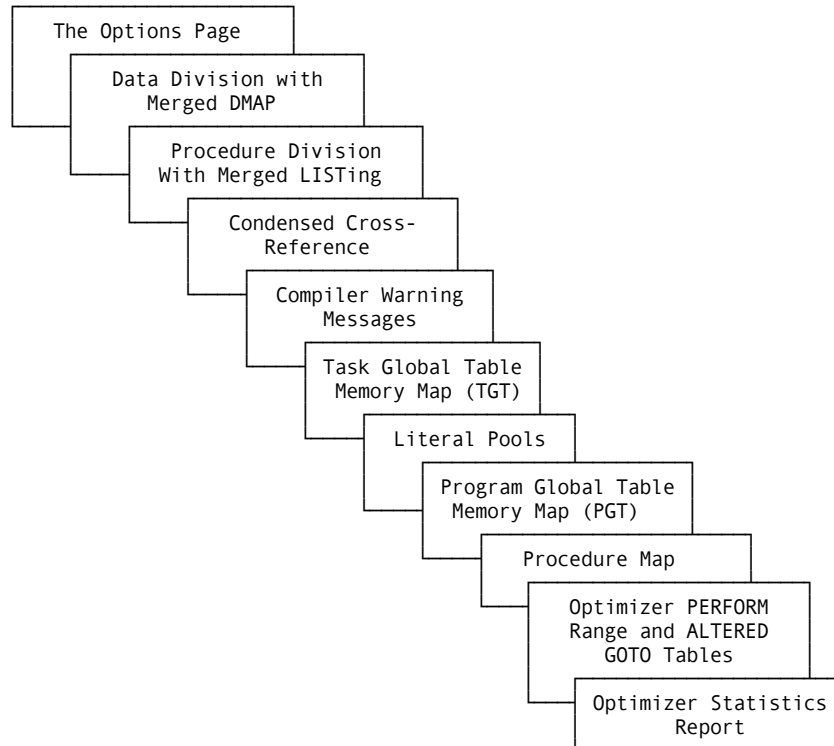
Inserting an EJECT where a SKIP was coded

A SKIP that falls at the end of a page in the compiled listing, appears exactly the same as an EJECT. Because the compiled listing has a premature page eject, CA-Optimizer must assume that an EJECT has been coded.

Optimizer Reports

Optimizer provides an optimized object module which can be submitted directly to the linkage editor. Optimizer also produces an Enhanced Source Listing with an accompanying report that reflects the optimization results.

The Enhanced Source Listing with MDMAP and MLIST contains the following reports. They are listed in the order that Optimizer generates them.



Source listing control options permit you to exclude the Merged Data Map (MDMAP) or the Merged Listing (MLIST) and to print the Compilers Data Map (DMAP), Condensed Listing (CLIST), and the Procedure Map (PMAP). You can also choose to print selected portions of the Procedure Map Object Listing by using the SELECT option.

Options Page

This is the first page of the Optimizer Enhanced Source Listing.

```

COMPUTER ASSOCIATESA
CA-Optimizer 7.0 0106A0M00
PP 5740-CB1 RELEASE 2.4B
CA-OPTIMIZER OPTIONS IN EFFECT:
NOCICSD
NOCLIST
COBCC=08
CSL
NOCSEXREF
CXREF
NODCI
NODECK
NODMAP
DTECT
NOEZTEST
FDCHECK
NOHOGAN
NOIDMSDC
LINECNT=54
MDMAP
MLIST
OPTRC=00
PFLOW
PMAP
NORDIS
NOSHADOW
SOURCE
NOSTATS
NOSUBRNGCK
NOTPHON
NOTRACE
VS
WSINIT
XCOUNT
NOXMDFMAP

```

PAGE 1
15 JUN,1901 14.29.26

IBM OS/VS COBOL JULY 1, 1982

- ^A Shows the customer name and location and identifies the CA-Optimizer version number.
- ^B Identifies the compiler version and release.
- ^C Shows the compilation date and the time of optimization.
- ^D Lists the Optimizer options in effect.

Merged Data Map (MDMAP)

The MDMAP option that you specify at compile time prints the data map information produced by the compiler beside the Data Division on the source listing.

The listing for the Merged Data Map is much shorter than the listings for the combined source and DMAP and contains the same information. The CA-Optimizer Detector feature provides reports which match these same base and displacement addresses.

The following is a partial source listing showing the MDMAP option's output.

COMPUTER ASSOCIATES		CAOMDEMO		PAGE 3	
CA-Optimizer		7.0 0106A0M00		15 JUN,1901 14.29.26	
CAOMDEMO		14.29.26		JUN 15,1901	
00088	000880	DATA DIVISION.		00088000	
00089	000890			00089000	
00090	000900	FILE SECTION.		00090000	
00091	000910	FD REPORT-OUT		00091000	DCB=01 4-208 QSAM
00092	000920	BLOCK CONTAINS 1 RECORDS		00092000	
00093	000930	LABEL RECORDS ARE STANDARD		00093000	
00094	000940	DATA RECORD IS LINE-OUT.		00094000	
00095	000950	01	LINE-OUT.	00095000	BL=1 000 4-232 0CL121
00096	000960	03	SLEW-CTL PIC X.	00096000	BL=1 000 4-253 1C
00097	000970	03	FILLER PIC X(120).	00097000	BL=1 001 4-271 120C
00098	000980			00098000	
00099	000990	WORKING-STORAGE SECTION.		00099000	
00100	001000	77	FILLER PIC X(80) VALUE 'THIS PROGRAM IS PROPRIETARY PROPERTY	00100000	BL=2 000 4-282 80C
00101	001010	-	'OF COMPUTER ASSOCIATES INTERNATIONAL, INC.'	00101000	
00102	001020	77	FILLER PIC X(80) VALUE 'COPYRIGHT (C) 1983, 1997 BY COMPUTER	00102000	BL=2 050 4-296 80C
00103	001030	-	'ASSOCIATES INTERNATIONAL, INC.'	00103000	
00104	001040	77	ERROR-FLAG PIC X VALUE '0'.	00104000	BL=2 0A0 4-310 1C
00105	001050	77	TVAR PIC S9(6) COMP-3.	00105000	BL=2 0A1 4-330 4P
00106	001060	77	ERROR-FLAG1 PIC X VALUE '0'.	00106000	BL=2 0A5 4-344 1C
00107	001070	77	ERROR-FLAG2 PIC X VALUE '0'.	00107000	BL=2 0A6 4-368 1C
00108	001080	77	ERROR-FLAG3 PIC X VALUE '0'.	00108000	BL=2 0A7 4-392 1C
00109	001090	77	MOVE-CODE PIC X.	00109000	BL=2 0A8 4-416 1C
00110	001100	77	SUM-CHK13 PIC S9(15) COMP-3.	00110000	BL=2 0A9 4-435 8P
00111	001110	77	SUM-CHK24 PIC S9(15) COMP-3.	00111000	BL=2 0B1 4-454 8P
00112	001120	77	SUM-CHK5 PIC S9(15) COMP-3.	00112000	BL=2 0B9 4-473 8P
00113	001130	77	ARY-COM PIC S9(15) COMP-3.	00113000	BL=2 0C1 4-491 8P
00114	001140	77	ARY-SUM PIC S9(15) COMP-3.	00114000	BL=2 0C9 5-000 8P
00115	001150	77	ARY-CK PIC S9(15) COMP-3.	00115000	BL=2 0D1 5-017 8P
00116	001160	77	DIVIDEND-D PIC S9(15) COMP-3.	00116000	BL=2 0D9 5-033 8P
00117	001170	77	DIVISOR-D PIC S9(03) COMP-3.	00117000	BL=2 0E1 5-053 2P
00118	001180	77	COMP-3-FIELD PIC S9(03) COMP-3.	00118000	BL=2 0E3 5-072 2P
00119	001190			00119000	
00120	001200	01	CONTROL-CARD.	00120000	BL=2 0E8 5-094 0CL24
00121	001210	03	PGM-ID PIC X(8).	00121000	BL=2 0E8 5-122 8C
00122	001220	03	FILLER PIC X(5).	00122000	BL=2 0F0 5-138 5C
00123	001230	03	DTE PIC X(11).	00123000	BL=2 0F5 5-152 11C
00124	001240			00124000	
00125	001250	01	VALUEX.	00125000	BL=2 100 5-168 0CL48
00126	001260	03	VALUE1 PIC S9(6) VALUE +2.	00126000	BL=2 100 5-187 6C
00127	001270	03	VALUE2 PIC S9(6) VALUE +4.	00127000	BL=2 106 5-203 6C
00128	001280	03	VALUE3 PIC S9(6) VALUE +3.	00128000	BL=2 10C 5-219 6C
00129	001290	03	VALUE4 PIC S9(6) VALUE +6.	00129000	BL=2 112 5-235 6C
00130	001300	03	A PIC S9(6) VALUE +1865.	00130000	BL=2 118 5-251 6C
00131	001310	03	B PIC S9(6) VALUE -1776.	00131000	BL=2 11E 5-262 6C
00132	001320	03	C PIC S9(6) VALUE +708.	00132000	BL=2 124 5-273 6C
00133	001330	03	D PIC S9(6) VALUE +150.	00133000	BL=2 12A 5-284 6C

The contents of the Merged Data Map include:

- A** The base locator numbers of the BL or BLL cells in the Task Global Table (TGT) which contain the base address of the data field.
- B** The displacement to be added to the base address for locating the data item in a dump.
- C** The compiler generated internal data name.
- D** The definition field in terms of its length in bytes.

E The data definition attribute.

If you use COPY with the SUPPRESS option, the data map information relating to the suppressed data name appears following the source listing. It is in the same location and format as the original compiler data map. If the XMDMAP option is specified, any data names defined from a COPY ... SUPPRESS statement do not appear.

There are two special cases in which the Merged Data Map facility is not able to resume processing. The first occurs after a suppressed COPY book. When the first data item following the COPY book has the same name and level number as one of the data items inside the suppressed COPY book, CA-Optimizer attempts to merge the data map information from the suppressed data item onto the nonsuppressed data item, which eventually results in an MDMAP error.

This commonly occurs when the COPY book is followed by a '01 FILLER' data item. In this case, the problem can be resolved by changing the name of the FILLER item to 'FILLERX' or some other unique name.

Note: While this limitation is circumventable, it is not correctable by PTF for two reasons:

- CA-Optimizer does not know how many data items are inside the suppressed COPY book.
- The compiler's data map report does not uniquely identify each data item by statement number.

The second condition occurs when optimizing a program with multiple data items on a line. CA-Optimizer is unable to merge all of the data map information within the data division under these circumstances. The data items that CA-Optimizer was unable to merge, along with the remaining data map information will be reported in the unmerged data map.

To circumvent this limitation, use a separate line for each data item.

Merged Listing (MLIST)

When you request an MLIST at compile time, the address of each verb prints alongside the Procedure Division statement on the source listing. An address prints for each referenced paragraph name.

COMPUTER ASSOCIATES			CAOMDEMO		PAGE	9
CA-Optimizer	7.0	0106A0M00			15 JUN,1901	14.29.26
	CAOMDEMO	14.29.26	JUN 15,1901			
00346	003460	PROCEDURE DIVISION				
00347	003470	USING PARMs UNUSED-PARM.				
00348	003480					
00349	003490	DEMO-BEGIN.				
00350	003500	GO TO DEMO-INIT.				
00351	003510					
00352	003520	DEMO-INIT.				
00353	003530	OPEN OUTPUT REPORT-OUT.				
00354	003540	MOVE HD1 TO LINE-OUT PERFORM WRITE-LINE.				
00355	003550	MOVE HD2 TO LINE-OUT PERFORM WRITE-LINE.				
00356	003560	MOVE 50 TO COUNT-TIMES.				
00357	003570	MOVE 30 TO SUB-MAX.				
00358	003580	MOVE BINARY-1 TO SUB.				
00359	003590	MOVE BINARY-1 TO TVAR.				
00360	003600					
00361	003610	DEMO-INIT-BYPASS.				
00362	003620	ALTER DEMO-BEGIN TO PROCEED TO DEMO-START-IT.				
00363	003630					
00364	003640	DEMO-START-IT.				
00365	003650	IF PARM-LENGTH = 8 AND PARM-S = 'SUBRNGCK'				
00366	003660	MOVE 'SUBRNGCK' TO PARM-SELECTED				
00367	003670	MOVE 'DTE' TO MODE-SELECTED				
00368	003680	ELSE				
00369	003690	IF PARM-LENGTH = 12 AND PARM-S-E = 'EDS,SUBRNGCK'				
00370	003700	MOVE 'SUBRNGCK' TO PARM-SELECTED				
00371	003710	MOVE 'EDS' TO MODE-SELECTED				
00372	003720	ELSE				
00373	003730	IF PARM-LENGTH = 7 AND PARM-F = 'FDCHECK'				
00374	003740	MOVE 'FDCHECK' TO PARM-SELECTED				
00375	003750	MOVE 'DTE' TO MODE-SELECTED				
00376	003760	ELSE				
00377	003770	IF PARM-LENGTH = 11 AND PARM-F-E = 'EDS,FDCHECK'				
00378	003780	MOVE 'FDCHECK' TO PARM-SELECTED				
00379	003790	MOVE 'EDS' TO MODE-SELECTED				
00380	003800	ELSE				
00381	003810	IF PARM-LENGTH = 6 AND PARM-W = 'WSINIT'				
00382	003820	MOVE 'WSINIT' TO PARM-SELECTED				
00383	003830	MOVE 'DTE' TO MODE-SELECTED				
00384	003840	ELSE				
00385	003850	IF PARM-LENGTH = 10 AND PARM-W-E = 'EDS,WSINIT'				
00386	003860	MOVE 'WSINIT' TO PARM-SELECTED				
00387	003870	MOVE 'EDS' TO MODE-SELECTED				
00388	003880	ELSE				
00389	003890	IF PARM-LENGTH = 4 AND PARM-N = 'NONE'				
00390	003900	MOVE 'NONE' TO PARM-SELECTED				
00391	003910	MOVE 'DTE' TO MODE-SELECTED				
00392	003920	ELSE				

The contents of the Merged CLIST include:

- A** The Merged Condensed Procedure Division Listing corresponds to the verb or procedure name mentioned on the source statement which identifies the hexadecimal location at which the statement begins. When Optimizer detects that a statement is unexecutable or that a statement is unconditionally true or false, it flags the statement here. An appropriate warning message is printed at the end of the Source Listing.

When Optimizer detects statements that cannot be reached for execution, or conditional statements that are unconditionally true or false, Optimizer prints UNEXEC or UNCOND on the source listing in place of the hex locations for any such verb, and removes the object code for all or part of those statements. Compound conditionals, where any one of the tests are found to be unconditional, are also flagged UNCOND. The appropriate warning messages print on the last page of the source listing along with the statement numbers of the source lines that are UNEXEC or UNCOND.

In conjunction with this report, the OPTRC option forces Optimizer to generate a particular return code if it identifies UNEXEC or UNCOND statements. Because of the way the compiler generates code for the WHEN verb, Optimizer cannot identify where its code begins. Therefore, an MLIST address of UNCOND or UNEXEC is never given for a WHEN verb. Instead, the verb prior to the WHEN shows the MLIST information for both itself and the WHEN.

- B** If more than one verb appears on a source line, the subsequent verbs are displayed to the right of the source. The MLIST of subsequent verbs is also listed on the line.

If Optimizer detects unexecutable or unconditional statements in the COBOL program, the following messages are printed at the end of the report.

```
*** WARNING ***  THE FOLLOWING STATEMENTS IN THIS PROGRAM ARE UNEXECUTABLE.
                  THEIR      16 HEX LOCATIONS PRINT AS -UNEXEC-
                  000627-000628 000631-000633 000636 000689-000690 000694-000695 000698-
                  000699 000704-000705

*** WARNING ***  THE FOLLOWING CONDITIONAL STATEMENTS ARE UNCONDITIONALLY TRUE OR FALSE.
                  THEIR      5 HEX LOCATIONS PRINT AS -UNCOND-
                  000653 000688 000693 000698 000704
```

Condensed Cross-Reference Dictionary (CXREF)

The Condensed Cross-Reference (CXREF) is the standard compiler cross-reference, with all unreferenced data items removed. Unreferenced paragraph names are not removed. Produce this report by specifying the CXREF option at compile time.

COMPUTER ASSOCIATES		CAOMDEMO										PAGE 25			
CA-Optimizer	7.0	0106AOM00											15 JUN,1901	14.29.26	
CAOMDEMO		14.29.26	JUN 15,1901												
CONDENSED CROSS REFERENCE DICTIONARY															
DATA NAMES		DEFN	REFERENCE												
REPORT-OUT		000086	000353	000492	000796										
LINE-OUT		000095	000354	000355	000398	000399	000492	000730	000733	000785	000791	000795			
			000862	000882	000891										
SLEW-CTL		000096	000492												
ERROR-FLAG		000104	000727	000783	000789	000794	000801	000808	000815	000824	000833	000842			
			000851												
TVAR		000105	000359	000446	000447										
ERROR-FLAG1		000106	000738	000742	000747										
ERROR-FLAG2		000107	000738	000743	000748										
ERROR-FLAG3		000108	000723	000736	000777	000782	000788								
MOVE-CODE		000109	000802	000809	000816	000825	000834	000843	000852	000870	000871	000872			
			000873	000874	000875	000876									
SUM-CHK13		000110	000766	000772	000774	000819	000837								
SUM-CHK24		000111	000768	000773	000775	000828	000846								
SUM-CHK5		000112	000770	000776	000855										
ARY-COM		000113	000870	000871	000872	000873	000874	000875	000876	000877					
ARY-SUM		000114	000722	000820	000829	000838	000847	000856	000860						
ARY-CK		000115	000722	000819	000828	000837	000846	000855	000861						
COMP-3-FIELD		000118	000894												
VALUE1		000126	000495	000496	000497	000498	000502	000503	000530	000541	000546				
VALUE2		000127	000495	000496	000497	000498	000502	000503	000530	000541	000546	000653			
VALUE3		000128	000495	000496	000498	000501	000503	000504	000743						
VALUE4		000129	000498	000641	000642	000643	000644	000645	000646	000647	000648	000658			
			000661	000664	000669	000672	000675	000678	000681	000687	000688	000693			
			000698	000704											
A		000130	000415	000428	000499	000500	000574	000579	000590	000595	000610	000766			
			000768	000770											
B		000131	000415	000499	000500	000574	000579	000590	000595	000610	000766	000768			
			000770												
C		000132	000415	000499	000500	000574	000579	000590	000595	000766	000768				
D		000133	000415	000428	000499	000574	000579	000590	000595	000766	000768				
SUB		000136	000358	000415	000444	000446	000448	000449	000450	000451	000452	000453			
			000456	000459	000460	000461	000570	000573	000574	000579	000581	000628			
			000631	000632	000633	000712	000715	000716	000717	000737	000741	000742			
			000743	000744	000754	000757	000758	000759	000760	000761	000762	000763			
SUB1		000137	000415	000602	000607	000610	000621	000863	000870	000871	000872	000873			
			000874	000875	000876	000878	000883	000887	000897						
SUB2		000138	000415	000603	000610	000618	000620	000867	000877	000879					
SUB3		000139	000415	000604	000610	000615	000617	000803	000810	000817	000826	000835			
			000844	000853	000878	000883									
C1		000140	000500	000574	000590	000766									
SUB-MAX		000141	000357	000444	000573	000589	000615	000618	000621	000757	000766	000768			
			000770	000817	000826	000835	000844	000853							
COUNT-TIMES		000142	000356	000481	000482										
BINARY-1		000143	000358	000359	000403	000421	000428								

Program Global Table (PGT) Memory Map

This map contains descriptions of the different areas of the Program Global Table (PGT). You should note that the PGT Memory Map is altered by CA-Optimizer. Produce this report by specifying any of these options at compile time: CSL, LIST, MLIST, CLIST, or SELECT.

A	B	C
PGT	01A28	0690
OVERFLOW CELLS	01A28	0690
VIRTUAL CELLS	01A28	0690
VIRTUAL EBCDIC NAMES	01A50	06B8
PROCEDURE NAME CELLS	01AA0	0708
GENERATED NAME CELLS	01AA0	0708
DCB ADDRESS CELLS	01AA0	0708
VNI CELLS	01AA4	070C
OPT LITERALS	01AB0	0718
SPACE LITERAL	01AB0	0718
ZERO LITERAL	01B30	0798
LITERALS	01B48	07B0
DISPLAY LITERALS	01D3A	09A2

The contents of the PGT Memory Map include:

- A** A description of each area of the PGT.
- B** Hexadecimal locations relative to the start of the program, for each area.
- C** Hexadecimal locations relative to the start of the PGT.

Program Global Table (PGT)

The Program Global Table also contains the literals used within the program. Produce this report by specifying any of these options at compile time: LIST, CLIST, or SELECT.

```

      B      C      A
LITERAL POOL (HEX)
01B48 (LIT+0)    0032001E 0008000C 0007000B 0006000A 00040000 00001C00
01B60 (LIT+24)  00000150 0C000000 00000004 0000000C 00000006 00000000
01B78 (LIT+48)  000C0001 00190099 999CF0F0 F0F0F0C8 F0F0F0F0 F0C30000
01B90 (LIT+72)  003C2C4C 5C6C7C8C 010C015C 020C014C 00050000 00000000
01BA8 (LIT+96)  000C0F40 20202020 20202020 20202020 20202060 00000012
01BC0 (LIT+120) F0212020 20202020 20202020 20202020 60000063 10000000
01BD8 (LIT+144) E2E4C2D9 D5C7C3D2 C4E3C5C4 E268E2E4 C2D9D5C7 C3D2C6C4
01BF0 (LIT+168) C3C8C5C3 D240C5C4 E268C6C4 C3C8C5C3 D2E6E2C9 D5C9E340
01C08 (LIT+192) 40C5C4E2 6BE6E2C9 D5C9E3D5 D6D5C540 404040C5 C4E26BD5
01C20 (LIT+216) D6D5C5C2 D3C9D5C4 40C3D6C4 C540F1C2 D3C9D5C4 40C3D6C4
01C38 (LIT+240) C540F2E3 E6D64040 40E3C8D9 C5C540C6 D6E4D940 40C6C9E5
01C50 (LIT+264) C54040E2 C9E74040 40E2C5E5 C5D540C5 C9C7C8E3 40C2C1C4
01C68 (LIT+288) 40E5C1D3 E4C56060 D3C7C5C2 C1C440E5 C1D3E4C5 6060E2D4
01C80 (LIT+312) D340E5C1 D3E4C540 40D5D6E3 40D6C4C4 E5C1D3E4 C54040C9
01C98 (LIT+336) E240F140 4040E5C1 D3E4C540 40C9E240 F3404040 E5C1D3E4
01CB0 (LIT+360) C54040C9 E240F540 4040E5C1 D3E4C540 40C9E240 F7404040
01CC8 (LIT+384) E5C1D3E4 C54040C9 E240F940 4040D5D5 40E8D540 D5E8E8D5
01CE0 (LIT+408) E8D5E8E8 E8D9C9C7 C8E3E6D9 D6D5C700 48140000 0040D3C9
01CF8 (LIT+432) E2E360F2 40D3C9E2 E360F340 D9C5E2E4 D3E360D3 60F140D9
01D10 (LIT+456) C5E2E4D3 E360D360 F240D9C5 E2E4D3E3 60D360F3 40D9C5E2
01D28 (LIT+480) E4D3E360 D360F440 D9C5E2E4 D3E360D3 60F5
      DISPLAY LITERALS (BCD)
01D3A (LIT+498) 'CAOMDEMO EXECUTION BEGINS.CAOMDEMO EXECUTION ENDS.'
.cm (pg 10 of SAMPLE Listing)

```

- A** Provides a listing of the entire Literal Pool. It contains the various literals (System, Program, Optimizer, and Character) used within the program.
- B** Indicates the hexadecimal program location.
- C** Provides the decimal location relative to the start of the program literal pool.

Task Global Table Memory Map (TGT)

The Task Global Table (TGT) contains program information needed by the COBOL program. It provides both fixed and variable TGT Memory Maps.

COMPUTER ASSOCIATES		CAOMDEMO		PAGE 32	
CA-Optimizer	7.0 0106A0M00	JUN 15,1901		15 JUN,1901	14.29.26
CAOMDEMO 14.29.26					
MEMORY MAP		TGT RELATIVE			
TGT	01398	0000			
SAVE AREA	01398	0000			
SWITCH	013E0	0048			
TALLY	013E4	004C			
SORT SAVE	013E8	0050			
ENTRY-SAVE	013EC	0054			
SORT CORE SIZE	013F0	0058			
RET CODE	013F4	005C			
SORT RET	013F6	005E			
WORKING CELLS	013F8	0060			
SORT FILE SIZE	01528	0190			
SORT MODE SIZE	0152C	0194			
PGT-VN TBL	01530	0198			
TGT-VN TBL	01534	019C			
RESERVED	01538	01A0			
LENGTH OF VN TBL	0153C	01A4			
LABEL RET	0153E	01A6			
RESERVED	0153F	01A7			
DBG R14SAVE	01540	01A8			
COBOL INDICATOR	01544	01AC			
A(INIT1)	01548	01B0			
DEBUG TABLE PTR	0154C	01B4			
SUBCOM PTR	01550	01B8			
SORT-MESSAGE	01554	01BC			
SYSOUT DDNAME	0155C	01C4			
RESERVED	0155D	01C5			
COBOL ID	0155E	01C6			
COMPILED POINTER	01560	01C8			
COUNT TABLE ADDRESS	01564	01CC			
RESERVED	01568	01D0			
DBG R11SAVE	01570	01D8			
COUNT CHAIN ADDRESS	01574	01DC			
PRBL1 CELL PTR	01578	01E0			
RESERVED	0157C	01E4			
TA LENGTH	01581	01E9			
RESERVED	01584	01EC			
PCS LIT PTR	0158C	01F4			
DEBUGGING	01590	01F8			
CD FOR INITIAL INPUT	01594	01FC			
OVERFLOW CELLS	01598	0200			
BL CELLS	01598	0200			
DECBADR CELLS	015B4	021C			
FIB CELLS	015B4	021C			
TEMP STORAGE	015B8	0220			
TEMP STORAGE-2	01628	0290			
TEMP STORAGE-3	01728	0390			
TEMP STORAGE-4	01728	0390			
BLL CELLS	01728	0390			
VLC CELLS	0173C	03A4			
SBL CELLS	0173C	03A4			
INDEX CELLS	0173C	03A4			

The contents of the TGT Memory Map include:

- A** A description of the field.
- B** The hexadecimal program location.
- C** The hexadecimal location relative to the start of the TGT.

PERFORM Range Table and ALTERed GO TO Table

Two tables which identify the PERFORM ranges and ALTERed GO TOs print on a separate page preceding the COBOL Diagnostic Messages and Compiler Statistics Report. If there are any PERFORM ranges or ALTERed GO TOs in the program, the PERFORM Range and ALTERed GO TO reports are automatically generated at the end of the source listing.

The PERFORM Range report is extremely useful in assessing the quality of the PERFORM ranges in the program. A glance at the summary line tells you if any of the PERFORM ranges do not conform to structured programming practices. The information related to the location of each PERFORM range in the source and object code, and the return cell associated with the PERFORM range is very useful for debugging. A description of how the PERFORM Range Table is used during debugging can be found in the Chapter “Using Detector” in this guide.

COMPUTER ASSOCIATES			CAOMDEMO			PAGE 110	
CA-Optimizer			7.0	0106A0M00		15 JUN,1901	14.29.26
CAOMDEMO			14.29.26	JUN 15,1901			
PERFORM RANGE TABLE							
LEGEND: A							
OVERLAP - THIS PERFORM RANGE OVERLAPS ANOTHER PERFORM RANGE.							
FALL THRU - THIS PERFORM RANGE EXIT MAY FALL THROUGH TO THE NEXT STATEMENT IN THE PROGRAM.							
BRANCH OUT - THERE IS A BRANCH IN THE PERFORM RANGE THAT BRANCHES OUT OF THE PERFORM RANGE TO THE ADDRESS SHOWN.							
RECURSIVE - THERE IS A PATH FROM THIS PERFORM RANGE ENTRANCE, BEFORE REACHING THE PERFORM RANGE EXIT, WHICH LEADS TO A PERFORM VERB OF THE SAME PERFORM RANGE. THE ENTRY AND EXIT ADDRESSES OF THE PERFORM RANGES CONTAINING THE RECURSIVE PERFORM VERBS ARE SHOWN.							
NO EXIT - THIS PERFORM RANGE CONTAINS NO PATH FROM THE PERFORM RANGE ENTRANCE TO THE PERFORM RANGE EXIT.							
NULL - THIS PERFORM RANGE CONTAINS NO EXECUTABLE STATEMENTS.							
B	C	D	E	F	G	H	I
SOURCE	ADDRESS	RETURN	OVER	FALL	BRANCH	RECURSIVE	NO
ENTRY - EXIT	ENTRY - EXIT	CELL	-LAP	THRU	OUT TO	CALL FROM	EXIT
000492-000492	00265A-0026B8	PSV=24					
000495-000504	0026D6-00278C	PSV=23					
000509-000516	0027AA-0027C2	PSV=22					
000530-000565	0027E0-00294A	PSV=21					
000570-000581	00296C-002A6A	PSV=20					
000602-000604	002BE8-002BF4	PSV=19	YES	YES			
000586-000622	002A90-002D4E	VN=1	YES	YES			
000626-000636	002D86-002D8A	PSV=18		YES			
000641-000648	002DB8-002ECA	PSV=17		YES			
000653-000653	002EF8-002F04	PSV=16		YES			
000658-000683	002F32-00308C	PSV=15		YES			
000687-000707	0030BA-003146	PSV=14		YES			
000712-000717	003178-0031DC	PSV=13		YES			
000722-000796	003212-003794	PSV=12		YES			
000801-000805	0037C6-0037E4	PSV=11		YES			
000808-000812	00380E-00382C	PSV=10		YES			
000815-000821	00385A-003884	PSV=9		YES			
000824-000830	0038AE-0038D8	PSV=8		YES			
000833-000839	003902-00392C	PSV=7		YES			
000842-000848	003956-003980	PSV=6		YES			
000851-000857	0039AA-0039D4	PSV=5		YES			
000860-000883	0039FE-003C74	PSV=4		YES			
000886-000894	003C9E-003D26	PSV=3		YES			
000897-000897	003D50-003D62	PSV=2		YES			
SUMMARY OF PERFORM RANGES: K			YES	YES	NO	NO	NO
ALTERED VN TABLE							
L	M	N					
STATEMENT	ADDRESS	CELL					
000350	00001DB4	VN=2					
000435	00002362	VN=3					

A Legend for items E through J described following.

B Source line numbers of the entrance and exit points for each PERFORM range in the program. The table is sorted by the line number of the exit.

C Hexadecimal addresses for each PERFORM range entrance and exit point.

D Provides, when possible, the address of the location to which control returns following the PERFORM.

To determine the actual status of PERFORM exits, look at the contents of their return cells in the TGT. With Detector, you find actual PERFORM exit settings very quickly. Find information on the significance and use of the VNs, PSVs, and R15 in Chapter 4, 'Using Detector'.

E Informs you that a PERFORM range shares some code with another range. The shared code may include the same entry or exit, be a PERFORM range within a PERFORM range, or be overlapping.

If you specify MLIST, the message ***EXIT FROM PERFORMED PARAGRAPH(S) prints on your program source listing to identify the exits for each of the PERFORM ranges in this report.

Either the characters YES or blanks appear beside the PERFORM range.

F Indicates if it is possible to fall through each PERFORM exit point.

In analyzing each PERFORM range, Optimizer determines whether it is logically possible to fall through the exit. In a structured PERFORM range, it should not be possible to fall through the exit point. However, if any of the statements within a PERFORM range can be reached by a "fall-in" to the entrance or a GO TO to any point within the range, then it is possible for control to fall through the PERFORM exit point.

While COBOL permits such fall-throughs, the program may be coded so that it happens unintentionally. It is then a hidden logic error and the cause may be in another part of the program. The exit analysis in the PERFORM Range Table helps you find and fix potential logic errors.

Note: There are a few rare conditions where an exit may be flagged as a possible fall-through when it is not possible. (For example, there may be an IF statement, not in a PERFORM range, that immediately precedes a PERFORM range entrance.) If the true condition of the IF results in a GO TO while the false condition results in NEXT SENTENCE then there is a logical path falling into the PERFORM range. In actual execution, however, the data may be such that the false condition of the IF can never be taken.

Either the characters YES or blanks appear beside the PERFORM range. Blanks indicate that a fall-through is not possible.

G Identifies PERFORM ranges that have one or more branches which branch out of the PERFORM range. In a structured PERFORM range, all branching should be to locations contained within the PERFORM range and the only branching out of the range should be at the exit. For each "branch out" branch in each PERFORM range, the branch destination is displayed as a hex address relative to the start of the program. The faulty branch can be found by locating the paragraph whose MLIST matches the hex address, and then identifying the branch in the PERFORM range that branches to that paragraph.

- H** Identifies PERFORM ranges that are called again while the PERFORM range is already active. For example, PERFORM range A calls B, which calls C, which calls A. While this can be done in COBOL, if one is careful, it is generally not considered a good programming practice. For each recursive PERFORM range, each PERFORM range containing a recursive call to this PERFORM range is listed by its entry and exit addresses.
- I** Identifies PERFORM ranges where there is no path from the entrance to the exit. Such PERFORM ranges are obviously not well structured and should be examined. A usual cause of this is a PERFORM range containing an unconditional GOBACK. Either the characters YES or blanks appear beside the PERFORM range.
- J** Identifies PERFORM ranges which contain no code. This may or may not indicate a problem. If the program is being implemented in a top-down fashion, then the code may be executable, but not complete. In this case, the null PERFORM range may be a stub not yet coded. On the other hand, if the coding is complete, this represents either a forgotten stub that needs to be coded or dead code that should be removed at the source level.
- K** Summary of PERFORM ranges. If this line says NO under each column, then all PERFORM ranges are well-structured.
- The location of each ALTERed GO TO in the program is given by its:
- L** Source line number.
- M** Hexadecimal address.
- N** Variable name (VN) cell number identifies where the ALTERed GO TO is currently pointing. This information is used in conjunction with the Detector Report. See the section “Perform Return Address Analysis” in Chapter 4.

Special Usage Note for CICS Programs

CA-Optimizer identifies all flow-of-control paths in the program. For CICS programs, this includes identifying implied branches from a CICS CALL to the GO TO DEPENDING ON clause of a HANDLE command. Because of these implied branches, there may be a number of recursive PERFORM ranges in the program that CA-Optimizer identifies. CA-Optimizer defines a recursive perform range in this way:

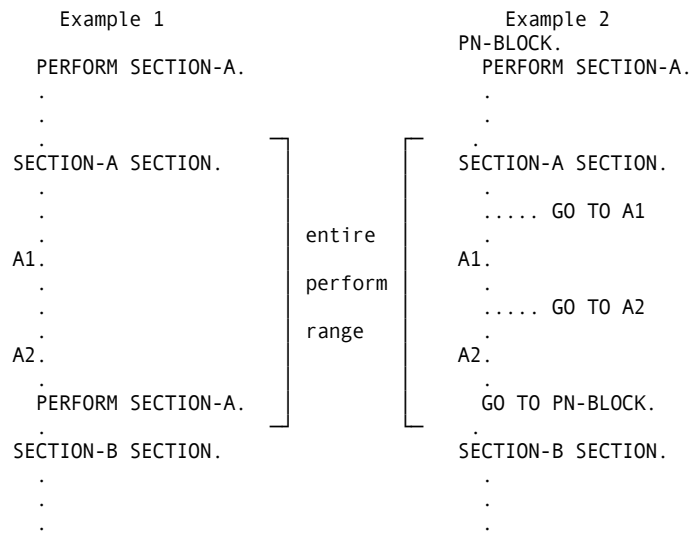
1. A PERFORM statement transfers control to a specific range of paragraphs. Call these paragraphs BLOCK A.

2. While performing BLOCK A, and before reaching BLOCK A's exit, a PERFORM statement which transfers control to that same range of paragraphs (BLOCK A) is executed.

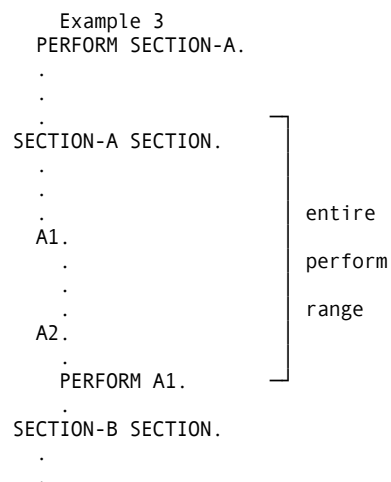
The above ensures that the entire perform range, BLOCK A, is recursive.

The following three examples show some types of recursion. Only the first two examples, however, are how CA-Optimizer defines recursion.

CA-Optimizer Defined Recursion Examples



CA-Optimizer Non-Recursion



This last example, while demonstrating simple recursion, does not conform to the CA-Optimizer recursion definition. It does not perform the entire block of the PERFORM range. CA-Optimizer identifies it as an overlapping PERFORM range.

There is a special option, HANDLIGN, for CICS programs which exhibit recursive or fall-through PERFORM ranges and which also do not have a path leading from the GO TO DEPENDING ON clause of a HANDLE command to the entrance of an active PERFORM range. The special option lets you tell the Optimizer component of CA-Optimizer to ignore the implied branches from the CICS calls in the program to the GO TO DEPENDING ON statements of HANDLE commands.

Caution! Use the HANDLIGN option with extreme care. Unpredictable execution results could occur if this special facility is used and there is a recursive path to the PERFORM call by way of a HANDLE command.

OS/VS COBOL Diagnostic Messages and Compiler Statistics Report

This report, specified at compile time, provides direct compiler output. It contains diagnostic messages regarding errors and warnings from the compilation. The messages are not altered by CA-Optimizer. However, they are folded onto eighty columns for terminal viewing.

COMPUTER ASSOCIATES	CAOMDEMO	PAGE 24
CA-Optimizer 7.0 0106A0M00		15 JUN,1901 14.29.26
CAOMDEMO 14.29.26	JUN 15,1901	
STATISTICS SOURCE	RECORDS = 897	DATA DIVISION STATEMENTS = 211
		PROCEDURE DIVISION STATEMENTS = 414
OPTIONS IN EFFECT	SIZE = 786432	BUF = 119925
	LINECNT = 51	SPACE1, FLAGW, NOSEQ, SOURCE
OPTIONS IN EFFECT	DMAP, PMAP, NOCLIST, SUPMAP, XREF, NOSXREF, LOAD, NODECK, APOST, NOTRUNC, NOFLOW	
OPTIONS IN EFFECT	NOTERM, NONUM, NOBATCH, NONAME, COMPILE=01, NOSTATE, RESIDENT, DYNAM, LIB, CSYNTAX	
OPTIONS IN EFFECT	NOOPTIMIZE, NOSYMDMP, NOTEST, VERB, ZWB, SYST, NOENDJOB, NOLVL	
OPTIONS IN EFFECT	NOLST, NOFDECK, NOCDECK, LCOL2, L120, DUMP, NOADV, NOPRINT,	
OPTIONS IN EFFECT	NOCOUNT, NOVBSUM, NOVBREF, LANGLVL(2)	

- D** If any Detector or Analyzer feature was specified for the compile/optimize step, then that feature's impact on the optimization results is described here.
- E** An estimation of the expected optimization results that might be obtained if you had not specified any Detector or Analyzer options.
- F** The first of these two lines represents the number of bytes by which you must increase your region at execution time so that Analyzer can store the count data it accumulates. This figure is different for each program. You should be concerned with the figure only if you intend to compute the exact storage requirements as outlined in *CA-Optimizer Getting Started*.

If you are using Analyzer or Detector, the second line indicates the reduction achieved when your program source listing was saved on the CSL (Condensed Source Listing Library).

Compiler Source Listing

The standard compiler source listing prints if you have not specified the Merged Data Map (MDMAP) and the Merged Listing (MLIST).

Merged Data Map (MDMAP)

Since the Merged Data Map (MDMAP) and DMAP options are mutually exclusive, you can only specify DMAP if you have not specified MDMAP. When you specify DMAP, the standard COBOL Compiler Data Map prints.

Compiler Data Map (MAP)

COMPUTER ASSOCIATES				CAOMDEMO				PAGE 20			
CA-Optimizer				7.0 0106A0M00				15 JUN,1901 15.15.28			
CAOMDEMO				15.15.28				JUN 15,1901			
INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	INTRNL NAME	DEFINITION	USAGE	R	O	Q	M
DNM=4-208	FD	REPORT-OUT	DCB=01		DNM=4-208		QSAM				F
DNM=4-232	01	LINE-OUT	BL=1	000	DNM=4-232	DS 0CL121	GROUP				
DNM=4-253	02	SLEW-CTL	BL=1	000	DNM=4-253	DS 1C	DISP				
DNM=4-271	02	FILLER	BL=1	001	DNM=4-271	DS 120C	DISP				
DNM=4-282	77	FILLER	BL=2	000	DNM=4-282	DS 80C	DISP				
DNM=4-296	77	FILLER	BL=2	050	DNM=4-296	DS 80C	DISP				
DNM=4-310	77	ERROR-FLAG	BL=2	0A0	DNM=4-310	DS 1C	DISP				
DNM=4-330	77	TVAR	BL=2	0A1	DNM=4-330	DS 4P	COMP-3				
DNM=4-344	77	ERROR-FLAG1	BL=2	0A5	DNM=4-344	DS 1C	DISP				
DNM=4-368	77	ERROR-FLAG2	BL=2	0A6	DNM=4-368	DS 1C	DISP				
DNM=4-392	77	ERROR-FLAG3	BL=2	0A7	DNM=4-392	DS 1C	DISP				
DNM=4-416	77	MOVE-CODE	BL=2	0A8	DNM=4-416	DS 1C	DISP				
DNM=4-435	77	SUM-CHK13	BL=2	0A9	DNM=4-435	DS 8P	COMP-3				
DNM=4-454	77	SUM-CHK24	BL=2	0B1	DNM=4-454	DS 8P	COMP-3				
DNM=4-473	77	SUM-CHK5	BL=2	0B9	DNM=4-473	DS 8P	COMP-3				
DNM=4-491	77	ARY-COM	BL=2	0C1	DNM=4-491	DS 8P	COMP-3				
DNM=5-000	77	ARY-SUM	BL=2	0C9	DNM=5-000	DS 8P	COMP-3				
DNM=5-017	77	ARY-CK	BL=2	0D1	DNM=5-017	DS 8P	COMP-3				
DNM=5-033	77	DIVIDEND-D	BL=2	0D9	DNM=5-033	DS 8P	COMP-3				
DNM=5-053	77	DIVISOR-D	BL=2	0E1	DNM=5-053	DS 2P	COMP-3				
DNM=5-072	77	COMP-3-FIELD	BL=2	0E3	DNM=5-072	DS 2P	COMP-3				
DNM=5-094	01	CONTROL-CARD	BL=2	0E8	DNM=5-094	DS 0CL24	GROUP				
DNM=5-122	02	PGM-ID	BL=2	0E8	DNM=5-122	DS 8C	DISP				
DNM=5-138	02	FILLER	BL=2	0F0	DNM=5-138	DS 5C	DISP				
DNM=5-152	02	DTE	BL=2	0F5	DNM=5-152	DS 11C	DISP				
DNM=5-168	01	VALUEX	BL=2	100	DNM=5-168	DS 0CL48	GROUP				
DNM=5-187	02	VALUE1	BL=2	100	DNM=5-187	DS 6C	DISP-NM				
DNM=5-203	02	VALUE2	BL=2	106	DNM=5-203	DS 6C	DISP-NM				
DNM=5-219	02	VALUE3	BL=2	10C	DNM=5-219	DS 6C	DISP-NM				
DNM=5-235	02	VALUE4	BL=2	112	DNM=5-235	DS 6C	DISP-NM				
DNM=5-251	02	A	BL=2	118	DNM=5-251	DS 6C	DISP-NM				
DNM=5-262	02	B	BL=2	11E	DNM=5-262	DS 6C	DISP-NM				
DNM=5-273	02	C	BL=2	124	DNM=5-273	DS 6C	DISP-NM				
DNM=5-284	02	D	BL=2	12A	DNM=5-284	DS 6C	DISP-NM				
DNM=5-295	01	SUBSCRIPTS	BL=2	130	DNM=5-295	DS 0CL28	GROUP				
DNM=5-318	02	SUB	BL=2	130	DNM=5-318	DS 2C	COMP				
DNM=5-334	02	SUB1	BL=2	132	DNM=5-334	DS 2C	COMP				
DNM=5-348	02	SUB2	BL=2	134	DNM=5-348	DS 2C	COMP				
DNM=5-362	02	SUB3	BL=2	136	DNM=5-362	DS 2C	COMP				
DNM=5-376	02	C1	BL=2	138	DNM=5-376	DS 2C	COMP				
DNM=5-388	02	SUB-MAX	BL=2	13A	DNM=5-388	DS 2C	COMP				
DNM=5-405	02	COUNT-TIMES	BL=2	13C	DNM=5-405	DS 2C	COMP				
DNM=5-426	02	BINARY-1	BL=2	13E	DNM=5-426	DS 2C	COMP				
DNM=5-444	02	BINARY-CNTR	BL=2	140	DNM=5-444	DS 4C	COMP				
DNM=5-465	02	BINARY-CNTR-SAV	BL=2	144	DNM=5-465	DS 4C	COMP				
DNM=5-493	02	FLAG1	BL=2	148	DNM=5-493	DS 4C	COMP				
DNM=6-000	01	LIST-2	BL=2	150	DNM=6-000	DS 0CL100	GROUP				
DNM=6-019	02	LIST-2-1	BL=2	150	DNM=6-019	DS 4P	COMP-3				
DNM=6-037	02	LIST-2-2	BL=2	154	DNM=6-037	DS 4P	COMP-3				

Procedure Map Object Listing (PMAP)

The Optimizer PMAP contains the same information as the compiler PMAP, but has additional refinements.

The object program is broken down into blocks of contiguous code. The requirement that governs the formation of these blocks is that any branch that goes to a point outside its own block must go to the beginning of the outside block. In other words, the only way to get into a block from a location outside it is by branching to its beginning.

The general registers used as base registers are assigned values, if necessary, at the beginning of a block.

The most obvious and significant refinement is that the entire source statement, not just the verb name, prints before the resulting object code. The procedure map and its enhancements are described below.

Produce the PMAP by specifying the PMAP option at compile time.

COMPUTER ASSOCIATES
CA-Optimizer

7.0 0106A0M00

CAOMDEMO

PAGE 34
15 JUN,1901 14.29.26

WORKING-STORAGE STARTS AT LOCATION 000A0 FOR A LENGTH OF 011D4.

PROCEDURE BLI VALUES

BLI 4=001BCC
BLI 5=001D6C
BLI 6=003B1E
BLI 7=003F06

* REF 1

A BLOCK NO 1 FROM HERE TO 2

BASE REGS 4=L3 10=B1 11=B2 5=B3 12=B4 8=B5 9=B6 3=B7

001D6C START EQU *

001D6C 58 40 D 398 L 4,398(0,13) BLL=3

001D70 58 A0 D 200 L 10,200(0,13) BLI=1

C 001D74 58 B0 D 204 L 11,204(0,13) BLI=2

001D78 58 50 D 208 L 5,208(0,13) BLI=3

001D7C 58 C0 D 20C L 12,20C(0,13) BLI=4

001D80 58 80 D 210 L 8,210(0,13) BLI=5

001D84 58 90 D 214 L 9,214(0,13) BLI=6

001D88 58 30 D 218 L 3,218(0,13) BLI=7

001D8C 45 E0 D 4EC BAL 14,4EC(0,13) FLOW

001D90 58 E0 D 004 L 14,004(0,13) SV1=2

001D94 58 10 E 018 L 1,018(0,14)

001D98 58 00 1 000 L 0,000(0,1)

001D9C 50 00 D 398 ST 0,398(0,13) BLL=3

001DA0 58 40 D 398 L 4,398(0,13) BLL=3

001DA4 58 00 1 004 L 0,004(0,1)

001DA8 50 00 D 39C ST 0,39C(0,13) BLL=4

* REF 2

A BLOCK NO 2 TO HERE FROM 1

FROM HERE TO 8 3 3

BASE REGS 4=L3 10=B1 11=B2 5=B3 12=B4 8=B5 9=B6 3=B7

001DAC GN=01 EQU *

001DAC 45 E0 D 4CC BAL 14,4CC(0,13) NOD=1

001DB0 0000 DC X'0000'

001DB2 0000 DC X'0000'

00347 USING PARMS UNUSED-PARM.

00349 DEMO-BEGIN. B

00350 GO TO DEMO-INIT.

001DB4 58 10 D 430 L 1,430(0,13) VN=02

A Block Cross-Reference Information and Base Register Assignments

The block cross-reference information facilitates debugging and provides additional documentation.

In this example, block number 2 has only one entrance point (block 1), and two exit possibilities (block 8 and 3).

Optimizer displays all the general registers that have assignments that must be maintained throughout a block at the beginning of that block along with their base locators.

In some cases, a base register assignment that appears at the beginning of a block is not used at all in that block. The reason for this is that when Optimizer analyzes the global logic of a COBOL program to develop the block structure described previously, it is able to set up base register assignments at the beginning of a block with foresight regarding how they shall be used in blocks further down the line.

Base registers are loaded at the beginning of the block only if these registers do not already contain the needed values.

This information is important if you are planning to patch a program and are looking for a register to use. You must not use any registers that are listed at the beginning of the block you are working in even if they are not used within that block. Optimizer requires that they remain as they have been assigned.

A shorthand notation designates the base locators as B for BL and L for BLL.

B Source Code

Each source listing statement prints before the object code for that statement.

C Object Code

The optimized object format is similar to the compiler-generated object code listing.

Note: For convenience in locating referenced symbols, PNs and GNs are always defined in ascending order on the listing. Also, PMAP suppresses SELECT.

Selected Procedure Map (SELECT)

Optimizer allows you to specify certain groups of statements for which you want a procedure map. This is very useful when you want a procedure map for only certain sections of a program.

Following is an example of a partial procedure map selected for statements 365 through 385.

COMPUTER ASSOCIATES				CAOMDEMO				PAGE 36	
CA-Optimizer				7.0 0106AOM00				15 JUN,1901 15.18.10	
CAOMDEMO				15.18.10 JUN 15,1901					
SELECTED PMAP									
BLOCK NO		8		TO HERE FROM		7		2	
				FROM HERE TO		34			
BASE REGS		9=L3 10=B1 11=B2 12=B3 7=B4 8=B5 5=B6							
00365	IF	001BF4				PN=02		EQU *	
		001BF4		48 E0 9 000				LH 14,000(0,9) DNM=11-329	
		001BF8		49 E0 D 584				CH 14,584(0,13) LIT+4	
		001BFC		47 70 7 0DE				BC 7,0DE(0,7) GN=06	
		001C00		D5 07 9 002 D 610				CLC 002(8,9),610(13) DNM=11-389 LIT+144	
00366	MOVE	001C06		47 70 7 0DE				BC 7,0DE(0,7) GN=06	
		001C0A		D2 07 B F3C D 610				MVC F3C(8,11),610(13) DNM=10-200 LIT+144	
00367	MOVE	001C10		D2 02 B FB5 D 618				MVC FB5(3,11),618(13) DNM=10-306 LIT+152	
		001C16		47 F0 7 1C8				BC 15,1C8(0,7) PN=03	
00369	IF	001C1A				GN=06		EQU *	
		001C1A		49 E0 D 586				CH 14,586(0,13) LIT+6	
		001C1E		47 70 7 100				BC 7,100(0,7) GN=07	
		001C22		D5 08 9 002 D 61A				CLC 002(12,9),61A(13) DNM=11-425 LIT+154	
		001C28		47 70 7 100				BC 7,100(0,7) GN=07	
00370	MOVE	001C2C		D2 07 B F3C D 610				MVC F3C(8,11),610(13) DNM=10-200 LIT+144	
		001C32		D2 02 B FB5 D 61A				MVC FB5(3,11),61A(13) DNM=10-306 LIT+154	
00373	IF	001C38		47 F0 7 1C8				BC 15,1C8(0,7) PN=03	
		001C3C				GN=07		EQU *	
		001C3C		49 E0 D 588				CH 14,588(0,13) LIT+8	
		001C40		47 70 7 122				BC 7,122(0,7) GN=08	
		001C44		D5 06 9 002 D 626				CLC 002(7,9),626(13) DNM=11-460 LIT+166	
00374	MOVE	001C4A		47 70 7 122				BC 7,122(0,7) GN=08	
		001C4E		D2 07 B F3C D 626				MVC F3C(8,11),626(13) DNM=10-200 LIT+166	
00375	MOVE	001C54		D2 02 B FB5 D 618				MVC FB5(3,11),618(13) DNM=10-306 LIT+152	
		001C5A		47 F0 7 1C8				BC 15,1C8(0,7) PN=03	
00377	IF	001C5E				GN=08		EQU *	
		001C5E		49 E0 D 58A				CH 14,58A(0,13) LIT+10	
		001C62		47 70 7 144				BC 7,144(0,7) GN=09	
		001C66		D5 0A 9 002 D 62E				CLC 002(11,9),62E(13) DNM=12-000 LIT+174	
		001C6C		47 70 7 144				BC 7,144(0,7) GN=09	
00378	MOVE	001C70		D2 07 B F3C D 626				MVC F3C(8,11),626(13) DNM=10-200 LIT+166	
		001C76		D2 02 B FB5 D 61A				MVC FB5(3,11),61A(13) DNM=10-306 LIT+154	
00379	MOVE	001C7C		47 F0 7 1C8				BC 15,1C8(0,7) PN=03	
		001C80				GN=09		EQU *	
		001C80		49 E0 D 58C				CH 14,58C(0,13) LIT+12	
		001C84		47 70 7 166				BC 7,166(0,7) GN=010	
		001C88		D5 05 9 002 D 639				CLC 002(6,9),639(13) DNM=12-038 LIT+185	
00382	MOVE	001C8E		47 70 7 166				BC 7,166(0,7) GN=010	
		001C92		D2 07 B F3C D 639				MVC F3C(8,11),639(13) DNM=10-200 LIT+185	
00383	MOVE	001C98		D2 02 B FB5 D 618				MVC FB5(3,11),618(13) DNM=10-306 LIT+152	
		001C9E		47 F0 7 1C8				BC 15,1C8(0,7) PN=03	
00385	IF	001CA2				GN=010		EQU *	
		001CA2		49 E0 D 58E				CH 14,58E(0,13) LIT+14	
		001CA6		47 70 7 188				BC 7,188(0,7) GN=011	
		001CAA		D5 09 9 002 D 641				CLC 002(10,9),641(13) DNM=12-071 LIT+193	

For each statement number, the entire Optimizer block that contains the number is formatted in procedure map form. Multiple blocks are formatted if required.

Note: Optimizer ignores the SELECT option if you have also specified PMAP. The CLIST and PMAP options are mutually exclusive, but if you specify the SELECT feature you can specify both CLIST and SELECT together and Optimizer integrates their outputs. The SELECTed procedure map statements override the CLIST. SELECT may also be specified with MLIST.

Using the COBOL Postprocessor

Use the CA-Optimizer COBOL postprocessor to load COBOL symbolic information into a Condensed Source Listing (CSL) library. Once loaded, the symbolic information can be used to enhance your Detector and Extended Detector reports.

COBOL Postprocessor Drivers

The COBOL postprocessor (CAOUCPP) can be invoked to load the symbolic information for a single COBOL program, or it can be driven to load an entire library at one time. A separate driver and JCL procedure is provided for each of the following library types:

Library Type	Driver Name	PROC Name
Sequential File	N/A	CAICPP
Partitioned Data Set	CAOUCPPO	CAICPPO
CA-Librarian	CAOUCPPL	CAICPPL
CA-Panvalet	CAOPCPPP	CAICPPP
CA-Endevor	CAOUCPPE	CAICPPE

JCL

This section lists the various JCL used with the COBOL postprocessor.

Sequential Listing Files

To create a CSL member from a COBOL compile listing residing in a sequential file:

```
// EXEC CAICPP,OPTLIB='CAI.CAILIB',  
// LPALIB='CAI.CAILPA',  
// LISTLIB='user.listlib',  
// CSLLIB='user.csllib',  
// CPPOPTS=  
/*
```

where:

OPTLIB	Is the CA-Optimizer CAILIB library
LPALIB	Is the CA-Optimizer CAILPA library
LISTLIB	Is the COBOL compile listing
CSLLIB	Is an existing Condensed Source Listing library
CPPOPTS	Contains the desired postprocessor options

* It is recommended that OPTLIB and LPALIB be defaulted in the PROC at installation time.

CAOUCPP uses the following DD statements:

ddname	Description
STEPLIB	Contains the CA-Optimizer executables
SYSUT1	Work data set
SYSUT2	Work data set
SYSUT5	Work data set
CAIUT1	Sequential file containing the COBOL listing
CAICSL	Condensed Source Listing (CSL) library
SYSPRINT	Output report from the postprocessor

Partitioned Listing Data Sets

To create CSL members from COBOL compile listings residing in a partitioned data set:

```
// EXEC CAICPP0,OPTLIB='CAI.CAILIB',
// LPALIB='CAI.CAILPA',
// LISTLIB='user.listlib',
// CSLLIB='user.csllib',
// CPPOPTS=CPPRC
//OPTIN DD*
FROM=member-name
TO=member-name
/*
```

where:

OPTLIB	Is the CA-Optimizer CAILIB library.
LPALIB	Is the CA-Optimizer CAILPA library.
LISTLIB	Is the COBOL compile listing.
CSLLIB	Is an existing Condensed Source Listing library.
CPPOPTS	Contains the desired postprocessor options.

* It is recommended that OPTLIB and LPALIB be defaulted in the PROC at installation time.

CAOUCPPO uses the following DD statements:

ddname	Description
STEPLIB	Contains the CA-Optimizer executables
SYSUT1	Work data set
SYSUT2	Work data set
SYSUT5	Work data set
CAIUT1	Sequential file containing the COBOL listing
CAICSL	Condensed Source Listing (CSL) library
SYSPRINT	Output from the postprocessor for each member
REPORT	Output from the postprocessor driver program
OPTIN	Used to define starting and ending member names (optional)

CA-Librarian

To create CSL members from COBOL compile listings residing in CA-Librarian:

```
// EXEC CAICPPL,OPTLIB='CAI.CAILIB',  
// LPALIB='CAI.CAILPA',  
// LISTLIB='user.listlib',  
// CSLLIB='user.csllib',  
// CPPOPTS=CPPRC  
//OPTIN DD*  
FROM=member-name  
TO=member-name  
/*
```

where:

OPTLIB	Is the CA-Optimizer CAILIB library
LPALIB	Is the CA-Optimizer CAILPA library
LISTLIB	Is the COBOL compile listing
CSLLIB	Is an existing Condensed Source Listing library
CPPOPTS	Contains the desired postprocessor options

* It is recommended that OPTLIB and LPALIB be defaulted in the PROC at installation time.

CAOUCPPL uses the following DD statements:

ddname	Description
STEPLIB	Contains the CA-Optimizer executables
SYSUT1	Work data set
SYSUT2	Work data set
SYSUT5	Work data set
CAIUT1	Sequential file containing the COBOL listing
CAICSL	Condensed Source Listing (CSL) library
SYSPRINT	Output from the postprocessor for each member
REPORT	Output from the postprocessor driver program
OPTIN	Used to define starting and ending member names (optional)

CA-Panvalet

To create CSL members from COBOL compile listings residing in CA-Panvalet:

```
// EXEC CAICPPP,OPTLIB='CAI.CAILIB',
// LPALIB='CAI.CAILPA',
// LISTLIB='user.listlib',
// CSLLIB='user.csllib',
// CPPOPTS=CPPRC
//CPPP.OPTIN DD*
FROM=member-name
TO=member-name
/*
```

where:

OPTLIB	Is the CA-Optimizer CAILIB library
LPALIB	Is the CA-Optimizer CAILPA library
LISTLIB	Is the COBOL compile listing
CSLLIB	Is an existing Condensed Source Listing library
CPPOPTS	Contains the desired postprocessor options

* It is recommended that OPTLIB and LPALIB be defaulted in the PROC at installation time.

CAOUCPPP uses the following DD statements:

ddname	Description
STEPLIB	Contains the CA-Optimizer executables
SYSUT1	Work data set
SYSUT2	Work data set
SYSUT5	Work data set
CAIUT1	Sequential file containing the COBOL listing
CAICSL	Condensed Source Listing (CSL) library
SYSPRINT	Output from the postprocessor for each member
REPORT	Output from the postprocessor driver program
OPTIN	Used to define starting and ending member names (optional)

CA-Endevor

To create CSL members from COBOL compile listings residing in CA-Endevor:

```
// EXEC CAICPPE,OPTLIB='CAI.CAILIB',  
// LPALIB='CAI.CAILPA',  
// LISTLIB='user.listlib',  
// CSLLIB='user.csllib',  
// CPPOPTS=CPPRC  
//OPTIN DD*  
FROM=member-name  
TO=member-name  
/*
```

where:

OPTLIB	Is the CA-Optimizer CAILIB library
LPALIB	Is the CA-Optimizer CAILPA library
LISTLIB	Is the COBOL compile listing
CSLLIB	Is an existing Condensed Source Listing library
CPPOPTS	Contains the desired postprocessor options

* It is recommended that OPTLIB and LPALIB be defaulted in the PROC at installation time.

CAOUCPPE uses the following DD statements:

ddname	Description
STEPLIB	Contains the CA-Optimizer executables
SYSUT1	Work data set
SYSUT2	Work data set
SYSUT5	Work data set
CAIUT1	Sequential file containing the COBOL listing
CAICSL	Condensed Source Listing (CSL) library
SYSPRINT	Output from the postprocessor for each member
REPORT	Output from the postprocessor driver program
OPTIN	Used to define starting and ending member names (optional)

The following DD statements are also made available in CAICPPE for use by CA-Endevor:

C1PRINT, NDVRMSG, BSTAPI, EN\$TRAPI, and NDVRLIST

COBOL Postprocessor Options

The following chart provides a list of all COBOL postprocessor options.

Specify	To have the COBOL postprocessor:
CLIST	Copy the Condensed Procedure Map report to the CSL. CLIST requires that the compiler's CLIST option be specified.
CPPRC	Indicate that the COBOL postprocessor should set RC=12 rather than abend if an error occurs during CSL generation.
CSXREF	Produce a Condensed Sorted Cross-Reference report. CSXREF requires that the SXREF option be specified. CSXREF is exclusive of the postprocessor's CXREF and XREF options.
CXREF	Produce a Condensed Cross-Reference report. CXREF requires that the compiler's XREF option be specified. CXREF is exclusive of the postprocessor's XREF option.
DCRDISP	Produce a relative decimal byte displacement on the merged data map, instead of a hexadecimal relative byte displacement.
DMAP	Copy the standard Data Map report to the CSL.
PMAP	Copy the Procedure Map object code listing to the CSL. PMAP requires that the compiler's PMAP option be specified.
SOURCE	Produce a source listing in the SYSPRINT data set. This listing will contain the same information as the generated CSL member.
XMMAP	Suppress the data map information which cannot be merged, such as statements within a COPY ... SUPPRESS.
XREF	Copy the standard Cross-Reference report to the CSL. XREF requires that the compiler's XREF option be specified. XREF is exclusive of CXREF, with CXREF taking precedence.

COBOL Postprocessor Reports

The output from the CA-Optimizer COBOL postprocessor utility consists of a CSL member and a listing that details the results of the postprocessing procedure. The listing will consist of the original COBOL compile listing, with the following modifications:

- The output from the compiler's DMAP option is merged into the Data Division section of the listing.
- The assembler offsets of each source statement, generated by the compiler's CLIST or PMAP options, is merged into the Procedure Division section of the listing.
- The compiler's CLIST, DMAP, PMAP, and XREF output is not written to the CSL unless specified by the user through the postprocessor options of CLIST, DMAP, PMAP, and XREF.

Options Page

This is the first page of the COBOL postprocessor output.

```
COMPUTER ASSOCIATESA                                PAGE      1
CA-Optimizer      7.0  0106ARX00                      15 JUN,1901  16.10.18C
PP 5740-CB1 RELEASE 2.4                               B      IBM OS/VS COBOL  JULY  1, 1982
CA-OPTIMIZER OPTIONS IN EFFECT:
  CLISTD
  NOCPPRC
  CSL
  NOCSXREF
  CXREF
  DMAP
  LINECNT=57
  NOLSTC
  MDMAP
  MLIST
  NOPMAP
  NORDIS
  SOURCE
  NOXMDFAP
  NOXREF
CA00144W INCREASED REGION SIZE OR UNBLOCKED SYSIN WILL IMPROVE PERFORMANCE.
```

- ^A Shows the customer name and location and identifies the CA-Optimizer version number.
- ^B Identifies the compiler version and release.
- ^C Shows the compilation date and time.
- ^D Lists the CA-Optimizer COBOL Postprocessor options in effect.

Merged Data Map

The COBOL postprocessor prints the data map information produced by the compiler beside the Data Division on the source listing. The first 88 print positions of the source statements appear similar to the way the compiler generates them. The COBOL postprocessor uses print columns 89 through 132 to add the data map information next to the associated source statements.

The listing for the merged data map is much shorter than the listings for the combined source and MAP or DMAP and contains the same information.

The following is a partial source listing showing the merged data map output.

00088	000880	DATA DIVISION.	00088000						
00089	000890		00089000						
00090	000900	FILE SECTION.	00090000						
00091	000910	FD REPORT-OUT	00091000	A	B	C	D	E	
00092	000920	BLOCK CONTAINS 1 RECORDS	00092000	DCB=01		4-208	QSAM	F	
00093	000930	LABEL RECORDS ARE STANDARD	00093000						
00094	000940	DATA RECORD IS LINE-OUT.	00094000						
00095	000950	01 LINE-OUT.	00095000	BL=1	000	4-232	0CL121		
00096	000960	03 SLEW-CTL PIC X.	00096000	BL=1	000	4-253	1C		
00097	000970	03 FILLER PIC X(120).	00097000	BL=1	001	4-271	120C		
00098	000980		00098000						
00099	000990	WORKING-STORAGE SECTION.	00099000						
00100	001000	77 FILLER PIC X(80) VALUE 'THIS PROGRAM IS PROPRIETARY PROPERTY	00100000	BL=2	000	4-282	80C		
00101	001010	'OF COMPUTER ASSOCIATES INTERNATIONAL, INC.'	00101000						
00102	001020	77 FILLER PIC X(80) VALUE 'COPYRIGHT (C) 1983, 1997 BY COMPUTER	00102000	BL=2	050	4-296	80C		
00103	001030	'ASSOCIATES INTERNATIONAL, INC.'	00103000						
00104	001040	77 ERROR-FLAG PIC X VALUE '0'.	00104000	BL=2	0A0	4-310	1C		
00105	001050	77 TVAR PIC S9(6) COMP-3.	00105000	BL=2	0A1	4-330	4P		
00106	001060	77 ERROR-FLAG1 PIC X VALUE '0'.	00106000	BL=2	0A5	4-344	1C		

The contents of the Merged Data Map include:

- A** The base locator number of the BL, BLL, or SBL cell in the TGT, which contains the base address of the data field.
- B** The displacement to be added to the base address for locating the data item in a dump.
- C** The compiler generated internal name.
- D** The definition field in terms of its length in bytes.
- E** A usage code defining:
 - D** The object of OCCURS DEPENDING ON clauses
 - E** EXTERNAL
 - F** The format of the records of a file as FIXED LENGTH
 - FB** FIXED LENGTH BLOCKED FILE

- G** GLOBAL
- O** OCCURS clauses
- OG** GROUP HAS OWN LENGTH DEFINITION
- R** REDEFINES statements
- S** SPANNED FILE
- U** UNDEFINED FORMAT FILE
- V** VARIABLE LENGTH FILE
- VB** VARIABLE LENGTH BLOCKED FILE

Note: For examples of how to use the information contained on the Data Map, see the chapter "Using Detector" in this guide. If you use COPY with the SUPPRESS option, the data map information relating to the suppressed data name will appear following the source listing. It is in the same location and format as the original compiler data map. If the XMMAP option is specified, any data names defined from a COPY ... SUPPRESS statement will not appear.

Merged CLIST

The COBOL postprocessor prints the address of each verb alongside the Procedure Division statement on the source listing. If the PMAP option was specified at compile time, an address also prints for each referenced paragraph name.

An example of a merged CLIST listing follows.

COMPUTER ASSOCIATES		CAOMDEMO		PAGE 9
CA-Optimizer	7.0 0106ARX00			15 JUN,1901 16.10.18
	CAOMDEMO 16.10.18	JUN 15,2001		
00346	003460	PROCEDURE DIVISION		00346000 000000
00347	003470	USING PARMs UNUSED-PARM.		00347000 001C44
00348	003480			00348000
00349	003490	DEMO-BEGIN.		00349000
00350	003500	GO TO DEMO-INIT.		00350000 001C60
00351	003510			00351000
00352	003520	DEMO-INIT.		00352000
00353	003530	OPEN OUTPUT REPORT-OUT.		00353000 001C66
00354	003540	MOVE HD1 TO LINE-OUT PERFORM WRITE-LINE.		00354000 001CA4
				PERFORM 001CAA
00355	003550	MOVE HD2 TO LINE-OUT PERFORM WRITE-LINE.		00355000 001CC6
				PERFORM 001CCC
00356	003560	MOVE 50 TO COUNT-TIMES.		00356000 001CE8
00357	003570	MOVE 30 TO SUB-MAX.		00357000 001CEE
00358	003580	MOVE BINARY-1 TO SUB.		00358000 001CF4
00359	003590	MOVE BINARY-1 TO TVAR.		00359000 001CFA
00360	003600			00360000
00361	003610	DEMO-INIT-BYPASS.		00361000
00362	003620	ALTER DEMO-BEGIN TO PROCEED TO DEMO-START-IT.		00362000 001D08
00363	003630			00363000
00364	003640	DEMO-START-IT.		00364000
00365	003650	IF PARM-LENGTH = 8 AND PARM-S = 'SUBRNGCK'		00365000 001D10
00366	003660	MOVE 'SUBRNGCK' TO PARM-SELECTED		00366000 001D2A
00367	003670	MOVE 'DTE' TO MODE-SELECTED		00367000 001D30
00368	003680	ELSE		00368000
00369	003690	IF PARM-LENGTH = 12 AND PARM-S-E = 'EDS,SUBRNGCK'		00369000 001D3A
00370	003700	MOVE 'SUBRNGCK' TO PARM-SELECTED		00370000 001D50
00371	003710	MOVE 'EDS' TO MODE-SELECTED		00371000 001D56
00372	003720	ELSE		00372000
00373	003730	IF PARM-LENGTH = 7 AND PARM-F = 'FDCHECK'		00373000 001D60
00374	003740	MOVE 'FDCHECK' TO PARM-SELECTED		00374000 001D76
00375	003750	MOVE 'DTE' TO MODE-SELECTED		00375000 001D7C
00376	003760	ELSE		00376000
00377	003770	IF PARM-LENGTH = 11 AND PARM-F-E = 'EDS,FDCHECK'		00377000 001D86
00378	003780	MOVE 'FDCHECK' TO PARM-SELECTED		00378000 001D9C
00379	003790	MOVE 'EDS' TO MODE-SELECTED		00379000 001DA2
00380	003800	ELSE		00380000
00381	003810	IF PARM-LENGTH = 6 AND PARM-W = 'WSINIT'		00381000 001DAC
00382	003820	MOVE 'WSINIT' TO PARM-SELECTED		00382000 001DC2
00383	003830	MOVE 'DTE' TO MODE-SELECTED		00383000 001DC8
00384	003840	ELSE		00384000
00385	003850	IF PARM-LENGTH = 10 AND PARM-W-E = 'EDS,WSINIT'		00385000 001DD2
00386	003860	MOVE 'WSINIT' TO PARM-SELECTED		00386000 001DE8
00387	003870	MOVE 'EDS' TO MODE-SELECTED		00387000 001DEE
00388	003880	ELSE		00388000
00389	003890	IF PARM-LENGTH = 4 AND PARM-N = 'NONE'		00389000 001DF8
00390	003900	MOVE 'NONE' TO PARM-SELECTED		00390000 001E0E
00391	003910	MOVE 'DTE' TO MODE-SELECTED		00391000 001E14
00392	003920	ELSE		00392000
00393	003930	IF PARM-LENGTH = 8 AND PARM-N-E = 'EDS,NONE'		00393000 001E1E
00394	003940	MOVE 'NONE' TO PARM-SELECTED		00394000 001E34
00395	003950	MOVE 'EDS' TO MODE-SELECTED.		00395000 001E3A
00396	003960			00396000
00397	003970	DEMO-PARM-MESSAGE.		00397000

The contents of the Merged CLIST include:

A The Merged Condensed Procedure Division Listing corresponds to the verb or procedure name mentioned on the source statement. This identifies the hexadecimal location at which the statement begins.

B If more than one verb appears on a source line, the subsequent verbs are displayed to the right of the source. The merged CLIST of subsequent verbs is also listed on the line.

Compiler Data Map (DMAP)

When you specify DMAP, the COBOL postprocessor outputs the standard compiler data map.

An example of a DMAP report follows.

COMPUTER ASSOCIATES			CAOMDEMO		PAGE 21	
CA-Optimizer 7.0 0106ARX00			JUN 15,2001		15 JUN,1901 16.10.18	
INTRNL NAME	LVL	SOURCE NAME	BASE	DISPL	INTRNL NAME	DEFINITION
DNM=4-208	FD	REPORT-OUT	DCB=01		DNM=4-208	
DNM=4-232	01	LINE-OUT	BL=1	000	DNM=4-232	DS 0CL121
DNM=4-253	02	SLEW-CTL	BL=1	000	DNM=4-253	DS 1C
DNM=4-271	02	FILLER	BL=1	001	DNM=4-271	DS 120C
DNM=4-282	77	FILLER	BL=2	000	DNM=4-282	DS 80C
DNM=4-296	77	FILLER	BL=2	050	DNM=4-296	DS 80C
DNM=4-310	77	ERROR-FLAG	BL=2	0A0	DNM=4-310	DS 1C
DNM=4-330	77	TVAR	BL=2	0A1	DNM=4-330	DS 4P
DNM=4-344	77	ERROR-FLAG1	BL=2	0A5	DNM=4-344	DS 1C
DNM=4-368	77	ERROR-FLAG2	BL=2	0A6	DNM=4-368	DS 1C
DNM=4-392	77	ERROR-FLAG3	BL=2	0A7	DNM=4-392	DS 1C
DNM=4-416	77	MOVE-CODE	BL=2	0A8	DNM=4-416	DS 1C
DNM=4-435	77	SUM-CHK13	BL=2	0A9	DNM=4-435	DS 8P
DNM=4-454	77	SUM-CHK24	BL=2	0B1	DNM=4-454	DS 8P
DNM=4-473	77	SUM-CHK5	BL=2	0B9	DNM=4-473	DS 8P
DNM=4-491	77	ARY-COM	BL=2	0C1	DNM=4-491	DS 8P
DNM=5-000	77	ARY-SUM	BL=2	0C9	DNM=5-000	DS 8P
DNM=5-017	77	ARY-CK	BL=2	0D1	DNM=5-017	DS 8P
DNM=5-033	77	DIVIDEND-D	BL=2	0D9	DNM=5-033	DS 8P
DNM=5-053	77	DIVISOR-D	BL=2	0E1	DNM=5-053	DS 2P
DNM=5-072	77	COMP-3-FIELD	BL=2	0E3	DNM=5-072	DS 2P
DNM=5-094	01	CONTROL-CARD	BL=2	0E8	DNM=5-094	DS 0CL24
DNM=5-122	02	PGM-ID	BL=2	0E8	DNM=5-122	DS 8C
DNM=5-138	02	FILLER	BL=2	0F0	DNM=5-138	DS 5C
DNM=5-152	02	DTE	BL=2	0F5	DNM=5-152	DS 11C
DNM=5-168	01	VALUEX	BL=2	100	DNM=5-168	DS 0CL48
DNM=5-187	02	VALUE1	BL=2	100	DNM=5-187	DS 6C
DNM=5-203	02	VALUE2	BL=2	106	DNM=5-203	DS 6C
DNM=5-219	02	VALUE3	BL=2	10C	DNM=5-219	DS 6C
DNM=5-235	02	VALUE4	BL=2	112	DNM=5-235	DS 6C
DNM=5-251	02	A	BL=2	118	DNM=5-251	DS 6C
DNM=5-262	02	B	BL=2	11E	DNM=5-262	DS 6C
DNM=5-273	02	C	BL=2	124	DNM=5-273	DS 6C
DNM=5-284	02	D	BL=2	12A	DNM=5-284	DS 6C
DNM=5-295	01	SUBSCRIPTS	BL=2	130	DNM=5-295	DS 0CL28
DNM=5-318	02	SUB	BL=2	130	DNM=5-318	DS 2C
DNM=5-334	02	SUB1	BL=2	132	DNM=5-334	DS 2C
DNM=5-348	02	SUB2	BL=2	134	DNM=5-348	DS 2C
DNM=5-362	02	SUB3	BL=2	136	DNM=5-362	DS 2C
DNM=5-376	02	C1	BL=2	138	DNM=5-376	DS 2C
DNM=5-388	02	SUB-MAX	BL=2	13A	DNM=5-388	DS 2C
DNM=5-405	02	COUNT-TIMES	BL=2	13C	DNM=5-405	DS 2C
DNM=5-426	02	BINARY-1	BL=2	13E	DNM=5-426	DS 2C
DNM=5-444	02	BINARY-CNTR	BL=2	140	DNM=5-444	DS 4C
DNM=5-465	02	BINARY-CNTR-SAV	BL=2	144	DNM=5-465	DS 4C
DNM=5-493	02	FLAG1	BL=2	148	DNM=5-493	DS 4C
DNM=6-000	01	LIST-2	BL=2	150	DNM=6-000	DS 0CL100

Compiler Condensed Procedure Map (CLIST)

When you specify CLIST and the CLIST option was specified at compile time, the COBOL postprocessor outputs the compiler's condensed procedure map listing.

An example of a CLIST report follows.

COMPUTER ASSOCIATES					SNAPPROG		PAGE 25	
CA-Optimizer		7.0	0106ARX00			12 FEB,1902	15.39.25	
SNAPPROG		15.39.25	FEB 12,1902					
CONDENSED LISTING								
347	ENTRY	001C44	350	GO	001C60	353	OPEN	001C66
354	MOVE	001CA4	354	PERFORM	001CAA	355	MOVE	001CC6
355	PERFORM	001CCC	356	MOVE	001CE8	357	MOVE	001CEE
358	MOVE	001CF4	359	MOVE	001CFA	362	ALTER	001D08
365	IF	001D10	366	MOVE	001D2A	367	MOVE	001D30
369	IF	001D3A	370	MOVE	001D50	371	MOVE	001D56
373	IF	001D60	374	MOVE	001D76	375	MOVE	001D7C
377	IF	001D86	378	MOVE	001D9C	379	MOVE	001DA2
381	IF	001DAC	382	MOVE	001DC2	383	MOVE	001DC8
385	IF	001DD2	386	MOVE	001DE8	387	MOVE	001DEE
389	IF	001DF8	390	MOVE	001E0E	391	MOVE	001E14
393	IF	001E1E	394	MOVE	001E34	395	MOVE	001E3A
398	MOVE	001E40	398	PERFORM	001E46	399	MOVE	001E5E
399	PERFORM	001E64	402	IF	001E80	403	MOVE	001E8E
404	MOVE	001E9C	405	MOVE	001EA6	406	MOVE	001EBA
407	MOVE	001EC8	409	MOVE	001EE0	410	MOVE	001EE8
411	MOVE	001EEE	412	MOVE	001EF4	414	CALL	001EFA
417	COMPUTE	001F24	423	ADD	0020C2	424	ADD	0020D4
425	ADD	0020DA	426	IF	0020EC	427	GO	0020F6
430	IF	0020FA	433	ADD	002134	434	GO	002146
437	GO	00214A	440	DISPLAY	002150	441	CALL	002164
444	ALTER	00218E	447	IF	002196	447	GO	0021A6
448	IF	0021AA	448	GO	0021BA	449	COMPUTE	0021C2
450	ADD	0021DA	451	MOVE	0021E4	452	MOVE	0021FC
453	MOVE	002218	454	MOVE	00223C	455	MOVE	002258
456	ADD	00227C	456	GO	002288	459	MOVE	00228C
462	IF	002292	462	GO	00229E	463	MOVE	0022A2
464	ADD	0022D2	464	GO	0022DE	469	MOVE	0022E2
472	PERFORM	0022EC	473	PERFORM	002304	474	PERFORM	002320
475	PERFORM	00233C	476	PERFORM	002358	477	PERFORM	002378
478	PERFORM	002398	479	PERFORM	0023B8	480	PERFORM	0023D8
481	PERFORM	0023F8	482	PERFORM	002418	483	PERFORM	002438
484	SUBTRACT	002458	485	IF	002464	486	GO	002474
489	PERFORM	002478	490	PERFORM	002494	491	DISPLAY	0024B4
492	GOBACK	0024C8	495	WRITE	0024F2	498	ADD	00253A
499	ADD	00255E	500	ADD	002582	501	SUBTRACT	00259A
502	COMPUTE	0025CA	503	COMPUTE	0025FA	504	MOVE	00262C
505	ADD	002632	506	ADD	00264A	507	MOVE	00266E
512	MOVE	00267A	519	MOVE	002710	533	IF	0027AC
533	GO	0027C2	534	GO	0027C6	536	GO	0027CA
538	GO	0027CE	540	GO	0027D2	542	GO	0027D6
544	IF	0027DA	544	GO	0027F0	545	GO	0027F4
547	GO	0027F8	549	IF	0027FC	549	GO	002812
550	GO	002816	552	GO	00281A	554	GO	00281E
556	GO	002822	558	GO	002826	560	GO	00282A
562	GO	00282E	564	GO	002832	566	GO	002836
568	GO	00283A	573	MOVE	002844	576	IF	00284A
576	GO	002856	577	COMPUTE	00285E	582	COMPUTE	00294A
584	ADD	00299E	584	GO	0029AA	589	SET	0029B4
592	IF	0029C6	592	GO	0029D8	593	COMPUTE	0029DC
598	COMPUTE	002AB4	600	SET	002AF6	600	GO	002B20
605	COMPUTE	002B24	606	COMPUTE	002B2A	607	COMPUTE	002B30

Compiler Procedure Map Object Listing (PMAP)

When you specify PMAP and the PMAP option was specified at compile time, the COBOL postprocessor outputs the compiler's procedure map object listing.

An example of a PMAP report follows.

COMPUTER ASSOCIATES				SNAPPROG				PAGE 25	
CA-Optimizer 7.0 0106ARX00								12 FEB,1902 15.35.39	
SNAPPROG 15.35.39				FEB 12,1902					
347	ENTRY	001C44		START	EQU *				
		001C44	58 40 D 004		L 4,004(0,13)				
		001C48	58 40 4 018		L 4,018(0,4)				
		001C4C	58 50 4 000		L 5,000(0,4)				
		001C50	50 50 D 350	ST	5,350(0,13)	BLL=3			
		001C54	58 90 D 350	L	9,350(0,13)	BLL=3			
		001C58	58 50 4 004	L	5,004(0,4)				
		001C5C	50 50 D 354	ST	5,354(0,13)	BLL=4			
349	*DEMO-BEGIN	001C60		PN=02	EQU *				
350	GO	001C60	58 10 D 438		L 1,438(0,13)	VN=01			
		001C64	07 F1		BCR 15,1				
352	*DEMO-INIT	001C66		PN=03	EQU *				
353	OPEN	001C66	58 10 C 084		L 1,084(0,12)	DCB=1			
		001C6A	58 40 1 024		L 4,024(0,1)				
		001C6E	D2 02 4 011 C 00D	MVC	011(3,4),00D(12)			V(ILBOEXT1)	
		001C74	50 10 D 508	ST	1,508(0,13)	SA3=1			
		001C78	92 0F D 508	MVI	508(13),X'0F'	SA3=1			
		001C7C	96 80 D 508	OI	508(13),X'80'	SA3=1			
		001C80	41 10 D 508	LA	1,508(0,13)	SA3=1			
		001C84	D2 03 D 060 C 1E4	MVC	060(4,13),1E4(12)	WC=01		LIT+140	
		001C8A	58 F0 C 010	L	15,010(0,12)	V(ILBOQI00)			
		001C8E	05 EF	BALR	14,15				
		001C90	58 10 C 084	L	1,084(0,12)	DCB=1			
		001C94	D2 03 D 060 C 16E	MVC	060(4,13),16E(12)	WC=01		LIT+22	
		001C9A	58 F0 C 010	L	15,010(0,12)	V(ILBOQI00)			
		001C9E	05 EF	BALR	14,15				
354	MOVE	001CA0	58 80 D 200	L	8,200(0,13)	BL =1			
354	PERFORM	001CA4	D2 78 8 000 6 D90	MVC	000(121,8),D90(6)	DNM=4-232		DNM=9-408	
		001CAA	D2 03 D 384 D 440	MVC	384(4,13),440(13)	PSV=1		VN=02	
		001CB0	58 80 C 390	L	11,390(0,12)	PBL=1			
		001CB4	41 00 B 07C	LA	0,07C(0,11)	GN=01			
		001CB8	50 00 D 440	ST	0,440(0,13)	VN=02			
		001CBC	47 F0 B 8AE	BC	15,8AE(0,11)	PN=019			
		001CC0		GN=01	EQU *				
		001CC0	D2 03 D 440 D 384	MVC	440(4,13),384(13)	VN=02		PSV=1	
355	MOVE	001CC6	D2 78 8 000 6 E09	MVC	000(121,8),E09(6)	DNM=4-232		DNM=9-483	
355	PERFORM	001CCC	D2 03 D 388 D 440	MVC	388(4,13),440(13)	PSV=2		VN=02	
		001CD2	58 80 C 390	L	11,390(0,12)	PBL=1			
		001CD6	41 00 B 09E	LA	0,09E(0,11)	GN=02			
		001CDA	50 00 D 440	ST	0,440(0,13)	VN=02			
		001CDE	47 F0 B 8AE	BC	15,8AE(0,11)	PN=019			
		001CE2		GN=02	EQU *				
		001CE2	D2 03 D 440 D 388	MVC	440(4,13),388(13)	VN=02		PSV=2	
356	MOVE	001CE8	D2 01 6 13C C 158	MVC	13C(2,6),158(12)	DNM=5-405		LIT+0	
357	MOVE	001CEE	D2 01 6 13A C 15A	MVC	13A(2,6),15A(12)	DNM=5-388		LIT+2	
358	MOVE	001CF4	D2 01 6 130 6 13E	MVC	130(2,6),13E(6)	DNM=5-318		DNM=5-426	
359	MOVE	001CFA	48 30 6 13E	LH	3,13E(0,6)	DNM=5-426			
		001CFE	4E 30 D 210	CVD	3,210(0,13)	TS=01			
		001D02	F8 32 6 0A1 D 215	ZAP	0A1(4,6),215(3,13)	DNM=4-330		TS=06	
361	*DEMO-INIT-BYPASS	001D08		PN=04	EQU *				
362	ALTER	001D08	58 00 C 07C		L 0,07C(0,12)	PN=05			

Condensed Cross-Reference (CXREF)

The condensed cross-reference (CXREF) is the standard compiler cross-reference, with all unreferenced data items removed. Unreferenced paragraph names are not removed. The COBOL postprocessor produces this report when the CXREF option is specified. CXREF requires that the compiler's XREF option was specified. CXREF is exclusive of the postprocessor's XREF option, with CXREF taking precedence.

An example of a CXREF report follows.

COMPUTER ASSOCIATES		CAOMDEMO								PAGE 34		
CA-Optimizer		7.0	0106ARX00	15 JUN,1901 16.10.18								
CAOMDEMO		16.10.18		JUN 15,2001								
CONDENSED CROSS REFERENCE DICTIONARY												
DATA NAMES		DEFN	REFERENCE									
REPORT-OUT		000086	000353	000492	000796							
LINE-OUT		000095	000354	000355	000398	000399	000492	000730	000733	000785	000791	000795
			000862	000882	000891							
SLEW-CTL		000096	000492									
ERROR-FLAG		000104	000727	000783	000789	000794	000801	000808	000815	000824	000833	000842
			000851									
TVAR		000105	000359	000446	000447							
ERROR-FLAG1		000106	000738	000742	000747							
ERROR-FLAG2		000107	000738	000743	000748							
ERROR-FLAG3		000108	000723	000736	000777	000782	000788					
MOVE-CODE		000109	000802	000809	000816	000825	000834	000843	000852	000870	000871	000872
			000873	000874	000875	000876						
SUM-CHK13		000110	000766	000772	000774	000819	000837					
SUM-CHK24		000111	000768	000773	000775	000828	000846					
SUM-CHK5		000112	000770	000776	000855							
ARY-COM		000113	000870	000871	000872	000873	000874	000875	000876	000877		
ARY-SUM		000114	000722	000820	000829	000838	000847	000856	000860			
ARY-CK		000115	000722	000819	000828	000837	000846	000855	000861			
COMP-3-FIELD		000118	000894									
VALUE1		000126	000495	000496	000497	000498	000502	000503	000530	000541	000546	
VALUE2		000127	000495	000496	000497	000498	000502	000503	000530	000541	000546	000653
VALUE3		000128	000495	000496	000498	000501	000503	000504	000743			
VALUE4		000129	000498	000641	000642	000643	000644	000645	000646	000647	000648	000658
			000661	000664	000669	000672	000675	000678	000681	000687	000688	000693
			000698	000704								
A		000130	000415	000428	000499	000500	000574	000579	000590	000595	000610	000766
			000768	000770								
B		000131	000415	000499	000500	000574	000579	000590	000595	000610	000766	000768
			000770									
C		000132	000415	000499	000500	000574	000579	000590	000595	000766	000768	
D		000133	000415	000428	000499	000574	000579	000590	000595	000766	000768	
SUB		000136	000358	000415	000444	000446	000448	000449	000450	000451	000452	000453
			000456	000459	000460	000461	000570	000573	000574	000579	000581	000628
			000631	000632	000633	000712	000715	000716	000717	000737	000741	000742
			000743	000744	000754	000757	000758	000759	000760	000761	000762	000763
SUB1		000137	000415	000602	000607	000610	000621	000863	000870	000871	000872	000873
			000874	000875	000876	000878	000883	000887	000897			
SUB2		000138	000415	000603	000610	000618	000620	000867	000877	000879		
SUB3		000139	000415	000604	000610	000615	000617	000803	000810	000817	000826	000835
			000844	000853	000878	000883						
C1		000140	000500	000574	000590	000766						
SUB-MAX		000141	000357	000444	000573	000589	000615	000618	000621	000757	000766	000768
			000770	000817	000826	000835	000844	000853				
COUNT-TIMES		000142	000356	000481	000482							
BINARY-1		000143	000358	000359	000403	000421	000428					
BINARY-CNTR		000144	000406	000411								
BINARY-CNTR-SAV		000145	000415	000431								
FLAG1		000146	000423	000426								

Compiler Cross-Reference (XREF)

When you specify XREF and the XREF option was specified at compile time, the COBOL postprocessor outputs the standard compiler cross-reference. XREF is exclusive of the CXREF option, with CXREF taking precedence.

Examples of the data name and procedure name XREF report follows.

COMPUTER ASSOCIATES		CAOMDEMO		PAGE 73	
CA-Optimizer	7.0	0106ARX00		15 JUN,1901	17.00.40
CAOMDEMO		17.00.40	JUN 15,2001		
CROSS-REFERENCE DICTIONARY					
DATA NAMES	DEFN	REFERENCE			
REPORT-OUT	000086	000353	000492	000796	
LINE-OUT	000095	000354	000355	000398	000399 000492 000730 000733 000785 000791 000795
		000862	000882	000891	
SLEW-CTL	000096	000492			
ERROR-FLAG	000104	000727	000783	000789	000794 000801 000808 000815 000824 000833 000842
		000851			
TVAR	000105	000359	000446	000447	
ERROR-FLAG1	000106	000738	000742	000747	
ERROR-FLAG2	000107	000738	000743	000748	
ERROR-FLAG3	000108	000723	000736	000777	000782 000788
MOVE-CODE	000109	000802	000809	000816	000825 000834 000843 000852 000870 000871 000872
		000873	000874	000875	000876
SUM-CHK13	000110	000766	000772	000774	000819 000837
SUM-CHK24	000111	000768	000773	000775	000828 000846
SUM-CHK5	000112	000770	000776	000855	
ARY-COM	000113	000870	000871	000872	000873 000874 000875 000876 000877
ARY-SUM	000114	000722	000820	000829	000838 000847 000856 000860
ARY-CK	000115	000722	000819	000828	000837 000846 000855 000861
DIVIDEND-D	000116				
DIVISOR-D	000117				
COMP-3-FIELD	000118	000894			
CONTROL-CARD	000120				
PGM-ID	000121				
DTE	000123				
VALUEX	000125				
VALUE1	000126	000495	000496	000497	000498 000502 000503 000530 000541 000546
VALUE2	000127	000495	000496	000497	000498 000502 000503 000530 000541 000546 000653
VALUE3	000128	000495	000496	000498	000501 000503 000504 000743
VALUE4	000129	000498	000641	000642	000643 000644 000645 000646 000647 000648 000658
		000661	000664	000669	000672 000675 000678 000681 000687 000688 000693
		000698	000704		
A	000130	000415	000428	000499	000500 000574 000579 000590 000595 000610 000766
		000768	000770		
B	000131	000415	000499	000500	000574 000579 000590 000595 000610 000766 000768
		000770			
C	000132	000415	000499	000500	000574 000579 000590 000595 000766 000768
D	000133	000415	000428	000499	000574 000579 000590 000595 000766 000768
SUBSCRIPTS	000135				
SUB	000136	000358	000415	000444	000446 000448 000449 000450 000451 000452 000453
		000456	000459	000460	000461 000570 000573 000574 000579 000581 000628
		000631	000632	000633	000712 000715 000716 000717 000737 000741 000742
		000743	000744	000754	000757 000758 000759 000760 000761 000762 000763
SUB1	000137	000415	000602	000607	000610 000621 000863 000870 000871 000872 000873
COMPUTER ASSOCIATES		CAOMDEMO		PAGE 77	
CA-Optimizer	7.0	0106ARX00		15 JUN,1901	17.00.40
CAOMDEMO		17.00.40	JUN 15,2001		
PROCEDURE NAMES	DEFN	REFERENCE			
DEMO-BEGIN	000349	000362			
DEMO-INIT	000352	000350			
DEMO-INIT-BYPASS	000361				

Using Detector

The Detector component of CA-Optimizer operates in both production and test environments as a productivity tool for COBOL programmers. It provides you not only with valuable diagnostic information, but also identifies problems that may otherwise remain hidden until the application goes into production. In production, two levels of diagnostic aids assist you in solving even the most difficult problems. The first level of diagnostic aids requires no changes to production JCL.

Detector generates three different reports:

- The Detector Abend Report eliminates much of the searching and guess work involved in debugging programs. The Detector Abend Report presents diagnostic information and key storage data information in a straightforward format.
- The SNAP Report provides snapshots of your program's data areas and other pertinent program information. To invoke the SNAP Report, place a CALL at the spot in your program where you wish the snapshot to be taken. You can specify multiple SNAPS in a single program execution.
- The Intercepts Report allows your program to continue executing after data or decimal divide exceptions occur. Each exception generates a short report for use in resolving it.

Detector Options

Detector is a family of testing and debugging facilities. Detector monitors application abends. When an abend occurs, Detector performs its reporting functions. Under Detector, there are a variety of tools you can use with your application.

The following chart provides a listing of all Detector options available to you at compile and execution Time.

Detector Options		
Compile-Time	Execution-time	View-Time
DTECT FDCHECK PFLOW SUBRNGCK WSINIT	Standard Options ACTONLY PARMCHK CEEDUMP PFLOW CSLDSN PGM DB2ACTIV PRTLIB DUMP PRTREPT INTERCEPTS SNAP MSTAE Formatting Options ASMINST MERGEDB BINFRMT MRGDATA FILES MRGFILES GRPADDR MRGLINK FIRST128 MRGLOCAL LINECOUNT MRGWORK LINKAGE OCCURS LOCALSTOR SAVEAREA MEMMAP WORKSTOR Data Mgmt Control Block BLOCKS IOB DCB UCB DEB VSAM Control Block ACB PLH AMB RPL AMBL VSAMIDX	Formatting Options ASMINST BINFRMT FILES GRPADDR FIRST128 LINECOUNT LINKAGE LOCALSTOR MEMMAP MERGEDB MRGFILES MRGLINK MRGLOCAL MRGWORK OCCURS SAVEAREA WORKSTOR

Detector has facilities to do the following:

- Enable and disable the XCOUNT, PFLOW, and SUBRNGCK options
- Detect invalid record area references
- Correct and intercept 0C7 and 0CB abends
- Check PARM list validity
- Abend or SNAP paragraph flow analysis of the last 'n' paragraphs
- Generate reports at specified points in the program
- Report customization
- Subscript range checking
- Uninitialized WORKING-STORAGE detection

Specify the SUBRNGCK option during compile time to use the subscript range checking facility. This option ensures that subscript and index values are within their specified range.

The facility that detects invalid record area references is controlled by the FDCHECK option. Specify FDCHECK at compile time to turn on this facility.

Specify INTERCEPTS at execution time to have the 0C7 and 0CBabend interception and correction facility in effect. This facility reports on any 0C7 and 0CB ABENDs, temporarily corrects the problem, and allows execution to continue. This lets programmers catch multiple problems with a single execution.

The PARM list validity checking facility can be used at either compile or execution time by specifying the PARMCHK option in the JCL. Using this facility, you are able to verify that the number of parameters passed to the application program is correct.

The Abend or SNAP paragraph flow analysis facility can be used at either compile or execution time. It checks the last 'n' paragraphs of the program. Specifying PFLOW turns on this facility.

Using SNAP at execution time allows you to analyze specific points in a program.

Reports are customized by specifying a variety of options at execution or view time. FILES, FIRST128, LINECOUNT, LINKAGE, MEMMAP, SAVEAREA, and WORKSTOR plus all the merging, VSAM control block, and data management control block options control report customization. Use WSINIT at compile time to detect uninitialized working storage.

You may freely combine Detector compile time options with OS/VS COBOL compiler options and other CA-Optimizer options. The preceding options have both compile time and execution time uses. The execution time applications are discussed later in this chapter. The chapter "Usage Options" in this guide contains a complete listing of all CA-Optimizer options, their defaults, usage, and special considerations.

The following chart summarizes how you can use Detector's more commonly used facilities.

At Compile Time specify	At Execution Time specify	To have Detector
SUBRNGCK		Report on subscript or index range violations.
WSINIT		Identify 0C7s and 0CBs that may have been caused by an uninitialized working storage item.
FDCHECK		Identify references to record areas of unopened files or at end-of-file.
	PARMCHK ON	Validity-check each option list passed to this optimized program.
A Call to CAODSNAP at each point where you want a snapshot	SNAP nnnnn	Collect snapshots of your program at specific points during execution. (Where nnnnn is the maximum number of snaps.)
DTECT or any other Detector option	INTERCEPTS nnnnn	Intercept data exceptions and decimal divide exceptions. (Where nnnnn is the maximum number of intercepts.)
PFLOW	PFLOW ON	List the names of the most recently executed paragraphs atabend or SNAP.
Any Detector option	MEMMAP OFF	Suppress: The display by Detector of the Memory Map (TGT and PGT).
	PFLOW OFF	Suppress: The display by Detector of the last 64 paragraphs executed even if PFLOW was specified at compile time.

JCL

Compile Time

To invoke Detector, you must first specify one or more Detector options in the compile/optimize step. To execute the Optimizer component, refer to a copy of your installation's cataloged procedures for a compile/optimize and link (or compile/optimize, link, and go).

We have provided a standard procedure called CAIOPT. The following example shows how to invoke this procedure for subscript range checking and detection of invalid record area references.

```
//STEP1 EXEC CAIOPT,COBOPT='SUBRNGCK,FDCHECK'
```

In the previous example, COBOPT is a symbolic option defined in the CAIOPT procedure. If the procedure at your data center does not have a symbolic option for the PARM options, you can pass Detector options in the same way you specify compiler options.

```
//STEP1 EXEC CAIOPT (or whatever your procedure name is)
// PARM.COB='SUBRNGCK,FDCHECK'
```

Execution Time

The standard Computer Associates procedure provides for compile and link only. If you are using this procedure, you have to execute your program directly. Use the following JCL:

```
//STEP2 EXEC PGM=your-program-name
//STEPLIB DD DSN=your-OPT-loadlib-name,DISP=SHR
//(concatenate the STEPLIB for your program's loadlib name to the
  previous STEPLIB.)
//Your program's DD statements.
//CAICSL DD DSN=your CSL library name,DISP=SHR
//CAIPRTL DD DSN=your repository library,DISP=SHR
//CAIPRINT DD SYSOUT=A
//CAIOPTS DD * (if desired)
/*
//SYSUDUMP DD SYSOUT=A
```

JCL Statement	Description
//STEPLIB DD	Points to the CA-Optimizer load library. This is not necessary if CA-Optimizer is in your concatenated LINKLIB.
//CAICSL DD	Points to the Condensed Source Listing Library. Detector's use of the CSL is optional. If you want Detector to use the CSL, you must do one of the following: ■ Code a CAICSL DD statement. ■ Specify the data set name in the CSLDSN option. ■ Let Detector use one of the default CSLs established by your systems programmer.
//CAIPRTL B DD	Points to the CAIPRINT Repository Library. Detector's use of the repository is based on the PRTREPT option. If you want Detector to use the repository, you must do one of the following: ■ Code a CAIPRTL B DD statement. ■ Code the repository library user exit, CAOCUPRT. ■ Specify the data set name in the PRTL B option. ■ Let Detector use the default repository library established by your systems programmer.
//CAIPRINT DD	For retaining Detector printed output. Detector's use of this DD statement is based upon the PRTREPT option. If you want Detector to use the CAIPRINT DD, but it is omitted from the JCL, it will be dynamically allocated. User exit CAOCUDSN can be used to control the destination of the dynamically allocated data set.
//CAIOPTS DD	For specifying Detector (and Analyzer) options. If no options need to be specified, this DD statement can be omitted.
//SYSUDUMP DD	The operating system produces a region dump. A SYSUDUMP DD statement is not required unless DUMP ON is the default or is specified in CAIOPTS. If it is required and you did not supply one, then it is dynamically allocated by Detector.

For more information on dynamic allocation, see Usage Considerations.

Example

Here is an example of the JCL required to use Detector with Paragraph Flow Analysis and produce the MDMAP from the optimization step.

```
//STEP1 EXEC CAIOPT,
//      COBOPT='PFLOW,MDMAP'
//STEP2 EXEC PGM=FICAEXTR,REGION=512K
//STEPLIB DD DSN=PAYROLL.LOADLIB,DISP=SHR
//        DD DSN=CAI.CAILIB,DISP=SHR          *
//        DD DSN=CAI.CAILPA,DISP=SHR          *
//FILEIN  DD DSN=PAYROLL.MASTER(0),DISP=OLD
//FILEOUT DD DSN=PAYROLL.MASTER(1),DISP=(NEW,CATLG),
//        UNIT=TAPE
//REPORT  DD SYSOUT=A
.
.      End of DD statements used by your application
.
//CAICSL DD DSN='your CSL library name',DISP=SHR      **
```

* This is only needed when the PDS is not part of LINKLIB.

** This is only needed if you want to use the CSL facility with Detector and your CSL is not in the installation's default CSL list.

Detector Reports

Five reports are described in this chapter. Two of these reports, the CAIOPTS File Processing Report and the Execution Monitor Summary, are produced by the Execution Monitor. The Execution Monitor is the component of the CA-Optimizer Runtime (RTL) that establishes the environment for Detector and Analyzer. The CAIOPTS File Processing Report displays the input options you specified, if any, to control Detector and Analyzer execution. The Execution Monitor Report displays the options in effect during the execution of your application under Detector or Analyzer. The other three reports, Abend, Snap and Intercepts are Detector reports; they are described following.

1. Detector Intercepts Report

Detector generates this report when a data exception (0C7) or decimal divide exception (OCB) occurs during the execution of your program. When you specify the INTERCEPTS feature at execution time, the 0C7 and OCB interrupts are handled by Detector.

Detector prints a short report describing the exception and its cause. Then Detector adjusts the data so that the program can continue executing.

2. Detector Abend Report

Detector prints this report whenever an abend occurs during program execution. This includes abends that were due to a subscript range violation (SUBRNGCK) and detection of invalid record area reference (FDCHECK). The report contains:

- a. Diagnostic information based on the type of abend.
- b. The actual optimized COBOL source statement on which the abend occurred which only happens when the Condensed Source Listing Library (CSL) is available. When the CSL is not available during execution time, the machine instruction is shown along with a short description of the abend.
- c. Addresses and other specifics pertinent to the abend itself.
- d. The module call sequence showing the hierarchy of programs which invoked the abending program.
- e. The contents of the abending program's registers at abend.
- f. An automatic interpretation and optional display of the following data management control blocks: DCB, DEB, UCB, IOB, ACB, AMBL, AMB, RPL, and PLH.
- g. Displays of the current record areas of all open files (optionally displays only the first 128 bytes), WORKING-STORAGE SECTION, and LINKAGE SECTION, if there is one.
- h. An optional display of the Memory Map (TGT, PGT).
- i. The contents of the PSV and VN cells and register 15, which are referred to in the PERFORM Range Table printed at compile time.
- j. A Paragraph Flow Report showing the last paragraphs executed up to the time of the abend (optional).
- k. The data control blocks of all open files that Detector has not yet reported on (optional).
- l. Save Area Trace Report showing the standard chain of register areas (optional).

3. **Detector Snap Report**

A Snap Report provides you with a snapshot of your optimized program at strategic points during execution.

To produce a Snap Report, you must:

- a. Add a CALL 'CAODSNAP' statement to your program at the point of execution that you want the SNAP to take place.
- b. Specify the SNAP option, 'SNAP nnnn', at execution time, where nnnn refers to the number of SNAP calls to be taken for all programs in the application.

The Snap Report contains the same information as the Abend Report except the Snap omits the data identifying the abend and the called module sequence for multimodule environments.

Intercepts Report

The output for the INTERCEPTS feature is brief, yet it contains everything you need to know to correct a data exception or decimal divide exception.

```
*****
* INTERCEPTS REPORT *
*****
INTERCEPTS PERMITTED = 2A
*****

INTERCEPTED COBOL STATEMENT      B PROGRAM-ID CAOMDEMO
SOURCE LOCATION: PAGE 10 LINE 00403 MLIST 002052C

MOVE MASK1 TO BINARY-1D

ITEM: MASK1E                      BL 002 DISP 000D80
VALUE: 5056F                      <-- NOT INITIALIZED - FOUND BY WSINIT

INTERCEPTED INSTRUCTION

4F00 D258          CONVERT

OPERAND: REG 0
VALUE: 00000000

OPERAND: TEMP STORAGE CELLG      TGT DISP 000258
VALUE: 00000000000000005H      I<-- NOT NUMERIC (SET TO ONE)

RELATIVE INTERRUPT 00002058J    INTERCEPT COUNTK1 CODE 0C7L
```

^A When using the INTERCEPTS feature, you must specify the maximum number of exceptions you wish Detector to intercept. This value must not be greater than 32,767. When the number of exceptions exceeds the specified number, the program is allowed to abend and a Detector Abend Report is produced. In this example, two intercepts are allowed prior to termination.

Note: In a multimodule environment, more than one program can use the INTERCEPTS feature concurrently. When more than one program is using the INTERCEPTS feature, the maximum number of exceptions refers to the maximum number of exceptions of all modules together rather than the maximum number of each individual module.

^B Identifies the program from which the exception originated.

^C The line number and the MLIST address are displayed with each COBOL statement. Use the line number and the MLIST address to locate the statement in your program.

- D** The COBOL statements from your program that caused the exception.
- E** The COBOL variable names which are involved with the exception. They correspond to the variable names in the displayed statement and your program listing.
- The Base Locator number and the displacement from the Base Locator is given and can be used to locate the variable names in your program.
- F** The machine instruction that caused the abend. If the ASMINST option is set to OFF, this output is suppressed.
- G** The instruction operands for the abending instruction. If possible, the operands are related back to an area in the COBOL program, such as a LITERAL or TEMP STORAGE CELL.
- If the ASMINST option is set to OFF, the instruction operands are suppressed.
- H** The hexadecimal representation of the actual storage value associated with the data item.
- I** A message area used to pinpoint the cause of the exception and indicate any action taken by Detector. If WSINIT was specified at compile time and the cause was due to uninitialized WORKING-STORAGE, Detector indicates this here.
- J** The location of the program exception relative to the beginning of the COBOL program.
- K** The interrupt count number that indicates the number of intercepts that have been reported.
- L** The interrupt code (0C7 or 0CB).
- After Detector generates the information for the Intercepts Report, it corrects the offending data item and reexecutes the instruction.
- In the case of a data exception (0C7), it sets the value of the invalid item to 1.
 - When a decimal-divide exception (0CB) occurs, Detector sets the divisor to 1.
- Detector corrects the bad data to prevent multiple intercepts for the same item. After correcting the data, Detector reexecutes the abending instruction so that predictions can be made concerning the outcome.

Detector Abend Report

The Detector Abend Report, which prints when an abend occurs, begins with some specific information pertaining to the abend itself.

```

COMPUTER ASSOCIATES          DETECTOR REPORT
CA-OPTIMIZER      7.0 0106ARX00
                                           PAGE 12
                                           15 JUN 2001 14.29.52

JOB: USRID02A  STEP: RUN      PROGRAM CAOMDEMO
*****
*              * S-00C7 *
*  A  A B E N D  *-----*
*              * CAOMDEMO *
*****

*****
*  DESCRIPTION:  S0C7 - A NUMERIC FIELD CONTAINED NON-NUMERIC DATA.  *
*  *
*  PROBABLE CAUSES:  1. NUMERIC DATA WAS NOT INITIALIZED.  *
*  B                2. A SUBSCRIPT OR INDEX CONTAINED AN INVALID VALUE.  *
*                  3. A COMP-3 FIELD HAD AN INVALID SIGN.  *
*                  4. A GROUP MOVE OVERLAYED A NUMERIC FIELD WITH NON-NUMERIC DATA.  *
*****

ABENDING COBOL STATEMENT          PROGRAM-ID CAOMDEMO
C SOURCE LOCATION: PAGE 23 LINE 00894 MLIST 003D20

      ADD 1 TO COMP-3-FIELD.

      ITEM: COMP-3-FIELDD          BL 002 DISP 0000E3
      VALUE: 5050                  <-- NOT INITIALIZED - FOUND BY WSINIT

ABENDING INSTRUCTION
E
FA10 B0E3 D7C6  ADD

OPERAND: COMP-3-FIELDF          BL 002 DISP 0000E3
      VALUE: 5050                G <-- NOT NUMERIC

OPERAND:                      CAOMDEM DISP 001B5E
      VALUE: 1C

RELATIVE INTERRUPT 00003D20H ABSOLUTE INTERRUPT 0000BD18I
RELATIVE ENTRY POINT 00001D6CJ NUMBER OF TIMES ENTERED 1K

```

- A** The system or user abend code along with the abending module name.
- B** Diagnostic information for the specified abend code. Lists the probable causes of the abend and suggests strategies to solve the problem.
- If the abend code indicates a DB2 abend (for example, S04E or S04F), the DB2 reason code is displayed with the appropriate description.
- If the abend occurs in a DB2 application, the DB2 return code, SQLCODE, is displayed with the associated messages.
- C** The abending COBOL source statement including the actual page and line number from the source listing of the COBOL compilation. This displays if the CSL library is available. If the CSL is not available, only the assembler information is displayed.
- D** The COBOL variable names which are involved with the exception. They correspond to the variable names in the displayed statement and your program listing.
- The Base Locator number and the displacement from the Base Locator is given and can be used to locate the variable names in your program.
- E** The machine instruction that caused the abend. If the ASMINST option is set to OFF, this output is suppressed.
- F** The instruction operands for the abending instruction. If possible, the operands are related back to an area in the COBOL program such as a LITERAL or TEMP STORAGE CELL.
- If the ASMINST option is set to OFF, the instruction operands are suppressed.
- G** The specific cause for the abend, if it is known.
- H** The relative interrupt address is the hexadecimal address of the abending instruction relative to the beginning of the program. Relating this address to MLIST or CLIST gives you the verb that caused the abend. Keep in mind that the address may not match exactly with MLIST or CLIST, because it represents object code lying between the addresses of two COBOL statements. MLIST and CLIST are located in the optimized listing.
- In a sequence of CALLED programs, the program currently being reported on may not be the abending program; it may have called or otherwise passed control to the program that abended. In this case, the words INTERRUPT ADDRESS become TRANSFER ADDRESS giving the relative hexadecimal address of where this program last had control.

- I** The absolute hexadecimal address of the abending instruction. This address is most relevant to non-COBOL programs.
- J** The point at which the Procedure Division was entered for the current program execution. It is either the start address, which is located just ahead of the first verb address in your program, or the relative address of an entry point which is defined by an ENTRY verb in your program. The start address can be located in the PMAP while an entry point address relates directly to your MLIST or CLIST.
- K** The number of times the abending program was entered. This entry count becomes significant when the program is one that is called by other programs a number of times.

The Offending Instruction

In the case of program check interruptions (0Cx), Detector reports on the particular instruction which caused theabend with as much supporting information as possible.

Detector may provide additional information for:

- Protection
- Addressing
- Data
- Fixed-point overflow
- Fixed-point divide
- Decimal overflow
- Decimal divide
- Exponent overflow
- Significance
- Floating-point divide

Detector cannot always provide further analysis of the instruction, especially with the following exceptions:

- Operation
- Privileged operation
- Execution
- Specification

Module Call Sequence

The next block of information is of interest if the application comprises multiple program modules. It lists the calling sequence of programs that resulted in the execution of the program that produced the abend.

MODULE CALL SEQUENCE:									
A	B	C	D	E	F	G		H	I
CALLING PROGRAM	RETURN DISP.	LOAD MODULE	CALLED PROGRAM	ACTUAL ADDR	PROGRAM LENGTH	DATE	TIME	LANGUAGE	OPTIMIZER VERSION/REL
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
SYSTEM		CALLED	SYSTEM	00000000	00			NOT COBOL	
CAOMDEMO	(ABENDED)	CALLED	CAOMDEMO	00007FF8	4006	JUL 18, 1901	15.20.19	COBOL VS	OPT 7.0
LOAD LIBRARY FOR CAOMDEMO IS 'USRID02.LOAD'.									
LOAD MODULE CAOMDEMO LENGTH IS 00004008 (d'16392').									
J									

A Specifies the name of the calling program. SYSTEM indicates the beginning of the current execution.

B Indicates the relative address in the calling program **A** to which control would normally have been returned.

C Identifies the load module name of the called program.

D Identifies the called program name.

E Specifies the hexadecimal address in storage of the called program.

F Shows the length of the called program.

G Indicates the compile date and time of the called program.

H Specifies the language in which the called program is coded and whether it was optimized by CA-Optimizer.

I Indicates the version number under which the calling program was optimized.

J Displays the data set name from which the abending program was loaded along with the length of the load-module in hexadecimal and decimal.

Note: The actual register Save Area Trace is at the end of the Detector Abend Report. Reference the Save Area Trace if you want to know the actual contents of the registers when control was passed from program to program.

File Section Display

Detector displays the status and the current record contents of each open data file defined to the COBOL program. If it is a sequentially accessed VSAM file, Detector will also display the contents of the previous record.

COMPUTER ASSOCIATES		DETECTOR REPORT		PAGE 3	
CA-OPTIMIZER 7.0 0106ARX00				15 JUN 2001 14.29.43	

* FILE SECTION *					

A					
DDNAME=SYSPRINT DSORG=PS QSAM BUFNO=1 UNIT=3390,UCB VOL=SER=MVR2AC					
DSN=USRID02.USRID02A.JOB03260.D0000106.? RECFM=FBA					
STATUS(OPEN)B USAGE(OUTPUT) LRECL=121 BLKSIZE=121					
RECORDS PROCESSED =4C					
CONTENTS OF CURRENT LOGICAL RECORD					

E					
LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION	

00095	01 LINE-OUT		GROUP	BL =0001+000000 (000248C8)	
00096	03 SLEW-CTL	X	DISPLAY	ZEROS	
00097	03 FILLER	X(120)	DISPLAY	*	*

The File Section Display portion of the Detector Report lists information for each file, when applicable:

- A** The data management control blocks are interpreted and displayed here in KEYWORD=value format.
- B** The current file's status (OPEN | CLOSED), usage (INPUT | OUTPUT | WORK), and access method I/O type (QSAM | VSAM | BSAM | BPAM, and so forth).
- C** The number of records processed. On a READ-WRITE file this number represents the total I/O count to this file. Report-Writer files display a zero here. This number actually represents the number of times a READ or WRITE verb was executed for this file.
- D** The current file status, for open VSAM files (not shown).
- E** The contents of the current logical record. If the FIRST128 option was specified, the option displays the first 128 bytes of the current logical record both in this report and the Open Files Report. If the MRGFILES option is specified, the data is displayed in merged format. Otherwise, it is displayed in dump format.
- F** If the file is a sequentially accessed VSAM file, the contents of the previous record is displayed (not shown).



Depending on the data management and VSAM control block options specified, I/O control blocks are optionally displayed along with their absolute addresses. A column of addresses, which are displacements relative to the blocks' beginning, also prints. This displacement is useful in locating offsets into the blocks.

The following blocks may be displayed for a Detector Abend Report: DCB, DEB, IOB, or UCB for non-VSAM files; ACB, AMB, AMBL, DEB, PLH, RPL, or UCB for VSAM files. These blocks are not displayed for a Detector Snap Report.

Note: END OF FILE HAS BEEN REACHED prints if the EOF status flag is set in the Data Extent Block (DEB) by system EOVS processing.

WORKING-STORAGE SECTION and LINKAGE SECTION Displays

Displays of the WORKING-STORAGE SECTION and the LINKAGE SECTION are given in the Detector Abend Report.

An example of a Linkage Section Display follows:

COMPUTER ASSOCIATES		DETECTOR REPORT		PAGE 8		
CA-OPTIMIZER		7.0 0106ARX00		15 JUN 2001 14.29.43		
JOB: USRID02A		STEP: RUN	PROGRAM CAOMDEMO			

* LINKAGE SECTION *						

LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION		
-----	-----	-----	-----	-----		
00319	01 PARMs		GROUP	BLL=0003+000000 (80006FF6)		
00320	03 PARM-LENGTH	S9(4)	BINARY	+6		
00321	03 PARM-DATA	X(12)	DISPLAY	? X'E6E2C9D5C9E3000000000000'		
00322	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00323	05 PARM-S	X(8)	DISPLAY	? X'E6E2C9D5C9E30000'		
00324	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00325	05 PARM-S-E	X(12)	DISPLAY	? X'E6E2C9D5C9E3000000000000'		
00326	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00327	05 PARM-F	X(7)	DISPLAY	? X'E6E2C9D5C9E300'		
00328	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00329	05 PARM-F-E	X(11)	DISPLAY	? X'E6E2C9D5C9E3000000000000'		
00330	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00331	05 PARM-W	X(6)	DISPLAY	*WSINIT*		
00332	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00333	05 PARM-W-E	X(10)	DISPLAY	? X'E6E2C9D5C9E30000000000'		
00334	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		
00335	05 PARM-N	X(4)	DISPLAY	*WSIN*		
00336	03 FILLER	REDEFINES PARM-DATA	GROUP	BLL=0003+000002 (80006FF8)		

If the MRGWORK (for Working-Storage) and MRGLINK (for Linkage Section) options are specified, the data is displayed in merged format. Otherwise, it is displayed in dump format.

See Merged vs. Dumped Data Displays for more information about the merged and dumped formats.

Register Contents

Detector displays the general registers and floating-point registers for the abending program.

The base address and a displacement are given if the contents of a register can be related directly to them. The base address may appear as:

- BL
- BLL
- Program Name
- PGT
- TGT

A blank line following the register display indicates that the origin of the register contents cannot be determined.

COMPUTER ASSOCIATES

CA-OPTIMIZER

7.0 0106ARX00

DETECTOR REPORT

PAGE 20

15 JUN 2001 14.29.52

JOB: USRID02A STEP: RUN

PROGRAM CAOMDEMO

* REGISTERS AT ABEND *

GENERAL REGISTERS:

R0 FFFFFFFF

R1 000000C4

R2 000092AC CAOMDEMO OFFSET 0012B4

R3 0000BEFE CAOMDEMO DISP 003F06

R4 80006FF6

R5 00009098 BL 0003 DISP 000000

R6 00001000

R7 00002000

R8 00009D64 CAOMDEMO DISP 001D6C

R9 0000BB16 CAOMDEMO DISP 003B1E

R10 01000000 BL 0001 DISP 000000

R11 00008098 BL 0002 DISP 000000

R12 00009BC4 PGT DISP 0001A4

R13 00009390 TGT DISP 000000

R14 8000BD14 CAOMDEMO DISP 003D1C

R15 00041B18

FLOATING-POINT REGISTERS:

FR0 00000000 00000000

FR2 00000000 00000000

FR4 00000000 00000000

FR6 00000000 00000000

If the basic floating-point extensions of OS/390 are installed on your system, Detector will display the floating-point control register and floating-point registers 0 through 15. Otherwise, Detector will display floating-point registers 0, 2, 4, and 6 (shown here).

Memory Map Display

The following is a display of the Memory Map, including both the TGT and PGT.

COMPUTER ASSOCIATES		DETECTOR REPORT		PAGE 10	
CA-OPTIMIZER		7.0 0106ARX00		15 JUN 2001 14.29.43	
JOB: USER002A		STEP: RUN		PROGRAM CAOMDEMO	

* MEMORY MAP *					

TGT	00009390	LENGTH	00000690	(d'1680')	
ADDRESS	OFFSET				
00009390	000000	00300000	00006FA0	00006B78	5000A2C6
000093B0	000020	00000058	00000000	00009BC4	00001000
000093D0	000040	00008098	00009A20	3042204B	00000000
000093F0	000060	00000003	51769824	00000000	5000A2C6
00009410	000080	00000058	00000000	00009BC4	00001000
00009430	0000A0	00008098	00009A20	60009DC8	000092AC
00009450	0000C0	800075AA	80006FF6	800075A6	00000001
00009470	0000E0	00009A44	10000000	00000000	0000E6E2
00009490	000100	40404040	40404040	40404040	40404040
000094B0	000120	40404040	40404040	00FD2BE8	000071B6
000094D0	000140	80006FF6	00009098	00001000	00002000
000094F0	000160	00009BC4	00000000	00000000	00000000
00009510	000180	00000000	00009490	00009480	E0000000
00009530	0001A0	00000000	00000000	00000000	4000DE9A
00009550	0001C0	E4E34040	E380000C	00008080	00000000
00009570	0001E0	00000000	00000000	00000000	00000000
00009590	000200	000248C8	00008098	00009098	00009BC4
000095B0	000220	00000000	00000000	00000000	9936358D
000095D0	000240	00000000	00000000	00035176	9824357D
000095F0	000260	00000000	0001775D	00000000	0001776D
00009610	000280	00000000	0001866C	00000000	0001865C
00009630	0002A0	00000000	00000000	00000000	00000000
LINES 000002C0-00000360 SAME AS ABOVE					
00009710	000380	00000000	00000000	00000000	00000000
00009730	0003A0	00000000	00000000	00000000	00000000
00009750	0003C0	00000000	00000000	00000000	00000000
00009770	0003E0	0000000F	0000000F	0000000F	0000000F
00009790	000400	0000000F	0000000F	0000000F	0000000F
000097B0	000420	0000000F	0000000F	A000A02C	0000AD64
000097D0	000440	00000000	00000000	68117001	0000F888
000097F0	000460	00003D92	00000001	00014F00	0000F8F4
00009810	000480	D4E3C1E0	00000000	00000000	00000000
00009830	0004A0	00840017	001803CC	0003042C	00070200
00009850	0004C0	0001021C	00020002	01630000	58F0D478
00009870	0004E0	06005000	F00047F0	E0040700	5810D500
00009890	000500	00000000	00000000	00000000	00000000
00009910	000520	00000000	00000000	00000000	00000000
00009930	000540	00000000	00000000	00000000	00000000
00009950	000560	00000000	00000000	00000000	00000000
00009970	000580	00000000	00000000	00000000	00000000
00009990	000600	00000000	00000000	00000000	00000000
00009A10	000620	00000000	00000000	00000000	00000000
00009A30	000640	00000000	00000000	00000000	00000000
00009A50	000660	00000000	00000000	00000000	00000000
00009A70	000680	00000000	00000000	00000000	00000000
00009A90	000700	00000000	00000000	00000000	00000000
00009AB0	000720	00000000	00000000	00000000	00000000
00009AD0	000740	00000000	00000000	00000000	00000000
00009AE0	000760	00000000	00000000	00000000	00000000
00009AF0	000780	00000000	00000000	00000000	00000000
00009B00	000800	00000000	00000000	00000000	00000000
00009B20	000820	00000000	00000000	00000000	00000000
00009B40	000840	00000000	00000000	00000000	00000000
00009B60	000860	00000000	00000000	00000000	00000000
00009B80	000880	00000000	00000000	00000000	00000000
00009BA0	000900	00000000	00000000	00000000	00000000
00009BC0	000920	00000000	00000000	00000000	00000000
00009BE0	000940	00000000	00000000	00000000	00000000
00009C00	000960	00000000	00000000	00000000	00000000
00009C20	000980	00000000	00000000	00000000	00000000
00009C40	000A00	00000000	00000000	00000000	00000000
00009C60	000A20	00000000	00000000	00000000	00000000
00009C80	000A40	00000000	00000000	00000000	00000000
00009CA0	000A60	00000000	00000000	00000000	00000000
00009CB0	000A80	00000000	00000000	00000000	00000000
00009CC0	000AA0	00000000	00000000	00000000	00000000
00009CE0	000AC0	00000000	00000000	00000000	00000000
00009CF0	000AE0	00000000	00000000	00000000	00000000
00009D00	000B00	00000000	00000000	00000000	00000000
00009D20	000B20	00000000	00000000	00000000	00000000
00009D40	000B40	00000000	00000000	00000000	00000000
00009D60	000B60	00000000	00000000	00000000	00000000
00009D80	000B80	00000000	00000000	00000000	00000000
00009DA0	000BA0	00000000	00000000	00000000	00000000
00009DC0	000BC0	00000000	00000000	00000000	00000000
00009DE0	000BE0	00000000	00000000	00000000	00000000
00009DF0	000C00	00000000	00000000	00000000	00000000
00009E00	000C20	00000000	00000000	00000000	00000000
00009E20	000C40	00000000	00000000	00000000	00000000
00009E40	000C60	00000000	00000000	00000000	00000000
00009E60	000C80	00000000	00000000	00000000	00000000
00009E80	000CA0	00000000	00000000	00000000	00000000
00009EA0	000CC0	00000000	00000000	00000000	00000000
00009EC0	000CE0	00000000	00000000	00000000	00000000
00009ED0	000D00	00000000	00000000	00000000	00000000
00009EE0	000D20	00000000	00000000	00000000	00000000
00009EF0	000D40	00000000	00000000	00000000	00000000
00009F00	000D60	00000000	00000000	00000000	00000000
00009F20	000D80	00000000	00000000	00000000	00000000
00009F40	000DA0	00000000	00000000	00000000	00000000
00009F60	000DC0	00000000	00000000	00000000	00000000
00009F80	000DE0	00000000	00000000	00000000	00000000
00009FA0	000E00	00000000	00000000	00000000	00000000
00009FB0	000E20	00000000	00000000	00000000	00000000
00009FC0	000E40	00000000	00000000	00000000	00000000
00009FE0	000E60	00000000	00000000	00000000	00000000
00009000	000E80	00000000	00000000	00000000	00000000
00009020	000EA0	00000000	00000000	00000000	00000000
00009040	000EC0	00000000	00000000	00000000	00000000
00009060	000EE0	00000000	00000000	00000000	00000000
00009080	000F00	00000000	00000000	00000000	00000000
000090A0	000F20	00000000	00000000	00000000	00000000
000090C0	000F40	00000000	00000000	00000000	00000000
000090E0	000F60	00000000	00000000	00000000	00000000
00009100	000F80	00000000	00000000	00000000	00000000
00009120	000FA0	00000000	00000000	00000000	00000000
00009140	000FC0	00000000	00000000	00000000	00000000
00009160	000FE0	00000000	00000000	00000000	00000000
00009180	001000	00000000	00000000	00000000	00000000
000091A0	001020	00000000	00000000	00000000	00000000
000091C0	001040	00000000	00000000	00000000	00000000
000091E0	001060	00000000	00000000	00000000	00000000
00009200	001080	00000000	00000000	00000000	00000000
00009220	0010A0	00000000	00000000	00000000	00000000
00009240	0010C0	00000000	00000000	00000000	00000000
00009260	0010E0	00000000	00000000	00000000	00000000
00009280	001100	00000000	00000000	00000000	00000000
000092A0	001120	00000000	00000000	00000000	00000000
000092C0	001140	00000000	00000000	00000000	00000000
000092E0	001160	00000000	00000000	00000000	00000000
00009300	001180	00000000	00000000	00000000	00000000
00009320	0011A0	00000000	00000000	00000000	00000000
00009340	0011C0	00000000	00000000	00000000	00000000
00009360	0011E0	00000000	00000000	00000000	00000000
00009380	001200	00000000	00000000	00000000	00000000
000093A0	001220	00000000	00000000	00000000	00000000
000093C0	001240	00000000	00000000	00000000	00000000
000093E0	001260	00000000	00000000	00000000	00000000
00009400	001280	00000000	00000000	00000000	00000000
00009420	0012A0	00000000	00000000	00000000	00000000
00009440	0012C0	00000000	00000000	00000000	00000000
00009460	0012E0	00000000	00000000	00000000	00000000
00009480	001300	00000000	00000000	00000000	00000000
000094A0	001320	00000000	00000000	00000000	00000000
000094C0	001340	00000000	00000000	00000000	00000000
000094E0	001360	00000000	00000000	00000000	00000000
00009500	001380	00000000	00000000	00000000	00000000

COMPUTER ASSOCIATES		CAOMDEMO		PAGE 32	
CA-Optimizer	7.0 0106A0M00			15 JUN,2001	14.29.26
	CAOMDEMO	14.29.26	JUN 15,2001		
MEMORY MAP			TGT RELATIVE		
TGT	01398		0000		
SAVE AREA	01398		0000		
SWITCH	013E0		0048		
TALLY	013E4		004C		
SORT SAVE	013E8		0050		
ENTRY-SAVE	013EC		0054		
SORT CORE SIZE	013F0		0058		
RET CODE	013F4		005C		
SORT RET	013F6		005E		
WORKING CELLS	013F8		0060		
SORT FILE SIZE	01528		0190		
SORT MODE SIZE	0152C		0194		
PGT-VN TBL	01530		0198		
TGT-VN TBL	01534		019C		
RESERVED	01538		01A0		
LENGTH OF VN TBL	0153C		01A4		
LABEL RET	0153E		01A6		
RESERVED	0153F		01A7		
DBG R14SAVE	01540		01A8		
COBOL INDICATOR	01544		01AC		
A(INIT1)	01548		01B0		
DEBUG TABLE PTR	0154C		01B4		
SUBCOM PTR	01550		01B8		
SORT-MESSAGE	01554		01BC		
SYSOUT DDNAME	0155C		01C4		
RESERVED	0155D		01C5		
COBOL ID	0155E		01C6		
COMPILED POINTER	01560		01C8		
COUNT TABLE ADDRESS	01564		01CC		
RESERVED	01568		01D0		
DBG R11SAVE	01570		01D8		
COUNT CHAIN ADDRESS	01574		01DC		
PRBL1 CELL PTR	01578		01E0		
RESERVED	0157C		01E4		
TA LENGTH	01581		01E9		
RESERVED	01584		01EC		
PCS LIT PTR	0158C		01F4		
DEBUGGING	01590		01F8		
CD FOR INITIAL INPUT	01594		01FC		
OVERFLOW CELLS	01598		0200		
BL CELLS	01598		0200		
DECBADR CELLS	015B4		021C		
FIB CELLS	015B4		021C		
TEMP STORAGE	015B8		0220		
TEMP STORAGE-2	01628		0290		
TEMP STORAGE-3	01728		0390		
TEMP STORAGE-4	01728		0390		
BLL CELLS	01728		0390		
VLC CELLS	0173C		03A4		
SBL CELLS	0173C		03A4		
INDEX CELLS	0173C		03A4		

If you wish to know the contents of any index cell, locate the displacement address from the start of the TGT, in this case this address is 3A4 (located in the TGT Relative column), and count four bytes for each index cell until you reach the desired one.

PERFORM Return Address Analysis

If your program abends in a PERFORM range and you need to establish which PERFORM verb you executed to get into that PERFORM range, do the following:

1. Using the relative address in your program where the abend occurred, go to the PERFORM Range Table that is part of every compile/optimize listing.

COMPUTER ASSOCIATES			CAOMDEMO			PAGE 110	
CA-Optimizer			7.0	0106AOM00	15 JUN,1901 14.29.26		
CAOMDEMO			14.29.26	JUN 15,1901			
PERFORM RANGE TABLE							
LEGEND:							
OVERLAP - THIS PERFORM RANGE OVERLAPS ANOTHER PERFORM RANGE.							
FALL THRU - THIS PERFORM RANGE EXIT MAY FALL THROUGH TO THE NEXT STATEMENT IN THE PROGRAM.							
BRANCH OUT - THERE IS A BRANCH IN THE PERFORM RANGE THAT BRANCHES OUT OF THE PERFORM RANGE TO THE ADDRESS SHOWN.							
RECURSIVE - THERE IS A PATH FROM THIS PERFORM RANGE ENTRANCE, BEFORE REACHING THE PERFORM RANGE EXIT, WHICH LEADS TO A PERFORM VERB OF THE SAME PERFORM RANGE. THE ENTRY AND EXIT ADDRESSES OF THE PERFORM RANGES CONTAINING THE RECURSIVE PERFORM VERBS ARE SHOWN.							
NO EXIT - THIS PERFORM RANGE CONTAINS NO PATH FROM THE PERFORM RANGE ENTRANCE TO THE PERFORM RANGE EXIT.							
NULL - THIS PERFORM RANGE CONTAINS NO EXECUTABLE STATEMENTS.							
SOURCE	ADDRESS	RETURN	OVER	FALL	BRANCH	RECURSIVE NO NULL	
ENTRY - EXIT	ENTRY - EXIT	CELL	-LAP	THRU	OUT TO	CALL FROM EXIT	
000492-000492	00265A-0026B8	PSV=24					
000495-000504	0026D6-00278C	PSV=23					
000509-000516	0027AA-0027C2	PSV=22					
000530-000565	0027E0-00294A	PSV=21					
000570-000581	00296C-002A6A	PSV=20					
000602-000604	002BE8-002BF4	PSV=19	YES	YES			
000586-000622	002A90-002D4E	VN=1	YES	YES			
000626-000636	002D86-002D8A	PSV=18		YES			
000641-000648	002DB8-002ECA	PSV=17		YES			
000653-000653	002EF8-002F04	PSV=16		YES			
000658-000683	002F32-00308C	PSV=15		YES			
000687-000707	0030BA-003146	PSV=14		YES			
000712-000717	003178-0031DC	PSV=13		YES			
000722-000796	003212-003794	PSV=12		YES			
000801-000805	0037C6-0037E4	PSV=11		YES			
000808-000812	00380E-00382C	PSV=10		YES			
000815-000821	00385A-003884	PSV=9		YES			
000824-000830	0038AE-0038D8	PSV=8		YES			
000833-000839	003902-00392C	PSV=7		YES			
000842-000848	003956-003980	PSV=6		YES			
000851-000857	0039AA-0039D4	PSV=5		YES			
000860-000883	0039FE-003C74	PSV=4		YES			
000886-000894	003C9E-003D26	PSV=3		YES			
000897-000897	003D50-003D62	PSV=2		YES			
SUMMARY OF PERFORM RANGES:			YES	YES	NO	NO NO NO NO	
ALTERED VN TABLE							
STATEMENT	ADDRESS	CELL					
000350	00001DB4	VN=2					
000435	00002362	VN=3					

2. Locate the PERFORM range that contains that relative address and note the return cell (R15, PSV, or VN) associated with the PERFORM range. This cell contains the address of the source statement following the PERFORM verb (the point to which control returns after the PERFORM range executes).
3. Look up that cell in the PERFORM RETURN ADDRESSES section of the Detector Report; it gives you the cell's contents in relative address format (the relative address of the statement following the PERFORM verb).

The PERFORM Return Addresses can also give you further information about the PERFORM Ranges in your program, depending on the type of cell used for linkage.

***** * PERFORM RETURN ADDRESSES * *****				
PSV CELLS				
2=NC	3=0025DE	4=NC	5=NC	
6=NC	7=NC	8=NC	9=NC	
10=NC	11=NC	12=0025D2	13=002592	
14=002586	15=00257A	16=00256E	17=002562	
18=002556	19=00254A	20=00251A	21=00250E	
22=002502	23=0024F6	24=0036F6		
VN CELLS				
1=FT	2=001E70	3=0023A6		
REGISTER 15 = 40404040 NOT A RETURN ADDRESS				

A

The PSV cells reveal the most information:

- If NC (Never Called) follows a cell number, the PERFORM range it represents was never executed.
- If an address prints next to the cell number, it is the return address for the last time the associated PERFORM range was executed. Match the address here with the address of the MLIST or CLIST in the CSL to find the last PERFORM CALL for the PERFORM range.
- If FT (Fall Through) follows the cell number, the last time you executed its PERFORM range you did not perform it. You either fell into it, through it, and out of it, or you entered it by a GO TO and then fell out.

Note: Fall-through refers to the process of logically ignoring the exit for a PERFORM range and continuing downwards, executing the following contiguous statements.

B

If there is an address for a VN cell, it can mean one of three things:

- You were in the associated PERFORM range when you abended, and the address is the return address.
- You were in the PERFORM range earlier in the execution, but somehow branched out instead of going through the normal exit.
- The VN cell represents an ALTERED GO TO. The cell contains the destination address for the next execution of the ALTERED GO TO.

Note: The NC code, Never Called, is not used because this condition cannot be determined with VN cells. An FT code means that the fall-through address was found there, and is meaningful if your program abended while a VN PERFORM range was active. FT means this PERFORM range was not entered by a PERFORM statement.



When youabend in an R15 PERFORM range, the address in register 15 points to the address you would have returned to after exiting the PERFORM range. In this example report, theabend did not take place in an R15 PERFORM range.

Paragraph Flow Report

The Paragraph Flow Report is an optional section of the Detector Abend Report. PFLOW lists the source name and address of the last 64 paragraphs executed up to the time of theabend. If the total number of paragraphs executed is less than 64, the report shows only what was executed. These addresses are relative to the beginning of the COBOL program and match up with the addresses given by MLIST and PMAP. If the Condensed Source Listing is not available, then only the addresses are printed; the PARAGRAPH NAME field is blank.

```
***** RELATIVE ADDRESS OF THE LAST    64 PARAGRAPHS
*  PARAGRAPH FLOW    * EXECUTED (IN ORDER OF EXECUTION, LEFT TO RIGHT)
***** 'COUNT' SHOWS CONSECUTIVE EXECUTIONS IF GREATER THAN 1

COUNT  OFFSET *-----PARAGRAPH NAME-----* COUNT  OFFSET *-----PARAGRAPH NAME-----*
  26    0032F0 CHECK-1-LOOP.                003370 CHECK-2.
  31    0033E4 CHECK-2-LOOP.                003458 CHECK-3.
        003646 CHECK-4.                    003688 CHECK-5.
        0036D0 CHECK-6.                    00264E WRITE-LINE
CAPD137I CSECT DUMP SUPPRESSED DUE TO EXCLUSION CRITERIA.
```

The paragraph names or addresses are listed from left to right in order of execution, meaning the one executed most recently appears last. A number to the left of an address indicates the number of consecutive times that the paragraph was executed, if it was executed more than once. If the paragraph was executed only once, the COUNT value remains blank.

The PFLOW feature is optional, but we strongly recommend that you use it for all debugging runs.

Open Files Report

In the Open Files Report, Detector prints the current logical record. Optionally, it also prints the data management control block for all files in the job step that remain open and have not been displayed earlier in the File Section of the Detector Abend Report. If the file is a sequentially accessed VSAM file, the previous logical record is also dumped. These records and blocks may be from COBOL programs not optimized with the DTECT option ON, or from any non-COBOL program in the job step.

The Open Files Report appears once, near the end of the Detector Report. The contents of this report are similar to the contents of the File Section Display. See the File Section Display Report for a description of the fields on the report. The following is an example of an Open Files Report.

```

COMPUTER ASSOCIATES          DETECTOR REPORT
CA-OPTIMIZER          7.0 0106ARX00
                                           PAGE    23
                                           15 JUN 2001  14.29.52

JOB: USRID02A  STEP: RUN          PROGRAM  CAOMDEMO
*****
* OPEN FILES REPORT *
*****
A
DDNAME=SYSOUT      DSORG=PS  QSAM  BUFNO=1  UNIT=3390,UCB  VOL=SER=MVR2AC
DSN=USRID02.USRID02A.JOB03260.D0000108.?          RECFM=FBA
STATUS(OPEN)      USAGE(OUTPUT)          LRECL=121  BLKSIZE=121

          CONTENTS OF CURRENT LOGICAL RECORDB
          -----
ADDRESS 00024840      LENGTH 00000079  (d'121')
      OFFSET
      +000000 40404040 40404040 40404040 40404040  *      *
      LINES 00000010-00000060 SAME AS ABOVE
      +000070 40404040 40404040 40      *      *

          DATA MANAGEMENT CONTROL BLOCKSB
          -----
DEB      000B5030      LENGTH 00000088  (d'136')
      OFFSET
      +000000 00000000 00000000 00000000 00000000
      LINES 00000010-00000070 SAME AS ABOVE
      +000080 00000000 00000000

DCB      00006010      LENGTH 00000060  (d'96')
      OFFSET
      +000000 00000000 00000000 00000000 01000000
      +000010 00000000 01024838 00794000 000069A0
      +000020 42000001 9400EF84 00900048 008B5030
      +000030 92D806F0 00000001 00000001 00090079
      +000040 00000000 00006840 000248B9 000248B9
      +000050 00000079 00000001 00000000 00000001

ACB      00006860      LENGTH 0000004C  (d'76')
      OFFSET
      +000000 A000004C 00000000 00BB4DC0 52000000
      +000010 00000000 00000000 4A040008 00000000
      +000020 00000000 00000000 00900041 008B5030
      +000030 12000000 00006784 00000000 00790079
      +000040 00000000 00000000 00000000

UCB      00F05198      LENGTH 00000040  (d'64')
      OFFSET
      +000000 00000000 00000000 00000000 00000000
      LINES 00000010-00000020 SAME AS ABOVE
      +000030 00000000 00000000 00000000 00000000

IOB      000069A0      LENGTH 0000003E  (d'62')
      OFFSET
      +000000 08000000 00000000 04000000 00000000
      +000010 00000000 00000000 00000000 00000000
      +000020 7F000000 008C3AB0 00000000 80000000
      +000030 00006A30 04020004 00000000 0000

```

A DDNAME is the filename and DD statement name.

B Displays the contents of any current logical record. For input files, the record just read is displayed in hexadecimal format followed on the right by a character representation.

The contents of an output record are not nearly as reliable. If the record is written using a WRITE statement, the record displayed by Detector contains what has been generated in it up to the time of the abend. If a WRITE FROM statement is used, the record more likely contains an old record that has already been written.

C

Depending on the data management and VSAM control block options specified, I/O control blocks are optionally displayed along with their absolute addresses. A column of addresses, which are displacements relative to the beginning of each block, also prints. This displacement is useful in locating offsets, into the DCB for example.

Save Area Trace Report

The Save Area Trace is a report that assists you in tracing and debugging programs in a multimodule environment. This report prints only once for any given abend reporting sequence. Whether the abend occurs in a multi-module run and Detector reports on more than one program, or it is a single-module run with only the Detector Abend Report, the Save Area Trace always prints just one time and is always the last item in the reporting sequence.

```

COMPUTER ASSOCIATES                               DETECTOR REPORT                               PAGE      24
CA-OPTIMIZER                                     7.0 0106ARX00                               15 JUN 2001  14.29.52

JOB: USRID02A  STEP: RUN                        PROGRAM  CAOMDEMO

*****
* SAVE AREA TRACE *
*****

_____ CALLED SYSTEM (NOT COBOL)
SAVE AREA FOR SYSTEM

  A      B      C      D      E
SA 00006FA0 WD1 00000000 BKL 00000000 FWL 00009390 RET 80FD2BE8
  F EPA 00007FF8 R0 FD00000E R1 00006FF0 R2 00000040
R3 008D2CD4 R4 008D2CB0 R5 008F9358 R6 008B9FF8
R7 FD000000 R8 008FFD90 R9 008E37F0 R10 00000000
R11 008F9358 R12 00C49002 G

SYSTEM CALLED CAOMDEMO (COBOL VS OPT)

CAOMDEMO ABENDED

```

When a program calls a subordinate program, the caller's registers are stored in an 18 fullword save area to establish the means of return. A series of calling and called programs establishes a chain of save areas, as shown in the example.

In this example, the operating system calls program CAOMDEMO.

A

The system's save area location.

B

WD1 is the first word of the eighteen-word save area. It is not used by the system and is usually zeros.

- C** BKL stands for backward link. It points backwards to the location of the prior save area if any.
- D** FWL is the forward link. It points forward to the location of the next save area, if any.
- E** RET represents the return address as an absolute hexadecimal location.
- F** EPA means entry point address. When control is passed from system to CAOMDEMO, this is the point where CAOMDEMO is entered.
- G** The next thirteen words are used to save the contents of registers 0 through 12.

This eighteen-word save area prints for each active program in the hierarchy leading to the program which abended. The save area for the abending program does not display.

Whenever Detector encounters a non-COBOL program in the chain, it designates the program in the Save Area Trace as UNKWNnnn where nnn is 1-999. Detector does not determine whether it is an Assembler program, PL/I, and so forth.

The information that Detector supplies in this detailed version of the Save Area Trace can be valuable to the user who understands register conventions and whose problem involves more than one module.

IMS Report

The IMS portion of the Detector report is produced whenever a program abends which accesses an IMS database.

```

COMPUTER ASSOCIATES          DETECTOR REPORT
CA-OPTIMIZER      7.0 0103ARX00
                                         PAGE    19
                                         22 JUN 2001  10.19.25

JOB: IMSTEST   STEP: G           PROGRAM DFSSAM01

*****
* IMS 6.1.0 RELATED INFORMATION *A
*****

*****
* LAST IMS CALL *
*****
B
CALL LOCATION:                      PROGRAM-ID DFSSAM01
      SOURCE LOCATION: PAGE    6 LINE 000151 MOFFSET 000ABA

CALL 'CBLTDLI' USING DL1-FUNCTION, PCB-AREA-1,
      SEG00060-INSERT-AREA, SEG00010-SSA,
      SEG00060-SSA.

IMS FUNCTION: ISRTC

```

The IMS report begins by displaying the following information:

A

The release of IMS.

B

The location of the last IMS call, if it can be determined. If a CSL member is available for the program, this information includes the program-id, the page number, line number and MOFFSET of the source statement, and the COBOL source statement.

If a CSL member is not available, this information includes the address of the last call, the program-id, and the offset of the call from the start of the program.

Note: Call location information is not available under the LE runtime.

C

The IMS function used on the last IMS call.

```

*****
* CURRENT/LAST DATABASE PCB - DI21PART *A
*****

PCB PREFIX:
ADDRESS  OFFSETB
00016CE8 000000 00500038 00020028 40404040 00000000 *.&.....*
00016CF8 000010 00000000 00000000 00000000 C4C2D3D6 *.....DBLO*
00016D08 000020 C1C44040 000067A0 00000000 00000000 *AD .....*
00016D18 000030 00016D70 00016CE8 *...%Y*

PCB:C

LINE#  LEVEL/FIELD NAME          PICTURE CLAUSE          USAGE          VALUE/LOCATION
-----
000096  01 PCB-AREA-1
000097  02 DBD-NAME                    X(08)           DISPLAY        BLL=0001+000000 (00016D20)
000098  02 SEGMENT-LEVEL              X(02)           DISPLAY        ? X'00C9F2F1D7C1D9E3'
000099  02 STATUS-CODES               X(02)           DISPLAY        *02*
000100  02 PROCESS-OPTIONS           X(04)           DISPLAY        SPACES
000101  02 FILLER                     S9(05)          DISPLAY        *L *
000102  02 SEG-NAME                   X(08)           DISPLAY        +93416
                                *STANINFO*

D E F G
STATUS  SEGMENT  NUMBER OF  SEGMENT  PROCESSING  KEY
CODE    LEVEL   SEGMENTS  NAME     OPTIONS    LENGTH
' '     02      5         STANINFO L      19

FEEDBACK FROM LAST CALL:D
ADDRESS  OFFSET
00016D44 000000  F0F2F9F8 F9F0F3F6 60F0F0F1 40404040 *02989036-001 *
00016D54 000010  40F0F2   * 02*

TRACE OF DATABASE CALLS (NEWEST ENTRY FIRST):
-0 ISRT      OK
-1 ISRT      OK
-2 ISRT      OK
-3 ISRT      OK
-4 ISRT      OK
-5 ISRT      OK

```

Information about the current or last database PCB is then displayed, if it is available:

- ^A PCB name.
- ^B The address and contents of the PCB prefix (IMS 4.1.0 and above).
- ^C The address and contents of the PCB. If the MERGEDB option is ON, the PCB is defined in the most active COBOL program, and a CSL is available for that program, the PCB is displayed in merged format. Otherwise, the PCB is displayed in dump format.
- ^D Status code.
- ^E Segment level.
- ^F Number of segments.
- ^G Name of last retrieved segment.

- H** Processing options.
- I** Key length.
- J** Key feedback.
- K** Call trace information for the JCB.

***** * CURRENT/LAST SEGMENT * *****				
A				
LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000081	01 SEG00060-INSERT-AREA		GROUP	BLW=0000+0004A8 (0005C880)
000082	02 FILLER	X(61)	DISPLAY	*02 742 *
			(+000032)	* 1200 96 *
000083	02 RIGHT-MAKE-SPAN	S9(03)	DISPLAY	+63
000084	02 FILLER	X(06)	DISPLAY	SPACES
000085	02 WRONG-MAKE-SPAN	9(03)	DISPLAY	? *06C*
000086	02 FILLER	X(12)	DISPLAY	SPACES
***** * SSA FOR LAST CALL * *****				
B				
ARGUMENT 1:				
LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000044	01 SEG00010-SSA		GROUP	BLW=0000+0003D0 (0005C7A8)
000045	02 SEG-NAME-00010	X(08)	DISPLAY	*PARTROOT*
000046	02 BEGIN-OP-00010	X(01)	DISPLAY	*(*
000047	02 KEY-NAME-00010	X(08)	DISPLAY	*PARTKEY *
000048	02 REL-OPER-00010	X(02)	DISPLAY	* =*
000049	02 KEY-VALUE-00010	X(17)	DISPLAY	*02989036-001 *
000050	02 END-OP-00010	X(01)	DISPLAY	*)*
ARGUMENT 2:				
LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000051	01 SEG00060-SSA		GROUP	BLW=0000+0003F8 (0005C7D0)
000052	02 SEG-NAME-00060	X(08)	DISPLAY	*STANINFO*
000053	02 BEGIN-OP-00060	X(01)	DISPLAY	*(*
000054	02 KEY-NAME-00060	X(08)	DISPLAY	*STANKEY *
000055	02 REL-OPER-00060	X(02)	DISPLAY	* =*
000056	02 KEY-VALUE-00060	X(02)	DISPLAY	*02*
000057	02 END-OP-00060	X(01)	DISPLAY	*)*

The remaining information about the last IMS call is displayed as follows:

- A** The current or last segment returned in the user I/O area.
- B** The Segment Search Argument (SSA) for the last database call, if available. If multiple search arguments were passed on the last call, they are displayed in the order in which they were passed.

For both areas above, if the MERGEDB option is ON, the area is defined in the most active COBOL program, and a CSL is available for that program, the area is displayed in merged format. Otherwise, it is displayed in dump format.

```

*****
* PROGRAM COMMUNICATION BLOCKS *
*****

***** A *****
* TERMINAL PCB FOR          - RELATIVE NUMBER 1 * (CURRENT OR LAST USED)
*****

PCB PREFIX: C
ADDRESS  OFFSET
00032084 000000  003C0038 00010018 40404040 00000000  *.....*
00032094 000010  00000000 00000000 00000000 C9D6D7C3  *.....IOPC*
000320A4 000020  C2404040 00000000 00000000 40404040  *B.....*
000320B4 000030  40404040 00032084                      *...d*

PCB: D
ADDRESS  OFFSET
000320BC 000000  40404040 40404040 10004040 40404040  * .. *
000320CC 000010  40404040 40404040 40404040 40404040  * .. *
000320DC 000020  40404040 40404040 40404040 40404040  * .. *
000320EC 000030  00000000 00000000 00000000 00000000  *.....*

E      F      G      H
STATUS  DATE AND TIME OF  MESSAGE  FORMATTING
CODE    LAST MESSAGE     COUNT
' '     N/A              N/A      N/A

```

All of the Program Communication Blocks (PCBs) used by the program are then displayed, beginning with the terminal PCBs, as follows:

- A** The PCB name and number.
- B** If it is the current or last used terminal PCB, it is marked as such.
- C** The address and contents of the PCB prefix (IMS 4.1.0 and above).
- D** The address and contents of the PCB. If the MERGEDB option is ON, the PCB is defined in the most active COBOL program, and a CSL is available for that program, the PCB is displayed in merged format. Otherwise, the PCB is displayed in dump format.
- E** The status code of the PCB.
- F** Date and time of last message.
- G** Message count.
- H** Formatting attributes.

```

*****
* DATABASE PCB FOR DI21PART - RELATIVE NUMBER 2 A * (CURRENT OR LAST USED)B
*****

PCB PREFIX:C
ADDRESS  OFFSET
00016CE8 000000  00500038 00020028 40404040 00000000  *.&.....*
00016CF8 000010  00000000 00000000 00000000 C4C2D3D6  *.....DBLO*
00016D08 000020  C1C44040 000067A0 00000000 00000000  *AD .....*
00016D18 000030  00016D70 00016CE8                *...%Y*

PCB:D

LINE#  LEVEL/FIELD NAME          PICTURE CLAUSE          USAGE          VALUE/LOCATION
-----
000096  01 PCB-AREA-1                  GROUP            BLL=0001+000000 (00016D20)
000097  02 DBD-NAME                     X(08)            DISPLAY        ? X'00C9F2F1D7C1D9E3'
000098  02 SEGMENT-LEVEL                X(02)            DISPLAY        *02*
000099  02 STATUS-CODES                 X(02)            DISPLAY        SPACES
000100  02 PROCESS-OPTIONS              X(04)            DISPLAY        *L *
000101  02 FILLER                       S9(05)           BINARY         +93416
000102  02 SEG-NAME                     X(08)            DISPLAY        *STANINFO*

E      F      G      H      I      J
STATUS  SEGMENT  NUMBER OF  SEGMENT  PROCESSING  KEY
CODE    LEVEL  SEGMENTS  NAME     OPTIONS    LENGTH
' '      02      5         STANINFO  L          19

FEEDBACK FROM LAST CALL:
ADDRESS  OFFSETK
00016D44 000000  F0F2F9F8 F9F0F3F6 60F0F0F1 40404040  *02989036-001 *
00016D54 000010  40F0F2                * 02*

TRACE OF DATABASE CALLS (NEWEST ENTRY FIRST):
-0 ISRT      OK
-1 ISRT      OK
-2 ISRT      OK
-3 ISRT      OK
-4 ISRT      OK
-5 ISRT      OK

```

The database PCBs are displayed as follows:

- A** PCB name and number.
- B** If it is the current or last used database PCB, it is marked as such.
- C** The address and contents of the PCB prefix (IMS 4.1.0 and above).
- D** The address and contents of the PCB. If the MERGEDB option is ON, the PCB is defined in the most active COBOL program, and a CSL is available for that program, the PCB is displayed in merged format. Otherwise, the PCB is displayed in dump format.
- E** Status code.
- F** Segment level.
- G** Number of segments.

- H** Name of last retrieved segment.
- I** Processing options.
- J** Key length.
- K** Key feedback.
- L** Call trace information for the JCB.

```

*****
* PARAMETER LIST * A
*****
ADDRESS  OFFSET
00006FC8 000000  C4D3C96B C4C6E2E2 C1D4F0F1 6BC4C6E2 *DLI,DFSSAM01,DFS*
00006FD8 000010  E2C1D4F0 F16BF76B F0F0F0F0 6B6BF06B *SAM01,7,0000,,0,*
00006FE8 000020  6BD56BF0 6BE36B6B 6BD56BD5 6B6BD56B *,N,0,T,,,N,N,,N,*
00006FF8 000030  6B6B                                     *,,*

*****
* MODULE DFSPRPX0 * B
*****
ADDRESS  OFFSET
00007A00 000000  00000000 00006FC0 00000000 00007A50 00007BA8 00007B00 00007A20 00000000 *. . . . . ? » . . . . . : & . . # y . . # . . . . . *
00007A20 000020  C4C6E2D7 C3C3F3F0 00000000 00018E90 00000000 0001A5F8 00000000 00000000 *DFSPCC30. . . . . v8. . . . . *
00007A40 000040  00000000 C4C6E2D4 D7D34040 00007DB0 00007A60 0000FFFF 00032060 008C81A8 *. . . . . DFSMPL . . ' . . . . - . . . . . - . . . . . ay *
00007A60 000060  E2C3C8C4 D3D6C1C4 E4D5D3C4 040000F0 C4C6E2E2 C1D4F0F1 C4C6E2E2 C1D4F0F1 *SCHDLOADUNLD. . . 0DFSSAM01DFSSAM01*
00007A80 000080  40404040 40404040 40404040 40404040 C1404040 00070000 00000000 00000000 *
00007AA0 0000A0  00000000 00000000 00007AA4 80007BD4 00000000 F0000122 00009B00 F080D5D5 *. . . . . : u . . # M . . . . . 0 . . . . . 0 . NN *

.
.
.
00007E00 000400  50F0F000 00000000                                     *&00. . . . . *

```

Finally, Detector displays the following areas:

- A** The parameter list passed to IMS.
- B** A dump of module DFSPRPX0.
- C** A dump of module DFSECP10 or DFSECP20, if available (not shown).

DB2 Reports

The DB2 portion of the Detector report is produced whenever a program abends that accesses a DB2 database using DB2 for MVS/ESA Version 4.1.0 or above.

```

COMPUTER ASSOCIATES          DETECTOR REPORT                      PAGE    20
CA-OPTIMIZER                7.0 0103ARX00                        23 JUN 2001 13.42.31

JOB: DB2610      STEP: RUN      PROGRAM DB2DEMO

A
*****
* DB2 6.1.0 RELATED INFORMATION FOR SUBSYSTEM D610 *B
*****

*****
* RETURN CODE FROM LAST SQL CALL *C
*****

DSNT400I SQLCODE = 000,  SUCCESSFUL EXECUTION

*****
* LAST SQL STATEMENT *
*****

STATEMENT LOCATION:D
COLLECTION-ID OPTIIDB2DEMO      PACKAGE DB2DEMO
CONSISTENCY TOKEN 16AACCF10F8D0459  LINE 00639

SQL STATEMENT:E
SELECT EMP , FIRSTNAME , LASTNAME , DEPT , MANAGER INTO :DCLEMPLOY.EMPLOY-EMP
:EMP-IND-1.EMP-IND-EMP , :DCLEMPLOY.EMPLOY-FIRSTNAME :EMP-IND-1.EMP-IND-FNAME ,
:DCLEMPLOY.EMPLOY-LASTNAME :EMP-IND-1.EMP-IND-LNAME , :DCLEMPLOY.EMPLOY-DEPT
:EMP-IND-1.EMP-IND-DEPT , :DCLEMPLOY.EMPLOY-MANAGER :EMP-IND-1.EMP-IND-MANAGER
FROM EMPLOY WHERE EMP = :DCLJOB.JOB-EMP

```

The first page of the DB2 report displays the following information:

- A The version of DB2.
- B The DB2 subsystem-ID.
- C The SQL return code, SQLCODE, and the message associated with that code. If SQLCODE is nonzero, the SQLSTATE value will also be displayed.
- D The location of the last SQL statement executed. (This is either the last SQL statement before the abend or the abending SQL statement.)

If the DB2 program that contains the last SQL statement is bound into a package, the location information includes the collection-id, package name, consistency token, and line number (as shown previously).

If the DB2 program is not bound into a package, the location information includes the plan name, DBRM name, consistency token, and line number.

In both cases, the line number refers to the listing generated by the SQL precompile, **not** the COBOL compile listing.

E The last or abending SQL statement. If there is an associated cursor declaration, it will also be displayed.

If the last SQL statement is not found in the DB2 catalog (for packages or plans which were bound remotely, for example), Detector will display the statement type.

```
*****
* HOST VARIABLES (OUTPUT) *
*****
```

LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000275	10 EMPLOY-EMP	S9(4)	BINARY	+47
000287	20 EMP-IND-EMP	S9(4)	BINARY	+0
000276	10 EMPLOY-FIRSTNAME	X(30)	DISPLAY	*C. EVERETT *
000288	20 EMP-IND-FNAME	S9(4)	BINARY	+0
000277	10 EMPLOY-LASTNAME	X(20)	DISPLAY	*MONSTER *
000289	20 EMP-IND-LNAME	S9(4)	BINARY	+0
000280	10 EMPLOY-DEPT	S9(4)	BINARY	+6
000292	20 EMP-IND-DEPT	S9(4)	BINARY	+0
000281	10 EMPLOY-MANAGER	S9(4)	BINARY	+48
000293	20 EMP-IND-MANAGER	S9(4)	BINARY	+0

```
*****
* HOST VARIABLES (INPUT) *
*****
```

LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000310	10 JOB-EMP	S9(4)	BINARY	+47

```
*****
* SQL COMMUNICATIONS AREA *
*****
```

LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000188	01 SQLCA		GROUP	BLW=0000+000848 (00098950)
000189	05 SQLCAID	X(8)	DISPLAY	*SQLCA *
000190	05 SQLCABC	S9(9)	BINARY	+136
000191	05 SQLCODE	S9(9)	BINARY	+0
000192	05 SQLERRM		GROUP	BLW=0000+000858 (00098960)
000193	49 SQLERRML	S9(4)	BINARY	+0
000194	49 SQLERRMC	X(70)	DISPLAY	SPACES
000195	05 SQLERRP	X(8)	DISPLAY	*DSN *

A The contents of the SQLDAs for the last SQL statement, if applicable (not shown). If the MERGEDB option is ON and a CSL is available for the program, the SQLDA will be mapped. Otherwise, the SQLDA is displayed in dump format.

B The contents of the host variables for the last SQL statement, if applicable. If relevant, the output host variables are displayed following the SQLDA for the output variables and the input host variables are displayed following the SQLDA for the input variables. If the MERGEDB option is ON and a CSL is available for the program, the host variables are mapped. Otherwise, the fully-qualified name of the host variables are displayed, along with the address, length, and data in dump format.

- C

The name and type of DB2 objects referenced by the last SQL statement, if applicable (not shown). If columns in a table or view are referenced, the contents of the referenced columns are displayed.
- D

The contents of the SQLCA. If the MERGEDB option is ON and the CSL is available for the program, data item information such as the data name, picture clause, and usage type is merged with the SQLCA storage. Otherwise, the address, length, and contents of the SQLCA is displayed in dump format.

```
*****
* CONNECTION INFORMATION *
*****

CONNECTION  CONNECTION  CONNECTION  CORRELATION  AUTHORIZATION
MODEA      TYPEB      IDC      IDD      IDE

BACKGROUND  BATCH      BATCH      DB2610      XYZAB02

*****
* APPLICATION PLAN *
*****

PLAN          BIND
NAMEF      DATE AND TIME      BOUND BY

OPT2DEMO 23 FEB,2001 13.42.28 XYZAB02

      BIND OPTIONS:G
      ACQUIRE(USE), CACHESIZE(1024), CURRENTDATA(YES), CURRENTSERVER(),
      DBPROTOCOL(DRDA), NODEFER(PREPARE), DEGREE(1), DISCONNECT(EXPLICIT),
      DYNAMICRULES(RUN), EXPLAIN(NO),
      FUNCTIONTS(2001-02-23-13.42.28.241736), GROUP MEMBER(),
      ISOLATION(CS), KEEPYNAMIC(NO), OPERATIVE(YES), OPTHINT(),
      OWNER(XYZAB02 ), PATHSCHEMAS(), QUALIFIER(WWDBLAB ),
      RELEASE(DEALLOCATE), NOREOPT(VARS), SQLRULES(DB2), VALID(YES),
      VALIDATE(RUN)

      OBJECT DEPENDENCIES:H
      TYPE          NAME          DB/AUTH-ID
      ----          -
      TABLE        DIVISN        WWDBLAB
      TABLE SPACE  DIVISN        WWDBLAB
      TABLE        DEPART        WWDBLAB
      TABLE SPACE  DEPART        WWDBLAB
      TABLE        EMPLOY        WWDBLAB
      TABLE SPACE  EMPLOY        WWDBLAB
```

The next page displays the connection and plan information, such as:

- A

The connection mode.
- B

The connection type.
- C

The connection ID.
- D

The correlation ID.
- E

The authorization ID.
- F

The application plan name, bind date and time and the authorization ID.

G The bind options.

H The names, types and authorization ID of the objects on which the plan is dependent. (If the object is a tablespace, its database is displayed instead of the authorization ID.)

```
*****
* PACKAGE SUMMARY FOR PLAN OPT2DEMO *
*****
```

PACKAGE ^A COLLECTION	PACKAGE ^B NAME	BIND ^C DATE AND TIME	PRE-COMPILE ^D DATE AND TIME	CONSISTENCY ^E TOKEN
OPTIIB2DEMO	DB2ADD	23 FEB,2001 13.42.26 02	FEB,2001 12.22.10	16AACCF206D23118

DBRM LIBRARY:^F
XYZAB02.DBRMLIB

BIND OPTIONS:^G
 CREATOR(XYZAB02), DBPROTOCOL(DRDA), CURRENTDATA(C), DEGREE(1),
 DYNAMICRULES(RUN), EXPLAIN(NO),
 FUNCTIONTS(2001-02-23-13.42.26.625172), GROUP MEMBER(),
 ISOLATION(CS), KEEP DYNAMIC(NO), OPERATIVE(YES), OPTHINT(),
 OWNER(XYZAB02), PATHSCHEMAS(), QUALIFIER(WWDBLAB),
 RELEASE(DEALLOCATE), REMOTE(NO), NOREOPT(VARS), SQLERROR(NOPACKAGE),
 TYPE(BINDPKG), VALID(YES), VALIDATE(RUN)

PRE-COMPILE OPTIONS:^H
 APOST, DEC(31), NOGRAPHIC, HOST(IBMCOB), NOTKATAKANA, PERIOD,
 VERSION()

OBJECT DEPENDENCIES:^I

TYPE	NAME	DB/AUTH-ID
----	----	-----
TABLE SPACE	DEPART	WWDBLAB
TABLE	DEPART	WWDBLAB
TABLE SPACE	DIVISN	WWDBLAB
TABLE	DIVISN	WWDBLAB
TABLE SPACE	EMPLOY	WWDBLAB
TABLE	EMPLOY	WWDBLAB

The package information is then displayed for each package in the plan. If the DB2ACTIV option is specified, package information is displayed only for the packages that are active at the time of the abend. The package information consists of:

A The collection ID.

B The package name.

C The bind date and time.

D The precompile date and time.

E The consistency token.

F The library where the DBRM resides.

G The bind options.

- H** The precompile options.
- I** The names, types and authorization IDs of the DB2 objects on which the package is dependent. (If the object is a table space, its database name is displayed instead of its authorization ID.)

* DBRM SUMMARY FOR PLAN OPT2DEMO *		

A	B	C
DBRM	PRE-COMPILE	CONSISTENCY
NAME	DATE AND TIME	TOKEN
DB2DEL	02 FEB,2001 12.22.17	16AACCF3012F55D9
DBRM LIBRARY: D		
XYZAB02.DBRMLIB		
PRE-COMPILE OPTIONS: E		
APOST, DEC(31), NOGRAPHIC, HOST(IBMCOB), NOTKATAKANA, PERIOD,		
VERSION()		
DB2UPD	02 FEB,2001 12.22.26	16AACCF40189869F
DBRM LIBRARY:		
XYZAB02.DBRMLIB		
PRE-COMPILE OPTIONS:		
APOST, DEC(31), NOGRAPHIC, HOST(IBMCOB), NOTKATAKANA, PERIOD,		
VERSION()		

The DBRM information is displayed for DBRMs that are not bound into a package within the plan. If the DB2ACTIV option is specified, DBRM information is displayed only for the DBRMs that are active at the time of the abend. The DBRM information consists of:

- A** The DBRM name.
- B** The precompile date and time.
- C** The consistency token.
- D** The library where the DBRM resides.
- E** The precompile options.

CA-IDMS Report

The CA-IDMS portion of the Detector report is produced whenever a program abends that accesses a CA-IDMS database using CA-IDMS Version 12.0 or above. The reports are generated based upon the SYSIDMS parameters specified. Documentation for these parameters can be found in the *CA-IDMS Features Summary guide*.

COMPUTER ASSOCIATES CA-OPTIMIZER	7.0 0103ARX00	EXTENDED DETECTOR REPORT	PAGE 14 23 JUN 2001 14.38.26
JOB: IDMSTEST	STEP: RUN	PROGRAM CA02IDMS	
***** * CA-IDMS REPORT * *****			
CA-IDMS abend trace	Tape Genlevel: E0GJ3M ^A	Release: 1400	
User=XYZAB02 ^B	Batch Local Job		
DBNODE=	DBNAME=EMPDEMO	DICTNODE=	DICTNAME= ^C
***** * SYSIDMS parms * *****			
^D	DMCL=R140DMCL		
	DBNAME=EMPDEMO		
	ABENDTRACE=ON		
	ABENDTRACE_ENTRIES=255		
	ABENDTRACE_VIBSNAP=ON		
	ABENDTRACE_SUBSCHEMA_DISPLAY=ON		
	ECHO=ON		
***** * DML trace for subschema=EMPSS01 * ^E *****			
^F			
VERB=59	BIND SUBSCHEMA-->EMPSS01	DBNAME=EMPDEMO	PROGRAM=CA02IDMS
VERB=48	BIND Record	REC-->STRUCTURE	ADDR=00041370
VERB=48	BIND Record	REC-->SKILL	ADDR=00041378
VERB=48	BIND Record	REC-->OFFICE	ADDR=000413C8
VERB=48	BIND Record	REC-->NON-HOSP-CLAIM	ADDR=00041418
VERB=48	BIND Record	REC-->JOB	ADDR=00041808
VERB=48	BIND Record	REC-->INSURANCE-PLAN	ADDR=00041930
VERB=48	BIND Record	REC-->HOSPITAL-CLAIM	ADDR=000419B8
VERB=48	BIND Record	REC-->EXPERTISE	ADDR=00041AE0
VERB=48	BIND Record	REC-->EMPOSITION	ADDR=00041AE8
VERB=48	BIND Record	REC-->EMPLOYEE	ADDR=00041B08
VERB=48	BIND Record	REC-->DEPARTMENT	ADDR=00041B80
VERB=48	BIND Record	REC-->DENTAL-CLAIM	ADDR=00041B88
VERB=48	BIND Record	REC-->COVERAGE	ADDR=00041F60
VERB=37	READY Area Retrieval	AREA->*** ALL AREAS ***	
VERB=32	OBTAIN CALC	REC-->DEPARTMENT	
VERB=19	OBTAIN First Record in Area	REC-->EMPLOYEE	AREA->EMP-DEMO-REGION
VERB=11	OBTAIN Next Record in Area	REC-->EMPLOYEE	AREA->EMP-DEMO-REGION
VERB=11	OBTAIN Next Record in Area	REC-->EMPLOYEE	AREA->EMP-DEMO-REGION
		Caller=CA02IDMS	DMLSEQ=000001 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000002 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000003 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000004 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000005 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000006 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000007 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000008 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000009 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000010 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000011 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000012 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000013 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000014 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000015 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000016 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000017 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000018 *** IDMS
		Caller=CA02IDMS	DMLSEQ=000018 *** IDMS

The first page of the CA-IDMS report displays the following information:

- ^A The genlevel and version of CA-IDMS.
- ^B The user ID that was used to submit the job and the job type.
- ^C The DBNODE, DBNAME, DICTNODE, and DICTNAME that were used.
- ^D The SYSIDMS parameters that were used.
- ^E The subschema name.



A trace of DML verbs for the subschema (the number of commands displayed in the trace is a user-specifiable option).

```
*****
* Currencies for subschema=EMPSS01 *
* Compiled: 2000-11-28 10.19.03 *
*****

Current DBKEY=X'01250501' (75013:1)      For Area EMP-DEMO-REGION
Current DBKEY=X'01259801' (75160:1)      For Area ORG-DEMO-REGION

Current DBKEY=X'01250501' (75013:1)      For Record EMPLOYEE
Current DBKEY=X'01259801' (75160:1)      For Record DEPARTMENT

Current DBKEY=X'01250501' (75013:1)      For Set DEPT-EMPLOYEE
Owner DBKEY=X'01259E01' (75166:1)        For Set DEPT-EMPLOYEE
Prior DBKEY=X'01252001' (75040:1)        For Set DEPT-EMPLOYEE
Next DBKEY=X'01259E01' (75166:1)        For Set DEPT-EMPLOYEE

Current DBKEY=X'01250501' (75013:1)      For Set EMP-COVERAGE
Owner DBKEY=X'01250501' (75013:1)        For Set EMP-COVERAGE
Prior DBKEY=X'01256401' (75108:1)        For Set EMP-COVERAGE
Next DBKEY=X'01256401' (75108:1)        For Set EMP-COVERAGE

Current DBKEY=X'01250501' (75013:1)      For Set EMP-EXPERTISE
Owner DBKEY=X'01250501' (75013:1)        For Set EMP-EXPERTISE
Prior DBKEY=X'01250503' (75013:3)        For Set EMP-EXPERTISE
Next DBKEY=X'01250504' (75013:4)        For Set EMP-EXPERTISE

Current DBKEY=X'0124FA04' (75002:4)      For Set EMP-NAME-NDX
IXCUR DBKEY=X'01250501' (75013:1)      For Set EMP-NAME-NDX
```

Currencies for all areas, records, and sets in the subschema at the time of the abend are displayed.

Record Name	Record Size	Bind Buff	'Prev' Buff	'Curr' Buff
A				
STRUCTURE	8	00041370	* none *	* none *
SKILL	76	00041378	* none *	* none *
OFFICE	76	000413C8	* none *	* none *
NON-HOSP-CLAIM	1008	00041418	* none *	* none *
JOB	296	00041808	* none *	* none *
INSURANCE-PLAN	132	00041930	* none *	* none *
HOSPITAL-CLAIM	292	000419B8	* none *	* none *
EXPERTISE	8	00041AE0	* none *	* none *
EMPOSITION	28	00041AE8	* none *	* none *
EMPLOYEE	116	00041B08	0D8AB8D0	0D8AB944
DEPARTMENT	56	00041B80	0D8AB9C8	0D8ABA00
DENTAL-CLAIM	932	00041BB8	* none *	* none *
COVERAGE	16	00041F60	* none *	* none *

* Previous image for record DEPARTMENT *				

ADDRESS	0D8AB9C8	LENGTH 56 (X'00000038')		
	OFFSET			
0D8AB9C8 000000	F2F0F0F0	40404040 40404040 40404040	*2000	*
0D8AB9D8 000010	40404040	40404040 40404040 40404040	*	*
0D8AB9E8 000020	40404040	40404040 40404040 40404040	*	*
0D8AB9F8 000030	40404040	40404040	*	*

* Current image for record DEPARTMENT *				

ADDRESS	0D8ABA00	LENGTH 56 (X'00000038')		
	OFFSET			
0D8ABA00 000000	F2F0F0F0	C1C3C3D6 E4D5E3C9 D5C740C1	*2000ACCOUNTING A*	
0D8ABA10 000010	D5C440D7	C1E8D9D6 D3D34040 40404040	*ND PAYROLL	*
0D8ABA20 000020	40404040	40404040 40404040 40404040	*	*
0D8ABA30 000030	40F0F0F1	F1000000	* 0011...	

* Bind image for record DEPARTMENT *				

B				
LINE#	LEVEL/FIELD NAME	PICTURE CLAUSE	USAGE	VALUE/LOCATION
000388	01 DEPARTMENT		GROUP	BLW=0000+000AF8 (00041B80)
000389	02 DEPT-ID-0410	9(4)	DISPLAY	9999
000390	02 DEPT-NAME-0410	X(45)	DISPLAY	*ACCOUNTING AND PAYROLL*
			(+000032)	*
000391	02 DEPT-HEAD-ID-0410	9(4)	DISPLAY	11
000392	02 FILLER	XXX	DISPLAY	

The next display shows:

A

Bind buffer size and address for each record in the subschema.

B

Current and previous buffer addresses for the records accessed.



Current, Previous, and Bind record images for the records accessed. If the MERGEDB option is ON, the record is defined to the most active COBOL program, and a CSL is available for the program, the record image is mapped. Otherwise, the address, length and contents of the record is displayed in dump format.

```
*****
* CA-IDMS VIB snap *
*****
  A
ADDRESS 0D8A85E0      LENGTH 5720 (X'00001658')
      OFFSET
0D8A85E0 000000      E5C2F5F0 00000005 0D886390 0D886416      *VB50....h...h..*
0D8A85F0 000010      0D8864E8 0D8864FA 0D886436 00000000      *.h.Y.h...h....*
0D8A8600 000020      00000000 00000000 00000000 00000001      *......*
0D8A8610 000030      0D886E60 0007C7F8 0D8A9DD4 0D8AA828      *.h>-.G8...M..y.*
0D8A8620 000040      00000000 00000000 00000000 00000000      *......*
.
.
.
0D8A9C30 001650      00000000 00000000      *......*

*****
* CA-IDMS VARS snap *
*****
  B
ADDRESS 0D8A9DD4      LENGTH 2644 (X'00000A54')
      OFFSET
0D8A9DD4 000000      E5C1D9E2 40404040 C5D4D7E2 E2F0F140      *VARS EMPSS01 *
0D8A9DE4 000010      00000000 0D8D6D1C 00000000 00000000      *......*
0D8A9DF4 000020      0D8D6D1C 00000000 04038003 01250501      *......*
0D8A9E04 000030      01250501 0007D2B0 0D8A85E0 0D88E1F0      *......K...e\h.0*
0D8A9E14 000040      00000000 00000000 00000000 00000000      *......*
0D8A9E24 000050      FFFFFFFF 000124F9 0001252A 0001252A      *......9.....*
.
.
.
0D8AA824 000A50      00000000      *....*
```

The next display show:



A Snap of the CA-IDMS VIB (Variable IB50 for subschema).



A Snap of CA-IDMS VARS.

SUBSCHEMA=EMPSS01						
Compiled=2000-11-28 10.19.03 ^A						
Subschema Structure is Network and Unbound ^B						
Area Name	Segment					
-----	-----					
^C						
EMP-DEMO-REGION	n/a					
INS-DEMO-REGION	n/a					
ORG-DEMO-REGION	n/a					
Record Name	Stored	Rec ID	Area Name	Data Length	Prefix Length	Procedures
-----	-----	-----	-----	-----	-----	-----
COVERAGE	VIA	400	INS-DEMO-REGION	16	20	
DENTAL-CLAIM	VIA	405	INS-DEMO-REGION	936	12	
DEPARTMENT	CALC	410	ORG-DEMO-REGION	56	16	
EMPLOYEE	CALC	415	EMP-DEMO-REGION	116	72	
EMPOSITION	VIA	420	EMP-DEMO-REGION	28	24	
EXPERTISE	VIA	425	EMP-DEMO-REGION	8	20	
HOSPITAL-CLAIM	VIA	430	INS-DEMO-REGION	292	8	
INSURANCE-PLAN	CALC	435	INS-DEMO-REGION	132	8	
JOB	CALC	440	ORG-DEMO-REGION	300	24	IDMSCOMP Before STORE IDMSCOMP Before MODIFY IDMSDCOM After GET
Chain Sorted-> CALC						
			Next,Prior			
Owner ----->	SR1		Next=00 Prior=04			
Member ----->	SR6		Next=00 Prior=04			
Member ----->	SR7		Next=00 Prior=04			
	Ckey	Offset=16	Length=16	Data	Type=Character	
Member ----->	SKILL			Next=00 Prior=04		
	Ckey	Offset=20	Length=4	Data	Type=Numeric (Unsigned)	
Member ----->	OFFICE			Next=00 Prior=04		
	Ckey	Offset=16	Length=3	Data	Type=Character	
Member ----->	JOB			Next=00 Prior=04		
	Ckey	Offset=28	Length=4	Data	Type=Numeric (Unsigned)	
Member ----->	INSURANCE-PLAN			Next=00 Prior=04		
	Ckey	Offset=8	Length=3	Data	Type=Character	
Member ----->	EMPLOYEE			Next=00 Prior=04		
	Ckey	Offset=72	Length=4	Data	Type=Numeric (Unsigned)	
Member ----->	DEPARTMENT			Next=00 Prior=04		
	Ckey	Offset=16	Length=4	Data	Type=Numeric (Unsigned)	

The final pages of the report display:

- ^A The compile date and time of the subschema.
- ^B The subschema structure and whether it is bound or unbound.
- ^C The associated segment for each area, if applicable.
- ^D Detailed information about each record in the subschema including storage mode, record ID, area name, data length, prefix length, and DB procedures.
- ^E Detailed information about each set in the subschema including owner record, member records, set type, and set pointers.

Snap Report

The SNAP feature of Detector produces nearly the same output as the Detector Abend Report. However, since snapshots do not result from an abend, those items that are concerned with the abend in the Detector Abend Report are omitted.

COMPUTER ASSOCIATES	DETECTOR REPORT	PAGE	3
CA-OPTIMIZER 7.0 0106ARX00		15 JUN 2001	14.29.43
JOB: USRID02A STEP: RUN	PROGRAM CAOMDEMO		

* SNAP OF CAOMDEMO *			

COBOL STATEMENT BEFORE SNAP:	PROGRAM-ID CAOMDEMO		
SOURCE LOCATION: PAGE 10 LINE 00426	MLIST 002292		
IF FLAG1 LESS THAN FLAG2			
RELATIVE SNAP 600022B0	SNAP COUNT 1		
RELATIVE ENTRY POINT 00001D6C	ENTRY COUNT 1		

A snapshot consists of the following output:

- The program's address information and registers at the time the SNAP was invoked.
- The contents of the File Section at SNAP time. The format is the same as the format of the Detector Abend Report.
- The Working-Storage Section and the Linkage Section.
- A SNAPped display of the Memory Map.
- The PERFORM Return Address Analysis.
- The last 64 paragraphs executed prior to the SNAP call.

CAIOPTS File Processing Report

The CA-Optimizer Execution Monitor produces this report at the start of application execution, after the Execution Monitor has read and processed the Detector or Analyzer runtime options. The report displays the input options read from the data set defined by the CAIOPTS DD statement, if present. The body of the report is a display of the 80-byte records in the CAIOPTS data set exactly as read by the Execution Monitor.

COMPUTER ASSOCIATES	CAIOPTS FILE PROCESSING	PAGE	1
CA-OPTIMIZER	7.0 0106ARX00	INPUT OPTIONS DISPLAY	15 JUN 2001 14.29.42
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8			
INTERCEPTS 2		00572006	
SNAP 1		00573013	
PFLOW ON		00574012	
PARMCHK ON		00575012	
XCOUNT ON		00576012	
XTIME FIRST		00577013	
XREPORT ON		00578012	
BLOCKS ON		00579027	
DUMP OFF		00579127	
INSTRMAP ALL		00580023	
CAPC015I ANALYZER REPORTER WILL BE CALLED AUTOMATICALLY IF ANY PROGRAMS REQUEST COUNTS AND/OR TIMINGS.			
CAPC012I INPUT OPTION DISPLAY COMPLETE.			

Execution Monitor Summary

This report is produced by the Execution Monitor at termination, just prior to returning to the operating system. It is divided into two major sections.

COMPUTER ASSOCIATES	EXECUTION MONITOR SUMMARY	PAGE	25
CA-OPTIMIZER	7.0 0106ARX00	15 JUN 2001	14.29.52
EXECUTION OPTIONS IN EFFECT:	EXECUTION ON: 05/390		
ACB	ON		
ACTONLY	OFF		
AMB	ON		
AMBL	ON		
ASMINST	ON		
BINDER	OFF		
BINFRMT	DEC		
CEEDUMP	OFF		
CSLDSN			
DB2ACTIV	OFF		
DCB	ON		
DEB	ON		
DUMP	OFF		
EXCEED	NOSTOP		
FILES	ON		
FIRST128	OFF		
GRPADDR	ON		
IMS	NO		
INTERCEPT	2		
IOB	ON		
LINKAGE	ALL		
LOCALSTOR	ALL		
MEMMAP	ON		
MERGEDB	ON		
MRGFILES	ON		
MRGLINK	ON		
MRGLOCAL	ON		
MRGWORK	ON		

MSEAPRE	Abbbb
MSTAE	ON
MTIME	0
OCCURS	1
PARMCHK	ON
PFLOW	ON
PLH	ON
PRTREPT	BOTH
RPL	ON
SAVEAREA	ON
SNAP	1
STIMERM	OFF
UCB	ON
VSAMIDX	ON
WORKSTOR	ALL
XCOUNT	ON
XREPORT	ON
XTIME	FIRST

LEGEND FOR OPTIMIZER OPTIONS:

D=DTECTC=XCOUNT

B	C	D	E	F	G	H	I	J	K
COBOL PROGRAM NAME	COMPILATION DATE AND TIME	COMPILER USED	REL	COUNT SPACE (BYTES)	NUMBER OF TIMES OF TIMES ALLOCATED EXECUTED	NUMBER OF TIMES OF TIMES ALLOCATED EXECUTED	RUN-TIME OPTIONS SPECIFIED AT COMPILE TIME	RUN-TIME OPTIONS SPECIFIED AT EXECUTION TIME	
CAOMDEMO	JUN 15,2001 14.29.26	05/VS	2.4	2048	1	1	,C,D,P		

The Execution Monitor Report provides the following information for every COBOL program running under Detector or Analyzer.

- A** The first section lists, in alphabetical order, the options that were in effect **during this execution** of the application.
- B** The second section shows various attributes of all the optimized COBOL programs encountered by the Execution Monitor during the execution of the application. Entry to any program compiled/optimized with Detector or Analyzer options is listed here.
- C** Compile date, which helps identify the version of the program.
- D** Compile time, which helps identify the version of the program.
- E** The COBOL compiler used for this program.
- F** The version of the COBOL compiler.
- G** The amount of main storage required to store XCOUNT data.
- H** Number of times the Execution Monitor was required to allocate XCOUNT space.
- I** Number of times the program was executed during the run.

- J** Compile time options in effect for the program.
- K** Runtime options in effect for the program.
- L** The operating system under which the application executed is shown as MVS, MVS/XA, MVS/ESA, OS/390, or z/OS.

Detector Facilities

Merged vs. Dumped Data Displays

Detector displays storage areas for different sections of a COBOL program. If a CSL is available for the program, the appropriate merging option is specified, and the storage area can be mapped to a Data Division area, the storage area is displayed in a merged format. Otherwise, the storage area is displayed in dump format.

Merged Display

The merging options are:

MRGDATA	Merge symbolic data information into all reports
MRGFILES	Merge symbolic information into the File Section report
MRGLINK	Merge symbolic information into the Linkage Section report
MRGWORK	Merge symbolic data information into the Working-Storage Section.
MERGEDB	Merge symbolic information for all Data Division sections into the database reports, if applicable.

If the merging options are not specified, they default to ON. The MRGFILES, MRGLINK, MRGWORK, and MERGEDB options can all be overridden by the MRGDATA option.

An example of a merged report follows.

```
*****
* WORKING-STORAGE SECTION *
*****
```

Using Detector 4-47

A merged report contains the following information:

- A** Header describing the report columns.
 - B** The source line number where the data item was defined.
 - C** The level number and name of the data item.
 - D** Any REDEFINES, OCCURS, or PICTURE clause specified on the data item definition.
 - E** The USAGE type of the data item.
 - COMP and COMP-4 items are identified as BINARY
 - COMP-1 items are identified as FLOATING-POINT
 - COMP-2 items are identified as LONG FLOATING-POINT
 - COMP-3 items are identified as PACKED-DECIMAL
 - F** For group items, the base locator and displacement of the item. The address of the item is also identified, in parentheses. (The base locator will print as BL, BLL, or SBL.) If the GRPADDR option is set to OFF, this information is suppressed.
 - G** For elementary items, the value of the item. The value is displayed in a format determined by the USAGE of the data item and the validity of the data.
 For example:
 - DISPLAY data is shown in display format.
 - PACKED-DECIMAL, BINARY, and INDEX data is shown in decimal display format or hexadecimal format, depending on the BINFRMT option.
 - FLOATING-POINT data is shown in hexadecimal format (for example, X'nnnnnnnn').
- If the value of a data item is invalid, a question mark (?) precedes the data value and the data is displayed in hexadecimal format.
- If the address of a data item is invalid, a question mark (?) precedes the data value column. The words 'INVALID ADDRESS' are displayed in the value column, followed by the address, in parentheses.
- H** If the data is too long to fit on the current line (32 bytes for display format, 16 bytes for hexadecimal format) it is continued on the next line. The offset of the continued data (from the beginning of the data item) is displayed, in decimal, on each successive line.

- I** If the COBOL program contains data that does not map to a data item (such as slack bytes or unmerged data items from a COPY...SUPPRESS statement) the string <UNMERGED DATA> is displayed in place of a data name, the length of the data is displayed in place of a picture clause, and the data is displayed in hexadecimal format.
- J** If the value specified in the OCCURS option is less than the number of occurrences defined to the table, the first n occurrences of the table is displayed in merged format (where n is the OCCURS option value) and the rest of the table is displayed in dump format. The OCCURS option was set to the default value of 1 for this report. The maximum value for the OCCURS option is 16777215.
- K** If an INDEXED clause is specified on an OCCURS data item, the index name, cell number, and current value is displayed on the line following the OCCURS data item. If the current index value cannot be determined, 'N/A' is displayed in the value column.

When displaying a variable-length table, if the size of the table has not yet been defined or cannot be determined, the maximum table size is used.

Dumped Display

An example of a dumped report follows.

COMPUTER ASSOCIATES		DETECTOR REPORT		PAGE 4					
CA-OPTIMIZER		7.0 0106ARX00		18 JUL 2001 15.20.23					
JOB: USRID02A		STEP: RUN	PROGRAM CA0MDemo						

* WORKING-STORAGE SECTION *									

C	A	B							
BL=0002	00008098	LENGTH 00001000	(d'4096')						
OFFSET									
D			E	F					
+000000	E3C8C9E2	40D7D9D6	C7D9C1D4	40C9E240	D7D9D6D7	*THIS PROGRAM IS PROPRIETARY PROP*			
+000020	C5D9E3E8	40D6C640	C3D6D4D7	E4E3C5D9	40C1E2E2	D6C3C9C1	E3C5E240	C9D5E3C5	*ERTY OF COMPUTER ASSOCIATES INTE*
+000040	D9D5C1E3	C9D6D5C1	D36B40C9	D5C34B40	C3D6D7E8	D9C9C7C8	E3404DC3	5D40F1F9	*RNATIONAL, INC. COPYRIGHT (C) 19*
+000060	F8F36B40	F1F9F9F7	40C2E840	C3D6D4D7	E4E3C5D9	40C1E2E2	D6C3C9C1	E3C5E240	*83, 1997 BY COMPUTER ASSOCIATES *
+000080	C9D5E3C5	D9D5C1E3	C9D6D5C1	D36B40C9	D5C34B40	40404040	40404040	40404040	*INTERNATIONAL, INC. *
+0000A0	F0000000	1CF0F0F0	00000000	00000000	00000000	00000000	00000000	00000000	*0...000.....*
+0000C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*
+0000E0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*
+000100	F0F0F0F0	F0C2F0F0	F0F0F0C4	F0F0F0F0	F0C3F0F0	F0F0F0C6	F0F0F1F8	F6C5F0F0	*00000B00000D00000C00000F00186E00*
+000120	F1F7F7D6	F0F0F0F7	F0C8F0F0	F0F1F5C0	00010001	00010001	000E001E	00320001	*177000070H00015.....*
+000140	00000001	18E2739B	00000001	00000000	00000000	00000000	00000000	00000000	*.....S.....*
+000160	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*
LINES 00000180-00000240 SAME AS ABOVE									
+000260	00000000	00000000	00000000	00000000	F0F0F0F0	F0F0C9F0	F0F0F0F0	F0C9F0F0	*.....000000I000000I00*
+000280	F0F0F0F0	C6F0F0F0	F0F0F0D3	F0F0F0F0	F6F4C7F0	F0F1F1F1	F5C8F0F0	F0F0F0F0	*0000F00000L000064G001115H000000*
+0002A0	C6F0F0F0	F0F0F1C4	00000000	00000000	00000000	00000000	00000000	00000000	*F000001D.....*
+0002C0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*

A dumped report contains the following information.

- A** Address of the dumped storage.

- B** Length of the dumped storage in hexadecimal and decimal.
- C** The base locator cell associated with the dumped storage. (The base locator will print as BL, BLL, or SBL.)
- D** The displacement of the storage from the specified base locator, in hexadecimal.
- E** The data in hexadecimal format.
- F** The data in display format.

Subscript Range Checking (SUBRNGCK)

The Subscript Range Checking feature of CA-Optimizer performs tests in the object code to verify the validity of subscripts and indexes before they are used. Invoke this feature by specifying the SUBRNGCK option in the CA-Optimizer option list at compile time.

Optimizer tests each subscript and index used in a COBOL program, compiled and optimized with the SUBRNGCK option, before it is used to reference the subscripted data item. A subscript value is considered valid if it is greater than zero and less than or equal to the number of occurrences specified for the data item in its OCCURS clause. An index value is considered valid if it is greater than or equal to zero, and less than or equal to the displacement corresponding to the number of occurrences specified for the data item's OCCURS clause. If the data item has an OCCURS DEPENDING ON clause, the current value of the depending-on variable is used for the upper bound test. If the depending-on variable does not contain a value prior to the reference to the subscripted or indexed table, then SUBRNGCK considers the current size of the table as undefined and flags an error referencing the subscripted or indexed table.

If a subscript or index is found to be invalid, the program terminates with an S0C1 system abend code. Under Detector, the S0C1 abend is intercepted and a special abend report is produced specifically for subscript range violations. Detector translates the S0C1 abend into a U1125 abend which is uniquely associated with subscript range violations. This feature stops the program from executing when the subscript is invalid before it destroys enough data to cause some other error.

Note: Only subscripts that are calculated at runtime are checked. Literal subscripts are not checked as these are resolved at compile time. The optimized program can only perform range checking tests on data items which follow ANSI COBOL rules. The optimized program only checks coincident tables for subscripts less than one because the upper range cannot be determined accurately. Coincident tables are subscripted data names whose first elements have the same base and displacement in the data map.

Detection of Uninitialized Working-Storage (WSINIT)

Many times there are problems in execution because data items are used before they are assigned a value. Often it is unclear whether the data item was never given a value or whether it was assigned a value of zero. Optimizer removes this confusion by causing all Working-Storage variables which are not initialized by a VALUE clause to start execution with each byte set to X'50'. If such a data item is printed, it appears as a string of ampersands (&). If you try to perform arithmetic operations on uninitialized data items other than type COMP, (binary), the program abends with an S0C7 abend code. Under Detector, when a S0C7 abend occurs, Detector analyzes the cause of the abend. If WSINIT was specified at compile time and the cause was due to uninitialized Working-Storage, Detector indicates this on the first page of the Detector Abend Report. This feature makes it much more difficult to use uninitialized data items. To invoke this feature, specify WSINIT in the CA-Optimizer option list at compile time.

Detection of Invalid Record Area References (FDCHECK)

When a COBOL program moves data into the record area of an unopened file or into an input file that is at end-of-file, several types of errors can occur. If the record area is not within the program (this is the usual case for files in the program which do not use a READ INTO option on a READ verb, or a WRITE FROM option on the WRITE verb), the destination of the data is unpredictable. Some possibilities are an S0C4 abend or destruction of actual program code resulting in a later S0C1 abend. Optimizer ensures that any attempt to reference the record of an unopened file or an input file at end-of-file, results in an abend.

For example, if a data item in a record area of an unopened file is used in an IF statement, the Detector FDCHECK facility intercepts this as an invalid record area reference. Under Detector, the abend is intercepted and a special abend report is produced specifically for invalid record area references. Detector translates the violation into a U1128 abend which is uniquely associated with FDCHECK. The report includes not only the source statement that caused the error, but also the FD whose record area is being referenced. To invoke this feature, specify FDCHECK in the CA-Optimizer option list at compile time.

Parameter List Validity Checking (PARMCHK)

Parameter List Validity Checking is a Detector feature that performs these functions:

- Compares the number of parameters that are passed to an optimized program with the number of parameters the optimized program is expecting. When a count violation occurs, a CAPD119W error message is produced.

- Checks the parameters list addresses to ensure references to them will not cause S0C4 abends at a later time. In the case of an address violation, a CAPD120W error message is produced.
- Issues a message identifying the calling and called programs.
- Reduces the addressing exceptions resulting from incorrect parameters lists being passed between programs. PARMCHK does not terminate the program execution when a violation occurs, but simply prints a warning message. Up to 16 occurrences of parameters list violations will be reported, after which the mechanism is automatically shut off for the duration of the application execution. To invoke this feature, specify PARMCHK at compile time.

Note: The PARMCHK execution time default for CA-Optimizer is ON. PARMCHK processing automatically occurs for any optimized OS/VS COBOL program if Detector is active.

Paragraph Flow Analysis (PFLOW)

PFLOW is optional. However, it is recommended that you use it in all debugging runs.

At compile time, PFLOW requests a Detector Paragraph Flow Report of the last 64 paragraphs executed. It follows the Detector Abend or SNAP report. You may override it by specifying the PFLOW OFF option at execution time. PFLOW turns on DTECT. If NODTECT, CICS, DCI, HOGAN, IDMSDC, SHADOW, or TPMON is specified, PFLOW is suppressed.

When used during execution time, it generates an optional section of the Detector Abend Report which lists the source name and address of the last 64 paragraphs executed up to the time of abend.

SNAP

SNAP is controlled by inserting CALLs to CAODSNAP at points you choose in your program and by the SNAP option at execution time. For programmers familiar with previous versions of CA-Optimizer, CALLs to CPXMSNAP are still valid. Remember, SNAP defines the sum of all SNAP CALLs taken for all programs in the application.

INTERCEPTS

Use of the INTERCEPTS option provides you with all the information you need to correct a data exception or decimal divide exception. After Detector generates the information for the Intercepts Report, it corrects the offending data item. In the case of a data exception, it plugs the item in error with the value 1. With a decimal divide exception, Detector changes the divisor to 1.

Detector corrects the bad data to prevent multiple intercepts for the same item. After correcting the data, Detector reexecutes the abending instruction so that predictions can be made concerning the outcome.

Usage Considerations

Detector with IMS

Detector can be used in IMS environments. A complete discussion on how to use Detector under IMS can be found in *CA-Optimizer Getting Started*.

Dynamic Allocation

This subsection covers dynamic allocation of DD statements for both Detector and Analyzer.

If you do not provide the required DD statements, the Execution Monitor uses the dynamic allocation facilities of the operating system to allocate them for you.

After reviewing the options' specifications used here as defaults, you ordinarily set up your application such that the data sets are dynamically allocated for you. If any of the options for any data set conflict with your requirements, simply override the specification by supplying a DD statement (or for a CLIST, an ALLOCATE statement).

The chart below describes the dynamic allocation specifications used in standard OS JCL keyword/value notation.

File	Allocation Specifications	Usage Notes
CAITIMSV	DISP=(NEW,DELETE,DELETE), DSN=&&TIMSV., SPACE=(TRK,(30,30),RLSE), UNIT=SYSDA	See Note 1. See Notes 1, 2.
CAIPRINT	SYSOUT=*	See Note 3.
CAIPRNT1	SYSOUT=*	
CAIPRNT2	SYSOUT=*	
SYSUDUMP	SYSOUT=*	
CAICSL	DISP=SHR,DSN=aaaaaaaa	See Note 4.

Notes

1. This data set is allocated if:
 - The DD statement for CAITIMSV is required (for counts or timings),
 - XREPORT ON is in effect, and
 - The DD statement is omitted.

The Analyzer opens this file for output, writes the counts or timings data to it during the application execution, and closes the file. Analyzer Reporter then opens the file for input, processes the counts and timings data records, and produces the Analyzer reports. When the job step ends in any fashion, the data set is deleted.
2. The amount of space required is a function of many variables. For dynamic allocation, the space allocated has been set high enough to accommodate any application.
3. The SYSTOUT class used during dynamic allocation is the asterisk (*). This is the default SYSOUT class specified in the JCL MSGCLASS. If you require a different SYSOUT class, consult your Systems Programmer (who can modify the default SYSOUT class in the CA-Optimizer installation defaults CSECT), or supply your own DD or ALLOCATE statement.
4. The data set name used during dynamic allocation is supplied as follows:
 - From the value of the CSLDSN keyword option in the CAIOPTS file, or
 - If CSLDSN is omitted, from the default option CSECT shipped with the product and maintained by your systems programmer.

In either of the above cases, the data set must be properly cataloged, or allocation will fail.

Multimodule Reporting

Detector supports any combination of programs that OS/VS COBOL's Runtime Environment supports, including: Optimized and non-optimized, IBM optimized and non-IBM optimized.

In a multimodule system, the programs in a calling sequence form a chain of active programs. Detector reports on any subsequently called modules that abend, whether they were optimized with any of the Detector features. The full Detector Abend Report prints for the abending program if it was running under Detector and was optimized with DTECT ON.

If the abending program was COBOL and not compiled with DTECT ON, or if it was not a COBOL program, the following information is produced:

- Diagnostics for the type of abend
- The absolute interrupt address
- The relative interrupt address
- Specifics about the offending instruction
- A module call sequence
- The register contents at abend
- A display of the program CSECT

After the information for the abending program is reported, a Detector Abend Report is generated for every other active Detector program in the calling sequence that was optimized with DTECT ON (unless ACTONLY ON was specified, which causes only the most active COBOL program optimized with DTECT to be reported). These reports for the other active programs in the calling sequence contain the transfer address, but do not contain any abend information such as interrupt addresses or details about offending instructions.

Restrictions

Detector cannot operate under CICS, HOGAN, IDMSDC, SHADOW, CMS or when multitasking is used, with the exception of IMS/VS message processing regions.

You must use standard IBM-type program linkage conventions to use Detector effectively. For information on setting up Detector for use in IMS environments, see the chapter "Special Environments" in *CA-Optimizer Getting Started*.

In an operating environment where your inhouse systems are responsible for catching abends, it is possible to obtain the full Detector Abend Report. See the *CA-Optimizer Getting Started*, "Special Usage Considerations" chapter.

Using Extended Detector

The Extended Detector component of CA-Optimizer operates both in the production and test environments as a productivity tool for COBOL programmers. Like Detector, it provides you with valuable diagnostic information. Unlike Detector, however, there are no compile time requirements or execution time overhead. Using the Extended Detector Support requires no changes to production JCL.

Once installed, Extended Detector Support is automatically available to every user. Exclusion criteria are determined at installation, so that a standard dump without Detector reports is always available if desired.

Extended Detector provides reports when a system dump has been requested as the result of an abend or when a SNAP request has been made by a call to CAODSNAP. Until that time, there is no intervention of any kind as there would be when using the standard Detector environment.

Extended Detector provides reports for all types of programs, COBOL or non-COBOL, CA-optimized or non-CA-optimized. The level of reporting, as well as the types of reports produced, is completely controlled by the installation.

Extended Detector Options

The following chart provides a listing of all Extended Detector options available to you at execution time. Each option has its own default value, established at installation. To override any of these options, include a CAIOPTS DD statement in the JCL of the executing job step. This allows the user, if necessary, to control the types of output reports produced.

Extended Detector Options					
Execution Time			View Time		
Standard Options		VSAM Control Blocks		Formatting Options	
BLLMAX	PRTLIB	ACB	PLH	ASMINST	
COBONLY	PRTREPT	AMB	RPL	BINFRMT	
CSLDSN	RPTSZ80	AMBL	VSAMIDX	FILES	
DB2ACTIV	SNAP			FIRST128	
DUMP				GRPADDR	
Formatting Options		Data Management Control Blocks		Formatting Options	
ASMINST	MERGEDB	BLOCKS	IOB	LINECNT	
BINFRMT	MRGDATA	DCB	UCB	LINKAGE	
FILES	MRGFILES	DEB		LOCALSTOR	
FIRST128	MRGLINK	Log and Notify Options		MEMMAP	
GRPADDR	MRGLOCAL	LOGROS	LOGUNI	MERGEDB	
LINECNT	MRGWORK	LOGTSO	WTO	MRGFILES	
LINKAGE	OCCURS			MRGLINK	
LOCALSTOR	SAVEAREA			MRGLOCAL	
MEMMAP	WORKSTOR			MRGWORK	
				OCCURS	
				SAVEAREA	
				WORKSTOR	

Extended Detector also supports the following compile time options:

FDCHECK	Detects invalid references to file record areas.
SUBRNGCK	Detects out of range subscripting or indexing.
WSINIT	Detects data exceptions resulting from uninitialized data in WORKING-STORAGE.

JCL Requirements At Execution Time

By default, Extended Detector Support does not require any additional JCL at execution time. All CA-Optimizer Runtime Library modules required for the use of Extended Detector Support should be made available at installation (see *CA-Optimizer Getting Started*) through LPA or LINKLIST.

Initialize Extended Detector Support using the CAIRIM component of Unicenter TNG Framework for OS/390 services when MVS is initialized. Your job step must include a SYSUDUMP DD statement if Extended Detector is being used.

The repository used by Extended Detector has a default name of CAIPRTLb, which may be customized at installation time. This file is dynamically allocated, if not included in the JCL, using the data set name specified at installation time. User exit CAOCUPRT may be used to select a repository library based upon criteria such as the job name or userid. The user may also override the default repository library by specifying the PRTLb execution option.

The report file (CAIPRINT) used by Extended Detector is dynamically allocated unless you wish to include it in the JCL. The CSL file (CAICSL) used by Extended Detector for symbolic reporting is also dynamically allocated if it is not included in the JCL. The name of the CSL file can be established at installation time or overridden at execution time using the CSLDSN option.

The options file (CAIOPTS) used by Extended Detector must be included in the JCL if you wish to override any installation options.

Following is an example of execution JCL that includes all of the files used by Extended Detector.

```
//STEP2   EXEC  PGM=your-program-name
//STEPLIB DD   DISP=SHR,DSN=user-load-library-name
//        DD   DISP=SHR,DSN=COBOL-load-library-name
//        DD   DISP=SHR,DSN=OPTIMIZER-load-library-name
//SYSUDUMP DD   SYSOUT=*
/*
// your program's DD requirements here
/*
//CAICSL   DD   DISP=SHR,DSN=OPTIMIZER-CSL-library-name
//CAIPRINT DD   SYSOUT=*
//CAIOPTS  DD   *
/*
```

Extended Detector Reports

Your Extended Detector reports are almost identical to the standard Detector reports. The only reports provided are the Abend and SNAP Reports, including each of the following:

For COBOL programs:

- Abend Report
- Module Call Sequence Report
- File Section Display
- Working-Storage Section Display
- Linkage Section Display
- Registers At Abend Report
- Memory Map Display

- Open Files Report
- Save Area Trace Report
- CAIOPTS File Processing Report
- SNAP Report

For non-COBOL programs:

- Abend Report
- Module Call Sequence Report
- Registers at Abend Report
- Open Files Report
- Save Area Trace Report
- CAIOPTS File Processing Report
- SNAP Report

For examples of each of these reports, see the PRTREPT option in the chapter “Using Options” in this guide.

Warning! *Each call to CAODSNAP generates a separate SNAP report in your central VSAM repository. If your program contains multiple SNAP calls or a SNAP call within a loop, you can write the output to the CAIPRINT DD alone, to avoid flooding the repository. See the PRTREPT option in the chapter “Using Options” in this guide.*

Hierarchy of Information

Unlike standard Detector, the Extended Detector Support provides reports even when the job step does not include any CA-optimized COBOL programs. Therefore, the information available in the report varies depending on the type of program.

If a CA-optimized COBOL program is compiled with the DTECT option, standard Detector provides the complete report as usual. If a CA-optimized COBOL program abends or SNAPS, but standard Detector was not used, Extended Detector provides the report. In this case, all of the Extended Detector reports are available.

If the CSL, MLIST, and MDMAP options are used at compile time and the correct CSL file is available, the report contains complete symbolic support.

For non-CA-optimized COBOL programs, a CSL can be created from the COBOL compiler listing using the COBOL postprocessor. (See the chapter “Using the COBOL Postprocessor” in this guide.) If a CSL is available, the Extended Detector report will contain complete symbolic support. If a non-COBOL program abends, only those reports that apply to non-COBOL programs are produced.

Usage Considerations

Dynamic Allocation

The following chart shows the allocation specifications for the files which may be dynamically allocated by Extended Detector:

File	Allocation Specifications
CAIPRINT	SYSOUT=* (See Note 1.)
CAICSL	DISP=SHR,DSN=csl.dsname (See Note 2.)

Notes:

1. The asterisk (*) is the default SYSOUT class, which results in the same class as your MSGCLASS in the JCL.

This may be changed at installation time to any other class desired using the SYSOUT= parameter when assembling CAOETABL.
2. The data set name allocated is obtained from the runtime defaults table, CAOUDFRX, or an override in the CAIOPTS file.

Restrictions

Extended Detector Support does not provide any reporting for COBOL programs whichabend under CICS, HOGAN, IDMSDC, or SHADOW. Extended Detector Support is not supported under CMS.

For effective use of Extended Detector Support, we suggest that your applications use standard IBM-type program linkage conventions whenever possible.

Extended Detector with Language Environment (LE)

If your application is executing under an LE environment, it is possible to receive Extended Detector reports even when an abend and system dump do not occur. Because LE applications can sometimes terminate abnormally without an abend or system dump, two callable interfaces have been provided, (CAOECEEX and CEEBXITA, respectively).

To use one of these interfaces, the appropriate LE user exit must be modified as described in *CA-Optimizer Getting Started*. For abending LE applications, Extended Detector functions as usual with no additional considerations.

Note: The LE option TERMTHDACT(UADUMP) will generate a U4039 abend when an unhandled condition occurs of severity 2 or greater. Extended

Using the CAIPRINT Repository Viewer

The CAIPRINT Repository Viewer can be used to view Detector and Extended Detector reports that have been written to your CAIPRINT repository. Each report is automatically written to the repository defined in your run time defaults member, CAOUDFRX.

The CAIPRINT Repository Viewer provides the following benefits:

- A single interface for viewing, printing, and deleting Detector reports
- Easy access to multiple CAIPRINT repositories
- View time control over the reporting options
- View time addition of symbolic information
- Shortcuts to COBOL postprocessor, CSL Display, and Installation Options Display utilities

Starting the Viewer

During installation, the ISPF Primary Option Menu should be updated to include a selection for the CAIPRINT Repository Viewer. If this has not been completed, please see *CA-Optimizer Getting Started* for additional information. Otherwise, select the appropriate option from the menu to start the viewer.

The Repository Index

When the viewer is started, the initial display is always the CAIPRINT Repository Index.

```
CA-Optimizer                      - CAIPRINT Repository Index -                      Row 1 to 6 of 6
Command ==>                      Scroll ==> PAGE
-----
Repository Dsname ==> CAI.PRTLIB
-----
Line Commands: S Select  D Delete  L Lock   U Unlock  P Print  O Options

Cmd  Job      Step      Userid  Comp    Program  Offset  Date      Time    Lock
--  --      --      --      --      --      --      --      --      --
--  XYZAB01A  RUN      XYZAB01  S=0C7   DB2DEMO  002FFC  2001/05/31  16.15  NONE
--  USER1D    OPTDEMO  USER1    S=0C7   CAOMDEMO 003D20  2001/05/31  15.07  NONE
--  USER1C    OPTDEMO  USER1    U=1128  CAOMDEMO 0040CC  2001/05/31  15.07  NONE
--  USER1B    OPTDEMO  USER1    U=1125  CAOMDEMO 004128  2001/05/31  15.07  NONE
--  XYZAB01A  G        XYZAB01  CEE3501 CEE      00005C  2001/05/31  15.06  NONE
--  USER1A    OPTDEMO  USER1    RC=0000  000000  2001/05/31  15.05  NONE
***** Bottom of data *****
```

When you start the viewer for the first time, the CAIPRINT repository dsname is determined using your installation defaults. For all subsequent executions, the value is retrieved from your profile. You can switch to another CAIPRINT repository at any time by overtyping the dsname field and pressing Enter.

The Repository Index display contains a sorted list of all Detector and Extended Detector reports that have been written to the specified repository. For each report, the following information is displayed:

Job	The name of the job that produced the report.
Step	The name of the step that produced the report.
Userid	The name of user that submitted the job that produced the report
Comp	The completion code from the step that produced the report.
Program	The name of the abending program that caused the report to be produced.
Offset	The offset in the abending program where the abend occurred.
Date	The date on which the report was produced.
Time	The time at which the report was produced

Lock The ID of the user, if any, that owns a lock on the report to prevent its accidental deletion.

The view is initially sorted in descending order by date and time, with the newest report appearing at the top of the list. The list can be resorted in either ascending or descending order on any column using the SORT command, described here.

Syntax: SORT column-name <A|D>

where: column-name Is the name that appears at the top of any column in the display

 A Causes the display to be sorted in ascending order (the default)

 D Causes the display to be sorted in descending order

You can refresh the index display at any time using the REFRESH command, described here:

Syntax: REFRESH

Abbreviation: REF

Note: This command has no arguments.

The Repository Index display is scrollable using your standard PF keys for UP and DOWN. The scroll amount can be modified by overtyping the Scroll field to one of the standard scroll amounts (PAGE, HALF, CSR, MAX, or nnn).

Above the scroll amount in the upper right corner of the display, the current position is displayed as **Row x to y of z** where x represents the position of the first report displayed, y the position of the last report displayed, and z the total number of reports in the selected repository.

Below the repository dsname, the valid line commands are listed. Line commands are one-character commands used to perform an action on one specific report in the repository. To execute a line command, place the appropriate character in the Cmd column next to the desired report, then press Enter. If you enter line commands on multiple lines before pressing Enter, they are executed in order from top to bottom. A detailed description of the valid line commands is provided in the following section.

Line Commands

This section describes the following line commands.

- Select
- Delete
- Lock
- Unlock
- Print
- Options

Select

Use the S line command to select a report for viewing.

The selected report is extracted from the CAIPRINT repository and written to a temporary disk file, where it is then opened for viewing using an ISPF BROWSE command.

```
BROWSE  SYS01164.T175139.RA000.XYZAB02.SYSUT1.H01  Line 00000559 Col 001 080
Command ==>                                         Scroll ==> CSR
COMPUTER ASSOCIATES                               DETECTOR REPORT
CA-OPTIMIZER                                     7.0 0106ARX00

JOB: XYZAB02A  STEP: RUN          PROGRAM CAOMDEMO
*****
*              * S-00C7 *
*  A B E N D  *-----*
*              * CAOMDEMO *
*****
*****
* DESCRIPTION:  S0C7 - A NUMERIC FIELD CONTAINED NON-NUMERIC DATA.
*
* PROBABLE CAUSES:  1. NUMERIC DATA WAS NOT INITIALIZED.
*                   2. A SUBSCRIPT OR INDEX CONTAINED AN INVALID VALUE.
*                   3. A COMP-3 FIELD HAD AN INVALID SIGN.
*                   4. A GROUP MOVE OVERLAYED A NUMERIC FIELD WITH NON-NUMERIC
*****
ABENDING COBOL STATEMENT                               PROGRAM-ID CAOMDEMO
SOURCE LOCATION:  PAGE   23  LINE 00894  MLIST 003D20
```

When the report is extracted from the repository, symbolic information is merged using the contents of the Condensed Source Listing libraries defined for your installation. You may optionally define additional CSL libraries to your search order during your session using the CSL command, described later in this chapter.

Note: It is not necessary for the symbolic information to be available at the time the Detector or Extended Detector report is written to the repository. Symbolic information can be added to the CSL library after the report has been created.

Delete

Use the D line command to delete a report from the repository.

A delete confirmation is requested.

```

CA-Optimizer                - Delete Confirmation -
Command ===>
-----
Job XYZAB02A Step RUN      User XYZAB02 Comp S=0C7   Date 2001/06/11 at 15.45
-----

      Delete Confirmation

      You have entered a delete request for the CAIPRINT
      report described above.

      Press ENTER to confirm this request or type END to
      cancel the request.

      ( _ ) Place an "X" here to suppress this confirmation
            for the remainder of this session.
  
```

Press Enter to confirm the request and delete the report from the repository, or END to cancel the request without deleting the report.

To suppress the delete confirmation for the remainder of your session, place an X in the space provided in the box before pressing Enter.

Notes:

1. A deleted report cannot be restored in your repository.
2. To prevent accidental deletion of a report, use the Lock line command.
3. Reports can also be deleted using the utility program CAOUPRTU.

Lock

Use the L line command to lock a report to prevent its accidental deletion from the repository.

When a report has been locked, the userid owning the lock is displayed in the Lock column for that report. If a report is already locked by another user at the time the command is issued, ownership of the lock is transferred to the issuer of the command.

Notes:

1. Ownership of a lock identifies the owner as a user that does not want a report to be deleted. It does not provide any other benefits to the owner.
2. Any user may unlock any locked report at any time using the Unlock line command.

Unlock

Use the U line command to unlock a locked report, allowing it to be deleted from the repository.

When a report is unlocked, the Lock column displays “None” and the report is free to be deleted. Any user may unlock any locked report.

Print

Use the P line command to print a report.

The Print CAIPRINT Report screen is displayed.

CA-Optimizer	- Print CAIPRINT Report -
Command ==>	

Job XYZAB02A Step RUN	User XYZAB02 Comp S=0C7 Date 2001/06/11 at 15.45

Enter a Printer Destination ==>	
OR SYSOUT Class ==>	
OR Output Dataset Name ==>	
and Volume ==>	(if not catalogued)
Place an X in this field to bypass symbolic post-processing ==>	
Press ENTER to print the report or END to cancel the request.	

Select one of the following:

Printer Destination	The printer ID or the nodename.userid where you would like the printed report to be sent.
SYSOUT Class	To send the printed report to a specific output class.
Output Dataset Name	To write the report to a permanent sequential data set, enter the fully-qualified data set name, without quotes.

Volume

A volume is required when the data set is not cataloged.

Press Enter to print the report, or End to cancel the print request.

Note:

1. The bypass option is provided for diagnostic purposes only. Do not place an X in this field unless directed by Computer Associates Technical Support.

Options

Use the O line command to view or update the formatting options for a report.

The Report Options screen is displayed.

```

CA-Optimizer                      - Report Options -
Command ==>

-----
Job XYZAB02A Step RUN           User XYZAB02 Comp S=0C7   Date 2001/06/11 at 15.45
-----

Display Options:                Merge Options:                Report Contents:
ASMINST    ==> ON                MRGDB      ==> ON                INTERCEPT ==> YES
FILES      ==> ON                MRGFILES   ==> ON                PARMCHK     ==> YES
FIRST128   ==> OFF              MRGLINK    ==> ON                SNAP        ==> YES
LINECNT    ==> 60              MRGLOCAL   ==> ON
LINKAGE     ==> ON              MRGWORK     ==> ON
LOCALSTOR  ==> ON              BINFRMT     ==> DEC
MEMMAP      ==> ON              GRPADDR     ==> ON
RPTSZ80     ==> OFF            OCCURS      ==> 1
SAVEAREA   ==> ON
WORKSTOR   ==> ON

Type END to update these options or CANCEL to return without updating.
```

Alter the Display or Merge options as desired for this report, then type End to return to the index display. Type CANCEL on the command line to exit this screen without updating the option settings.

Each report maintains its own set of options in the CAIPRINT repository. Changing one or more option settings for a report does not affect other reports in the repository, but it does affect other users viewing the same report.

For a complete description of the Display and Merge options, see the chapter “Using Options” in this guide.

Adding Symbolic Information

When a report is printed or selected for viewing, symbolic information may be added to the report for COBOL programs that were active when the report was created. Symbolic information is added automatically for any program whose listing resides in one of the Condensed Source Listing (CSL) libraries defined in the runtime defaults member, CAOUDFRX.

Additionally, if a specific CSL library was available to Detector (or Extended Detector) at the time the report was created, that CSL would always be searched first for symbolic information.

Finally, any user of the CAIPRINT repository viewer may specify additional CSL libraries to be used only for their own viewing without affecting any other users of the viewer. These libraries are searched in front of any installation default libraries.

To specify additional CSL libraries, type CSL on the command line and press Enter. The Supplemental CSL Dsnames screen is displayed.

```
CA-Optimizer          - Supplemental CSL Dsnames -
Command ===>
-----
Enter dataset names for any Condensed Source Listing (CSL) libraries
that are not already defined in your installation defaults:

    -----
    | Enter all dataset names fully qualified, without quotes. |
    -----

===>
===>
===>
===>
===>

Type END to update this list of dsnames or CANCEL to return without updating.
```

Enter up to five additional CSL library names to be used whenever you view reports from your own userid. Use fully-qualified data set names without quotes. These library names are stored in your profile and remain in effect until you remove them.

Type End to return to the index display.

Invoking the COBOL Postprocessor

Symbolic information for any program can be added to a CSL library at any time, even if one or more CAIPRINT reports for that program already exist in the repository.

There are two basic methods for adding symbolic information to a CSL library:

- Using the CSL option during a CA-Optimized compile
- Using the COBOL postprocessor after a standard COBOL compile

Sample JCL is provided for each of these methods, including JCL to load entire COBOL listing libraries into a CSL library. These are the recommended methods for loading your CSL libraries.

There may also be times when you notice the absence of symbolic information while viewing a CAIPRINT report. You may prefer to load the symbolic information for one or more COBOL programs into a CSL library without exiting the viewer or submitting a batch job.

To load symbolic information for one or more COBOL programs into a CSL library, type CPP on the command line and press Enter. The Create CSL Members screen is displayed.

CA-Optimizer - Create CSL Members -
Command ==>

Use this panel to create CSL members from listing files using the
listing post-processor.

Enter all dataset names fully qualified, without quotes.

Target CSL Library ==> USER.CSL
Options ==> CPPRC

Source Listing Library ==> USER.COBOL.LISTINGS
Library Type ==> PDS (PDS, SEQ, LIB, PAN, or NDV)

From Member ==>
To Member ==>

Press ENTER to create CSL members or END to terminate this operation.

The fields on this panel are:

Target CSL Library	The fully-qualified (no quotes) data set name for the CSL library into which you are loading the symbolic information.
Options	Any COBOL postprocessor options that you may require. For a detailed description of the CPP options, see COBOL Postprocessor Options in Chapter 3.

Source Listing Library The fully-qualified (no quotes) data set name for the COBOL listing library. Do not include a member name for partitioned data sets.

Library Type One of the following supported library types:

PDS	Partitioned data set
SEQ	Sequential data set
LIB	CA-Librarian library
PAN	CA-Panvalet library
NDV	CA-Endevor library

From Member The name of the first (or only) member to be loaded into the CSL. For all members or to specify a prefix, a single trailing asterisk "*" may be used in this field.

To Member Only if a From Member has been specified and does not contain a trailing asterisk (*), this field can be used to specify the last member of the range of members to be loaded into the CSL.

Fill in the fields on this screen and press Enter to load the symbolic information into your CSL library, or press End to cancel the request and return to the index display.

Note: The CPP command provides a foreground interface to the COBOL postprocessor that is functionally equivalent to submitting a batch job. When loading a large number of members, loading symbolic information in batch is recommended.

Invoking the CSL Display Utility

To review the contents of your CSL libraries, type LISTCSL on the command line and press Enter. The CSL Display Utility CAOUCSLD is invoked, producing a report on the contents of all CSL libraries defined to your installation. The report is written to a temporary file and displayed using BROWSE.

```

BROWSE      SYS01164.T181916.RA000.USRID02.SYSPRINT.H0 Line 00000000 Col 001 080
Command ==>
                                           Scroll ==> CSR
***** Top of Data *****
COMPUTER ASSOCIATES
CA-OPTIMIZER      7.0  0106ARX00      13 JUN,2001      PAGE      1
                                           18:19:17
                  CA-OPTIMIZER CONDENSED SOURCE LIBRARY
                  DATA SET NAME = CAI.CSL
PROGRAM ID      DATE COMPILED      TIME COMPILED      GENLEVEL      RELEASE      COMPILER
-----
PG01            31 MAY,2001          11:54:47          0106AOM00      7.0          VS COBOL
PG06            30 MAY,2001          16:31:24          0106AOM00      7.0          VS COBOL
PLGP0W10        25 MAY,2001          17:50:29          0106AOM00      7.0          VS COBOL
TOTAL NUMBER OF VS COBOL PROGRAMS      =      3
TOTAL NUMBER OF PROGRAMS                =      3
COMPUTER ASSOCIATES
CA-OPTIMIZER      7.0  0106ARX00      13 JUN,2001      PAGE      2
                                           18:19:17
                  CA-OPTIMIZER CONDENSED SOURCE LIBRARY
                  DATA SET NAME = CAI.QA.CSL
PROGRAM ID      DATE COMPILED      TIME COMPILED      GENLEVEL      RELEASE      COMPILER
-----
CAOMDEMO        11 JUN,2001          17:46:35          0106ARX00      7.0          VS COBOL
COREINDX        30 MAY,2001          15:50:21          0106AOM00      7.0          VS COBOL

```

The LISTCSL command also accepts a single (optional) argument allowing you to specify the fully-qualified (no quotes) name of a CSL library. If specified, the report is limited to only that one CSL library.

Example: LISTCSL MY.CSL.LIBRARY

Displaying Installation Options

Type OPTS on the command line to view a report of your installation default options.

```

BROWSE      SYS01164.T183400.RA000.USRID02.SYSPRINT.H0 Line 00000000 Col 001 080
Command ==>                                     Scroll ==> CSR
***** Top of Data *****
CA-OPTIMIZER      7.0 0106ARX00
*
* EXECUTION TIME DEFAULTS:      REPORT TIME DEFAULTS:      COMPILE TIME DEFAULTS:
* GENLEVEL 0106ARX00           GENLEVEL 0106ARX00           GENLEVEL 0106AOM00
*
* CA-OPTIMIZER RUNTIME ACTIVE - CAOCCEMCK LOADED AT 00A466F8
*
* ACB      = OFF           CNTXREF  = YES           BUFMAX    = 0
* ACTONLY  = OFF           INSTRMAP = NO           CICS      = NO
* AMB      = OFF           PROCDIV  = NO           CLIST     = NO
* AMBL     = OFF           SOURCEMAP = YES          COBCC     = 8
* ASMINST  = ON           *           COMPILER  = IKFCB
* BINDER   = OFF           *           COPT      = YES
* BINFRMT  = DEC           *           CSL       = NO
* BLOCKS   = OFF           *           CSLDD    = CAICS
* CEEDUMP  = OFF           *           DCSL     = YES
* CSLDD    = CAICSL        *           DECK     = NO
* DBGAWAKE = OFF           *           DMAP     = NO
* DB2ACTIV = OFF           *           DTECT    = N

```

An options report is written to a temporary file and displayed using BROWSE.

Note: The report has a record length of 121 bytes. Depending on your terminal the display may be both horizontally and vertically scrollable.

Procedure CAIOPTNS is also provided to produce this report in batch. See the Options Summary in the chapter “Utilities” in this guide for more information.

Help

Help is available for the CAIPRINT Repository Viewer at any time by pressing your HELP key. The main menu for Help is displayed.

```

CA-Optimizer          - CAIPRINT Repository Index -          Help
Command ==>
-----
                                Help Menu
-----
Select a Help Topic ==>
    1 Selecting a CAIPRINT Repository
    2 Repository Index Fields
    3 Commands
    4 Line Commands
    5 Printing Reports
    6 Report Options
    7 Error and Reason Codes

```

Help can be found for any of the following topics:

Selecting a
Repository

Option #1 describes how to switch the index display to a new repository.

Index Fields

Option #2 describes the fields on the repository index display.

Commands

Option #3 describes all of the commands recognized on the command line.

Line Commands

Option #4 describes the line commands available on the index display.

Printing Reports

Option #5 describes how to print a CAIPRINT report from the repository.

Report Options

Option #6 displays the acceptable values for the report options.

Error and Reason
Codes

Option #7 explains all of the error codes and reason codes that you may encounter while using the repository viewer.

Use your UP arrow key to jump to the previous Help topic (that is, from option #2 to option #1) and your DOWN arrow key to jump to the next topic.

Most topics display as a single panel, although some topics require multiple panels (use Enter to step from one panel to the next) and other topics provide additional selection menus. Specific directions for maneuvering through each panel can be found at the bottom of each panel.

Using Analyzer

Analyzer helps you debug your COBOL programs with speed and assurance. When your programs go into production, you will have confidence that testing was thorough.

The count and timing data for each statement reveals exactly how many times the statement was executed and how much CPU time was required. The count and timing data may be representative of a single execution or of the consolidation of multiple executions. In any case, all of this information prints beside each statement on your source listing.

For **testing and debugging**, Analyzer:

- Flags the statement causing the abend
- Locates the statement where the time out occurred
- Identifies:
 - Untested verbs
 - IF tests that are always true or always false
 - The path of an infinite loop
 - All possible control paths into each paragraph
 - Related logic (groups of code that have the same count)
- Shows:
 - What additional test data is needed
 - The number of records read and written
 - PERFORM ranges that can be fallen into or out of
 - PERFORM ranges that can be branched out of

In **source level tuning**, Analyzer:

- Locates the most frequently executed code
- Helps locate unexecutable code that can be removed

- Identifies:
 - Data distribution
 - Code being executed more often than necessary (for instance, code which can be moved outside a loop)
 - Infrequently used code that can be overlaid
 - Subroutines in need of tuning and modules in a multimodule program in need of tuning
- pinpoints:
 - Statements and paragraphs where the most CPU time is spent
 - Statements or groups of statements in non-COBOL CSECTS which should be tuned
- shows:
 - Inefficient ordering of nested IF tests
 - Time differences resulting from different data usages
 - Time differences resulting from the use of various COBOL verbs

For **maintenance**, Analyzer:

- Verifies that new code is tested
- Shows:
 - Whether additional test data is needed after a source code change
 - Obsolete code
 - The number of records read and written
- Helps identify initialization, error, exception, and termination routines
- Identifies:
 - Main line logic
 - Inner loops
 - Data distribution (record types, number and types of errors, and so forth)

With **quality assurance**, Analyzer:

- Verifies that all code has been tested
- Indicates that the program is doing what it is supposed to do
- Identifies any unexpected changes in the processing performed by the program

In **programmer training**, Analyzer:

- Reinforces good programming techniques through:
 - Timings on inefficient verbs
 - Generated statements showing the PERFORM logic is always exited
 - Comparative timings on various data usages
- Clarifies what a new program does through counts that identify main logic: initiation, termination, and exception routines, and so forth

For **dynamic program documentation**, Analyzer documents:

- What a program does
- Where time is spent
- Data distribution when modifications are made to program code changing the data in the files

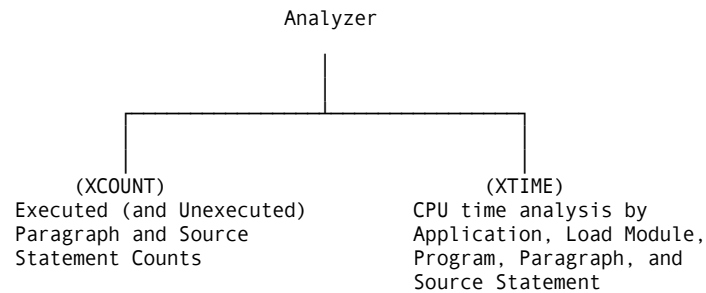
The Counts facility lets you identify executed (and unexecuted) paragraphs and source statements in one program, several programs, or the whole application. Counts can be used with or without Timings. In addition, the Counts data can be accumulated so that a single set of reports can be produced summarizing the execution results of a whole battery of test cases.

The Timings facility lets you identify CPU time concentrations in a hierarchical fashion. You can look at the CPU time for the whole application, one load module, one program, one paragraph, or one source statement. Timings can be used with or without Counts. Additionally, the Timings data can be accumulated so that a single set of reports can be produced summarizing the Timings results of a whole battery of test cases.

Analyzer Options

Overview

Analyzer consists of two facilities: Counts and Timings. The two facilities may be used independently or together. Pictorially:



Analyzer control may be needed at compile time, execution time, and report time. The following chart provides a listing of Analyzer control options available to you at compile, execution, and report time.

Compile Time	Execution Time	Report Time
XCOUNT	XCOUNT	CNTXREF
	XREPORT	INSTRMAP
	XTIME	PROC DIV
		SOURCEMAP
		USERID

You may freely combine Analyzer compile time options with OS/VS COBOL compiler options and with other CA-Optimizer options. The options shown here have both compile time and execution time uses. Their execution time applications are discussed later in this chapter. See the chapter “Using Options” in this guide for a complete listing of all CA-Optimizer options, their defaults, usage, and special considerations.

The following chart summarizes how you may use Analyzer's more commonly used facilities.

To:	At compile time specify:	At execution time specify:
Report on the paragraphs and source statements that were executed, or not executed.	XCOUNT	
Report on where the CPU time is concentrated in your program.		XTIME FIRST
Turn OFF Counts and Timings, even if they were specified at compile time.	XCOUNT OFF XTIME OFF	
Defer Counts or Timings Reports to a separate Analyzer Reporter job step.	XCOUNT	XREPORT OFF
Begin Timings with the first program optimized with any Analyzer or Detector facility.	Any Analyzer or Detector option	XTIME FIRST
Produce the Execution Count Cross-Reference Report for a single program only (for example PGMNAME).	XCOUNT	CNTXREF PGMNAME
Suppress the Source Map Report for all programs.		NOSOURCEMAP ALL

Report Time Considerations

The Analyzer Reporter can be run as a separate job step any time after the application has executed, or it can be automatically invoked in the same job step as the application. The option that controls this is XREPORT.

The options you use to control the Analyzer Reporter are the same whether XREPORT is ON or OFF. When using XREPORT ON, the report time options can be mixed freely with other execution time options for Analyzer or Detector. However, when XREPORT OFF is in effect at execution time, the Analyzer Reporter is not automatically invoked in the same job step when the application is terminated. Instead, the Analyzer Reporter must be executed as a separate job step. In this case, the JCL for the separate job step includes a CAIOPTS DD statement in which only report time options are specified.

All report time options, except USERID, are intended for use in special circumstances and are not needed in straightforward applications of Analyzer where default values suffice. The USERID option permits you to associate a unique page heading line with each Analyzer run.

Analyzer correlates all count and timing measurement data it has gathered and uses it to produce various Analyzer reports. The measurement data prints next to the source statements of the programs, if listings have been saved in compressed form by the Condensed Source Listing Library from compile time. If the source listings are not available, you can only obtain the Run Summary, Module Summary, and Instruction Detail Reports.

At report time, you can consolidate the measurement data from several executions of the same programs to produce a single set of reports. The Analyzer Run Summary Report identifies the various executions that are consolidated for the reporting run. The Run Summary shows how report time options were used and provides additional diagnostic information on report time input.

JCL

Compile Time

To invoke Analyzer, you must specify one or more of the Analyzer options in the compile/optimize step. To execute the Optimizer component, refer to a copy of your installation's cataloged procedures for a procedure to compile/optimize/and link (or compile/optimize/link, and go). The standard Computer Associates procedure is called CAIOPT. Use this JCL to invoke the Optimizer component.

```
//STEP1      EXEC CAIOPT,LISTLIB='Your CSL library name',  
//              COBOPT='XCOUNT'
```

In the previous example, LISTLIB and COBOPT are symbolic options defined in the CAIOPT procedure. LISTLIB gives the library name of the Condensed Source Listing Library which is used to produce the Analyzer reports. COBOPT passes the PARM options for the compile/optimize step. If the cataloged procedure at your installation does not have these symbolic options, you can specify the CSL library name and pass the Analyzer options using the JCL here.

```
//STEP1      EXEC CAIOPT,  
//              PARM.COB='XCOUNT'  
//COB.CAICSL DD DSN='your CSL library name',DISP=SHR
```

Execution Time

The standard Computer Associates procedure is for compile and link only. If you are using this procedure, you need to execute your program directly. The following JCL executes your application and produces the Analyzer reports in the same job step.

```
//STEP2      EXEC PGM=your-program-name,REGION=xxxK
//STEPLIB    DD DSN=your-OPTIMIZER-loadlib-name,DISP=SHR
//           (Concatenate the STEPLIB for your program's
//           loadlib name to the previous STEPLIB.)
//
//           .
//           .
//           .
//           Your program's DD statements
//
//CAIPRINT    DD SYSOUT=A
//CAITIMSV    DD DSN=&&TIMSV., DISP=(MOD, PASS),
//              UNIT=SYSDA, SPACE=(TRK,(1,1),RLSE),
//              DCB=(LRECL=20,BLKSIZE=4360,RECFM=FB)
//CAIOPTS     DD *
//              XCOUNT ON
//              XTIME FIRST
//              XREPORT ON
//              /*
//CAICSL      DD DSN=your CSL library name,DISP=SHR
//CAIPRNT1    DD SYSOUT=A
//CAIPRNT2    DD SYSOUT=A
//CAISTATS    DD DSN=your installation's OPT statistics file,DISP=SHR
```

These are descriptions for the JCL statements used by Analyzer and Analyzer Reporter. The region size on the EXEC statement shown here should be increased a minimum of 1 MB to accommodate the Analyzer Reporter when it is automatically invoked following the termination of the application.

ddname	Description
//STEPLIB DD	Points to the CA-Optimizer load library. This is not necessary if CA-Optimizer is in your concatenated LINKLIB.
//CAIPRINT DD	Retains Analyzer (and Detector) printed output. This DD statement is optional and is dynamically allocated, if omitted.
//CAITIMSV DD	Holds Analyzer's count and timing measurement data; a MODed, OLD, or NEW data set can be specified. If you only want Analyzer reports for data from this run, omit the TIMSV DD statement. It is then dynamically allocated.
//CAIOPTS DD	Specifies Detector, Analyzer, and Analyzer Reporter options. You can omit this DD statement if no options need to be specified.

ddname	Description
//CAICSL DD	Defines the Condensed Source Listing library for Analyzer Reporter. This DD statement is required unless: ■ An installation default CSL is used, or ■ The data set name of the CSL is specified in the CSLDSN option in the CAIOPTS DD statement.
//CAIPRNT1 DD //CAIPRNT2 DD	These two DD statements define the two report data sets for Analyzer Reporter. If one or both are omitted, they are dynamically allocated.
//CAISTATS DD	Specify this DD statement if you want Analyzer Reporter to put the name of your installation on the top of each page of Analyzer Reporter output. If you do not want this facility, omit the DD statement.

For a detailed discussion of dynamic allocation for Analyzer and Detector, see the chapter “Usage Considerations” in this guide.

Analyzer Reporter Job Step

Most of the discussion thus far has presented Analyzer Reporter as a reporting function that is automatically invoked at either a normal or abnormal end-of-job. This automatic invocation is triggered by XREPORT ON and the reports produced by Analyzer Reporter reflect that single execution only.

For advanced performance measurement studies, or for quality assurance purposes, it may be desirable to combine many executions of an application. This is achieved by concatenating the counts and timings data with prior runs. (For a more detailed discussion of Analyzer Reporter execution, either as a job step or automatically, see Automatic Invocation of Analyzer Reporter). When reporting is deferred in this manner, the Analyzer Reporter function must be run as a separate job or step.

Refer to a copy of your installation's cataloged procedure for Analyzer Reporter. The cataloged procedure supplied with CA-Optimizer, is as follows:

```
//stepname    EXEC CAIARII,TIMSV='counts and timings data set name',
//  LISTLIB='your Condensed Source Listing Library name'
//CAIOPTS     DD *
//              Analyzer Reporter options (if required)
/*
```

TIMSV and LISTLIB are symbolic options defined in the CAIARII procedure. TIMSV provides the data set name of the counts and timings data set and LISTLIB gives the library name of the Condensed Source Listing Library. If the cataloged procedure at your installation does not have these symbolics, specify these two data sets in the execution JCL:

```
//stepname    EXEC your-procedure-name
//CAITIMSV    DD DSN=counts and timings data set name,DISP=SHR
//CAICSL      DD DSN=your CSL library name,DISP=SHR
//CAIOPTS     DD *
              Analyzer Reporter options (if required)
/*
```

Example

This JCL is an example of how to use both Analyzer (with Counts and Timings) and Detector (with subscript range checking) and produce the MDMAP from the Optimization step (see STEP 1 following). Analyzer Reporter runs in the same job step (STEP 2) immediately after the application terminates.

```
//STEP1      EXEC CAIOPT,
// COBOPT='XCOUNT,SUBRNGCK,MDMAP',
// LISTLIB='PAYROLL.LISTLIB'
//STEP2      EXEC PGM=FICAEXTR,REGION=512K
//STEPLIB    DD DSN=PAYROLL.LOADLIB,DISP=SHR
//           DD DSN=CAI.CAILIB,DISP=SHR
//           DD DSN=CAI.CAILPA,DISP=SHR
//FILEIN     DD DSN=PAYROLL.MASTER(0),DISP=OLD
//FILEOUT     DD DSN=PAYROLL.MASTER(1),DISP=(NEW,CATLG),
//           UNIT=TAPE
//PAYREPT    DD SYSOUT=A
//CAICSL     DD DSN=PAYROLL.LISTLIB,DISP=SHR
//CAIOPTS    DD *
              XREPORT ON
              XTIME FIRST
              XCOUNT ON
/*
```

*This is only needed when the PDS is not part of LINKLIB.

**This is only needed if your CSL is not the installation's default CSL.

Analyzer Reports

This section explores each field in each report. Reference this section if you have any questions concerning the fields in any of the reports.

CAIOPTS File Processing Report

The Execution Monitor produces this report at the start of application execution, after the Execution Monitor has read and processed the Detector or Analyzer runtime options. The report displays the input options read from the data set defined by the CAIOPTS DD statement, if present. The body of the report is a display of the 80-byte records in the CAIOPTS data set exactly as read by the Execution Monitor.

COMPUTER ASSOCIATES	ANALYZER/REPORTER CAIOPTS FILE PROCESSING REPORT	PAGE 1
CA-OPTIMIZER 7.0 0106ARX00		15 JUN,2001 14.52.53
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8		
INTERCEPTS 2	00572006	
SNAP 1	00573013	
PFLOW ON	00574012	
PARMCHK ON	00575012	
XCOUNT ON	00576012	
XTIME FIRST	00577013	
XREPORT ON	00578012	
BLOCKS ON	00579027	
DUMP OFF	00579127	
INSTRMAP ALL	00580023	
INPUT OPTION DISPLAY COMPLETE		

Execution Monitor Report

The Execution Monitor Report, which is generated at execution time, is separate from the Analyzer reports, but accompanies each Analyzer run. The Execution Monitor Report verifies that Analyzer was in operation during the run and gives the version of each program running under Analyzer and the CA-Optimizer execution options specified.

COMPUTER ASSOCIATES		EXECUTION MONITOR SUMMARY		PAGE 25				
CA-OPTIMIZER		7.0 0106ARX00		15 JUN 2001 14.29.52				
EXECUTION OPTIONS IN EFFECT: A		EXECUTION ON: 05/390 K						
ACB	ON							
ACTONLY	OFF							
AMB	ON							
AMBL	ON							
ASMINST	ON							
BINDER	OFF							
BINFMT	DEC							
CEEDUMP	OFF							
CSLDSN								
DB2ACTIV	OFF							
DCB	ON							
DEB	ON							
DUMP	OFF							
EXCEED	NOSTOP							
FILES	ON							
FIRST128	OFF							
GRPADDR	ON							
IMS	NO							
INTERCEPT	2							
IOB	ON							
LINKAGE	ALL							
LOCALSTOR	ALL							
MEMMAP	ON							
MERGEDB	ON							
MGRFILES	ON							
MRGLINK	ON							
MRGLOCAL	ON							
MRGWORK	ON							
MREPORT	ALL							
MSAMPLE	10000							
MSTAE	ON							
MTIME	0							
OCCURS	1							
PARMCHK	ON							
PFLOW	ON							
PLH	ON							
PRTREPT	BOTH							
RPL	ON							
SAVEAREA	ON							
SNAP	1							
STIMERM	OFF							
UCB	ON							
VSAMIDX	ON							
WORKSTOR	ALL							
XCOUNT	ON							
XREPORT	ON							
XTIME	FIRST							
LEGEND FOR OPTIMIZER OPTIONS: D=DETECT C=XCOUNT								
B	C	D	E	F	G	H	I	J
COBOL	COMPILATION	COMPILER	COUNT	NUMBER	NUMBER	RUN-TIME OPTIONS		RUN-TIME OPTIONS
PROGRAM	DATE AND TIME	USED REL	SPACE	OF TIMES	OF TIMES	SPECIFIED AT		SPECIFIED AT
NAME			(BYTES) ->	ALLOCATED	EXECUTED	COMPILE TIME		EXECUTION TIME
CAOMDEMO	JUN 15,1901 14.29.26	05/VS 2.4	2048		1	1	,C,D,P	

The Execution Monitor Report provides the following information for every COBOL program running under Detector or Analyzer:

A	Lists in alphabetical order the options that were in effect during this execution of the application.
B	Lists various attributes of all the Optimized COBOL programs found by the Execution Monitor during the execution of the application. Entry to any program compiled/optimized with the Detector or Analyzer Option, will cause it to be listed here.
C	Compile date, which helps identify the version of the program.
D	Compile time, which helps identify the version of the program.
E	COBOL compiler and version number used to compile this program.
F	Amount of main storage required to store XCOUNT data.
G	Number of times the Execution Monitor was required to allocate XCOUNT space.
H	Number of times the program was executed during the run.
I	Compile time options in effect for the program.
J	Runtime options in effect for the program.
K	Operating system under which the application executed is shown as MVS, MVS/XA, MVS/ESA, OS/390, or z/OS.

Run Summary Report

This report prints every time an Analyzer reporting run is executed. It provides EXCP counts for each ddname, as well as a summary of the options specified, runs consolidated, and other information. The example following shows the format for a single execution:

```

COMPUTER ASSOCIATES              ANALYZER RUN SUMMARY              PAGE    1
CA-OPTIMIZER                     7.0  0106ARX00                    15 JUN,2001 14.52.53
      A ANALYZER RUN IDENTIFICATION 15 JUN,2001 14.52.53  COMPUTER ASSOCIATES

EXECUTION OPTIONS IN EFFECT: B

XTIME      FIRST
XCOUNT     ON
XREPORT     ON
SOURCEMAP   ALL
CNTXREF     ALL
INSTRMAP    ALL
LINECOUNT  60
            TOTAL SAMPLES TAKEN                245 C
            IDENTIFICATION OF EXECUTIONS CONSOLIDATED D

COUNT  JOB      STEP  PROC STEP  DATE      TIME      SAMPLES

    001 USRID02A RUN                06/15/2001 14.52.47      245

            EXCP COUNT BY DDNAME E
            DDNAME      UNIT      EXCP COUNT
            CAICSL      UCB        203      *
            CAITIMSV    UCB         2      *
            STEPLIB     UCB        151      *
            -----
            TOTAL              356

            INTERRUPT IN PGM CAOMDEMO EXECUTION  1 OFFSET 003D20 PAGE  23 LINE 00894 008940

CAPR326W CSECT ILBOQIO HAS INSUFFICIENT SAMPLES TO PRODUCE AN INSTRMAP REPORT.
CAPR326W CSECT CAODSNAP HAS INSUFFICIENT SAMPLES TO PRODUCE AN INSTRMAP REPORT.

```

A

Identifies the reporting run: the current date and time, and any user identification information that was specified in the USERID control statement.

B

Displays the Analyzer reporting control statements (options) which are in effect for this run. When certain statements have not been specified, the default controls are listed.

The options that were specified to control the measurement gathering process during program execution are listed for your reference. When multiple executions have been consolidated, the measurement options could be different for each execution. In this case, line B shows the options specified for the first execution step. If any execution steps after the first have different options, the options that are different from the original ones print in the report section entitled Identification of Executions Consolidated.

- C** Shows the total number of time samples taken for all of the executions consolidated in this set of reports. If no time measurement data is available, this line does not print.
 - D** Identifies the executions that are consolidated in this report. Each line represents a single execution step. The job name, step name, procedure name, date, time, and total number of samples print for each execution step. If there is no time measurement data available, the Samples column remains empty. If the next job step has different execution options, the options and their differing values precede the next identification line for the job step.
 - E** Lists EXCP count and the unit addresses for every DDname. The EXCP count is listed for each unit and its associated DDname, if the data set has multiple volumes. If you have SYSIN/SYSOUT data sets under JES, the EXCPs are added together and appear as a sum labeled JES.
- The EXCP counts can give you some important insights into the efficiency of your blocking and overlay structures. A blocking factor that is too low shows up as a higher EXCP count. A high number in the EXCP COUNT field for the STEPLIB DDname may indicate an inefficient overlay structure. This section of the Run Summary Report also shows how far your programs progressed in terms of the number of records read in and the number that were written out.
- Note:** The EXCP counts are taken from SMF internal counters.

Module Summary Report

The Module Summary Report makes it possible to accomplish CPU time tuning on a broader scale: at the load module and CSECT level. This report isolates those CSECTs within load modules that consume the most execution time. To generate the Module Summary, XTIME (for timings) must have been used.

The Module Summary Report shows how the total CPU time for the set of executions in this Analyzer reporting run is distributed across all of the CSECTs within each load module that has samples. The report lists timings for any system modules such as SORT, access methods, and so forth. Supervisor modules invoked by SVCs, such as OPEN, CLOSE, and EXCP, are not shown, but their CPU time is charged to the SVC instructions within the CSECTs.

The following example shows a multimodule execution:

COMPUTER ASSOCIATES		MODULE SUMMARY		PAGE 2	
CA-OPTIMIZER		7.0 0106ARX00		15 JUN,2001 14.52.53	
EXECUTIONS	LOAD	DATE	TIME	CPU TIME	PERCENT OF
CONSOLIDATED	MODULE	CSECT	COMPILED	COBOL COPT	TOTAL
1				MIN.SEC.HUN	PERCENT OF TOTAL
	CAODSNAP			0.00.96	100.00 .XXXXXXXXXXXXXXXXXXXXXXXXXXXX
	CAODSNAP			0.00.00	100.00 .51* .
	CAOMDEMO			0.00.00	100.00 .51 .
	CAOMDEMO	JUN 15,2001	14.29.26	VS OPT DC	0.00.95 100.00 98.99* .XXXXXXXXXXXXXXXXXXXXXXXXXXXX
	ILBOQIO			0.00.00	100.00 .51* .
	ILBOQIO			0.00.00	100.00 .51 .

A Identifies each reporting run, the date and time of the reporting run, and the user program-ID information (if provided).

B Shows the total number of sets of separate execution measurement data consolidated for this run.

C Identifies all load modules in this reporting run and the CSECTs within those load modules.

A load module may consist of one or more programs (CSECTs) grouped as one entity by the linkage editor. It is possible that the same CSECT may appear in more than one load module, and is therefore listed separately for each module.

Samples attributed to the user private area or the operating system are resolved to the 16 MB boundary of virtual storage in which they occurred. These samples are reported as *REGN-*nn* or *SYSTM*nn*. In both cases *nn* is the hex value of the high order byte of the 4-byte address in which the sample occurred.

D Shows the date the program was compiled and optimized.

E Indicates the time the program was compiled and optimized.

F Since the Analyzer reporting facilities are oriented to COBOL execution environments, each individual CSECT is identified as COBOL or NOT, meaning not a COBOL CSECT. The compiler version (ANS2, ANS3, ANS4, VS) for each COBOL CSECT is displayed.

G Further classifies each COBOL CSECT as Optimized (OPT) or Not Optimized (NOPT). This shows at a quick glance which COBOL programs have yet to be compiled with CA-Optimizer. Also, if a COBOL module has been optimized with any of the Detector features, a D prints next to OPT. If it has been optimized with any of the Analyzer features, an A prints next to OPT. A blank indicates that this is not a COBOL program.

H Displays the CPU time for each load module and the CSECTs within each load module in minutes, seconds, and hundredths of seconds.

The first entry in the column is the total CPU time for the run (or set of consolidated runs). This total CPU time is the sum of the time-weighted samples and therefore approximates, and is less than, the job step CPU time displayed by MVS on the JES job log. It is less because CPU time is not included for processing:

- The application before timings are started.
- The application after timings have ended.
- Analyzer to process and write the timings data.

The CPU time for each load module is the sum total of the CPU time measured for all CSECTs within the load module.



The first percentage column gives the percentage of this load module's time that is consumed by each CSECT.

The percentage in the second column relates each CSECT to the total CPU time reported. The load module lines give the percentage of the total CPU execution time that was devoted to each load module. The line for each CSECT tells what percent of the total CPU time was devoted to that CSECT.



The percent histogram depicts the CPU time distribution by individual CSECT and load module. This graph shows at a quick glance which CSECTs and load modules used large amounts of CPU time.

Source Listing Count and Timing Report (SOURCEMAP)

On the COBOL source listing, both the count and timing measurement data print in columns to the right of the source code.

Note: Heading lines pertain to the date and time of compilation, not the report time. This data is obtained from the CAICSL file. Also, the page numbering reflects that of compilation too, rather than the page number associated with Analyzer (Reporter) output.

COMPUTER ASSOCIATES CA-Optimizer		7.0 0106A0M00	ANALYZER SOURCEMAP FOR PROGRAM - CAOMDEMO		PAGE 9		15 JUN,1901 14.29.26
SENTENCES		CAOMDEMO	MLIST	COUNTS	PERCENT	OF TIME	SPENT
00346	003460	PROCEDURE DIVISION	00346000 001D6C	1	*		
00347	003470	USING PARMS UNUSED-PARM.	00347000		*		
00348	003480		00348000		*		
00349	003490	DEMO-BEGIN.	00349000		*		
00350	003500	GO TO DEMO-INIT.	00350000 001DB4	1	*		
00351	003510		00351000		*		
00352	003520	DEMO-INIT.	00352000 001DC2	1	*		
00353	003530	OPEN OUTPUT REPORT-OUT.	00353000 001DCE	1	.5*		
00354	003540	MOVE HD1 TO LINE-OUT PERFORM WRITE-LINE.	00354000 001E1C	1	*		
			PERFORM 001E22	1	*		
00355	003550	MOVE HD2 TO LINE-OUT PERFORM WRITE-LINE.	00355000 001E2E	1	*		
			PERFORM 001E34	1	*		
00356	003560	MOVE 50 TO COUNT-TIMES.	00356000 001E40	1	*		
00357	003570	MOVE 30 TO SUB-MAX.	00357000 001E46	1	*		
00358	003580	MOVE BINARY-1 TO SUB.	00358000 001E4C	1	*		
00359	003590	MOVE BINARY-1 TO TVAR.	00359000 001E52	1	*		
00360	003600		00360000		*		
00361	003610	DEMO-INIT-BYPASS.	00361000		*		
00362	003620	ALTER DEMO-BEGIN TO PROCEED TO DEMO-START-IT.	00362000 001E60	1	*		
00363	003630		00363000		*		
		---FALL-THROUGH TO NEXT PARAGRAPH.		1	*		
00364	003640	DEMO-START-IT.	00364000 001E70	1	*		
00365	003650	IF PARM-LENGTH = 8 AND PARM-S = 'SUBRNGCK'	00365000 001E7C	1	*		
00366	003660	MOVE 'SUBRNGCK' TO PARM-SELECTED	00366000 001E9A	0	*		
00367	003670	MOVE 'DTE' TO MODE-SELECTED	00367000 001EA0	0	*		
00368	003680	ELSE	00368000		*		
00369	003690	IF PARM-LENGTH = 12 AND PARM-S-E = 'EDS,SUBRNGCK'	00369000 001EB2	1	*		
00370	003700	MOVE 'SUBRNGCK' TO PARM-SELECTED	00370000 001ECC	0	*		
00371	003710	MOVE 'EDS' TO MODE-SELECTED	00371000 001ED2	0	*		
00372	003720	ELSE	00372000		*		
00373	003730	IF PARM-LENGTH = 7 AND PARM-F = 'FDCHECK'	00373000 001EE4	1	*		
00374	003740	MOVE 'FDCHECK' TO PARM-SELECTED	00374000 001EFE	0	*		
00375	003750	MOVE 'DTE' TO MODE-SELECTED	00375000 001F04	0	*		
00376	003760	ELSE	00376000		*		
00377	003770	IF PARM-LENGTH = 11 AND PARM-F-E = 'EDS,FDCHECK'	00377000 001F16	1	*		
00378	003780	MOVE 'FDCHECK' TO PARM-SELECTED	00378000 001F30	0	*		
00379	003790	MOVE 'EDS' TO MODE-SELECTED	00379000 001F36	0	*		
00380	003800	ELSE	00380000		*		
00381	003810	IF PARM-LENGTH = 6 AND PARM-W = 'WSINIT'	00381000 001F48	1	*		
00382	003820	MOVE 'WSINIT' TO PARM-SELECTED	00382000 001F62	1	*		
00383	003830	MOVE 'DTE' TO MODE-SELECTED	00383000 001F68	1	*		
00384	003840	ELSE	00384000		*		
00385	003850	IF PARM-LENGTH = 10 AND PARM-W-E = 'EDS,WSINIT'	00385000 001F7A	0	*		
00386	003860	MOVE 'WSINIT' TO PARM-SELECTED	00386000 001F94	0	*		
00387	003870	MOVE 'EDS' TO MODE-SELECTED	00387000 001F9A	0	*		
00388	003880	ELSE	00388000		*		
00389	003890	IF PARM-LENGTH = 4 AND PARM-N = 'NONE'	00389000 001FAC	0	*		
00390	003900	MOVE 'NONE' TO PARM-SELECTED	00390000 001FC6	0	*		
00391	003910	MOVE 'DTE' TO MODE-SELECTED	00391000 001FCC	0	*		
00392	003920	ELSE	00392000		*		

A The MLIST addresses print along with the count and timing measurement data on the source listing.

B This column contains the total execution count (XCOUNT) for each verb. Each count represents the number of times the COBOL verb itself was executed and not the number of times a loop was executed. Using the PERFORM UNTIL as an example, Analyzer counts the number of times the PERFORM verb was executed and not the number of times the PERFORM Range was executed as a result of UNTIL.

C Two types of paragraph exit indicators may appear. They are EXIT FROM PERFORMED PARAGRAPH(S) and FALL-THROUGH TO NEXT PARAGRAPH messages.

The EXIT FROM PERFORMED PARAGRAPH(S) message appears at the end of each PERFORM Range. The execution count represents the number of times control entered the PERFORM exit sequence as a result of a PERFORM verb. Unlike the fall-through message, this message appears at each PERFORM Range exit point even if its applicable count is zero.

The fall-through message indicates that control is passed from the statement preceding the message to the paragraph name following the message. If control does not fall through or the paragraph name is not referenced, the message is omitted.

For example, suppose you have a paragraph that has a conditional GO TO verb in it that transfers control to the EXIT verb for that paragraph. A FALL THROUGH TO NEXT PARAGRAPH message will be inserted before this label. This will be used to count the number of times that the exit paragraph has fallen through as opposed to the number of times it was branched to. To determine the number of times a label was branched to, subtract the counts on the FALL THROUGH TO NEXT PARAGRAPH message from the counts on the EXIT verb.

If both the fall-through and PERFORM exit messages appear together, their respective counts indicate how many times the paragraphs exited through the PERFORM linkage versus passing to the next statement.

D The CPU time for each source statement is expressed as a percentage of the program's total CPU time. When the percentage for a statement is less than one-tenth of one percent, it is not listed.

If the program calls another program, all of the CPU time used in the called program is charged to the CALL verb. All CPU time used by system functions is charged to the verbs which invoked those functions, such as OPEN, CLOSE, READ, WRITE, and so forth.

Thus, all CPU time used by this program, including that of any program subroutines and operating system functions it invokes, is ascribed to the appropriate source statement.

There is only one exception to this, when READY TRACE is used. When there is an unreferenced paragraph name, the TRACE time that normally would be attributed to that paragraph name falls instead on the preceding statement.

E At a quick glance, the histogram shows any high concentrations of CPU time. It also indicates which statements received little or no CPU time concentration. The source statement containing the highest concentration of CPU time shows a full histogram line, and all other source statements are proportional to it.

F If the program abends during execution, a message prints at the far right, flagging the statement that caused the abend. If the abend occurred in a called program or system module, Analyzer flags the statement responsible for the call.

If multiple executions are being consolidated and multiple abends have occurred, each abend is identified by the message to the right containing an abend number. The abend number relates directly back to the Run Summary Report and matches the number assigned to the execution step in which the abending program is contained.

Change Summary Report

This report contains a brief extract from the SOURCEMAP Report that targets selected source statements that have been tagged with a change.

The Change Summary Report is generated immediately following the SOURCEMAP Report for each program. It contains only those statements that have been tagged as changed during application development.

COMPUTER ASSOCIATES CA-OPTIMIZER		7.0	0106ARX00	YEAR 2000 CHANGE SUMMARY REPORT			PAGE 1 15 JUN,2001 12.01.29
A PAGE#	B STMT#	C TYPE	D COUNTS	E SOURCE IMAGE	F %	HISTOGRAM	
---	---	---	---	..1...+...2...+...3...+...4...+...5...+...6...+...7..	---	---	-----
11	000391	CAWINUP	3	CALL 'CAWINUP'	17.5	*XXXXXXXXXXXXX	
11	000397	CAWINUP	3	CALL 'CAWINUP'		*	
11	000403	Y2K	3	COMPUTE DAYS-BETWEEN = 365 *		*	

The report displays the following information.

- A** The page number where the statement resides in the SOURCEMAP report or the listing file for the program.
- B** The sequence number for the statement.
- C** The token that was used to tag the statement.
- D** The number of times that the statement was executed.
- E** The 80-byte source image for the statement.
- F** The CPU percentage, including the histogram report, if timings were taken for the program during execution.

The entries in the tag table are processed sequentially, so the first tag to be identified on any line will be reported in the Change Summary Report. Even if a source statement matches more than one tag, only the first matching tag will be reported.

The percentage and histogram fields in the Change Summary Report are not available if XTIME is not in effect for the execution, or if none of the source statements in the report have any timings attributed to them.

The Change Summary Report cannot be produced without a SOURCEMAP Report or without the XCOUNT option in effect.

Report of Unexecuted Paragraphs

This report lists each paragraph name that did not execute (received no counts) for this Analyzer run. This information can be very valuable in debugging and testing. There are many possible reasons why a paragraph may not have executed, including logic errors or inadequate test data. The report is organized in a dual column fashion from left to right.

COMPUTER ASSOCIATES
CA-OPTIMIZER

7.0 0106ARX00

UNEXECUTED PARAGRAPHS

PAGE 1
15 JUN, 2001 11.45.18

PROGRAM-ID = (CAOMDEMO)

A	B	C	D	A	B	C	D
PAGE	STATEMENT #			PAGE	STATEMENT #		
NO	COMP/SOURCE *-----PARAGRAPH NAME-----*			NO	COMP/SOURCE *-----PARAGRAPH NAME-----*		
13	00536 /005360 GTNN3			13	00538 /005380 GTY3		
13	00543 /005430 GTY1			13	00545 /005450 GTY4		
13	00550 /005500 GTNN4			13	00552 /005520 GTY2		
13	00556 /005560 GTYY5			14	00560 /005600 GTYN5		
14	00562 /005620 GTYY6			17	00697 /006970 ID4-2		
18	00726 /007260 CHECK-0			20	00798 /007980 PRINT-ARRAY-ENTRIES		
20	00806 /008060 PRINT-LIST-3			21	00814 /008140 PRINT-RL1		
21	00823 /008230 PRINT-RL2			21	00832 /008320 PRINT-RL3		
21	00841 /008410 PRINT-RL4			21	00850 /008500 PRINT-RL5		
22	00859 /008590 PRINT-ARRAY			22	00865 /008650 LINE-LOOP		
22	00869 /008690 ELEM-LOOP			22	00881 /008810 LINE-RITE		
23	00896 /008960 LOAD-TABLE						
TOTALS FOR (CAOMDEMO):				23 UNEXECUTED PARAGRAPHS			
E				50 EXECUTED PARAGRAPHS			
				73 TOTAL PARAGRAPHS			

This report displays the following information:

- A** The source listing page number where the paragraph begins.
- B** The compiler assigned statement number.
- C** The programmer assigned statement number.
- D** The paragraph name.
- E** The total number of executed and unexecuted paragraphs.

Paragraph Execution Time Summary

This report identifies the 10 most time-consuming paragraphs in the program. Analyzer lists them in order by the amount of time consumed, with the paragraphs consuming the most time appearing first. These may be the paragraphs most worthwhile to tune. The report shows less than 10 paragraphs, when the program itself contains less than 10, or when less than 10 have executed.

When the Sourcemap Report consolidates multiple executions of a program, the time measurement data is consolidated to show the overall 10 most time-consuming paragraphs.

COMPUTER ASSOCIATES			PARAGRAPH EXECUTION TIME SUMMARY			PAGE 2	
CA-OPTIMIZER			7.0 0106ARX00			15 JUN,2001 14.52.53	
PROGRAM-ID = (CAOMDEMO)							
A	B	C	D	E	F	G	
PAGE	STATEMENT #			CPU TIME			
NO	COMP/SOURCE *-----PARAGRAPH NAME-----*			(SECONDS)	PERCENT		
15	00609	/006090	LOOP-3	.38	39.8	* XX	
10	00397	/003970	DEMO-PARM-MESSAGE	.01	1.0	*	
15	00606	/006060	LOOP-3-S	.01	.8	*	
9	00352	/003520	DEMO-INIT	.00	.5	*	
10	00414	/004140	LOOP-COMPUTE	.00	.5	*	
23	00885	/008850	FORCING-THE-ABEND	.00	.4	*	
CPU OVERHEAD FOR PARAGRAPH FLOW (PFLOW) SUPPORT				46.1%			
CPU OVERHEAD FOR EXECUTION COUNT (XCOUNT) SUPPORT				10.9%			
END OF PARAGRAPH EXECUTION TIME SUMMARY FOR (CAOMDEMO)							

These are the contents of the Paragraph Execution Time Summary.

- A** The source listing page number where the paragraph begins.
- B** The compiler statement number.
- C** The COBOL line number. If the source sequence numbers are not coded in the program identified by PROGRAM-ID, this field is blank.
- D** The paragraph name.
- E** The CPU time (in seconds) consumed by each paragraph.
- F** The percentage of the total program CPU time devoted to each paragraph.
- G** The histogram showing the CPU time distribution. The paragraph name with the highest time consumption shows a full histogram line, the other lines are proportional to it.

Execution Count Cross-Reference Report (CNTXREF)

This report groups all the source statement numbers by the number of times they were executed. You can tell at a glance which groups of statements were executed the same number of times and are therefore possibly related. You can use the Execution Count Cross-Reference Report in maintenance by comparing counts between runs to make sure changes were made correctly.

This report is your major quality assurance tool. If the count field on the first line of the report is **not zero**, then every program statement was executed at least once. Thus, a quick look at the first line of the report's first page tells you how thoroughly the program has been tested.

COMPUTER ASSOCIATES CA-OPTIMIZER		7.0 0106ARX00	EXECUTION COUNT CROSS REFERENCE (CNTXREF) PROGRAM-ID = (CAOMDEMO)						PAGE 1 15 JUN,2001 14.52.53
A COUNT	B TOTAL	C COBOL LINE NUMBERS							
0		00366--00367 00370--00371 00374--00375 00378--00379 00385--00387 00389--00391							
.		00393--00395 00409--00412 00425 00445 00488--00489 00530							
.		00536 GTNN3							
.		00537							
.		00538 GTY3							
.		00539 00542							
.		00543 GTY1							
.		00544							
.		00545 GTY4							
.		00546--00547							
.		00550 GTNN4							
.		00551							
.		00552 GTY2							
.		00553							
.		00556 GTYY5							
.		00557							
.		00560 GTYN5							
.		00561							
.		00562 GTYY6							
.		00563 00627--00628 00631--00633 00636 00641 00642 00644 00645 00646 00647							
.		00648 00659--00660 00662--00663 00667--00668 00670--00671 00675--00683 00688--00690							
.		00694--00695							
.		00697 ID-2							
.		00698--00701 00704--00705 00723							
.		00726 CHECK-0							
.		00727--00733 00736 00742 00743 00747 00748 00772 00773 00774 00775 00776							
.		00777--00779 00783--00785 00789--00791							
.		00798 PRINT-ARRAY-ENTRIES							
.		00801--00805							
.		00806 PRINT-LIST-3							
.		00808--00812							
.		00814 PRINT-RL1							
.		00815--00821							
.		00823 PRINT-RL2							
.		00824--00830							
.		00832 PRINT-RL3							
.		00833--00839							
.		00841 PRINT-RL4							
.		00842--00848							
.		00850 PRINT-RL5							
.		00851--00857							
.		00859 PRINT-ARRAY							
.		00860--00863							
.		00865 LINE-LOOP							
.		00866--00867							
.		00869 ELEM-LOOP							
.		00870--00879							
.		00881 LINE-RITE							
.		00882--00883 00887 00891							
.		00896 LOAD-TABLE							
0	194	00897							
1		00346 PROCEDURE							
.		00350							

The report fields are as follows.

A

This column is the execution count (XCOUNT) value. A series of lines showing the same count values begins and ends with the count number. Intervening dots indicate that the same count prevails.

Note: When a count range extends beyond a page, the new page repeats the count column digit in parentheses indicating that the count is continued from the previous page.

- B** The last line of a count group gives the total number of program statements having the specified execution count.
- C** The six-digit compiler statement numbers belonging to each count category are listed. Groups of consecutive statements with the same counts are shown by the first statement number, two hyphens (--), and the last statement number.

When the statement number represents a paragraph name, that name prints following the statement number on a separate line.

If the NUM option is specified at compile time, Analyzer uses six-digit source statement numbers.

Instruction Detail Report (INSTRMAP)

This address-oriented report is language-independent and can be produced for any CSECT. The Instruction Detail Report lists only those instructions, known as hot spots, that use a significantly greater amount of CPU time than the others in the program. A hot spot may be a single, high CPU time instruction, such as a move long (MVCL) or it may be a group of instructions performing some very frequently executed function. Knowing the location of these instruction hot spots can help you reduce the program's total execution time.

COMPUTER ASSOCIATES			INSTRUCTION DETAIL REPORT						PAGE 1						
CA-OPTIMIZER			7.0 0106ARX00						15 JUN,2001 14.52.53						
REPORTER RUN IDENTIFICATION			15 JUN,2001 14.52.53			COMPUTER ASSOCIATES									
CPU TIME BY INSTRUCTION IN CSECT (CAOMDEMO)															
C			D			E			F			G		H	
- 6 HEX LOC + 6 HEX			CSECT ADDRESS		PERCENT OF CSECT MODULE TOTAL			CUMULATIVE PERCENT OF CSECT MODULE TOTAL			INSTRUCTION CLUSTERING		PERCENT OF CSECT		
												.X			
												.XX			
												.X			
												.XXXXXXXXXXXXXXXXXXXX			
												.X			
												.X			
												.XXXXXXXX			
												.			
												.			
												.XXXXXXXX			
												.XXX			
												.XXXXXXXX			
												.XXX			
												.XXXXXXXX			
												.X			
												.X			
												.XXXXXXXX			
												.XX			
												.XXXXXXXX			
												.X			
												.XX			
												.XXXXXXXX			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			
												.X			

The previously shown Instruction Detail Report has the following fields.

- A** Identifies the Analyzer reporting run that produced this report.
- B** The CSECT identification.
- C** The contents of storage, six bytes before and six bytes after, help to further identify the instruction.

Except for branch destinations, the -6 HEX column contains the hexadecimal instruction that used the CPU.

The hexadecimal data should match the hexadecimal instruction of that part of the code on your CSECT listing. If not, the code may be from a different listing, or the program may have modified its own instruction code during execution.

If your program contains instructions modified during runtime, and if both the original and modified instructions use large amounts of CPU time, Analyzer prints the original and modified instructions on separate lines.

- D** Analyzer identifies each instruction appearing in this report by its offset relative to the CSECT origin.
- E** The percent columns relate the CPU time spent at each CSECT address as a percentage of the total CPU time spent in the CSECT, load module, and entire execution run.
- F** The data in the Cumulative Percent of columns shows the amount of CPU time spent at all addresses up to and including this one. As a result, the last figure in the CSECT column of the Cumulative Percent of heading is always 100 percent. The last figure in the MODULE column under the Cumulative Percent of heading is the percentage of the load module CPU time used by the CSECT. The percentage of the entire execution run consumed by the CSECT appears at the foot of the Cumulative Percent of TOTAL column.
- G** The Instruction-Clustering histogram shows tight clusters of hot spot instructions in the program's code. There may be groups of instructions that perform some heavily executed function. If a group of instructions appearing in this report forms a cluster of consecutive addresses, then the instruction-clustering histogram shows high values for these instructions. On the other hand, an instruction that appears on this report and is relatively isolated from any of the others, indicates a single high CPU time instruction in an otherwise low CPU time set of code. The clustering histogram value for this instruction is low.
- H** The Percent of CSECT histogram graphically represents the CSECT column in the Percent Of heading.

Usage Considerations

Analyzer with IMS

Analyzer can be used in IMS environments. A complete discussion on how to use Analyzer under IMS can be found in *CA-Optimizer Getting Started*.

Analysis of Counts Results

Each process in a program is usually associated with a distinctive count, and each distinctive count is usually associated with only one process. If a statement exhibits a distinctive count, it is probably part of the indicated process.

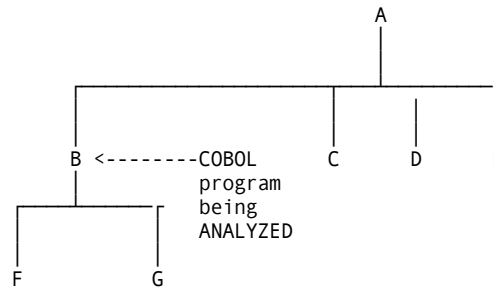
The patterns of the counts as they relate to the various routines in your programs are significant and can help you bridge the gap between what your program is supposed to be doing and what it is doing. Certain types of routines normally exhibit certain count characteristics.

Types of Routines	Count Values
Initialization	1
Termination	1
Some Error	0
Main Loop	A distinctive count usually tied to the number of input records.
I/O	A distinctive count equivalent to the number of records read or written, or the number of lines printed.
Inner Loops	Distinctive counts considerably higher than the Main Loop Count.
Record Type, Exception Case, and Control Break	Distinctive counts lower than the Main Loop Count.
Other Routines	Often sums or multiples of distinctive counts.

Interpretation of Timings

For Multimodule Environments

The timings on the Analyzer Sourcemap Report take on a different meaning in a multi-module environment than in single-module executions. For example, assume that in the diagram below, program B is the only COBOL program compiled using XTIME.



The Execution Monitor begins to gather CPU time samples when program B gets control. It collects sampling information on program B and its CALL statements. It only gathers CPU time samples on A, C, D, or E if they are given control after B first gained control. The total time reported is the CPU time consumed by program B, the programs downstream from it (programs F and G), and possibly some portion of A, C, D, and E. This means that when a verb in program B is reported to have used 10 percent of the CPU time, it has used 10 percent of the total CPU time consumed by programs B, F, and G. The CPU time for a verb which includes a transfer of control out of the program, for example CALL, WRITE, or SORT, includes the CPU time consumed by the routines that were called by that verb. It is important to remember this when you use the Sourcemap Report in conjunction with the Module Summary Report which shows CPU time by CSECT and load module.

Using the same example, assume that you want to get timings for the entire application starting with program A. If A was optimized with **any** of the Detector or Analyzer options then simply include the following in the JCL at execution time:

```
// CAIOPTS DD *
           XTIME FIRST
/*
```

Finally, using the same example, assume you want to get timings for the entire application starting with program A, but A is an Assembler program. You can accomplish this by executing the Execution Monitor (CAOCEM00) and then passing the application job step program name through the PGM option. Assuming the application job step program name is APP1, the execution time JCL is as displayed following.

- When you wish to perform in-depth CPU time analysis. Once the data has been captured the Analyzer Reporter can be run any number of times against that same input data. This is very useful for large systems consisting of tens, even hundreds, of modules and where there is a mix of COBOL and Assembler. In this case, in the initial run you might specify NOCNTXREF ALL and NOSOURCEMAP ALL to get just the summary reports. These reports tell you what modules to look at in more detail. If they are COBOL modules, use the SOURCEMAP option to get one or more detail reports. If they are Assembler modules, use INSTRMAP to get your detail reports.

The conditions under which automatic invocation occurs in the same job step as the application are as follows:

- When Counts or Timings (XCOUNT or XTIME) are active for one or more programs in the application.
- When XREPORT is ON. (This is the default.)

Automatic invocation is designed primarily for a one-time use of Analyzer where TIMSV data is not kept when the job is completed. If you do not supply a TIMSV DD statement in the execution JCL, one is dynamically allocated for you. Therefore, if you wish to save the TIMSV data you must supply your own DD statement. For more details on the TIMSV file and on dynamic allocation, consult the 'Usage Considerations' section of the "Using Detector" chapter.

Tagging a Source Statement for the Change Summary Report

When source changes are made during application development, the affected statements may be tagged by placing a user-defined token in the sequence (1 through 6) or serial (73 through 80) columns. Tags must be placed on executable COBOL statements, not on separate comment lines.

In the sequence columns, a tag can consist of any characters and can be any length from 1 through 6 characters. It may begin in any of the first six columns.

In the serial columns, a tag can consist of any characters and can be any length from 1 through 8 characters. It can begin in any of the last eight columns, 73 through 80.

A typical tag may contain the programmer's initials followed by the year and month in which the changes were made. This type of tag is particularly useful when isolating changes made by a specific programmer, or in a specific month or year. In the example following, the tag SD9806 is used for a new paragraph that was added in June, 1998.

```
SD9806 NEW-PARAGRAPH.  
SD9806     CALL 'NEWSUB' USING NEW-DATA NEW-RC.  
SD9806     IF NEW-RC > ZERO  
SD9806         DISPLAY 'ERROR IN NEW SUBROUTINE, RC=' NEW-RC  
SD9806         MOVE SPACES TO LS-DATA  
SD9806         MOVE 8      TO LS-RC  
SD9806     ELSE  
SD9806         MOVE NEW-DATA TO LS-DATA  
SD9806         MOVE ZERO    TO LS-RC.
```

To include this enhancement in the Change Summary Report, you could define any number of tags, including the entire tag (SD9806), the programmer's initials (SD), or the date (9806). Any of these tag definitions would cause the statements to be included in the Change Summary Report.

Restrictions

Analyzer cannot operate under CICS, CMS, HOGAN, IDMSDC, SHADOW, or when using multitasking, with the exception of IMS/VS message processing regions. For more information on using Analyzer in IMS environments, see “Special Environments” in *CA-Optimizer Getting Started*.

The Analyzer Reporter is upwardly compatible with TIMSV files and CSL libraries produced by CA-Optimizer Version 4.2 and higher.

CMS, TSO, and CA-Roscoe Support Facilities

This chapter describes the various display panels used with:

- CMS Support
- TSO Support
- CA-Roscoe Support

CMS Support

The CMS support feature allows you to Optimize COBOL programs interactively in the CMS environment. The object code generated by CA-Optimizer on CMS may be executed on either CMS or MVS. The Analyzer and Detector features are not supported under CMS.

Once the product has been installed on CMS, use the CAOPT EXEC provided to access the CA-Optimizer Utilities Menu.

```

CA-OPTIMIZER 7.0 0106A0M00                                UTILITIES

1) OPTIMIZE                                                6) BROWSE ERROR LISTING
2) UTILITIES                                              7) PRINT OPTIMIZED LISTING
3) HELP                                                  8) ERASE LISTINGS
4) EDIT COBOL PROGRAM                                    9) EXECUTE OPTIMIZED PROGRAM
5) BROWSE OPTIMIZED LISTING                             A) TRANSFER CSL TO MVS
X) EXIT

ENTER SELECTION NUMBER AND PROGRAM NAME
- - - - - DEFAULT PROGRAM NAME IS CAOMDEMO - - - - -

```

The Utilities Menu allows you to EDIT, OPTIMIZE, and EXECUTE a COBOL program and also BROWSE and ERASE the output listings. Simply select the desired function from the menu along with a program name. For example to Optimize a program called TESTPGM simply enter:

```
1  TESTPGM
```

If no program name is specified, the current default is used. Each time a program name is specified it becomes the default name for the next time. When the EXEC is first invoked the default name is CAOMDEMO, the CA-Optimizer demonstration program.

Following is a description of each of the functions.

1) OPTIMIZE	Invokes CA-Optimizer to compile and optimize the specified COBOL program.
2) UTILITIES	Displays the UTILITIES menu.
3) HELP	Invokes the product HELP facility.
4) EDIT COBOL PROGRAM	Invokes the CMS editor to edit the specified COBOL program.
5) BROWSE OPTIMIZED LISTING	Invokes the CMS BROWSE facility to browse the specified optimized listing.
6) BROWSE ERROR LISTING	Invokes the CMS BROWSE facility to browse the specified ERROR listing.
7) PRINT OPTIMIZED LISTING	Prints the Optimized listing of the specified program.
8) ERASE LISTINGS	Erases the Optimized listing of the specified program.
9) EXECUTE OPTIMIZED PROGRAM	Executes the specified COBOL program.
A) TRANSFER CSL TO OS	Transfers the specified CMS file to OS.
X) EXIT	Terminates the CAOPT EXEC.

Option 1: OPTIMIZE

If you select Option #1 (OPTIMIZE) the Default Options panel is displayed. This panel contains a list of the installation defaults and their values.

You are prompted to enter any COBOL or OPTIMIZER option overrides, separated by spaces (not commas).

CA-OPTIMIZER 7.0 0106A0M00		DEFAULT OPTIONS
BUFMAX.....0	LINECNT.....57	SOURCE.....YES
CICS.....NO	MDMAP.....NO	STATS.....YES
CLIST.....NO	MLIST.....NO	SUBRNGCK.....NO
COPT.....YES	OBJCMS.....NO	SUPMAP.....NO
CSL.....NO	OBJMVS.....YES	TPMON.....NO
DECK.....NO	OPTRC.....0	TRACE.....NO
DMAP.....NO	PFLOW.....NO	VS.....YES
DTECT.....NO	PMAP.....NO	V2.....NO
FDCHECK.....NO	RDIS.....NO	WSINIT.....NO
HOGAN.....NO	SHADOW.....NO	XCOUNT.....NO
IDMSDC.....NO	SIZEMAX.....0	XMDMAP.....YES
OPTIMIZE CAOMDEMO COBOL *		
ENTER OPTIONS		
ENTER QUIT TO DISCONTINUE OPTIMIZATION		

If an option keyword requires the equal (=) sign, append the equal sign to the end of the keyword, but enter the operand as a separate token, separated from the equal sign by a space.

Example:

SELECT= (3-20)

The following limitations apply to options:

- Limit options to eight-bytes in length, but do not break them into two separate words.
- A dash (-) at the end of the options line indicates that another line of options follows.
- Enter a maximum of 25 tokens on any one line.

Option 9: EXECUTE OPTIMIZED PROGRAM

Option 9 (EXECUTE) displays the Execution panel. This panel prompts you to enter any FILEDEF statements that are required to execute the optimized program.

```
CA-OPTIMIZER 7.0                                EXECUTION

ENTER ANY FILEDEF COMMANDS WHICH ARE NEEDED TO EXECUTE THE APPLICATION.
CP COMMANDS MAY BE ENTERED BUT MUST BE PRECEDED BY "CP".

ENTER "QUIT" TO DISCONTINUE EXECUTION.
ENTER "GO" TO BEGIN EXECUTION.
```

Option A: TRANSFER CSL TO OS

If you select Option A (TRANSFER CSL TO OS) the CSL Transfer panel is displayed. This panel prompts you to enter the name of the receiving virtual machine.

The OSJCL file should be received and submitted to OS to transfer the CSL file.

```
CA-OPTIMIZER 7.0                                TRANSFER CSL

ENTER THE NAME OF THE RECEIVING VIRTUAL MACHINE WHERE THE JCL FILE
'CAPASS OSJCL' WILL BE PUNCHED.

ENTER "QUIT" TO DISCONTINUE TRANSFER.
```

TSO Support

The TSO Support feature allows you to optimize COBOL programs interactively. To invoke the CA-Optimizer TSO facility, select it from your ISPF Master Menu panel. The facility guides you in optimizing, linking, and executing COBOL programs.

CA-Optimizer Primary Option Menu

```

----- CA-OPTIMIZER PRIMARY OPTION MENU -----CA-OPTIMIZER
SELECT OPTION ===>

 1 OPTIMIZE      - Invoke CA-OPTIMIZER (foreground or background)
 2 BROWSE        - Browse Optimized Listing
 3 ERRORS        - Browse COBOL Error Listing
 4 PRINT         - Print Optimized Listing
 5 DELETE        - Delete Optimized Listing and Error Listing
 6 EXECUTE       - Invoke Optimized Program in the background
 7 MRS          - Execute the Management Reporting System in background
 X EXIT         - Terminate CA-OPTIMIZER Dialogs

Enter END command to terminate CA-OPTIMIZER DIALOGS

```

The Primary Option Menu is displayed when you first enter the CA-Optimizer TSO Facility and after you have completed each function. The return code from the last requested function is displayed unless the return code was zero, in which case the return code display area is left blank.

These are the function codes:

- | | |
|------------|--|
| 1 OPTIMIZE | Initiates an optimization run. You can supply options to override those set in the default option CSECT before proceeding with optimization. |
| 2 BROWSE | Displays the CA-Optimizer listing file on the terminal. |
| 3 ERRORS | Displays the COBOL compiler's error listing file on the terminal. |
| 4 PRINT | Sends the CA-Optimizer listing file to the printer. |
| 5 DELETE | Deletes the CA-Optimizer listing file. |
| 6 EXECUTE | Invokes the optimized program in the background. |

7 MRS Executes the CA-Optimizer Management Reporting System in the background.

X EXIT Terminates ISPF using LOG and LIST defaults.

You can invoke HELP for CA-Optimizer panels at any time. Simply press PF1 or PF13, or type **HELP** and press Enter and HELP for the current panel displays.

To return to the panel from which you called help, press PF3 or Enter.

Invocation Panel

You invoke this panel from the Primary Option Panel (option 1). It allows you to tell CA-Optimizer whether you want to run the product in the foreground or in the background. Additionally, you can specify the name of the CA-Optimizer Load Library that is required for foreground processing, but optional for background processing. However, the OS/VS COBOL Load Library is optional for background processing, and is ignored for foreground processing. The CSL and STATS files are also optional.

```
----- CA-OPTIMIZER INVOCATION -----CA-OPTIMIZER
SELECT OPTION ==>
  1 BACKGROUND   - BACKGROUND COMPILE/OPTIMIZE
  2 FOREGROUND   - FOREGROUND COMPILE/OPTIMIZE

                        CA-OPTIMIZER DATA SETS
OPTIMIZER      LOAD LIBRARY ==>
                VOLUME SERIAL ==>          (IF NOT CATALOGED)
OPTIMIZER LPA  LOAD LIBRARY ==>
                VOLUME SERIAL ==>          (IF NOT CATALOGED)
OS/VS COBOL    LOAD LIBRARY ==>
                VOLUME SERIAL ==>          (IF NOT CATALOGED)

Condensed SOURCE LISTING ==>
                VOLUME SERIAL ==>          (IF NOT CATALOGED)
Statistics FILE  ==>
                VOLUME SERIAL ==>          (IF NOT CATALOGED)

                        OS/VS COBOL COPY LIBRARIES
LIB1 ==>
LIB2 ==>
LIB3 ==>
Enter END COMMAND TO RETURN TO PRIMARY OPTION PANEL
```

Output File Information Panel

This panel is displayed when you select either BACKGROUND or FOREGROUND on the Invocation Panel. It allows you to specify the type of file output you desire: hard copy or disk, as well as the name of the data set which is to contain the OBJECT module, and the JOB statement information which is required for background submissions.

```

----- CA-OPTIMIZER OUTPUT FILE INFORMATION -----
COMMAND ==>

Enter DSNNAME which will contain the OBJECT module (Required):
Object Data Set ==> 'CA0000.PDS.OBJ(VLC301)'

To print your listing, enter the SYSOUT Class:
SYSOUT Class ==>

To save the output listing to disk, enter the DSNNAME:
Listing Data Set ==> 'CA0000.'

JOB STATEMENT INFORMATION: (Required for background compile)
==> //CA0000A JOB (ACCOUNT),'NAME'
==> //*
==> //*
==> /*

Enter END to exit

```

Defaulted Options Panel

The Defaulted Options panel is the last panel displayed before CA-Optimizer is invoked. It displays the CA-Optimizer Version and Genlevel numbers, as well as many important user-defined defaults. It also prompts you for the input data set that contains the OS/VS COBOL source code, and any option overrides to those defaults set at installation. It is possible to override these options by entering the alternative options as shown on the following panel.

Note: Separate options with commas as shown in the following example.

Example: APOST,MDMAP

Do not code apostrophes ('); the dialog automatically inserts them for you.

```

----- CA-OPTIMIZER RELEASE 6.1 GENLEVEL 97110M61 -----CA-OPTIMIZER
COMMAND ==>

      Enter Options

PARM Information ==>
      ==>

Source Data Set ==>
Volume Serial ==>      (if not cataloged)

      DEFAULTED OPTIONS

CICS..... No    EZTEST... No    PMAP..... Yes    SUPMAP... No
CLIST.... Yes    FDCHECK.. No    RDIS .... No    TPMON.... No
COPT..... Yes    HOGAN.... No    SHADOW... No    TRACE.... No
CSL..... No    IDMSDC... No    SOURCE... Yes    WSINIT... No
DECK..... No    MDMAP.... No    STATS.... No    XCOUNT... No
DMAP..... Yes    MLIST.... No    SUBRNGCK. No    XMDMAP... No
DTECT.... No    PFLOW.... No

Enter END to exit

```

Note: The Defaulted Options are set at installation time by your Systems Programmer.

Browse CA-Optimizer Output Panel

This panel allows you to browse CA-Optimizer output (Option 2). However, you can only use this panel if the output data set has been placed in a disk data set. The dialog remembers the name of the last output data set used, and displays it on the panel prior to invoking BROWSE. If you would like to BROWSE a different data set, simply provide the data set name at the Listing Data Set prompt before pressing Enter.

```

----- BROWSE CA-OPTIMIZER OUTPUT -----
COMMAND ==>

To use this option, output from the compilation step (option 1) must have been
routed to a data set (not a SYSOUT Class).

Verify the listing data set information below, then press ENTER to BROWSE
output

Listing Data Set ==> 'CA0000.'

Enter END to exit

```

Browse CA-Optimizer Errors

This panel (Option 3) allows you to browse any errors that may have occurred during compilation. The CA-Optimizer dialogs automatically generate a data set name and allocate it to the system. This data set name is then displayed on the panel. You may change the name, if you wish, before pressing Enter.

```
----- BROWSE CA-OPTIMIZER ERRORS -----
COMMAND ==>

Verify the data set name listed below. If you want to BROWSE another data
set enter its name at the prompt

Errors Data Set ==> 'CA0.OM32767.ERRLIST'

Enter END to exit
```

Print CA-Optimizer Listing

This panel (Option 4 PRINT) prompts you for information necessary to print the listing data set. The listing must have been saved to disk. You must enter the following information: a SYSOUT class, and JOB statement information. When you press Enter, an IEBGENER job is submitted to allow you to print the data set.

Note: The END command allows you to EXIT without submitting the job.

```
----- PRINT CA-OPTIMIZER LISTING -----
COMMAND ==>

Verify the information below. The data set will be printed (but not deleted).

Listing Data Set ==> 'CA0000.'
Enter SYSOUT class ==> B

JOB STATEMENT INFORMATION:
==> //JOHNB JOB ('11700000B.SARAH',423),'JOHNB Q.',CLASS=K,
==> // MSGCLASS=B,MSGLEVEL=(1,1),NOTIFY=JOHNB
==> /*ROUTE PRINT MVSE ==>

Enter END to exit
```


Note: If any modules are to be dynamically loaded, they must be present in the linklist/link pack area, STEPLIB, or ISPLLIB.

Enter **END** or press PF3 or PF15 to execute the program. Use **CANCEL** to exit.

Invoke Management Reporting System Panel

The following panel (Option 7) prompts you for information needed to invoke the Management Reporting System (MRS). Information requested includes: the name of the CA-Optimizer Load Library (optional), the Load Library to be analyzed (required), the SYSOUT class or listing data set, and JOB card information. When you press Enter the job is submitted for background processing.

```

----- INVOKE MANAGEMENT REPORTING SYSTEM -----
COMMAND ==>
                                RELEASE      GENLEVEL
OPTIMIZER Load Library ==> 'CDE.OPTU.PROD.OPT60.CAILIB'
                                Volume Serial ==> (if not cataloged)
OPTIMIZER LPA Load Library ==> 'CDE.OPTU.PROD.OPT60.CAILPA'
                                Volume Serial ==> (if not cataloged)

Load Library for Analysis ==>
                                Volume Serial ==> (if not cataloged)

Enter SYSOUT class (if you desire the listing to be printed):
==>
Enter DSNAME (if you desire the output listing to be saved on disk):
==>

JOB STATEMENT INFORMATION:
==> //USRID02A JOB (ACCOUNT),'NAME'
==> /*
==> /*
==> /*

Enter END to exit

```

CA-Roscoe Support

The ROSCOE support provides the same functionality as in TSO/ISPF. To invoke the CA-Optimizer ROSCOE facility, select it from your Master Menu panel. Prior to using this feature, the member named OPTISPFJ, containing job statements or route statements, must be created under the users key.

CA-Optimizer Primary Option Menu

Depending on full-panel or split-panel mode, one of the Primary Option menus displays when you first enter the CA-Optimizer ROSCOE facility and after you have completed each function.

```
----- CA-OPTIMIZER PRIMARY OPTION MENU ----- CA-OPTIMIZER
SELECT OPTION ==>

  1 OPTIMIZE      - Invoke CA-OPTIMIZER (background)
  2 BROWSE        - Browse Optimized Listing
  3 ERRORS        - Browse COBOL Error Listing
  4 PRINT         - Print Optimized Listing
  5 DELETE        - Delete Optimized Listing and Error Listing
  6 EXECUTE       - Invoke Optimized Program in the background
  7 MRS          - Execute the Management Reporting System in background
  X EXIT         - Terminate CA-OPTIMIZER Dialogs

Enter END command to terminate CA-OPTIMIZER DIALOGS
```

These are the function codes:

- | | |
|------------|--|
| 1 OPTIMIZE | Initiates an optimization run. You can supply options to override those set in the default option CSECT before proceeding with optimization. |
| 2 BROWSE | Displays the CA-Optimizer listing file on the terminal. |
| 3 ERRORS | Displays the COBOL compiler's error listing file on the terminal. |
| 4 PRINT | Sends the CA-Optimizer listing file to the printer. |
| 5 DELETE | Deletes the CA-Optimizer listing file. |
| 6 EXECUTE | Invokes the optimized program in the background. |

- 7 MRS Executes the CA-Optimizer Management Reporting System in the background.
- X EXIT Terminates ISPF using LOG and LIST defaults.

You can invoke CA-Optimizer HELP panels for CA-Roscoe at any time. Simply press PF1 or PF13, or type **HELP** and press Enter and HELP for the current panel displays. PF3 or Enter brings you back to the panel from which you called HELP.

Invocation Panel

Invoke this panel from the Primary Option Menu (Option 1). This panel allows you to specify object data sets, input data sets containing source code, and compile options. The full panel also accepts the names of the CA-Optimizer load library, OS/VS COBOL library, SYSLIB, CSL, and STATS file. All entered information is saved in member OPTMPROF so that it can be recalled whenever needed.

```

----- CA-OPTIMIZER INVOCATION ----- CA-OPTIMIZER
COMMAND ==>                                RSP01F00
                                Libraries
OPTIMIZER Load Library      ==>
                                Load Library (LPA) ==>
OS/VS COBOL Load Library   ==>
OS/VS COBOL COPY Library   1 ==>
                                Library 2 ==>
                                Library 3 ==>
                                Output Datasets
Object Data Set             ==>
Listing Data Set (to browse) ==>
                                Sysout Class (to print) ==> (default to * if both blank)
Condensed Source Listing    ==>
                                Volume Serial ==> (if not cataloged)
Statistics File              ==>
                                Volume Serial ==> (if not cataloged)
                                Input Information
Source Data Set              ==>
                                Volume Serial ==> (If not cataloged)
PARM Information ==>
                                ==>

Enter END command to return to Primary Option Panel

```

Note: A pause prompt for the SUBMIT command, allows you to exit (by entering GO) without submitting the job.

Browse CA-Optimizer Output Panel

This panel allows you to browse CA-Optimizer output (Option 2 from the Main menu). However, you can only use this panel if the output data set has been placed in a disk data set. The dialog remembers the name of the last output data set used, and displays it on the panel prior to invoking BROWSE. If you would like to browse a different data set, provide the data set name at the Listing Data Set prompt before pressing Enter.

```
----- BROWSE CA-OPTIMIZER OUTPUT ----- CA-OPTIMIZER
COMMAND ==>

To use this option, output from the compilation step (Option 1) must have been
routed to a data set (not a SYSOUT Class).

Verify the listing data set information below, then press ENTER to BROWSE
output

Listing Data Set ==>

Enter END to exit
```

Browse CA-Optimizer Errors

This panel (Option 3) allows you to browse any errors that may have occurred during compilation. The CA-Optimizer dialogs automatically generate a data set name and allocate it to the system. This data set name is then displayed on the panel. You can change the name, if you wish, before pressing Enter.

```
----- BROWSE CA-OPTIMIZER ERRORS ----- CA-OPTIMIZER
COMMAND ==>

Verify the data set name listed below. If you want to BROWSE another data
set enter its name at the prompt

Errors Data Set ==>

Enter END to exit
```

Print CA-Optimizer Listing

This panel (Option 4) prompts you for information necessary to print the listing data set. The listing must have been saved to disk. You must enter the following information: a SYSOUT class, and JOB statement information. When you press Enter, an IEBGENER job is submitted to allow you to print the data set.

Note: A pause prompt for the SUBMIT command allows you to exit (by entering GO) without submitting the job.

```
----- PRINT CA-OPTIMIZER LISTING ----- CA-OPTIMIZER
COMMAND ==>

Verify the information below. The data set will be printed (but not deleted).

Listing Data Set ==>
Enter SYSOUT Class ==>

Enter END to exit
```

Delete CA-Optimizer Listing Data Sets

At this panel (Option 5) you may specify either one or two data set listing files for deletion. The data sets are deleted after you press Enter. The listing data set is optional.

```
----- DELETE CA-OPTIMIZER LISTING DATA SETS ----- CA-OPTIMIZER
COMMAND ==>

Verify the two data set names listed below.

If you do not want to delete these data sets: Enter the END command,
or press ENTER to delete them both.

Errors Data Set ==>
Listing Data Set ==>

Enter END to exit
```

Execute Optimized Program Panel

From this panel (Option 6) you can test (execute) a program that you optimized earlier. The system prompts you for your load library and load module that you have linked.

----- CA-OPTIMIZER EXECUTION -----		CA-OPTIMIZER
SELECT OPTION ==>		RSP06F00
E X E C U T I O N P A N E L		
Your Load Module	==>	
Your Loadlib	==>	
Your Syslib (if any)	==>	
List Data Set (to browse)	==>	
Sysout Class (to print)	==>	(default to * if both blank)
CAI Option Dataset	==>	
Step Name ==>	Volume Serial ==>	(if not cataloged)
	Region Size ==>	(in K)
Please fill in INPUT & OUTPUT DD statement at prompt of SUBMIT		

Note: At the pause, first attach the AWS and fill in the input and output data sets that are needed to execute this program. Then, submit the job. You may enter GO to exit this panel at any time.

Invoke Management Reporting System Panel

The following panel (Option 7) prompts you for information needed to invoke the Management Reporting System (MRS). Information requested includes: the name of the CA-Optimizer Load Library (optional), the Load Library to be analyzed (required), the SYSOUT class or listing data set, and JOB card information. When you press Enter the job is submitted for background processing.

```
----- INVOKE MANAGEMENT REPORTING SYSTEM ----- CA-OPTIMIZER
COMMAND ==>                                         RSP07F00

CA-OPTIMIZER      Load   Library:
                  Volume Serial ==>                (if not cataloged)

CA-OPTIMIZER (LPA) Load Library:
                  Volume Serial ==>                (if not cataloged)

Load   Library   for   Analysis:
                  Volume Serial ==>                (if not cataloged)

Enter DSNAME (if you desire the output listing to be saved on disk):
=====
Enter SYSOUT class (if you desire the listing to be printed):
=====
                                   (both blank default to class *)

Enter END to exit
```


CA-Optimizer provides several reporting facilities to measure the use and effectiveness of the product. They are as follows.

- The Management Reporting System (MRS) provides a means for you to audit your load module libraries.
- The Statistics Report allows you to view the overall effectiveness of the optimization in your data center.
- The Initialization Summary generates a report describing which CA-Optimizer modules are currently loaded in the CSA area of LPA.
- The Options Summary reports on the options set as installation defaults.
- The CSL Summary reports on the contents of a given CSL or the on all the CSLs defined as installation defaults.
- The CAIPRINT Repository Utility provides a facility to list and maintain the contents of the central VSAM repository.

Management Reporting System (MRS)

The Management Reporting System (MRS) scans any load module library and reports on the COBOL programs contained within each member. MRS produces two reports, a detail report and a summary report, indicating the size and characteristics of each load module in a selected library.

MRS reads the members of a specified library, and identifies and lists COBOL CSECTs and load modules that contain them. It also provides compilation and size information for these CSECTS, and indicates whether a CA-Optimizer product has processed them. The program also checks for and flags certain CA-Optimizer options that you selected at compile/optimize time.

JCL

Computer Associates provides a procedure called CAIMRS to invoke MRS. To obtain detail and summary reports for library PROD1.LOADLIB, specify:

```
// EXEC CAIMRS,LIBRARY='PROD1.LOADLIB'
```

The MRS program, CAOUMRS, uses the following DD statements:

STEPLIB	Defines the library containing CA-Optimizer MRS.
SYSPRINT	Defines the MRS report file.
SYSLIB	Defines the library to be analyzed.

Reports

MRS provides two reports for each load library that is scanned. The first report is a detailed listing of all COBOL CSECTs in the library and the second is a summary of all of the library's statistics.

Detail Report

The Load Library Analysis Detail Report lists all load modules in the library that contain COBOL CSECTs.

COMPUTER ASSOCIATES				LOAD LIBRARY ANALYSIS: DETAILED REPORT							PAGE 1	
CA-OPTIMIZER 7.0 0106ARX00				USRID02.LOAD (SPD001)							21 SEP 2001 12.14.57	
LEGEND FOR OPTIMIZER OPTIONS: A												
D=DTECT C=XCOUNT I=CICS												
B	C	D	E	F	G	H	I	J	K			
MODULE	MODULE-SIZE	CSECT	CSECT-SIZE	PROC-SIZE	DATA-SIZE	COMPILER-ID	REL	DATE	TIME	CA-OPT	REL	DCI
CAOMDEMO	16,390	CAOMDEMO	16,390	8,858	7,532	VSR1 5740-CB1	2.4	07/18/2001	15.20.19	OPT/OS	7.0	DC.
CAOMTC2K	15,542	CAOMTC2K	15,542	8,372	7,170	VSR1 5740-CB1	2.4	01/17/2000	16.56.43	OPT/OS	6.1	.C.
LSTTOSRC	3,178	MDOGCOB	3,186	1,436	1,750	VSR1 5740-CB1	2.4	09/04/2001	14.32.03	OPT/OS	6.1	...
SNAPPROG	15,798	SNAPPROG	15,798	8,562	7,236	VSR1 5740-CB1	2.4	07/18/2001	14.41.13	NONE		
TOTALS: L	51,512		50,908	27,220	23,688							

A Defines the meaning of each letter that can appear under column **K**. These are all the options for execution time facilities that can be specified at compile/optimize time.

B Lists the names of the load modules that contain at least one COBOL CSECT.

C Specifies the total number of bytes in each load module listed.

Note: For overlay modules, the size specifies the maximum amount of memory used by the module and not the sum of the CSECTs contained within it.

- D** Lists all the COBOL CSECT names in each load module. Multiple COBOL CSECTs within a given load module print on a separate line.
- E** Specifies the total number of bytes in each COBOL CSECT.
- F** Specifies the number of bytes for the program's Procedure code. See Usage Considerations later in this chapter for more information.
- C** Specifies the number of bytes of storage generated by the compiler for the COBOL program's Data Division. See 'Usage Considerations' later in this chapter for more information.
- H** Identifies the type of COBOL compiler used and the compiler's program product number and version number. CA-Optimizer MRS copies the program product number and the version number directly from the COBOL module.
- I** Specifies the version of the COBOL compiler used and the date and time of the program's compilation (copied directly from the COBOL module).
- J** Verifies whether the COBOL CSECT has been processed with a CA-Optimizer product. The version number indicates the version with which the CSECT was optimized. YES indicates that the CSECT was optimized with an earlier version that did not retain release numbers in the load module. NONE indicates that the CSECT has not been processed with CA-Optimizer. In some instances, the compiler program product number, version number, and the date and time of compilation may not be available. In this case, an UNAVAILABLE message prints in their place.
- K** CA OPTIONS shows whether certain CA-Optimizer execution time options were specified at compile/optimize time. If the program was optimized using CA-Optimizer OS, then only the letters for DTECT, PFLOW, or XCOUNT may appear even though the options FDCHECK, SUBRNGCK, or WSINIT may also have been specified at compile time.
- L** This line specifies the following totals in KB (1024):
- Load modules listed on the report
 - COBOL CSECTs
 - Number of data bytes
 - Number of procedure bytes from the COBOL CSECTs

Summary Report

The Load Library Analysis Summary Report follows the Detail Report and provides summarized statistical information about the library being scanned.

COMPUTER ASSOCIATES CA-OPTIMIZER				LOAD LIBRARY ANALYSIS: SUMMARY REPORT USRID02.LOAD (SPD001)					PAGE 2 21 SEP 2001 12.14.57	
	7.0	0106ARX00								
COMPILER-ID	REL	MODULES	CSECTS	OPTOS	TC	II	DTE	XCO	CIC	
VSR1 5740-CB1 2.4		4	4	3	0	1	2	0		
TOTALS: COBOL		4	4	3	0	1	2	0		
ILBO		3	3							
CA0		33	33							
OTHER		72	694							
ALL		76	734							

A The number of modules containing COBOL CSECTs, the total number of COBOL CSECTs, the number of modules containing ILBO, IGZ, and DFH, CSECTs, and the total number of ILBO, IGZ, and DFH, CSECT's respectively. Also listed here are any other modules containing a CSECT and the total number of these CSECTs (not shown).

B The total number of modules and CSECTs in the library.

The next part of the Summary Report provides a summary of CA-Optimizer options in effect, by compiler type and version.

C COMPILER VERSION lists the compiler type and version used.

D The number of programs compiled under the listed compiler version, which were compiled and optimized with CA-Optimizer.

E The number of programs from column **B** that were optimized using the DTECT or XCOUNT options, respectively.

Usage Considerations

MRS analyzes any standard IBM load module library but does not support concatenated load library data sets. Specify each load library to be analyzed in a separate CA-Optimizer MRS execution run.

MRS recognizes COBOL CSECTs that have been compiled using the following versions of IBM COBOL compilers: V2, V3, V4, and OS/VS.

OS/VS COBOL Reporting

MRS makes a distinction between data bytes and procedure bytes in the Detail Report. Data bytes include the storage size in the CSECT that is directly generated by the COBOL CSECTs Data Division plus the INIT1 code. (Data bytes are figured from the beginning of the program to the TGT.) Procedure bytes include the storage size of the remainder of the CSECT. The sections of the CSECT that make up the procedure bytes are:

- TGT
- PGT
- Report Writer Routines
- COBOL Procedure Division
- Q Routines
- INIT2
- INIT3

Statistics Report

The CA-Optimizer Statistics Report lets you monitor the results obtained by the Optimizer component. Each time a program is optimized, the one-track statistics data set is updated. An independent procedure, CAOMSRPT, retrieves this data upon request and prints a one-page report.

JCL

Computer Associates provides a procedure called CAOMSRPT to invoke the Statistics Report program. To obtain a statistics report for your data center, specify:

```
// EXEC CAOMSRPT
```

The Statistics Report program, CAOMSRPT, uses the following DD statements:

STEPLIB	Defines the library containing the CA-Optimizer Statistics Report program.
SYSPRINT	Defines the report file.
CAISTATS	Defines the permanent statistics file

Report

This sample shows the information and the format of the Statistics Report.

COMPUTER ASSOCIATES DEVELOPMENT		CA-OPTIMIZER STATISTICS & USAGE SUMMARY		PAGE 1
CA-OPTIMIZER 7.0 0106AOM0				21 SEP,2001 12.59.34
LICENSED TO :		COMPUTER ASSOCIATES DEVELOPMENT		
		ISLANDIA, NY		
A	INSTALLED	06/01/2001		
OPTIMIZATION RESULTS				
		CURRENT	CURRENT	SINCE
		MONTH (NOV)	YEAR (1997)	INSTALLATION
AVERAGE PROGRAM SIZE *				
B	BEFORE OPTIMIZATION (BYTES)	50,596	50,596	50,596
	AFTER OPTIMIZATION (BYTES)	50,840	50,840	50,840
	PERCENTAGE SAVINGS	.4-	.4-	.4-
AVERAGE PROCEDURE SIZE *				
C	BEFORE OPTIMIZATION (BYTES)	1,196	1,196	1,196
	AFTER OPTIMIZATION (BYTES)	1,440	1,440	1,440
	PERCENTAGE SAVINGS	20.3-	20.3-	20.3-
AVERAGE INSTRUCTION COUNT *				
	BEFORE OPTIMIZATION	169	169	169
D	AFTER OPTIMIZATION	170	170	170
	PERCENTAGE SAVINGS	.2-	.2-	.2-
* THE TOTALS SHOWN DO NOT INCLUDE THOSE PROGRAMS OPTIMIZED WITH THE ANALYZER FEATURE.				
CA-OPTIMIZER USAGE SUMMARY				
NUMBER OF PROGRAMS OPTIMIZED				
		CURRENT	CURRENT	SINCE
	OPTIMIZATION TYPE	MONTH (JUN)	YEAR (2001)	INSTALLATION
E	DETECTOR	1	1	1
	ANALYZER	0	0	0
	DETECTOR AND ANALYZER	3	4	12
	STANDARD OPTIMIZATION	0	0	0
F	TOTAL PROGRAMS OPTIMIZED	4	5	13
		=====	=====	=====

- A** Date the statistics data set was initialized.
- B** Average number of bytes in the program (includes Data Division) before and after Optimization and the percentage difference. The numbers are the before and after compile time sizes. All figures are reported for month-to-date, year-to-date, and since installation.
- C** Average number of bytes in only the Procedure Division before and after Optimization, and the difference in the percentages.
- D** Average number of machine instructions in the program before and after Optimization and the difference in the percentages.

E Breaks down the number of programs optimized by the combination of Detector or Analyzer features specified during Optimization. It shows the relative usage of each of the components.

F Total number of programs using CA-Optimizer.

Note: These statistics are derived from all executions of the Optimizer component without regard to the options specified at compile/optimize time. Because the CA-Optimizer optimization algorithms sometimes trade space for speed, smaller than expected percentage savings do not necessarily reflect the actual reduced CPU time when the applications are executed.

Initialization Summary

The CA-Optimizer Initialization Summary allows you to verify that the modules loaded into the CSA at initialization time are the correct versions.

JCL

Computer Associates provides a procedure called CAILPAQ to invoke the Initialization Summary program. To obtain an initialization summary for your data center, specify:

```
// EXEC    CAILPAQ
```

The initialization summary program, CAOULPAQ, uses the following DD statements:

STEPLIB	Defines the library containing CA-Optimizer/II Initialization Reporting program.
SYSPRINT	Defines the report file.

Report

This sample shows the information and the format of the Initialization Report.

COMPUTER ASSOCIATES						PAGE 1
CA-OPTIMIZER 7.0 0106ARX00						
INITIALIZATION SUMMARY REPORT						
MODULE	ADDRESS	REL	DATE	TIME	SIZE	
A	B	C	D	E	F	
CAOESNAP	8094B7B8	3.0	06/04/01	16.41	003848	
CAOEWRTR	00A6B8B8	3.0	06/04/01	16.42	000210	
CAOECEEI	8AB95A08	3.0	06/04/01	16.41	0003D8	
CAOMID	00C1AA38				000018	
CAOCEMCK	0099A078	3.0	06/04/01	16.34	000238	
CAOEPROL	80A04900	3.0	06/04/01	16.41	0001C8	
CAOESQAT	8B3D1FC8				000038	
7 MODULES WERE FOUND. G						
CONTENTS OF CAOESQAT H						
+0 00000000 00000000 809F7B28 000003F8						
+10 84762000 8094B7B8 80A04900 00A6B8B8						
+20 FFFFFFD13 00000000 00000000 00000000						

- A** Module name.
- B** Address where module resides.
- C** Version of module.
- D** Date module was assembled.
- E** Time module was assembled.
- F** Size of module, in bytes.
- G** Number of CA-Optimizer modules found.
- H** Contents of module CAOESQAT (for support purposes).

CSL Summary

CA-Optimizer CSL Summary produces a report of all members in the Condensed Source Listing library.

JCL

Computer Associates provides a procedure called CAOUCSLD to invoke the Condensed Source Listing Summary program. To obtain a CSL Summary for your data center, specify:

```
// EXEC CAOUCSLD
```

The CSL summary program, CAOUCSLD, uses the following DD statements:

STEPLIB Defines the library containing CA-Optimizer CSL Summary program.

SYSPRINT Defines the report file.

CAICSL Defines the Condensed Source Listing library to be reported on.

Note: The ddname for the CSL is an installation option that may be modified. If it has been modified at your installation, be sure that procedure CAOUCSLD has been modified as well. Also note that if this DD is omitted, the default list of CSLs are reported on.

Report

This sample report shows the information and the format of the Condensed Source Listing Summary.

COMPUTER ASSOCIATES				PAGE	1
CA-OPTIMIZER	7.0 0106ARX00	14 JUN,2001		17:36:47	
CA-OPTIMIZER CONDENSED SOURCE LIBRARY					
DATA SET NAME = CDE.OPTU.PROD.OPT61.CSL					
A	B	C	D	E	F
PROGRAM ID	DATE COMPILED	TIME COMPILED	GENLEVEL	RELEASE	COMPILER
-----	-----	-----	-----	-----	-----
BOMB7	06 JUN,1901	14:42:24	01030M610	6.1	VS COBOL
CAOMDEMO	06 JUN,1901	13:44:41	01030M610	6.1	VS COBOL
TOTAL NUMBER OF VS COBOL PROGRAMS		=	2		
TOTAL NUMBER OF PROGRAMS		=	2		

- A** Program ID.
- B** Date the program was compiled.
- C** Time the program was compiled.

- D** The genlevel of CA-Optimizer used to create the CSL member.
- E** The version of CA-Optimizer used to create the CSL member.
- F** The compiler used.
- G** Totals for each CSL reported on.

CAIPRINT Repository Utility

This section covers the JCL, commands, and usage considerations of the CAIPRINT Repository Utility.

JCL

Computer Associates provides a procedure called CAIPRTU to invoke the CAIPRINT Repository Utility, CAOUPRTU. To display or maintain the contents of your repository, specify:

```
// EXEC CAIPRTU
//PRTU.SYSIN DD *
    (commands go here)
/*
```

CAOUPRTU uses the following DD statements:

- | | |
|----------|--|
| STEPLIB | Defines the library containing CA-Optimizer. |
| SYSPRINT | Defines the report file. |
| SYSUT1 | Defines a work file used for some requests. |

Commands

The syntax of any command is:

keyword <selection criteria>

where:

keyword Is one of the valid command keywords listed in this section.

selection criteria Is comprised of an argument and optionally an operator and a value.

Example:

```
LIST        DATE    GT   2001/03/19
```

In this example, the keyword LIST is used to display a summary of reports that were created in the repository after March 19, 2001. The selection criteria is comprised of the argument DATE, the operator GT (for Greater Than) and the value 2001/03/19.

Supported command keywords are:

COPY	Copy reports from a secondary repository (ODSN) to your primary repository (DSN). Requires that DSN and ODSN both be specified.
DELETE	Delete selected reports from a repository (DSN). Requires that DSN be specified.
DSN	Define the dsname for your primary CAIPRINT repository. Must be specified prior to using any other commands.
INIT	Initialize a newly defined repository (DSN). Requires that DSN be specified.
LIST	List a summary of reports contained within a repository (DSN). Requires that DSN be specified.
LOCK	Lock one or more reports in a repository (DSN) to prevent deletion. Requires that DSN be specified.
ODSN	Define the dsname for a secondary CAIPRINT repository. Must be specified prior to using the COPY command.
UNLOCK	Unlock one or more locked reports in a repository (DSN) to allow deletion. Requires that DSN be specified.

Valid arguments for selection criteria include:

AGE	Select reports by the number of days since they were created. This argument requires an operator and a one- to 3-digit value. Example: DELETE AGE GT 5
ALL	Select every report unconditionally, with no operator or value specified. Example: UNLOCK ALL
CODE	Select reports by the completion code of the step that produced the report. This argument requires an operator and a four- to seven-character value. Examples: LIST CODE EQ S0C7 (or) LIST CODE EQ U1125 (or) LIST CODE EQ RC0004 (or) LIST CODE EQ CEE3250

System abend codes must begin with S followed by three hexadecimal digits.

User abend codes must begin with U followed by four decimal digits. Return codes must begin with RC followed by four decimal digits. IBM message codes must be one- to seven- digits.

DATE Select reports by the date on which they were created. This argument requires an operator and a value in the format YYYY/MM/DD. Example:

```
DELETE      DATE  LT  2001/03/01
```

JOB Select reports by the name of the job that produced the report. This argument requires an operator and a one- to eight-character value. Example:

```
UNLOCK      JOB  NE  OPTJOB1
```

LOCK Select reports by the ID of the user that owns a lock on the report. This argument requires an operator and a one- to seven-character value. Example:

```
LIST        LOCK  EQ  OPTUSER
```

PGM Select reports by the name of the abending program that caused the report. This argument requires an operator and a one- to eight-character value. Example:

```
DELETE      PGM  EQ  CA02DEMO
```

STEP Select reports by the name of the step that produced the report. This argument requires an operator and a one- to eight-character value. Example:

```
LIST        STEP  NE  GO
```

USER Select reports by the ID of the user that submitted the job that produced the report. This argument requires an operator and a one- to seven-character value. Example:

```
LOCK        USER  EQ  OPTUSER
```

Valid operators include:

EQ Select reports if the value of the argument is equal to the value specified.

GE Select reports if the value of the argument is greater than or equal to the value specified.

GT Select reports if the value of the argument is greater than the value specified.

LE Select reports if the value of the argument is less than or equal to the value specified.

LT Select reports if the value of the argument is less than the value specified.

NE Select reports if the value of the argument is not equal to the value specified.

RANGE Select reports if the value of the argument is between the two values specified.

Usage Considerations

None.

Using Options

The initial sections of this chapter summarize CA-Optimizer component option usage; the last section of this chapter, Alphabetized Option Listing, is an alphabetical reference for all CA-Optimizer options.

Whenever an option has both a compile time and an execution time usage, the compile time usage is presented first. The term “option” refers to the control of functions at compile, postprocess, execution, report, and view time.

Compile Time Overview

Coding Options

Optimizer options, like OS/VS COBOL options, may be specified as follows:

1. PARM='opt1,opt2,opt3,...' on your EXEC statement, or
2. As specified in your installation defaults

You may freely combine your Optimizer options with OS/VS COBOL options on the EXEC statement. The processing of Optimizer options takes place on separate levels. The order of precedence for specified options is the same as for the OS/VS COBOL compiler:

- PARM='OPTION'
- Default options

Any conflicting options are processed on each level individually. The levels are then processed together for conflicts. See Order of Precedence in Establishing Option Values for more information.

Note: Any options not specified on the EXEC statement are determined by their values in the installation defaults.

Overriding Defaults

You can override the default values at compile/optimize time by specifying options in the parameter list on the EXEC statement.

Order of Precedence in Establishing Option Values

At compile time, once all options have been scanned, their processing takes place in the following order of precedence:

1. FDCHECK PFLOW SUBRNGCK or WSINIT	forces	DTECT
2. CICS DCI HOGAN IDMSDC SHADOW or TPMON	forces	NODTECT NOXCOUNT and NOXTIME
3. NODTECT	forces	NOPFLOW
4. XCOUNT	forces	CSL and MLIST
5. MLIST	forces	NOCLIST
6. CLIST	forces	NOPMAP
7. PMAP	forces	NOSELECT
8. CXREF	forces	NOXREF
9. MDMAP	forces	NODMAP

COBOL Postprocessor Time

Coding Options

The COBOL postprocessor options may be specified in two ways:

1. PARM='opt1,opt2,opt3,...' on your EXEC statement, or
2. as specified in your installation defaults.

The order of precedence for specified options is:

- PARM='OPTION'
- Default options

Any conflicting options are processed on each level individually. The levels are then processed together for conflicts.

Note: Any options not specified on the EXEC statement are determined by their values in the installation defaults.

Overriding Defaults

You can override the default values at postprocessor time by specifying options in the parameter list on the EXEC statement.

Execution, Report, and View Time

Coding Options

These are the general rules for coding keywords and their options at execution and report time.

You can use:

- One or more spaces as delimiters for options.
- An equal sign (=) to separate a keyword from its option value (for example, DUMP=ON and DUMP ON are both acceptable).
- Positions 1-72 and ignore positions 73-80.

You can specify:

- Options in any order (if you repeat an option, the last specified occurrence is applied).
- Any number of options on a line (however, do not split options between two lines)

You can also:

- Denote a comment line with an asterisk in position 1.
- Input an option associated with a keyword when you specify the keyword. To obtain the total default value of the keyword, omit the entire keyword and option.

Overriding Defaults

You can override option default values by specifying options in the card-image data set defined by the CAIOPTS DD statement. You can specify this DD statement at execution and report time. If the Analyzer Reporter is to be automatically invoked at execution time, the report time options for Analyzer can be mixed freely with the execution time options for both Detector and Analyzer.

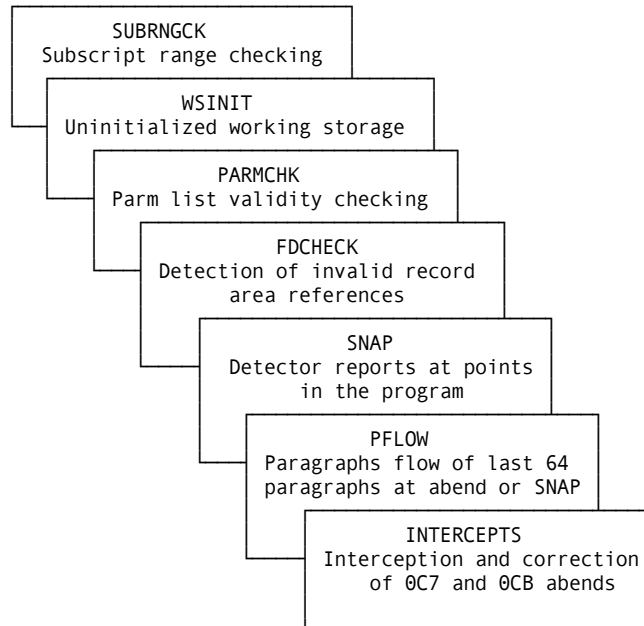
At view time, you can override options through the Report Options panel of the CAIPRINT Repository Viewer. See Chapter 6, “Using the CAIPRINT Repository Viewer”, for more information.

Optimizer Options

Compile-Time Options			
Optimizer		Detector	
To invoke Optimizer:	Environmental Options:	To invoke Detector features:	
COPT	CICS IDMSDC DCI SHADOW HOGAN TPMON	DTECT PFLOW FDCHECK SUBRNGCK WSINIT	
Producing Output:	Other Options:	Analyzer	
CLIST RDIS CXREF SELECT DMAP SOURCE MDMAP XMDMAP MLIST XREF PMAP	COBCC LINECNT CSL OPTRC DECK STATS	To invoke Analyzer features:	
		XCOUNT	

Detector Options

Detector consists of the following facilities:



The chart below lists all Detector options available at compile, execution, and view time.

Detector Options		
Compile Time	Execution Time	View Time
DTECT FDCHECK PFLOW SUBRNGCK WSINIT	Standard Options ACTONLY PARMCHK CEEDUMP PFLOW CSLDSN PGM DB2ACTIV PRTLIB DUMP PRTREPT INTERCEPTS SNAP MSTAE	Formatting Options ASMINST BINFRMT FILES FIRST128 GRPADDR LINECOUNT LINKAGE LOCALSTOR MEMMAP MERGEDB MRGFILES MRGLINK MRGLOCAL MRGWORK OCCURS SAVEAREA WORKSTOR
	Formatting Options ASMINST MERGEDB BINFRMT MRGDATA FILES MRGFILES FIRST128 MRGLINK GRPADDR MRGLOCAL LINECOUNT MRGWORK LINKAGE OCCURS LOCALSTOR SAVEAREA MEMMAP WORKSTOR	
	Data Mgmt Control Block BLOCKS DCB DEB IOB UCB	
	VSAM Control Block ACB AMB AMBL PLH RPL VSAMIDX	

The PFLOW Detector feature has both a compile time and execution time option.

Specifying Detector Compile Time Options

Specify the Detector compile time options in the PARM list in the EXEC statement. You may freely combine Detector compile time options with OS/VS COBOL compiler options and other CA-Optimizer options. The following lists Detector compile time options. Their execution time applications are discussed in detail in the chapter “Using Detector”.

DTECT	SUBRNGCK
FDCHECK	WSINIT
PFLOW	

Indicating Detector Execution Time and View Time Options

Detector execution time options are provided to the Execution Monitor through the CAIOPTS data set. The view time options are specified through the Report Options panel in the CAIPRINT Repository Viewer.

Standard Detector Execution Time Options

The following is a list of standard Detector execution time options. Example applications are presented in Chapter 4, ‘Using Detector’.

ACTONLY	PARMCHK
CEEDUMP	PFLOW
CSLDSN	PGM
DB2ACTIV	PRTLIB
DUMP	PRTREPT
INTERCEPTS	SNAP
MSTAE	

Detector Formatting Options

The following is a list of Detector formatting options. These options can be specified either at execution time or at view time. See Chapter 6, “Using the CAIPRINT Repository Viewer”, for information on how to override these options at view- time.

ASMINST	MERGEDB
BINFRMT	MRGDATA
FILES	MRGFILES
FIRST128	MRGLINK
GRPADDR	MRGLOCAL
LINECOUNT	MRGWORK
LINKAGE	OCCURS
LOCALSTOR	SAVEAREA
MEMMAP	WORKSTOR

Data Management Control Block Options

On the File Section Display (which is discussed in the chapter "Using Detector". Detector optionally displays Data Management Control Blocks, including VSAM Control Blocks. The following options control which I/O blocks are displayed on the Detector Open Files Report.

```
BLOCKS  
DCB  
DEB  
IOB  
UCB
```

Note: To print all control blocks except one, specify BLOCKS ON with the block you do not want printed specified as OFF. For example, to display all Data Management Control Blocks, except UCBs, specify:

```
BLOCKS ON  
UCB OFF
```

VSAM Control Block Options

The following options control which VSAM Control Blocks are displayed on the Detector Open Files Report.

```
ACB  
AMB  
AMBL  
PLH  
RPL  
VSAMIDX
```

Detector optionally displays Data Management Control Blocks, including VSAM Control Blocks, for all files which are open in the Job Step.

Note: BLOCKS ON/OFF turns ON/OFF all VSAM control block options. For example, if you wish to get a display of open ACBs and no other I/O control blocks simply specify:

```
BLOCKS OFF ACB ON.
```

No Data Management Control Blocks, except open ACBs, are displayed.

Extended Detector Options

The chart below lists all Extended Detector options available at compile, execution, and view time.

Extended Detector Options		
Compile Time	Execution Time	View Time
FDCHECK * SUBRNGCK * WSINIT *	Standard Options	Formatting Options
	CSLDSN PRTLIB	ASMINST
	DB2AVTIV PRTREPT	BINFRMT
	DUMP SNAP	FILES
		FIRST128
		GRPADDR
		LINECOUNT
		LINKAGE
		LOCALSTOR
		MEMMAP
	Formatting Options	MERGEDB
	ASMINST MERGEDB	MRGFILES
	BINFRMT MRGDATA	MRGLINK
	FILES MRGFILES	MRGLOCAL
	FIRST128 MRGLINK	MRGWORK
	GRPADDR MRGLOCAL	OCCURS
	LINECOUNT MRGWORK	SAVEAREA
	LINKAGE OCCURS	WORKSTOR
	LOCALSTOR SAVEAREA	
	MEMMAP WORKSTOR	
	VSAM Control Block	
	ACB AMB	
	AMBL PLH	
	RPL VSAMIDX	
	Data Management Control Block	
	BLOCKS DCB	
	DEB IOB	
	UCB	
	Log and Notify Options	
	LOGROS LOGUNI	
	LOGTSO WTO	

* These options force DTECT unless the EDS option is ON in the compile time default CSECT, CAOUDFOX.

Specifying Extended Detector Compile Time Options

Specify the Extended Detector compile time options in the PARM list in the EXEC statement. You may freely combine Extended Detector compile time options with COBOL compiler options and other CA-Optimizer options. The following is a list of Extended Detector compile time options.

Note: The EDS option must be specified in the compile time default CSECT, CAOUDFOX, to prevent these options from forcing DTECT.

FDCHECK
SUBRNGCK
WSINT

Indicating Extended Detector Execution Time and View Time Options

Extended Detector execution time options are provided to the Execution Monitor by the CAIOPTS data set. The view time options are specified on the Report Options panel in the CAIPRINT Repository Viewer.

Standard Extended Detector Execution Time Options

The following is a listing of standard Extended Detector execution time options.

CSLDSN
DB2ACTIV
DUMP
PRTLIB
PRTREPT
SNAP

Extended Detector Formatting Options

The following is a list of Extended Detector formatting options. These options can be specified at execution time or view time. See Chapter 6, “Using the CAIPRINT Repository Viewer”, for information on how to override these options at view time.

ASMINST	MERGEDB
BINFRMT	MRGDATA
FILES	MRGFILES
FIRST128	MRGLINK
GRPADDR	MRGLOCAL
LINECOUNT	MRGWORK
LINKAGE	OCCURS
LOCALSTOR	SAVEAREA
MEMMAP	WORKSTOR

Data Management Control Block Options

On the File Section Display, Extended Detector will optionally display Data Management Control Blocks, including VSAM Control Blocks. The following options control which I/O blocks are displayed on the Extended Detector Open Files Report.

BLOCKS
DCB
DEB
IOB
UCB

Note: To print all control blocks except one, specify BLOCKS ON with the block you do not want printed specified as OFF. For example, to display all Data Management Control Blocks, except UCB, specify:

BLOCKS ON
UCB OFF

VSAM Control Block Options

The following options control which VSAM Control Blocks are displayed on the Extended Detector Open Files Report.

ACB
AMB
AMBL
PLH
RPL
VSAMIDX

Extended Detector will optionally display Data Management Control Blocks, including VSAM Control Blocks, for all files in the Job Step that are open.

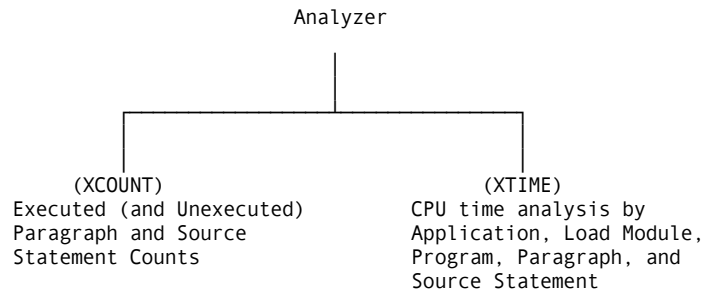
Note: BLOCKS ON/OFF turns ON/OFF all VSAM control block options. For example, if you wish to get a display of open ACBs and no other I/O control blocks simply specify:

BLOCKS OFF ACB ON

No Data Management Control Blocks, except open ACBs, are displayed.

Analyzer Options

Analyzer consists of two facilities: Counts and Timings. The two facilities can be used independently or together. Pictorially:



You control Analyzer at compile time, execution time, and report time. The following chart lists the Analyzer Control options whose use is discussed in the "Using Analyzer" chapter.

Compile Time	Execution Time	Report Time
XCOUNT	BINDER XCOUNT XREPORT XTIME	STIMERM CNTXREF INSTRMAP PROCDIV SOURCEMAP USERID

Compile Time Options

Specify the Analyzer compile time options in the PARM list in the EXEC statement.

You may freely combine Analyzer compile time options with OS/VS COBOL compiler options and with other CA-Optimizer options.

XCOUNT is the only Analyzer compile time option. Its execution time application is discussed in detail in the chapter "Using Analyzer" in this guide.

Execution Time Options

Analyzer execution time options are provided to the Execution Monitor by the CAIOPTS data set. You may freely combine Analyzer execution time options with Detector execution time options.

The following is a list of Analyzer execution time options. Their applications are discussed in the chapter “Using Analyzer” in this guide.

BINDER
STIMERM
XCOUNT
XREPORT
XTIME

Report Time Options

Analyzer report time options are provided to Analyzer Reporter by the CAIOPTS data set at report time. If the Analyzer Reporter is to be automatically invoked at execution time using XREPORT, you may freely combine Analyzer report time options with both Detector and Analyzer execution time options in the execution time CAIOPTS data set.

The following is a list of the Analyzer report time options. Their applications are discussed in detail in the chapter “Using Analyzer” in this guide.

CNTXREF
INSTRMAP
PROCDIV
SOURCEMAP
USERID

Guidelines for Using Report Time Options

Generally, you can supply options in any order, but sometimes the order in which keywords appear can be significant. For example, if the same PROGRAM-ID appears on both the SOURCEMAP and PROCDIV options, the last keyword processed is the controlling option.

The defaults for the keyword options cover all programs. Any overriding specification, such as SOURCEMAP PGMA, overrides the default, SOURCEMAP ALL in this instance. Of all the programs in the reporting step, only the one specified, in this case PGMA, produces full source listing reports. Once you override an options' default, you must specify each parameter you want for the given option.

To reverse the option request, prefix each keyword option with NO. Do not put a space between NO and the keyword. The following are some examples showing how to reverse the option request.

- If you want SOURCEMAP reports for all programs except PGMA, you may specify:

SOURCEMAP ALL
NOSOURCEMAP PGMA

- Or if you want CNTXREF reports for all programs except PGMA, you can specify:

```
CNTXREF ALL  
NOCNTXREF PGMA
```

This feature is very useful when you are running several programs.

Relationship of PROGRAM-ID and CSECT Name

It is important to be aware of the possible differences between options using PROGRAM-ID and those using CSECT to identify programs.

For those reports relating to the COBOL source listing (SOURCEMAP, CNTXREF, and PROCDIV), the name specified following the keyword must be identical to the name specified on the first PROGRAM-ID statement of the COBOL program.

The CSECT name is the control section name actually assigned by the linkage editor. It is used mainly for reporting time measurement data for non-COBOL programs (INSTRMAP).

CSECT names are not necessarily associated with load module names. For COBOL programs, the first program-ID in the source program is CSECT name. There are exceptions, however, where the load module name may be changed during link-edit. For this reason, the three source options SOURCEMAP, PROCDIV, and CNTXREF must use the COBOL program-ID for 'name,' and the INSTRMAP option must use the CSECT name for 'CSECT.'

Alphabetized Option Listing

The underlined options represent the default values. There are no abbreviations for any of these options.

Option	Description
ACB {ON <u>OFF</u> }	Prints access method control blocks for open VSAM files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
ACTONLY {ON <u>OFF</u> }	Used primarily for initial debugging in a multi-module environment. During Abend reporting, it indicates that reports are to be printed only for the program in the calling sequence that was most recently executed and is still active (when it has not yet executed its GOBACK). Component: Detector, Extended Detector When used: Execution time
AMB {ON <u>OFF</u> }	Prints access method blocks for open VSAM files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
AMBL {ON <u>OFF</u> }	Prints an access method block list for open VSAM files in the Detector or Extended Detector report. Component: Detector, Extended Detector When used: Execution time
ASMINST { <u>ON</u> OFF}	Displays the abending assembler instruction and associated operands on the Detector or Extended Detector Report. Component: CAIPRINT Formatter When used: Execution time, view time

BINDER {ON <u>OFF</u> }	Indicates whether Analyzer should access user load modules through the binder API function calls or the standard I/O macros. This option must be set to ON if the programs being Analyzed reside in PDSE libraries. Component: Analyzer When used: Execution time
BINFRMT { <u>DEC</u> HEX }	Indicates whether binary data values should be displayed in decimal display or hexadecimal format. Component: CAIPRINT Formatter When used: Execution time, view time
BLOCKS {ON <u>OFF</u> }	A Detector or Extended Detector data management control block option that sets all control block options, including all VSAM control block options, either ON or OFF. This option overrides any previous control block options. Component: Detector, Extended Detector When used: Execution time
BUFMAX={ <u>0</u> nnnn}	Indicates the maximum value of the compiler BUF parameter that Optimizer should assign. The default 0 will allocate the optimum buffer size for the devices assigned. By altering the BUFMAX value, you risk increasing I/O activity and degrading compiler or Optimizer performance. Specify a value between 0 and 1024 to override the default buffer size allocated for the compiler workfile. Component: Optimizer When used: Compile time
CEEDUMP {ON <u>OFF</u> }	This option indicates whether Detector should force the LE option TERMTHDACT(MSG) after writing a Detector report for an application executing under an LE environment. If TERMTHDACT(QUIET) or TERMTHDACT(MSG) are already in effect this option is ignored. Component: Detector When used: Execution time

CICS <u>NOCICS</u>	Determines if the program is optimized for a CICS environment. CICS suppresses DTECT, PARMCHK, PFLOW, XCOUNT, and XTIME. Component: Optimizer When used: Compile time
CLIST <u>NOCLIST</u>	Specifies whether Optimizer produces a condensed procedure map listing. CLIST is mutually exclusive of MLIST and PMAP. You can, however, use CLIST and SELECT together, and Optimizer integrates the two outputs. For CPP, the CLIST option must have been specified at compile time for this option to take effect. Component: Optimizer, CPP When used: Compile time, postprocessor time
CNTXREF NOCNTXREF{name} {(name1,name2...)} {ALL}	Indicates by program-ID the programs for which the Execution Count Cross-Reference Report prints. You can produce this report only for those programs in which you are also obtaining a SOURCEMAP. Component: Analyzer When used: Report time
COBCC={8 nnnn}	This option is normally set at installation. It specifies the highest return code from the compiler that is acceptable to Optimizer. Optimizer must ordinarily work only with executable modules (no severe or terminating diagnostics). For this reason, you should not change this option except under special conditions. Component: Optimizer When used: Compile time
<u>COPT</u> NOCOPT	Instructs the Run Executive to invoke Optimizer after the compile. If you specify NOCOPT, only the compiler is invoked. Component: Optimizer When used: Compile time

CPPRC <u>NOCPPRC</u>	Indicates that the COBOL postprocessor should set RC=12 rather than abending if an error occurs during CSL generation. Component: CPP When used: Postprocessor time
CSL <u>NOCSL</u>	Determines if a condensed source listing should be stored in the Condensed Source Listing Library. If CICS is specified the CSL is suppressed. Analyzer options at compile time force CSL on. For details, see the Order of Precedence in Establishing Option Values section in this chapter. Component: Optimizer When used: Compile time
CSLDSN=CSL library name}	Defines to the Execution Monitor the data set name of the CAICSL file for dynamic allocation. When used, this option must name a fully qualified 'PDS' that has already been created and cataloged. The member name retrieved is the program's program-ID. Component: Detector, Extended Detector When used: Execution time
CSXREF <u>NOCSXREF</u>	Indicates that Optimizer should produce an alphabetically sorted condensed cross-reference of referenced data and procedure names. CSXREF is exclusive of XREF, CXREF, and SXREF. For CPP, SXREF must have been specified at compile time for this option to take effect. Component: Optimizer, CPP When used: Compile time, Postprocessor time
CXREF <u>NOCXREF</u>	Specifies that Optimizer should produce a condensed cross-reference of referenced data items. CXREF is exclusive of XREF, CSXREF, and SXREF and produces the procedure name XREF with all procedure names listed. For CPP, XREF must have been specified at compile time for this option to take effect. Component: Optimizer, CPP When used: Compile time, postprocessor time

DB2ACTIV {ON <u>OFF</u> }	Indicates whether Detector or Extended Detector should display summary information for only those packages and DBRMs that are active at the time of the abend. If the option is set to OFF, Detector will display summary information for all of the packages and DBRMs defined to the DB2 plan. Component: Detector, Extended Detector When used: Execution time
DCB {ON <u>OFF</u> }	Prints data control blocks for all files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
DCI <u>NODCI</u>	Determines whether the program is optimized for an MSA DCI environment. DCI suppresses XCOUNT, PFLOW, XTIME, DTECT, and PARMCHK. Component: Optimizer When used: Compile time
DEB {ON <u>OFF</u> }	Prints data extent blocks for all open files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
DECK <u>NODECK</u>	Determines whether Optimizer produces the optimized object module on SYSPUNCH. Component: Optimizer When used: Compile time
DMAP <u>NODMAP</u>	Produces a compiler data map. The DMAP option is mutually exclusive of the MDMAP option. The MDMAP option takes precedence. Component: Optimizer, CPP When used: Compile time, postprocessor time

DTECT <u>NODTECT</u>	Specifies that the CA-Optimizer Detector feature is to be invoked at execution time. If you select DTECT, Detector automatically generates the CSL option so that the source is available for the Abend, Snap, and Intercepts Reports. Unless NOCSL is specified, DTECT forces on CSL, MOFFSET, and MMAP. NODTECT suppresses PARMCHK and PFLOW. If CICS, DCI, HOGAN, IDMSDC, SHADOW, or TPMON is specified, DTECT is suppressed. Component: Detector When used: Compile time
DUMP {ON <u>OFF</u> }	Instructs Detector or Extended Detector to force or suppress a system dump after it has completed its processing. If DUMP ON is in effect, the SYSUDUMP DD statement is dynamically allocated, if necessary. Component: Detector, Extended Detector When used: Execution time
FDCHECK <u>NOFDCHECK</u>	Indicates that Optimizer is to cause a user 1128 abend at execution time if a program attempts to move data to (or from if MVS/XA or higher) a record or a file which is not OPEN or, for an input file, is at end-of-file. This option forces the DTECT option except when CICS, DCI, HOGAN, IDMSDC, SHADOW, TPMON or EDS=YES is specified in the defaults. For more details, see the chapter “Using Detector” in this guide. Component: Detector, Extended Detector When used: Compile time
FILES { <u>ON</u> OFF}	Instructs Detector or Extended Detector to display all open files not already shown in the File Section displays of the Detector or Extended Detector Report. Component: CAIPRINT Formatter When used: Execution time, view time
FIRST128 {ON <u>OFF</u> }	Controls whether Detector or Extended Detector prints only the first 128 bytes of the current record of each open file or the entire record. This option affects both the FILE SECTION and Open Files Report. Component: CAIPRINT Formatter When used: Execution time, view time

GRPADDR { <u>ON</u> OFF}	Controls whether Detector or Extended Detector displays the base locator, displacement and address of a group item on the merged Data Division displays. Component: CAIPRINT Formatter When used: Execution time, view time
HOGAN <u>NOHOGAN</u>	Indicates whether the program is optimized for a HOGAN environment. HOGAN suppresses XCOUNT, DTECT, and PFLOW. Component: Optimizer When used: Compile time
IDMSDC <u>NOIDMSDC</u>	Indicates whether the program is optimized for an IDMSDC environment. IDMSDC suppresses XCOUNT, DTECT, and PFLOW. Component: Optimizer When used: Compile time
INSTRMAP <u>NOINSTRMAP</u> { name } { (CSECT1,CSECT2...) } { <u>ALL</u> }	Indicates by CSECT name the programs for which the Instruction Detail Report prints. Component: Analyzer When used: Report time
INTERCEPTS { <u>0</u> nnnnn}	Indicates that Detector intercepts program interrupt processing when data exceptions (0C7s) or decimal divide-by-zero exceptions (0CBs) occur. Detector prints the Intercepts Report and continues processing. You must set nnnnn to the total number of INTERCEPTS in the run that you will permit. The maximum number allowed is 32,767. If the number of exceptions goes beyond the limit you have specified, a Detector Abend Report is generated and the run is terminated. To use INTERCEPTS, the DTECT, FDCHECK, PFLOW, SUBRNGCK, or WSINIT options must be specified at compile time. You may use the INTERCEPTS feature in the same run with the SNAP feature. Component: Detector When used: Execution time

IOB {ON <u>OFF</u> }	Prints input/output blocks for all open files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
LINECOUNT= { <u>57</u> nn}	Indicates the number of lines printed on each page of output. nn may be any integer from 10 to 255, inclusive. Component: CAIPRINT Formatter When used: Execution time, view time
LINKAGE { <u>ALL</u> NONE}	Instructs Detector to display (ALL) or suppress (NONE) the program's Linkage Section. Component: CAIPRINT Formatter When used: Execution time, view time
LOGROS { <u>ON</u> OFF}	Indicates whether a message describing the abend should be sent to the CA-Roscoe user who submitted the job. A list of CA-Roscoe job names must be established in the Extended Detector defaults using the ROSCOE= parameter. Component: Extended Detector When used: Execution time
LOGTSO { <u>ON</u> OFF}	Indicates whether a message describing the abend should be sent to the TSO user who submitted the job. Component: Extended Detector When used: Execution time
LOGUNI {ON <u>OFF</u> }	Indicates whether a message describing the abend should be sent to the CA-Unicenter/TNG console. Component: Extended Detector When used: Execution time

LOWER <u>NOLOWER</u>	Specify LOWER only under the direction of CA Technical Support. LOWER may cause a decrease in optimization results. It is necessary only under certain conditions which are identifiable by CA Technical Support. Component: Optimizer When used: Compile time
MDMAP <u>NOMDMAP</u>	Indicates whether the compiler data map is merged with the source listing. MDMAP is mutually exclusive of the DMAP option, with MDMAP taking precedence. MDMAP may be forced on by Detector or Analyzer options. For details, see the section Order of Precedence in Establishing Option Values in this chapter. Component: Optimizer When used: Compile time
MEMMAP { <u>ON</u> OFF}	Produces a display of the memory map (TGT and PGT) in the Detector Report or Extended Detector. Component: CAIPRINT Formatter When used: Execution time, view time
MERGEDB { <u>ON</u> OFF}	Indicates whether Detector or Extended Detector should symbolically map DB2, IDMS, and IMS. If the option is set to ON, data item information such as the name, picture clause and usage type are merged with the dump output for those areas. Component: CAIPRINT Formatter When used: Execution time, view time
MLIST <u>NOMLIST</u>	Determines whether the Optimized condensed object listing is merged onto the source listing. MLIST is exclusive of CLIST, with MLIST taking precedence. You can use MLIST in addition to PMAP or SELECT. MLIST is forced on by Analyzer options. For details, see the section Order of Precedence in Establishing Option Values in this chapter. Component: Optimizer When used: Compile time

MRGDATA { <u>ON</u> OFF}	A Detector or Extended Detector option that sets all merging options either ON or OFF. This option overrides any previous merging options. Component: CAIPRINT Formatter When used: Execution time, view time
MRGFILES { <u>ON</u> OFF}	Indicates whether Detector or Extended Detector should symbolically map the FILE SECTION display. If the option is set to ON, data item information such as the name, picture clause and usage type is merged with the dump output for the current logical record. Component: CAIPRINT Formatter When used: Execution time, view time
MRGLINK { <u>ON</u> OFF}	Indicates whether Detector or Extended Detector should symbolically map the LINKAGE SECTION display. If the option is set to ON, data item information such as the name, picture clause and usage type is merged with the dump output for the LINKAGE SECTION. Component: CAIPRINT Formatter When used: Execution time, view time
MRGWORK { <u>ON</u> OFF}	Indicates whether Detector or Extended Detector should symbolically map the WORKING-STORAGE SECTION. If the option is set to ON, data item information such as the name, picture clause and usage type is merged with the dump output for the WORKING-STORAGE SECTION. Component: CAIPRINT Formatter When used: Execution time, view time

MSTAE { <u>ON</u> OFF}	This option is normally set at installation. It indicates whether the Execution Monitor is to capture abends and program checks during execution. The Detector captures abends using the ESTAE and ESPIE macros issued during the initialization of the Execution Monitor. It is compatible with other Abend handling products including COBOL's own runtime ESTAE processing and Abend Aid *. Refer to <i>CA-Optimizer Getting Started</i> for more detailed information on the use of this option. * Abend Aid is a registered trademark of Compuware Corp. If you have specified MSTAE OFF and are using Analyzer, your own Abend-catching facility must call the Execution Monitor if an Abend occurs. This is to allow Analyzer to perform its termination processing including the closing of the TIMSV file. For a detailed description on how to do this, see <i>CA-Optimizer Getting Started</i>. Component: Detector When used: Execution time
OBJMVS <u>OBJCMS</u>	Indicates whether the optimized object module is to be used under MVS or CMS. This option does not cause any change in the generated object module. It is used internally at compile time to determine whether or not the options selected are valid. Specifying OBJCMS results in a warning message if XCOUNT or DTECT is specified. Component: Optimizer When used: Compile time
OCCURS { <u>1</u> nnnnnnnn MAX}	Indicates to Detector or Extended Detector the maximum number of table occurrences to be symbolically mapped when using the merging options. If the option is set to a value lower than the number of occurrences in the table, the remainder of the table occurrences are dumped in hexadecimal format. If OCCURS MAX is specified, the option is set to its maximum value of 16777215. Component: CAIPRINT Formatter When used: Execution time, view time

OPTRC= { <u>0</u> nn}	This Optimizer parameter is normally set at installation. It specifies the return code that the Optimizer returns if any statements have been flagged unexecutable (UNEXEC) or unconditional (UNCOND). You can specify any integer from 0 to 99 inclusive. Component: Optimizer When used: Compile time
PARMCHK { <u>ON</u> OFF}	Used to indicate whether option list validity checking is ON at execution time. It is ON by default, which means that PARMCHK processing occurs for any OS/VS COBOL program if the DTECT option was specified. Component: Detector When used: Execution time
PFLOW <u>NOPFLOW</u>	Requests a Detector Paragraph Flow Report of the last 64 paragraphs executed. It follows the Detector Abend or SNAP report. You may override it by specifying the PFLOW OFF option at execution time. PFLOW turns on DTECT. If NODTECT, CICS, DCI, HOGAN, IDMSDC, SHADOW, or TPMON is specified, PFLOW is suppressed. Component: Detector When used: Compile time
PFLOW { <u>ON</u> OFF}	Turns PFLOW on or off at execution time provided it was specified at compile time. Component: Detector When used: Execution time

PGM xxxxxxxx	Specifies the job-step program name of the application to be executed under the CA-Optimizer Execution Monitor. The Execution Monitor is that part of the CA-Optimizer runtime system which sets up the execution environment for applications using any of the Detector or Analyzer features. This option is only used in special circumstances. It is only specified when the Execution Monitor (CAOCEM00) is the program being executed as the job-step program. It identifies the load module name of the program you ordinarily specify as the job-step program. See the chapter “Using Analyzer” and <i>CA-Optimizer Getting Started</i> for a description of the PGM Detector application. Component: Detector When used: Execution time
PLH {ON <u>OFF</u> }	Prints place holders for open VSAM files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
PMAP <u>NOPMAP</u>	Specifies whether Optimizer produces an optimized object code listing. The PMAP and CLIST options are mutually exclusive; PMAP and MLIST are not. If both PMAP and CLIST are specified, CLIST takes precedence. For CPP, the PMAP compiler option must have been specified for this option to take effect. Component: Optimizer, CPP When used: Compile time, postprocessor time
PROCDIV <u>NOPROCDIV</u> {ALL} {name} {name1,name2...}	Indicates by program-ID the program for which only the Procedure Division prints. All measurement data for all programs specified with this option prints next to the Procedure Division; the other divisions are omitted. This option allows you to make several Analyzer runs for the same program without having to produce the full program source listing each time. Component: Analyzer When used: Report time

PRTREPT { <u>BOTH</u> REPOS PRTDD }	Indicates whether the CAIPRINT report should be written to the central VSAM repository (REPOS), the CAIPRINT DD (PRTDD), or both. Component: Detector, Extended Detector When used: Execution time
RDIS <u>NORDIS</u>	Specifies whether Optimizer produces a relative decimal byte displacement on the data map or merged data map listing instead of the hex relative byte displacement produced by the compiler. Component: Optimizer, CPP When used: Compile time, postprocessor time
RPL {ON <u>OFF</u> }	Prints a Request Parameter List for open VSAM files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
RPTSZ80 { <u>ON</u> OFF}	Indicates whether Extended Detector should force all data in dump format to appear in an 80-column format. Component: CAIPRINT Formatter When used: Execution time, view time
SAVEAREA { <u>ON</u> OFF}	Produces the Save Area Trace Report which displays the chained register save areas starting at the abending module. Component: CAIPRINT Formatter When used: Execution time, view time
<u>SBCALC</u> NOSBCALC	Indicates whether Optimizer should calculate SIZE and BUF values which are passed to the COBOL compiler. Specifying NOSBCALC suppresses these calculations. SIZE and BUF parameters must follow the NOSBCALC parameter. Failure to do this results in the SIZE and BUF parameters not being passed down to the compiler. Component: Optimizer When used: Compile time

SELECT= (stmnt1,stmnt2-stmnt6,stmnt9,...)	Specifies that a procedure map (PMAP) prints for only certain(selected) statements. When you indicate the statement number, Optimizer formats, in procedure map form, the entire Optimizer block of code that contains the number. It is possible to format multiple blocks. If PMAP is specified, SELECT is suppressed. Notes: ■ You cannot specify more than 10 statement numbers in the option, but you can specify a range of statements by indicating the starting and ending statement numbers. For example, 600-650 is counted as two statement numbers but provides PMAP information for 51 lines. ■ The statement numbers must appear in ascending order. ■ If you specify only one statement or one group statement, omit the parenthesis. ■ The statement numbers are the compiler-generated statement numbers, unless you have specified the compiler NUM option, in which case they are the user source statement numbers. ■ Optimizer ignores the SELECT option if you have also specified PMAP. You can use SELECT with CLIST or MLIST, and Optimizer integrates the two outputs. Component: Optimizer When used: Compile time
SHADOW <u>NOSHADOW</u>	This Optimizer option indicates whether or not the program is Optimized for a SHADOW environment. SHADOW suppresses XCOUNT, DTECT, PFLOW, XTIME, and PARMCHK. Component: Optimizer When used: Compile time
SIZEMAX={0 nnnnK}	Indicates the maximum region size that the Optimizer (and the compiler using Optimizer) will use. If SIZEMAX=0, CA-Optimizer will use all storage available in the region up to 1 megabyte. SIZEMAX values may range between 0 and 1024 KB. SIZEMAX should only be used to limit the amount of space the Optimizer is to use. Component: Optimizer When used: Compile time

SNAP { <u>0</u> nnnnn ON OFF}	Indicates that Detector or Extended Detector is to support the SNAP option in a program. For Detector, you must specify the number of SNAPs you will allow using nnnnn. The maximum allowed by CA-Optimizer is 32,767. To use SNAP, you must include a call to CAODSNAP in your source code wherever you want a snapshot to be taken. You may use the SNAP feature in the same run with the INTERCEPTS feature. If your program contains SNAP calls and you wish to run it without them, you do not need to recompile; simply do not specify the SNAP option at all. Warning! When using SNAP with Extended Detector, each call to CAODSNAP generates a separate SNAP report in your central VSAM repository. If your program contains multiple SNAP calls or a SNAP within a loop, you can write the output to the CAIPRINT DD alone to avoid flooding the repository. See the PRTREPT option. Component: Detector, Extended Detector When used: Execution time
<u>SOURCE</u> NOSOURCE	This option is used to indicate whether Optimizer is to produce a source listing on the standard SYSPRINT data set. NOSOURCE suppresses the source listing. However, the options, diagnostic messages, compiler statistics, and Optimizer statistics pages still appear. Component: Optimizer, CPP When used: Compile time, postprocessor time
<u>SOURCEMAP</u> NOSOURCEMAP { ALL } { name } { (name1,name2...) }	Indicates by program-ID the programs for which the full Analyzer source listing prints. Component: Analyzer When used: Report time
<u>STATS</u> NOSTATS	This option is normally set at installation. It indicates whether the Optimizer component updates the statistics data set with the optimization results. Component: Optimizer When used: Compile time

STIMERM {ON <u>OFF</u> }	This option indicates whether Analyzer should use STIMERM instead of STIMER for time sampling. If XTIME OFF is specified this option is ignored. Component: Analyzer When used: Execution time
SUBRNGCK <u>NOSUBRNGCK</u>	Specifies that Optimizer generates code which tests each subscript or index value before it is used to assure that it lies within the range defined for the subscripted data item. A subscript or index range violation results in a User 1125 abend if DTECT is ON or a S0C1 if NODTECT is in effect at execution time. Consult the section on subscript range checking in the "Using Detector" chapter for details on usage. This option also forces the DTECT option except when CICS, DCI, HOGAN, IDMSDC, SHADOW, TPMON or EDS=YES is specified in the defaults. Component: Detector, Extended Detector When used: Compile time
<u>SUPMAP</u> NOSUPMAP	Indicates that the recovery program should suppress the compiler PMAP or CLIST in the event that the program is not optimized due to source errors. Specifying SUPMAP suppresses PMAP or CLIST. Component: Optimizer When used: Compile time
SXREF <u>NOSXREF</u>	Determines whether Optimizer should produce an alphabetically sorted cross-reference of data and procedure names. SXREF is exclusive of XREF, CXREF, and CSXREF. Component: Optimizer When used: Compile time
TPMON <u>NOTPMON</u>	This Optimizer option indicates that the program is Optimized for a teleprocessing environment other than CICS, DCI, HOGAN, IDMSDC, or SHADOW. TPMON need not be specified for programs to run in an IBM environment. This option is designed for programs to be run in user-written teleprocessing environments or in other teleprocessing monitors such as INTERCOM. TPMON suppresses XCOUNT, DTECT, and PFLOW. Component: Optimizer

	When used: Compile time
TRACE <u>NOTRACE</u>	Indicates whether READY/RESET TRACE code is to be left in the optimized object module. If you specify NOTRACE, the READY/RESET TRACE code is removed. Component: Optimizer When used: Compile time
UCB {ON <u>OFF</u> }	Prints unit control blocks for all open files in the Detector or Extended Detector Report. Component: Detector, Extended Detector When used: Execution time
USERID {'your text' ' <u>Computer Associates</u> '}	Indicates a user identification statement (1-60 bytes). The default title is specified during product installation. This is included in all report headings if a STATS file is not available. Component: Analyzer, Detector, Extended Detector When used: Report time, Execution time
<u>VS</u> NOVS	Indicates whether the COBOL compiler is IBM's OS/VS compiler. VS is exclusive of V2. Component: Optimizer When used: Compile time
V2 <u>NOV2</u>	Indicates that the COBOL compiler is IBM's ANS version 2 compiler. V2 is exclusive of VS. Component: Optimizer When used: Compile time
VSAMIDX {ON <u>OFF</u> }	Prints the VSAM indexed control blocks for the index portion of an open VSAM file along with the control blocks for the data portion in the File Section and Open Files Reports. Component: Detector, Extended Detector When used: Execution time

WORKSTOR { <u>ALL</u> NONE}	Indicates that Detector displays (ALL) or suppresses the program's WORKING-STORAGE display. Component: CAIPRINT Formatter When used: Execution time, view time
WSINIT <u>NOWSINIT</u>	Initializes to X'50' all WORKING-STORAGE items which are not initialized by a VALUE clause. This option also forces the DTECT option, except when CICS, DCI, HOGAN, IDMSDC, SHADOW, TPMON, or EDS=YES is specified in the defaults. For details on usage, consult the 'Detection of Uninitialized Working-Storage' section in the "Using Detector" chapter. Component: Detector, Extended Detector When used: Compile time
WTO { <u>ON</u> OFF}	Indicates whether a message describing the abend should be sent to the job log (using WTO with ROUTCDE=11). Component: Extended Detector When used: Execution time
XCOUNT <u>NOXCOUNT</u>	Indicates whether Analyzer's Count feature is to be invoked at execution time. If you specify XCOUNT, then MLIST and CSL are forced. If you specify CICS, DCI, HOGAN, IDMSDC, SHADOW, or TPMON then XCOUNT is suppressed. Component: Analyzer When used: Compile time
XCOUNT { <u>ON</u> OFF}	Specifies whether count measurement data is to be collected. If you specify XCOUNT at compile time, you can specify XCOUNT either ON or OFF at execution time. Component: Analyzer When used: Execution time

XDMAP <u>NOXDMAP</u>	The XDMAP Optimizer option indicates that data map information that cannot be merged with the source listing is to be suppressed. An example of data map information which cannot be merged is COPY...SUPPRESS statements. XDMAP may only be used when MDMAP is specified. Component: Optimizer When used: Compile time
XREF <u>NOXREF</u>	Indicates that Optimizer should produce a standard compiler cross-reference of all data items and procedure-names. XREF is mutually exclusive of CXREF, SXREF, and CSXREF with XREF having the lowest priority of these options. For CPP, the XREF compiler option must have been specified for this option to take effect. Component: Optimizer, CPP When used: Compile time, postprocessor time
XREPORT { <u>ON</u> OFF}	If this option is specified at execution time and Counts or Timings were taken, the Analyzer Reporter is automatically invoked upon termination of the application. This occurs for normal terminations, as well as abending applications. Note: You can still execute Analyzer Reporter as a separate job step if you wish by supplying a CAITIMSV DD statement and specifying XREPORT OFF. See "Using Analyzer", in this guide for further usage details. Component: Analyzer When used: Execution time

XTIME {FIRST <u>OFF</u> }	Using this option at execution time causes timings to be taken during the execution of the application program. You may specify XTIME at execution time in one of the following ways: ■ XTIME FIRST starts timings at the first call to the Execution Monitor from any program in the application. The Execution Monitor is automatically invoked during execution from any program compiled with DTECT, FDCHECK, PFLOW, SUBRNGCK, WSINIT, or XCOUNT. ■ XTIME OFF indicates that no timings are to be taken. Component: Analyzer When used: Execution time
-----------------------------	---

Data Sets and Block Sizes

This appendix lists the compile step DD statements and the execution time statement.

Compile Step DD Statements

SYSUT1 SYSUT2 SYSUT3 SYSUT4 SYSUT5 SYSUT6 SYSUT7 SYSUT8	Block size is automatically determined based on the amount of available storage and device type. CA-Optimizer controls the block size.
CAISTATS	Block size is fixed at 136 and must not be altered.
CAICSL	Block size is set to the multiple of 121 which yields the best disk space utilization for the device type on which it resides without exceeding the block size specified in the DD statement (or 4096 if none was specified in the DD statement). For a pre-existing file, the block size with which it was created is used.

Execution Time DD Statement

CAITIMSV

Default block size is 1020. You may specify any multiple of 20. If you specify any valid number, it is rounded down to a multiple of 20 with 20 being the smallest block size allowed.

Note: Analyzer Reporter accepts concatenated CAITIMSV files with mixed block sizes in any order.



Glossary

Analyzer

A CA-OPTIMIZER component that processes the execution-time data gathered in the CAITIMSV file and produces a variety of useful reports for performance analysis and quality assurance. One of the reports is a copy of your source listing displaying the number of times each statement was executed and the amount of CPU time each statement required.

base locator cell

A one-word entry in the TGT that contains the address of a portion of Data Division. A base locator is assigned to each file, each 01 item in the Linkage Section, and to Working-Storage. If any one of these areas are greater than 4096 bytes, additional base locators will be allocated. BLWs are used for Working-Storage, BLFs are used for files, and BLLs are used for the Linkage Section.

BLF

See base locator cell.

BLL

See base locator cell.

BLW

See base locator cell.

CBL

See constant base locator.

compile/optimize-time

The phase of program development when the COBOL compiler and CA-OPTIMIZER are together invoked to translate and optimize the program (as opposed to execution-time, when the program is executed).

constant base locator

For optimized programs, a one-word entry in the PGT that contains the address of a portion of the PGT if the PGT is greater than 4096 bytes. A CBL cell is assigned for each 4 KB portion of the PGT starting 4 KB from the beginning of the PGT. The last CBL may be assigned for less than a 4 KB portion.

control parameters/options

Parameters or options that are generally governed by their installation defaults and will only need to be specified in special cases.

CSECT

The control section, the smallest unit of code that can be assembled or compiled. CSECTs are translated into object modules and combined with other CSECTs to form load modules.

CSL

A compressed copy of the optimized COBOL source listing with enhancements. The CSL is created and saved by the Optimizer for use by Analyzer and Detector.

default parameter/option

A parameter or option that has already been set for the system according to the way it is assumed to be most generally used. You may change any default option to your own specifications.

default parameter/option (fixed)

A parameter or option that has already been set for the system according to the way it is wanted to be used. You cannot override a fixed default option.

Detector

A feature of CA-OPTIMIZER that is used to debug COBOL programs in the production and test environments. Detector prints out reports on diagnostics, key storage data, snapshots of data areas, and data and decimal divide exceptions.

diagnostic

A message that is displayed or printed out by the compiler or Optimizer when it detects an error in your program.

dump

A hexadecimal display of the information currently in memory.

Execution Monitor

A CA-OPTIMIZER component that becomes a logical subroutine to a COBOL program whenever Detector or Analyzer features are used. The Execution Monitor provides the foundation for all runtime debugging and timing facilities. It monitors the entry and exit from all COBOL programs in the run unit and provides the logic for processing certain system events related to timings, interrupts, and a formattedabend report by Detector.

execution-time

The point at which your program is run or tested (as opposed to compile/optimize-time when the program is compiled and CA-optimized). This term is synonymous with "runtime."

file

An ordered collection of data stored on an auxiliary storage device.

fixed parameter/option

A parameter or option that has been set at installation and that may not be altered by the programmer at compile/optimize-time or at runtime.

forced parameter/option

A parameter or option that has been set or reset by CA-OPTIMIZER because of a conflict with another requested option.

intercept

A CA-OPTIMIZER feature that provides for the interception of a program check violation as the result of a data or decimal divide exception. Rather than abend, the program is restarted with a value of 1 to circumvent the "bad" data situation during testing.

memory map

A breakdown of the global tables within a COBOL program and their corresponding program locations. This can be a useful debugging tool. The memory maps are displayed in full by Optimizer with the object listing.

object code

The machine language output by the compile/optimize step after it has translated your source program.

optimize time

The time it takes to execute the compile and optimize step.

Optimizer

The Optimizer is a component of CA-OPTIMIZER that reduces the size and execution-time of your program's object code by eliminating redundant machine instructions without changing the processing or altering user-defined data areas.

Optimizer Stamp

A short section of each optimized COBOL program located immediately before the Program Global Table (PGT). It identifies the CA-OPTIMIZER product code, version number, data and time of compilation, and other information related to the compilation.

option

A variable that is assigned a constant value at runtime or report time for use in controlling Analyzer or Detector.

options page

The first page of the Optimizer Enhanced Source Listing which identifies the compiler version and version number, compilation date and time of optimization, Optimizer release number, selected and default options, and customer name and location.

PBL

See procedure base locator

PGT

See Program Global Table.

Procedure Base Locator

For optimized programs, a one-word entry in the PGT that contains the address of a portion of the executable code in the Procedure Division. The portion of code can be a maximum of 8 KB to 12 KB depending on the size of the program and the options used.

procedure map

The procedure map printed out by Optimizer contains the following information about your COBOL program: block cross-reference data, base register assignments, interleaved source code, and object code.

Program Global Table (PGT)

A section of a COBOL program that contains constants, literals, and certain base locators that are used during program execution. The areas in the PGT will not be modified when the program is executed.

report time

The phase during which the Analyzer Reporter reports on the counts or timings data gathered while the application was running. Report time can occur automatically after normal or abnormal termination of the application in the same job step or as a separate job step.

runtime

See execution-time.

snapshot

A report about your COBOL program that is output each time a SNAP is invoked. A snapshot consists of information about your program's address and registers, contents of the file, working storage and linkage sections, memory map, PERFORM Address Analysis, and the last nnnnn paragraphs executed prior to the SNAP call.

sort

To place records in a predetermined sequence, such as in alphabetical order, by means of a sort key (a field or a hierarchy of fields) within the record.

Task Global Table (TGT)

A section of a COBOL program that contains work areas, information, and base locators that are used during execution of the program. Some areas in the TGT can be modified when the program is executed.

text deck

See object code.

TGT

See Task Global Table.

timings

Data captured during application program execution using a technique called time sampling. This occurs when the XTIME function is enabled. The timing data is formatted by Analyzer into reports that indicate which parts of your program use the most CPU time.

Index

A

Abend

- cause of, 4-12, 4-50
- interception and correction, 4-3
- pinpointing cause, 4-12
- report
 - debugging, 4-1
 - description, 4-11
 - Extended Detector Support, 5-1, 5-4
 - locating instruction address, 4-12
 - PFLOW, 4-22
 - symbolic support, 5-4
 - tools, 4-1
- statement location, 7-1

abend page, 1-4

ACB

- VSAM control block, 10-8, 10-11, 10-15

ACTONLY Detector option

- Abend Report, 4-55
- execution-time
 - description, 10-15

Address

- control return, 2-16
- entry point, 4-13
- MLIST, 4-22
- MOFFSET, 4-22

Allocating - DD statements, 4-53

ALTERed GO TO Table

- report description, 2-15

ALTERED GO TO Table

- locations, 2-18
- VN pointer, 2-18

AMB

- Detector Options
 - open files report, 10-8, 10-11
- VSAM control block, 10-15

AMBL

- Detector Options
 - open files report, 10-8, 10-11
- VSAM control block, 10-15

Analyzer

- available options, 10-5
- compile-time options
 - chart, 10-12
 - XCOUNT, 10-33
- count data storage, 2-22
- counts, 10-11, 10-12
- execution-time options
 - BINDER, 10-16
 - chart, 10-12
 - STIMERM, 10-31
 - XCOUNT, 10-33
 - XREPORT, 10-34
 - XTIME, 10-34

JCL

- Optimizer component, 7-6
- options
 - BINDER, 10-16
 - CNTXREF, 7-5, 10-17
 - execution-time, 10-12
 - INSTRMAP, 10-21
 - NOSOURCEMAP ALL, 7-5
 - PROCDIV, 10-27
 - report-time guidelines, 10-13
 - SOURCEMAP, 10-13, 10-30
 - STIMERM, 10-31
 - USERID, 10-32
 - with Detector options, 10-12
 - XCOUNT, 7-5, 10-12
 - XREPORT, 7-5, 10-13, 10-34
 - XTIME, 7-5, 10-13, 10-34

Reporter
 execution considerations, 7-28
reports
 CAIOPTS File Processing, 7-10
Reports
 Execution Count Cross Reference, 7-23
 INSTRMAP, 7-24
report-time options
 chart, 10-12
 CNTXREF, 10-17
 INSTRMAP, 10-21
 PROCDIV, 10-27
 SOURCEMAP, 10-13, 10-30
 USERID, 7-5, 10-32
 XREPORT, 7-5
source listing data, 7-3
timings, 10-11, 10-12
under CMS, 8-1
usage considerations
 counts, 7-26
 multi-module environments, 7-27
 multi-run reports, 7-28
 restrictions, 7-30
 timings, 7-27, 7-28
ASMINST
 abending assembler instruction, 10-15

B

Base locators
 displacement, 4-10
 number, 4-10
 unreferenced cells, 2-7
Base register assignments, 2-25
Benefits
 enhancing performance, 1-2
 improving program reliability, 1-2
 maximizing productivity, 1-2
 programming techniques, 1-2
BINDER
 binder API function calls, 10-16
BINFRMT
 binary data values, 10-16
BLF - cells with SUBRNGCK, 4-51
BLL - cells with SUBRNGCK, 4-51

BLOCKS
 Detector option
 execution-time, 10-16
 Open Files Report, 10-8, 10-11
 VSAM control block options, 10-8, 10-11
BLV base locators, 2-25
BLW - cells with SUBRNGCK, 4-51
BRANCH-OUT TO PERFORM ranges, 2-17
BROWSE
 Errors panel, 8-9, 8-14
 Output panel, 8-8

C

CAIAR PROC, 7-8
CAICSL
 allocation, 5-3
 definition, 7-7
 JCL, 7-9
 library pointer, 4-6
CA-IDMS, 1-4
CAILPAQ, 9-7
CAIMRS, 9-2
CAIOPT
 execute Optimizer
 FDCHECK, 4-5
 procedure, 2-4
 SUBRNGCK, 4-5, 7-6
 job step, 7-9
CAIOPTNS procedure
 options summary, 9-9
CAIOPTS
 Analyzer and Detector Example, 7-8
 DD statement - card-image data set, 10-4
 description, 4-6
 file processing report, 4-43, 7-10
 overriding options, 5-3
CAIPRINT
 dynamic allocation, 5-2
 printed output statement, 4-6, 7-7

CAIPRINT Formatter
 execution-time options
 ASMINST, 10-15
 BINFRMT, 10-16
 GRPADDR, 10-21
 MERGEDB, 10-23
 MRGDATA, 10-24
 MRGFILES, 10-24
 MRGLINK, 10-24
 MRGWORK, 10-24
 OCCURS, 10-25
 options
 ASMINST, 10-15
 BINFRMT, 10-16
 GRPADDR, 10-21
 MERGEDB, 10-23
 MRGDATA, 10-24
 MRGFILES, 10-24
 MRGLINK, 10-24
 MRGWORK, 10-24
 OCCURS, 10-25

CAIPRINT reports, 1-4

CAIPRINT Repository
 utility, 9-11
 commands, 9-11
 JCL, 9-11

CAIPRINT Repository Viewer, 6-1, 6-8, 6-13

CAIPRNT1 - report data set, 7-8

CAIPRNT2 - report data set, 7-8

CAISTATS
 Analyzer Reporter header, 7-8
 definition, 9-5

CAITIMSV
 definition, 7-7
 execution JCL, 7-7, 7-9

CALL to CAODSNAP at compile-time, 4-4

CAOCEM00 - Execution Monitor JCL, 7-27

CAODSNAP - making a SNAP call, 4-8

CAODSNAP program, 1-4

CAOMSRPT - procedure, 9-5

CAOMSRPT - statistics report, 9-5

CAOPT EXEC
 accessing CA-Optimizer Utilities, 8-1
 default program names, 8-2

CA-Optimizer load library, 8-6

CAOUMRS - DD statements, 9-2

CAOUOPTS program
 options summary, 9-9

CA-Roscoe
 DELETE listing data sets, 8-15
 option menu codes, 8-12

CA-Roscoe - Primary Option Menu, 8-12

CA-SymDump/Batch
 options summary, 9-9

Cause ofabend, 4-50

CBL base locator, 2-25

CEEDUMP
 LE environment, 10-16

Change summary report
 tagging source statement, 7-29
 Y2K source changes, 7-19

Chart
 Analyzer
 compile-time options, 2-2, 7-4
 execution-time options, 7-4
 runtime options, 7-4
 Detector
 compile-time options, 2-2
 Optimizer compile-time options, 2-2

CICS
 flow-of-control paths, 2-18
 Optimizer compile-time options, 10-17
 restrictions, 7-30

CLIST
 Optimizer option
 compile-time, 2-24
 description, 10-17

CMS
 Analyzer restriction, 7-30
 BROWSE facility, 8-2
 Execute, 8-4
 Optimize, 8-3
 TRANSFER CSL, 8-4

CNTXREF
 Analyzer options
 execution-time, 7-5
 report-time, 7-22, 10-13, 10-17
 COBOL source listing, 10-13

COBCC compile-time option, 10-17

COBOL

compilers, 9-2, 9-3, 9-4

CSECT

Analyzer Module Summary, 7-14

Detail Report, 9-2, 9-3

MRS, 9-1, 9-4

Summary Report, 9-4

line number, 2-24

OS/VS, 2-4

OS/VS options with Detector options, 10-7

postprocessor, 1-4

invoking, 6-9

support, 1-3

verb location, 2-24

COBOL II

options

processing precedence, 10-2

Codes - data definition attribute, 2-8

Coding

options, 10-3

rules, 7-6

Combining Options

Detector

and CA-Optimizer, 10-7

and COBOL OS/VS, 10-7

with Analyzer options, 10-12

Optimizer with COBOL II, 10-2

Optimizer with OS/VS COBOL, 10-1

OS/VS COBOL with Analyzer, 10-12

Comparison - NOOPT, 2-21

Compiler

data map (MAP), 2-23

source listing, 2-22

statistics report, 2-20

Compile-time

Optimizer options

CICS, 10-17

CLIST, 10-17

COBCC, 10-17

COPT, 10-17

CSL, 10-18

CXREF, 10-18

DCI, 10-19

DECK, 10-19

DMAP, 10-19

HOGAN, 10-21

IDMSDC, 10-21

LOWER, 10-22

MDMAP, 10-23

MLIST, 10-23

OPTRC, 10-25

PMAP, 10-27

RDIS, 10-28

SELECT, 10-28

SHADOW, 10-29

SOURCE, 10-30

STATS, 10-30

SUPMAP, 10-31

SXREF, 10-31

TPMON, 10-31

TRACE, 10-31

V2, 10-32

VS, 10-32

XDMAP, 10-33

XREF, 10-34

options

CALL to CAODSNAP, 4-4

chart summary, 4-1

CICS, 10-17

CLIST, 2-24

CXREF, 2-11

DTECT, 4-4

FDCHECK, 4-3, 4-4, 5-2

JCL, 4-5

MDMAP, 2-6

MLIST, 2-9

NOCICS, 10-17

NUM, 7-24

OPTRC, 2-10

PARMCHK, 4-3, 4-4

PFLOW, 4-3, 4-4

PMAP, 2-25

SUBRNGCK, 4-2, 4-4, 5-2

WSINIT, 4-3, 4-4, 4-10, 5-2

XCOUNT, 10-12

options summary, 2-2

overview, 10-1

Condensed

Cross-reference report (CXREF), 2-11

Optimized object code listing (CLIST), 2-24

Condensed Source Listing Library, 4-8

Control

Detector Open Files Report options, 10-11

return address, 2-16

Control Detector Open Files Report options, 10-8

COPT Optimizer option
 compile-time description, 10-17

Count and Timing Report
 example, 7-16
 field descriptions, 7-17, 7-18

Counts, 7-1, 7-26

CPP
 options
 CPPRC, 10-18
 postprocessor time options
 CPPRC, 10-18

CPPRC
 CSL generation, 10-18

CPU
 performance, 2-21
 time analysis, 7-28

Create CSL members screen, 6-9

Cross-Reference
 CXREF, 10-18
 debugging, 2-25
 documentation, 2-25
 XREF, 10-34

CSECT
 COBOL, 9-4
 INSTRMAP, 7-25
 Non-COBOL, 9-4

CSL
 summary, 9-10

CSL Display Utility
 invoking, 6-11

CSL libraries
 symbolic information, 1-4

CSL Optimizer option
 compile-time, 10-18
 execution-time, 4-8

CSLDSN Detector option
 description, 10-18

CXREF
 sample report, 2-11

CXREF - Optimizer compile-time option, 10-18

D

Data bytes, 9-5

Data Division
 MDMAP contents, 2-7
 Optimizer option
 CXREF, 10-18
 MAP, 2-23
 MDMAP, 2-6
 XREF, 10-34
 Sample Listing
 MDMAP, 2-6

Data Management Control Block
 Abend report, 4-8
 report customization, 4-3

Data name, 2-7

Data set
 Analyzer Reporter, 7-8
 counts and timings - TIMSV, 7-9
 dynamic allocation, 4-53
 using the CSL, 4-6

Data Size, 9-3

DB2
 report
 example, 4-32

DB2 reporting, 1-4

DB2ACTIV
 summary information, 10-19

DCB Detector
 Abend report, 4-8
 execution-time option, 10-19

DCI Optimizer compile-time option, 10-19

DD statement
 execution-time, A-2

DD Statement
 compile step, A-1

DEB - Detector execution-time option, 10-19

Debugging facility - Detector, 4-1

DECK Optimizer compile-time option, 10-19

Defaulted Options panel, 8-8

- Defaults
 - CMS, 8-3
 - program name, 8-2
- DELETE CA-Optimizer Listing Data Sets, 8-10, 8-15
- Detail Report, 9-2
- Detector
 - available options, 10-5
 - compile-time options
 - chart summary, 4-1
 - FDCHECK, 4-3, 10-20
 - INTERCEPTS, 4-3
 - PARMCHK, 4-3
 - PFLOW, 4-3
 - SNAP, 4-3
 - SUBRNGCK, 4-2, 10-31
 - WSINIT, 4-3
 - Debugging facilities
 - INTERCEPTS, 4-53
 - PFLOW, 4-52
 - SNAP, 4-52
 - diagnostics - SNAP report, 4-1
 - execution-time options
 - ACB, 10-15
 - ACTONLY, 10-15
 - AMB, 10-15
 - AMBL, 10-15
 - BLOCKS, 10-16
 - CEEDUMP, 10-16
 - chart summary, 4-1
 - DB2ACTIV, 10-19
 - INTERCEPTS, 4-4, 10-21
 - IOB, 10-21
 - LINKAGE, 10-22
 - list, 10-7, 10-10
 - MEMMAP, 10-23
 - MSTAE, 10-24
 - PRTREPT, 10-28
 - facilities, 4-2, 4-4
 - JCL compile-time example, 4-5
 - Open Files Report, 10-8, 10-11
 - options
 - ACB, 10-15
 - ACTONLY, 10-15
 - AMB, 10-15
 - AMBL, 10-15
 - and OS/VS COBOL, 10-7
 - BLOCKS, 10-16
 - CEEDUMP, 10-16
 - Data Mgmt Control Block, 10-8, 10-11
 - DB2ACTIV, 10-19
 - DTECT, 10-19
 - DUMP, 10-20
 - FDCHECK, 4-3, 10-20
 - INTERCEPTS, 4-3, 4-4, 10-21
 - IOB, 10-21
 - LINKAGE, 10-22
 - MEMMAP, 10-23
 - MSTAE, 10-24
 - PARMCHK, 4-3
 - PFLOW, 4-3
 - PRTREPT, 10-28
 - SNAP, 4-3
 - SUBRNGCK, 4-2, 10-31
 - VSAM control block, 10-8, 10-11
 - with Analyzer options, 10-12
 - WSINIT, 4-3
 - reports
 - abend, 4-7
 - DB2, 4-32
 - IMS, 4-26
 - instruction analysis, 4-13
 - intercepts, 4-7, 4-9
 - SNAP, 4-8
 - runtime JCL, 4-7
 - under CMS, 8-1
 - usage considerations
 - dynamic allocation, 4-53
 - multi-module reporting, 4-55
 - restrictions, 4-55
 - with IMS, 4-53
- Detector reports, 1-4, 6-1
- Diagnostic
 - messages, 2-20
- Diagnostic reports, 4-1
- Director control routine, 2-2
- Displaying the Utilities menu, 8-2
- Distinctive counts, 7-26
- DMAP - Optimizer compile-time option, 10-19
- DTECT
 - Detector option
 - summary description, 4-4
 - Detector Option
 - detailed description, 10-19
 - Extended Detector, 5-4

DUMP Detector option
description, 10-20

Dynamic allocation, 4-53

E

Editing COBOL programs, 8-2

Enhanced source listing
report
features, 2-2

Enhanced Source Listing
Options Page, 2-6
report
sequence, 2-5

Environments, 1-2

Erasing Optimized listing, 8-2

Establishing option values, 10-2

Estimated results
without Analyzer, 2-22
without Detector, 2-22

EXCP counts, 7-14

EXECUTE Optimized Program panel, 8-10, 8-16

Executing
COBOL program, 8-2
Optimizer, 8-4

Execution Monitor
Analyzer Reporter
CNTXREF, 7-22
producing application timings, 7-27
reports
description, 4-7
file processing, 4-43, 7-10
summary, 4-43
setting at installation, 10-24
summary report field descriptions, 4-43, 4-44
using dynamic allocation facilities, 4-53

Execution statistics - sample numbers, 7-28

Execution-time
coding options, 10-3
Detector options, 10-7, 10-10
Detector Options
LINECOUNT, 10-22
JCL, 5-2

options
ACB, 10-15
ACTONLY, 10-15
AMB, 10-15
AMBL, 10-15, 10-19, 10-21
ASMINST, 10-15
BINDER, 10-16
BINFRMT, 10-16
BLOCKS, 10-16
CEEDUMP, 10-16
INTERCEPTS, 4-3, 4-4, 4-9
JCL, 4-5
LOGROS, 10-22
LOGTSO, 10-22
LOGUNI, 10-22
MEMMAP, 4-4
MERGDB, 10-23
MRGDATA, 10-24
MRGFILES, 10-24
MRGLINK, 10-24
MRGWORK, 10-24
OCCURS, 10-25
PARMCHK, 4-3
PFLOW, 4-3, 4-4
RPTSZ80, 10-28
SNAP, 4-3, 4-4
STIMERM, 10-31
WTO, 10-33
overview, 10-3

Execution-Time
reduction, 2-1

Exits - fall-through, 2-17

Extended Detector
Abend Reports, 5-1
execution-time options
LOGROS, 10-22
LOGTSO, 10-22
LOGUNI, 10-22
PRTSZ80, 10-28
WTO, 10-33
options
LOGROS, 10-22
LOGTSO, 10-22
LOGUNI, 10-22
PRTSZ80, 10-28
WTO, 10-33
restrictions, 5-5

Extended detector reports, 6-1

Extended Detector reports, 1-4

F

Facilities

- abend interception and correction, 4-3, 10-21
- detecting
 - invalid references, 4-3, 10-20
 - uninitialized working storage, 4-3
- enabling options, 4-3
- paragraph flow analysis, 4-3, 10-26
- parm list validity checking, 4-3
- program spot analysis, 4-3, 10-29
- report customization, 4-3
- subscript range checking, 4-2, 10-31

Facilities overview, 1-1

Fall-through exit - PERFORM range, 2-17

FDCHECK Detector Option

- description, 4-3, 4-4, 10-20
- error analysis, 4-51

Features

- enhances source listing, 2-2
- unsupported, 1-3

Field descriptions - TGT, 2-14

File section display, 4-15, 4-16

FILEDEF, 8-4

FILES

- Detector Options
 - description, 10-20
- report customization, 4-3

FIRST128

- Detector execution-time option, 10-20
- report customization, 4-3

floating-pointing register display, 1-4

Flow-of-control paths - CICS, 2-18

G

GOBACK - PERFORM ranges, 2-18

GRPADDR

- group address, 10-21

H

HANDLE

- commands, 2-18
- implied branches, 2-18, 2-20

HELP

- for CA-Roscoe, 8-13
- panels, 8-6

Hexadecimal

- address - branch destination, 2-17
- location - TGT, 2-14

HOGAN

- Analyzer restriction, 7-30
- Optimizer compile-time option, 10-21

I

IBM COBOL compilers, 9-4

Identifying logic errors - PERFORM range, 2-17

IDMSDC

- Analyzer restriction, 7-30
- Optimizer compile-time option, 10-21

Implied branches - PERFORM Ranges, 2-18

IMS

- Detector, 4-53
- report
 - example, 4-26

IMS reporting, 1-4

Initialization

- report, 9-8
- summary, 9-7

Installation - defaults, 8-3

Installation options

- displaying, 6-12

INSTRMAP

Analyzer Option

- chart**, 10-12
- printing by CSECT name, 10-21
- report-time, 10-13
- sample listing, 7-24

Instruction
 detail report
 example, 7-24, 7-25
 fields, 7-25
 hot spots, 7-24, 7-25

INTERCEPTS

 Detector option
 description, 4-1
 execution-time, 4-4
 reporting, 4-3
 Detector Option
 execution-time, 10-21
 report
 description, 4-7
 example, 4-9
 fields, 4-9, 4-10

Invalid file reference, 4-51

Invocation panel, 8-6

Invoking

 CAIOPT, 2-4
 CA-Optimizer, 8-2
 CMS BROWSE, 8-2
 CMS editor, 8-2
 HELP, 8-2
 Management Reporting System Panel, 8-11, 8-17

IOB Detector Option

 execution-time
 description, 10-21

ISPF interface, 1-4

J

JCL

 Analyzer
 compile-time, 7-6
 execution-time, 7-7, 7-27
 report-time, 7-7, 7-8, 7-9
 CAIOPT, 4-5
 CAIOPTS, 4-6
 CAIPRINT, 4-6
 CAISCL, 4-6
 compile-time, 4-5
 execution-time, 4-5
 management reporting facilities
 MRS, 9-2, 9-5
 statistics, 9-5

 optimizing, 2-4
 runtime, 4-5, 4-7
 STEPLIB, 4-6
 SYSUDUMP, 4-6

L

Libraries

 CA-Optimizer Load, 8-6
 OS/VS COBOL Load, 8-6
 statistics, 9-4

Line commands

 delete, 6-5
 lock, 6-5
 options, 6-7
 print, 6-6
 select, 6-4
 unlock, 6-6

LINECNT - Detector execution-time option, 10-22

LINECOUNT

 Detector execution-time option, 10-22
 report customization, 4-3

LINKAGE

 Detector execution-time option, 10-22
 report customization, 4-3

LINKAGE-SECTION display, 4-16

LIST - object code, 2-26

Load

 libraries
 CA-Optimizer, 8-6
 OS/VS COBOL, 8-6
 module, 7-14

Locating

 data definition attribute, 2-8
 internal data name, 2-7

LOGROS

 log ROSCOE, 10-22

LOGTSO

 log to TSO, 10-22

LOGUNI

 log to Unicenter, 10-22

LOWER

 Optimizer compile-time option, 10-22

M

Management reporting facilities

- JCL, 9-2

- MRS

 - audit load module libraries, 9-1

 - OS/VS COBOL reporting, 9-5

 - usage, 9-4

- reports, 9-5

Management Reporting System, 8-11, 8-17, 9-1

MAP - Optimizer sample listing, 2-23

MDMAP

- contents, 2-7

- data definition attribute, 2-8

- NORDIS, 2-7

- Optimizer option

 - merged data map, 10-23

 - reports, 2-22

- options page legend, 2-6

- RDIS, 2-7

- reports generated, 2-5

- sample listing, 2-6

- source listing report features, 2-2, 4-7

MEMMAP

- Detector execution-time option, 4-4, 10-23

- report customization, 4-3

Memory map

- cells, 2-12, 2-13, 2-14, 4-19

- display

 - CGT, 4-18

 - PGT, 4-18

 - TGT, 4-18

- Table

 - PGT, 2-12, 2-13

 - TGT, 2-14

Merged

- Condensed Procedure Division - hexadecimal location, 2-10

- Data Map

 - description, 2-7

 - MDMAP, 2-6

- Listing (MLIST), 2-9

MERGEDB

- detector mapping, 10-23

MLIST

- address references, 4-22

- branching, 2-17

Optimizer option

- merged source listing, 10-23

- PERFORM ranges, 2-17

- reports generated, 2-5

- sample listing, 2-9

- source listing report features, 2-2

Module

- call sequence, 4-8, 4-14

- summary report, 7-14

MOFFSET

- address references, 4-22

- entry point address, 4-13

MRGDATA

- merging options, 10-24

MRGFILES

- map the FILE SECTION display, 10-24

MRGLINK

- LINKAGE SECTION, 10-24

MRGWORK

- WORKING-STORAGE-SECTION, 10-24

MRS

- invoking, 8-11, 8-17

- JCL, 9-2

MSTAE Detector option

- description, 10-24

Multi-module

- environment, 7-27

- reporting, 4-55

Multi-run reports, 7-28

MVS, 7-30

N

New features, 1-4

NO EXIT - unconditional GOBACK, 2-18

NOCICS - description, 10-17

NOCOPT

- description, 10-17

Non-COBOL CSECTs, 9-4

NOSOURCE

- source listing option, 10-30

NOSOURCEMAP

- Analyzer option
 - description, 10-30
 - execution-time, 7-5
 - program reports, 10-13

NULL

- code, 2-18

O

Object Code

- GNs, 2-26
- PNs, 2-26

OCCURS

- maximum table occurrences, 10-25

OFFSET - entry point address, 4-13

OFL base locator, 2-25

Open Files Report, 4-22

Optimization techniques, 2-1

Optimizer

- compile-time options
 - chart, 2-2
 - CICS, 10-17
 - CLIST, 10-17
 - COBCC, 10-17
 - LOWER, 10-22
 - MDMAP, 2-6, 10-23
 - MLIST, 10-23
 - NOCICS, 10-17
 - PMAP, 10-27
 - SUPMAP, 10-31
 - SXREF, 10-31
 - TRACE, 10-31
 - V2, 10-32
 - VS, 10-32
- Detector features, 2-2
- options
 - analyzer features, 2-2
 - detector features, 2-2
 - environmental, 2-2
 - in effect, 2-6
 - MDMAP, 2-6
 - output producing, 2-2
- overview, 2-1

Optimizing a program, 8-2

Options

- ACTONLY, 10-15
- Alphabetized Listing, 10-15
- AMB, 10-15
- AMBL, 10-15
- Analyzer
 - selecting, 10-11, 10-12
 - summary table, 7-5
- Analyzer compile-time
 - chart, 2-2
 - XCOUNT, 10-33
- Analyzer execution-time
 - XCOUNT, 10-33
 - XREPORT, 10-34
 - XTIME, 10-34
- Analyzer report-time
 - CNTXREF, 10-13, 10-17
 - consolidation, 7-28
 - INSTRMAP, 7-24, 10-21
 - PROCDIV, 10-13, 10-27
 - SOURCEMAP, 10-13, 10-30
 - SOURCEMAP sample listing, 7-17, 7-18
 - USERID, 7-5, 10-32
 - XREPORT, 7-5
- ASMINST, 10-15
- BINDER, 10-16
- BINFRMT, 10-16
- BLOCKS, 10-8, 10-11, 10-16
- BUFMAX, 10-16
- CEEDUMP, 10-16
- chart, 5-1
- CICS, 10-17
- CLIST, 10-17
- CNTXREF, 7-22, 10-13, 10-17
- COBCC, 10-17
- COBOL II, 10-2
- COBOL overrides, 8-3
- coding, 10-1, 10-3
- combining Optimizer with COBOL II, 10-2
- combining Optimizer with OS/VS COBOL, 10-1
- compile-time
 - SXREF, 10-31
 - TRACE, 10-31
 - V2, 10-32
 - VS, 10-32
- CPPRC, 10-18
- CSL, 10-18
- CSLDSN, 10-18
- CXREF, 2-11, 10-18
- data management control block
 - BLOCKS, 10-8, 10-11, 10-16
 - DCB, 10-8, 10-11, 10-19

- DEB, 10-8, 10-11, 10-19
- IOB, 10-8, 10-11, 10-21
- UCB, 10-8, 10-11, 10-32
- DB2ACTIV, 10-19
- DCB, 10-19
- DCI, 10-19
- DEB, 10-19
- DECK, 10-19
- Detector compile-time
 - chart, 2-2
 - chart summary, 4-1
 - descriptions, 4-3, 4-4
 - DTECT, 10-19
 - FDCHECK, 4-3, 10-20
 - PARMCHK, 4-3
 - PFLOW, 4-3, 10-26
 - SSRANGE, 10-31
 - SUBRNGCK, 4-2, 4-50, 10-31
 - uninitialized working storage, 4-51
 - WSINIT, 4-3, 4-10, 10-33
- Detector execution time
 - ACB, 10-11
 - AMB, 10-11
 - AMBL, 10-11
 - PLH, 10-11
 - RPL, 10-11
 - VSAMIDX, 10-11
- Detector execution-time
 - abend interception & correction, 4-3
 - ACB, 10-8
 - ACTONLY, 10-15
 - AMB, 10-8
 - AMBL, 10-8
 - BLOCKS, 10-16
 - chart summary, 4-1
 - correcting interrupts, 4-53
 - CSLDSN, 10-18
 - descriptions, 4-4
 - DUMP, 10-20
 - FILES, 10-20
 - FIRST128, 10-20
 - INTERCEPTS, 4-9, 4-53, 10-21
 - LINECOUNT, 10-22
 - LINKAGE, 10-22
 - MEMMAP, 10-23
 - MSTAE, 10-24
 - PARMCHK, 4-3, 10-26
 - PFLOW, 4-52, 10-26
 - PGM, 10-26
 - PLH, 10-8
 - reporting on interrupts, 4-7
 - RPL, 10-8
 - SAVEAREA, 10-28
 - SNAP, 4-3, 4-52, 10-29
 - VSAMIDX, 10-8
 - WORKSTOR, 10-32
- Detector execution-time - LINECOUNT, 10-22
- DMAP, 10-19
- DTECT, 10-19
- DUMP, 10-20
- Enhanced Source Listing, 2-5, 2-6
- establishing precedence, 10-1, 10-2
- FDCHECK, 10-20
- FILES, 10-20
- FIRST128, 10-20
- GRPADDR, 10-21
- HOGAN, 10-21
- IDMSDC, 10-21
- INSTRMAP, 10-21
- INTERCEPTS, 10-21
- IOB, 10-21
- length checking, 4-51
- LINECOUNT, 10-22
- LINKAGE, 10-22
- LOGROS, 10-22
- LOGTSO, 10-22
- LOGUNI, 10-22
- LOWER, 10-22
- MEMMAP, 10-23
- MERGEDB, 10-23
- MLIST, 10-23
- MRGDATA, 10-24
- MRGFILES, 10-24
- MRGLINK, 10-24
- MRGWORK, 10-24
- MSTAE, 10-24
- NOSOURCE, 10-30
- NUM, 7-24
- OCCURS, 10-25
- Optimizer compile-time
 - chart, 2-2
 - CICS, 10-17
 - CLIST, 2-24, 10-17
 - CNTXREF, 7-22
 - COBCC, 10-17
 - CSL, 10-18
 - CXREF, 2-11, 10-18
 - DCI, 10-19
 - DECK, 10-19
 - DMAP, 10-19
 - Enhanced Source Listing reports, 2-5
 - HOGAN, 10-21
 - IDMSDC, 10-21
 - LIST, 2-25
 - LOWER, 10-22
 - MAP, 2-23
 - MDMAP, 2-22, 10-23
 - MLIST, 2-9, 10-23

- NOCICS, 10-17
- NOSOURCE, 10-30
- OPTRC, 2-10, 10-25
- PMAP, 2-5, 2-25, 10-27
- RDIS, 10-28
- SELECT, 2-5, 2-26, 10-28
- SHADOW, 10-29
- SOURCE, 10-30
- STATS, 10-30
- SUPMAP, 10-31
- TPMON, 10-31
- XDMAP, 10-33
- XREF, 10-34
- OPTIMIZER overrides, 8-3
- OPTRC, 10-25
- OS/VS COBOL, 10-1
- overriding defaults
 - card-image data set, 10-4
 - CBL statement, 10-2, 10-3
 - EXEC statement, 10-2, 10-3
 - in JCL, 5-1
- PFLOW, 10-26
- PGM, 10-26
- PLH, 10-27
- PMAP, 10-27
- precedence, 2-24, 2-26, 2-27
- PROCDIV, 10-27
- PRTREPT, 10-28
- PRTSZ80, 10-28
- RDIS, 10-28
- RPL, 10-28
- Runtime report, 4-7
- SAVEAREA, 10-28
- SELECT, 10-28
- SHADOW, 10-29
- SIZEMAX, 10-29
- SNAP, 10-29
- SOURCE, 10-30
- SOURCEMAP, 10-13, 10-30
- SSRANGE, 10-31
- STATS, 10-30
- STIMERM, 10-31
- SUBRNGCK, 10-31
- TPMON, 10-31
- UCB, 10-32
- unspecified, 10-1
- use, 10-5
- USERID, 10-32
- VSAM
 - ACB, 10-15
 - AMB, 10-15
 - AMBL, 10-15
 - PLH, 10-27

- RPL, 10-28
- VSAMIDX, 10-32
- VSAMIDX, 10-32
- WORKSTOR, 10-32
- WSINIT, 10-33
- WTO, 10-33
- XCOUNT, 10-33
- XDMAP, 10-33
- XREF, 10-34
- XREPORT, 10-34
- Options summary
 - CAIOPTNS procedure, 9-9
 - CAOUEOPTS program, 9-9
 - CA-SymDump/Batch, 9-9
- OPTRC
 - Optimizer option
 - compile-time, 2-10
 - description, 10-25
- OS/VS COBOL
 - combining options with Analyzer options, 10-12
 - compatibility, 2-4
 - options
 - in effect, 2-6
 - options - processing precedence, 10-1
 - Reporting, 9-5
- OS/VS COBOL load library, 8-6
- Output File Information panel, 8-7
- Overlay modules, 9-2
- Overriding
 - CSL file, 5-3
 - option defaults
 - card-image data set, 10-4
 - CBL statement, 10-2, 10-3
 - EXEC statement, 10-2, 10-3
 - options, 5-1, 5-3

P

- Panels
 - BROWSE CA-Optimizer
 - Errors, 8-9
 - Output, 8-8
 - ROSCOE Errors, 8-14
 - Defaulted Options, 8-7
 - DELETE CA-Optimizer Listing Data Sets, 8-10, 8-15
 - EXECUTE Optimized Program, 8-10, 8-16

- INVOKE Management Reporting System, 8-11, 8-17
 - Optimizer Invocation, 8-6
 - Output File Information, 8-7
 - Primary Option Menu, 8-5
 - PRINT CA-Optimizer Listing, 8-9, 8-15
- Paper saving features - MDMAP, 2-6
- Paragraph
 - execution-time summary, 7-21
 - flow report, 4-22
- PARMCHK
 - compile-time, 4-3, 4-4
 - execution-time, 10-26
 - purpose, 4-51
- PBL base locators, 2-25
- PERFORM
 - Range Table and ALTERed GO TO Table, 2-15
- PERFORM Range
 - branch out, 2-17
 - exits fall-through, 2-16, 2-17
 - logic error Identification, 2-17
 - NO EXIT, 2-18
 - NULL, 2-18
 - recursive, 2-18
 - RECURSIVE CALL, 2-17
 - return address analysis, 4-20, 4-21
 - return cell, 2-16
 - shared code, 2-17
- PFLOW
 - compile-time - Abend & SNAP facility, 4-3
 - description, 4-4
 - Detector Option
 - compile-time, 10-26
 - execution-time, 10-26
 - source listing report features, 4-7
- PGM Detector Option
 - execution-time description, 10-26
- PGT
 - hexadecimal locations, 2-12, 2-13
 - Memory Map, 2-12, 2-13
 - Program Literal Pool, 2-13
 - System Literal Pool, 2-12
- PLH Detector Option
 - execution-time, 10-27
 - open files report, 10-8, 10-11
- PMAP
 - description, 10-27
 - Optimizer
 - compile-time option, 2-25
 - reports, 2-5
- Postprocessor time
 - options
 - CPPRC, 10-18
- Primary Option Menu, 8-5, 8-12
- PRINT CA-Optimizer Listing panel, 8-9, 8-15
- Printing
 - PMAP selections, 2-5
- Printing an Optimized listing, 8-2
- Problem solving techniques - Detector, 4-1
- PROCDIV**
 - Analyzer report-time chart**, 10-12
 - description, 10-27
 - guidelines, 10-13
 - list, 10-13
- Procedure
 - CAIOPT, 2-4
 - code - size, 9-3
 - Division - MLIST sample listing, 2-9
 - map
 - object listing (PMAP), 2-25
 - SELECT, 2-26
 - size - how to measure, 2-21
- Processing
 - background, 8-6
 - foreground, 8-6
- Program
 - documentation, 7-3
 - exceptions, 4-13
 - Global Table (PGT), 2-13
 - location - TGT, 2-14
 - size measurement, 2-21
- PRTREPT
 - CAIPRINT report, 10-28
- PRTSZ80
 - 80-column format, 10-28
- PSV cell, 4-20

R

RDIS

- merged data map contents, 2-7

RDIS Optimizer compile-time option, 10-28

Recursion

- CA-Optimizer, 2-19
- simple, 2-19

Recursive

- CALL FROM - field description, 2-17
- PERFORM range
 - definition, 2-18

Register contents, 4-17

Report

- Change summary, 7-19
- tagging source statement, 7-29

Report features

- CPU time, 7-5
- customization, 4-3
- for abends, 4-1
- paragraph executions, 7-5
- source listing, 2-2
- suppressing Source Map, 7-5
- XCOUNT
 - deferring, 7-5
 - for single program, 7-5
- XREPORT - deferring, 7-5

Reports

Analyzer

- CNTXREF, 7-22
- Execution count cross-reference, 7-22
- fields, 7-10, 7-11
- Instruction Detail Report, 7-24, 7-25
- Module summary, 7-14, 7-15
- Paragraph execution-time summary, 7-21
- Run summary, 7-6, 7-13
- SOURCEMAP, 7-16, 7-17, 7-20, 7-28
- Unexecuted paragraphs, 7-20

CAIPRINT, 5-2

COBOL II Diagnostic Messages & Compiler Statistics, 2-20

Detector

- Abend, 4-1, 4-7, 4-11
- DB2, 4-32
- Execution Monitor, 4-43
- File section display, 4-15
- IMS, 4-26
- Intercepts, 4-1, 4-7, 4-9

Open Files, 4-22, 10-8, 10-11

Save area trace, 4-24

SNAP, 4-1, 4-8, 4-42

diagrams - Options Page, 2-6

Execution Monitor

- CAIOPTS, 4-7, 4-43, 7-10

- Module summary, 7-27

- summary, 4-7, 4-43

management reporting facilities

- MRS, 9-2, 9-4

- statistics, 9-5

MRS

- detail report, 9-2

- Summary, 9-4

Multi-module, 4-55

Optimization Results, 2-21

Optimizer

- ALTERed GO TO table, 2-15

- Compiler Data Map (MAP), 2-23

- Compiler Source Listing, 2-22

- Condensed Cross-Reference (CXREF), 2-11

- Condensed Optimized Object Code (CLIST), 2-24

- Merged Data Map (MDMAP), 2-6

- Merged Listing (MLIST), 2-9

- Options Page, 2-6

- PERFORM Range table, 2-15

- Procedure Map Object Listing (PMAP), 2-25

- Program Global Table (PGT) Memory Map, 2-12, 2-13

- Selected Procedure Map (SELECT), 2-26

- sequence, 2-5

- Task Global Table memory map (TGT), 2-14

Report-time

- coding options, 10-3

- options

- descriptions, 7-5

- guidelines, 10-13

- overview, 10-3

Repository index, 6-2, 6-3, 6-13

Repository viewer

- Starting, 6-1

- Using, 6-1

Return

- cell, 2-16

Return code, 10-17

RPL Detector option

- Execution-time

- description, 10-28

- open files report, 10-8, 10-11

Run summary report
description, 7-13
fields, 7-13

Runtime
INTERCEPTS, 4-7, 4-9
JCL, 4-5, 4-7
options
INTERCEPTS, 4-3
PARMCHK, 4-3
PFLOW, 4-3
SNAP, 4-3
Options
Report, 4-7

S

Save area trace report, 4-24

SAVEAREA
Detector Option
description, 10-28
report customization, 4-3

SELECT
Optimizer option
Compile-time, 10-28
groups of statements, 2-26
printing specific items, 2-5
procedure map (SELECT), 2-26

Selecting statement groups - LIST, 2-26

SHADOW
Analyzer restriction, 7-30
Optimizer option
compile-time, 10-29

SIZEMAX, 10-29

SNAP Detector option
diagnostics, 4-1
example, 4-42
execution-time
description, 4-3, 4-4, 10-29
SNAP CALL, 4-8

Source listing
count and timing report, 7-16

Source Listing
control options, 2-5

SOURCE Optimizer option
compile-time description, 10-30

SOURCEMAP

Analyzer

option chart, 10-12
report-time option, 10-13, 10-30
description, 7-16
explaining consolidation, 7-28
report example, 7-16

Specifying Analyzer options, 10-12, 10-13

Standard Detector options, 10-7, 10-10

Statistics report
monitoring results, 9-5
performance evaluation, 9-1

Statistics Report
compile-time, 2-20
instruction count, 2-21
total program size, 2-21

STATS Optimizer Option
description, 10-30

STEPLIB, 4-6, 7-9

STIMERM
time sampling, 10-31

SUBRNGCK
compile-time, 4-4
Detector Option
valid subscripts, index values, 4-2

SUBRNGCK/SSRANGE, 4-50, 10-31

Summary of new features, 1-4

Summary Report, 9-4

SUPMAP, 10-31

Support
BROWSING errors for ROSCOE, 8-14
CMS, 8-1
MRS for ROSCOE, 8-17
ROSCOE, 8-12, 8-15

Suppressing - counts and timings, 7-5

SXREF, 10-31

Symbolic information, 6-8

SYSPRINT, 10-22

SYSPUNCH
Optimizer compile-time option, 10-19

System Literal Pool - PGT, 2-12, 2-13

SYSUDUMP, 4-6

T

- Tagging source statement
 - change summary report, 7-29
- Task Global Table memory map (TGT), 2-14
- Techniques - optimization, 2-1
- Testing
 - and debugging with Analyzer, 7-1
 - Detector facility, 4-1
- TGT - memory map, 2-14
- Timings, 7-1, 7-27, 7-28
- TPMON Optimizer option
 - compile-time, 10-31
- TRACE, 10-31
- Transferring CMS file, 8-2
- TSO - support, 8-5

U

- UCB Detector execution-time option, 10-32
- UNCOND - MLIST, 2-9
- UNEXEC - MLIST, 2-9
- Unexecuted paragraphs report, 7-20
- Unspecified options, 10-1
- Unsupported features, 1-3
- Usage considerations
 - Detector, 4-53
 - MRS, 9-4
 - OS/VS COBOL reporting, 9-5
- Usage Considerations
 - Analyzer
 - Counts Results, 7-26
 - Reporter, 7-28
- USERID
 - Analyzer**
 - description, 10-32
 - report-time option**, 10-12, 10-13
 - unique page heading, 7-5

- Using
 - Analyzer options, 10-11, 10-12
 - options, 10-5
- Utilities Menu
 - functions, 8-2
 - panel, 8-1

V

- V2, 10-32
- Version 7.0 new features, 1-4
- VN cell, 2-18, 4-20, 4-21
- VS, 10-32
- VSAM control block
 - Detector option
 - display control, 10-8, 10-11
 - options, 10-15
- VSAM repository, 1-4
- VSAMIDX
 - Detector option
 - open files report, 10-11
 - Detector Option
 - open files report, 10-8
 - execution-time option, 10-32

W

- Working storage
 - variables, 4-51
- WORKING-STORAGE
 - section display, 4-16
- WORKSTOR
 - Detector Option
 - execution-time, 10-32
 - report customization, 4-3
- WSINIT
 - compile-time, 4-4
 - debugging tool, 4-51
 - Detector option
 - compile-time, 10-33
 - INTERCEPTS Report, 4-10
 - uninitialized working storage, 4-3

WTO
 abebd ti job log, 10-33

X

XCOUNT
 Analyzer option
 chart, 10-12
 compile-time, 7-5, 10-12, 10-33
 execution-time, 7-5, 10-13, 10-33
XDMAP - Optimizer compile-time option, 10-33
XMDMAP
 with COPY...SUPPRESS, 2-8
XREF
 description, 10-34
XREPORT
 Analyzer option
 description, 10-34
 execution-time, 7-5, 10-13
 summary chart, 10-12
 invoking Analyzer Reporter, 7-5
XTIME
 Analyzer option
 chart summary, 10-12
 execution-time, 7-5, 10-13
 description, 10-34

Y

Y2K source changes
 change summary report, 7-19