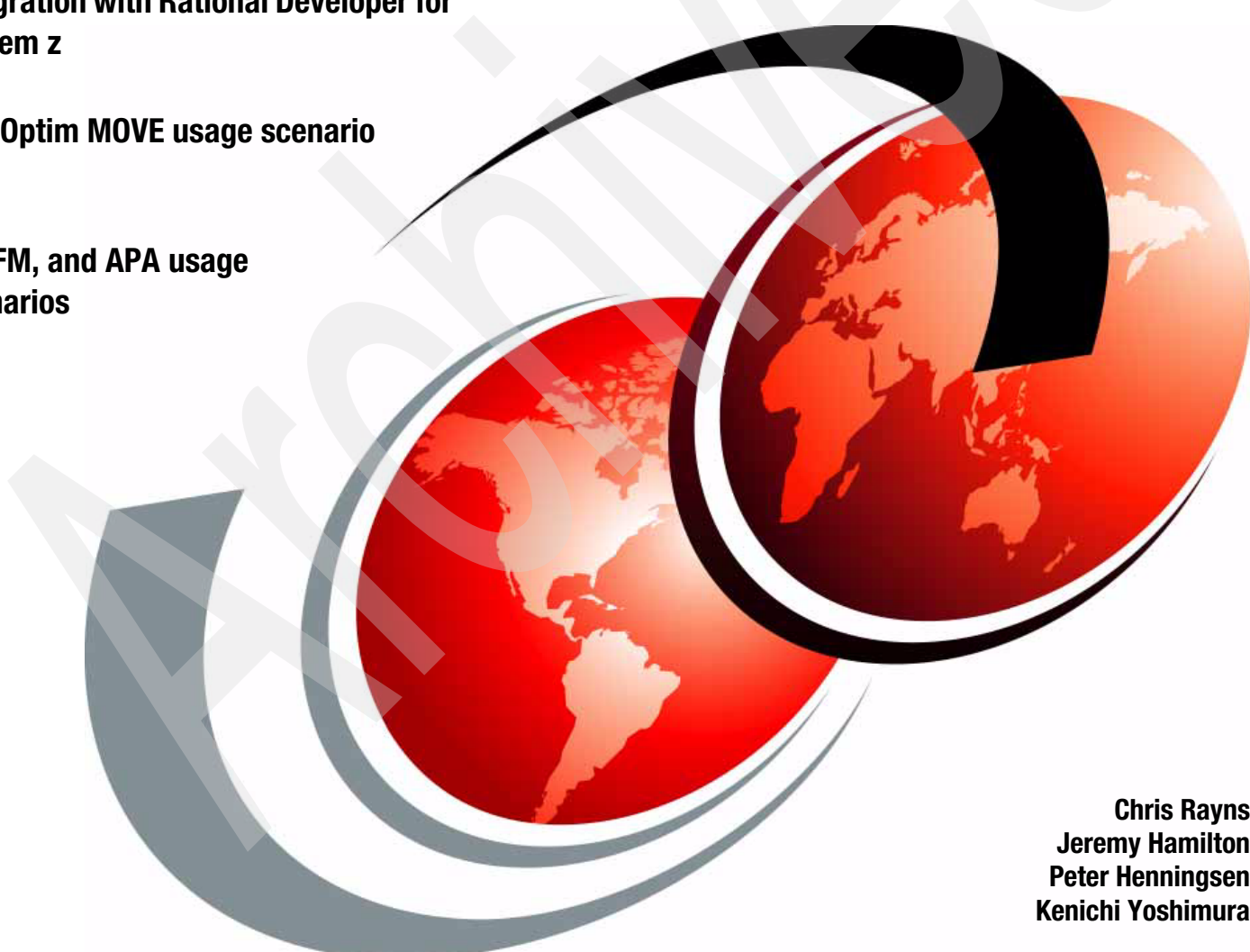


IBM Application Development and Problem Determination

Integration with Rational Developer for System z

IBM Optim MOVE usage scenario

FA, FM, and APA usage scenarios



Chris Rayns
Jeremy Hamilton
Peter Henningsen
Kenichi Yoshimura

Redbooks



International Technical Support Organization

**IBM Application Development and
Problem Determination**

September 2008

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

Archived

First Edition (September 2008)

This edition applies to Version 8, Release 1, Application Performance Analyzer, Version 8, Release 1, Debug Tool for z/OS, Version 8, Release 1, Fault Analyzer for z/OS, Version 8, Release 1, and File Manager for z/OS.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
Become a published author	x
Comments welcome	xi
Chapter 1. Application development tools for System z: An introduction	1
1.1 Introduction	2
1.2 IBM Application Performance Analyzer for z/OS	2
1.2.1 Highlights	3
1.2.2 Support for IBM subsystems	4
1.3 IBM Debug Tool Utilities and Advanced Functions for z/OS	5
1.3.1 Highlights	5
1.4 IBM Fault Analyzer for z/OS	6
1.4.1 Highlights	6
1.4.2 Support for IBM subsystems	7
1.4.3 Analysis options	7
1.5 IBM File Manager for z/OS	8
1.5.1 Highlights	8
1.5.2 File Manager components	9
1.5.3 SOA support	10
1.6 Optim MOVE	11
1.6.1 Test data management	11
1.6.2 Processing flow	12
1.7 Workload Simulator	14
1.8 Rational Developer for System z	15
1.8.1 Support for the IBM problem determination tools	15
1.9 IBM problem determination tools at a glance	16
Chapter 2. Integration of the Problem Determination Tools	17
2.1 Debug Tool Integration with Rational Developer for System z	18
2.1.1 Debugging COBOL, PL/I, Assembler, and C++ Programs	18
2.1.2 Right-clicking in the code space	23
2.2 Fault Analyzer Integration with Rational Developer for System z	24
2.2.1 Fault Analyzer Artifacts view	24
2.2.2 Detailed view	27
2.2.3 Working with older fault entries	28
2.2.4 Fault Analyzer Report browser	28
2.2.5 Fault Analyzer Dump browser	30
2.2.6 Source code display	31
2.2.7 Lookup view	33
2.3 File Manager Integration for Rational Developer for System z	34
2.3.1 Editing data set records using RDz File Manager	34
2.3.2 Editing the sample data set in the Formatted Data Editor	38
2.4 Across platform utilization	44
2.5 APA Integration with OMEGAMON	46
2.5.1 Tool integration	48
2.5.2 Using OMEGAMON with APA	48

2.6	APA and CICS PA in tandem	51
2.7	Workload Simulator with Rational Test Manager	54
Chapter 3. Program preparation		
3.1	Introduction	58
3.1.1	SYSDEBUG	60
3.1.2	Compiler listing	60
3.1.3	LANGX	61
3.1.4	SYSADATA	62
3.1.5	DWARF	62
3.2	COBOL	62
3.2.1	Enterprise COBOL V4.1 (5655-S71)	62
3.2.2	Enterprise COBOL V3 (5655-G53) and COBOL for OS/390 and VM (5648-A25)	63
3.2.3	COBOL for MVS and VM (5688-197)	64
3.2.4	VS COBOL II (5668-958)	65
3.2.5	OS/VS COBOL (5740-CB1)	66
3.3	PLI	67
3.3.1	Enterprise PLI V3.7 and later (5655-H31)	67
3.3.2	Enterprise PLI V3.5 and V3.6 (5655-H31)	68
3.3.3	Enterprise PLI V3.4 and earlier (5655-H31)	69
3.3.4	PLI for MVS and VM (5688-235), and OS/PLI (5668-909 and 5734-PL1)	69
3.4	Assembler	70
3.5	C and C++ (5694-A01) and later	71
3.6	Starting Debug Tool automatically	72
Chapter 4. Using Debug Tool on CICS programs		
4.1	What happened?	74
Chapter 5. Scenario for PD Tools with RDz		
5.1	Submitting the job	96
Chapter 6. Using APA to analyze CICS DB2 performance		
6.1	Getting started	118
6.1.1	Starting APA	118
6.1.2	Building the observation session	119
Chapter 7. Using APA to analyze batch performance		
7.1	Setting the scene	148
7.1.1	Starting APA	148
7.1.2	Building the observation session	148
7.2	Summary	163
Chapter 8. Optim MOVE test data capture		
8.1	Introduction	166
8.2	Provide IMS environment parameters	170
8.3	Define IMS segments as Legacy Tables	171
8.4	Define VSAM and QSAM files as Legacy Tables	176
8.5	Identify extract data sources and data source relationships	180
8.5.1	Define extract selection criteria	187
8.5.2	Specify parameters for Legacy Tables	188
8.6	Extract the data	189
8.7	View extract report	192
8.8	Use table lookup to convert extracted data	194
8.9	Delete previously defined objects	201

Related publications	209
IBM Redbooks	209
Other publications	209
Online resources	209
How to get Redbooks	210
Help from IBM	210
Index	211

Archived

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®	OMEGAMON®	REXX™
DB2 Universal Database™	OS/390®	System z™
DB2®	Parallel Sysplex®	Tivoli®
IBM®	QMF™	VTAM®
IMS™	Rational®	WebSphere®
Language Environment®	Redbooks®	z/OS®
MVS™	Redbooks (logo)  ®	zSeries®

The following terms are trademarks of other companies:

Adobe, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

J2SE, Java, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Expression, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The Problem Determination toolset, or PD Tools for short, consists of a core group of IBM® products designed to work in conjunction with compilers and run times to provide a start-to-finish development solution for the IT professional. This IBM Redbooks® publication can provide you with an introduction to the tools, guidance for program preparation for use with them, an overview of their integration, and several scenarios for their use.

When developing a system, an iterative process is usually employed. You can decide to create a starting set of test data using Optim MOVE to strip or modify existing databases, files, and data relationships. File Manager can also scramble, alter, or otherwise process this data. Once created, this data can be used to test the newly developed or maintained software.

Should an abend during testing occur, Fault Analyzer enables the programmer to quickly and easily pinpoint the abending location and optionally, the failing line of code. Many times, this is all the programmer requires to correct the problem. But occasionally, it might be necessary to delve a little deeper into the code to figure out the problem. Debug Tool allows the programmer to step through the code at whatever level is required to determine where the error was introduced or encountered.

After the code or data is corrected, the same process is followed again until no errors are encountered. But, this is not the whole story. Volume testing or testing with multiple terminals is sometimes required to ensure real-world reliability. Workload Simulator can be utilized to perform this type of testing.

After all tests have been completed, running the application using Application Performance Analyzer will ensure that no performance bottlenecks are encountered. It also provides a baseline to ensure that future enhancements do not introduce new performance degradation into the application.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the Silicon Valley Laboratory, in San Jose, California, USA.

Chris Rayns is an IT Specialist and Project Leader at the ITSO, Poughkeepsie Center in New York. Chris writes extensively on all areas of CICS TS. Before joining the ITSO, he worked in IBM Global Services in the United Kingdom (UK) as a CICS IT Specialist.

Jeremy Hamilton is a System z™ Technical Sales Specialist from Atlanta, GA. He has five years of experience with IBM, four at IBM's Almaden Research Center in California working on Molecular Electronics and Nano Fabrication, and one as a vitality hire for System z PD tools technical sales team. He is of the Cherokee nation and holds a Bachelors of Science in Engineering/Physics from Fort Lewis College in Durango, CO, and a Masters of Science in Information Systems from Santa Clara University in Santa Clara, CA.

Peter Henningsen is a Software Engineer in Australia Development Laboratory based in Perth, Australia. Since joining IBM in 2004, he has worked on several System z software products, including IBM Fault Analyzer for z/OS® and IBM SCLM Developer Toolkit. His main areas of expertise are Java™ on z/OS and Eclipse plug-in development. He holds a Masters Degree in Engineering Science from The University of Melbourne.

Kenichi Yoshimura joined IBM in 1977, having previously worked in the industry in various customer sites. His current role is as a Senior IT Specialist in a technical sales role in the WebSphere® Pillar of the A/NZ (Australia/New Zealand) zSeries® Software Group. This involves working with customers and IBM colleagues in the marketing and deployment of transaction and messaging systems (CICS®, WMQ, WAS), and the tools and languages that support the development and operation of applications in this space.

Thanks to the following people for their contributions to this project:

Tim Robertson
IBM Silicon Valley Laboratory

Peter Costigan
IBM Silicon Valley Laboratory

Jim Hildner
IBM US

Alan Schwartz
IBM US

John Leake
IBM Silicon Valley Laboratory

Francisco Anaya
IBM Silicon Valley Laboratory

Marty Shelton, Product Line Manager, Problem Determination Tools
IBM Silicon Valley Laboratory

Mark Duckworth
IBM US

Graham Hannington
IBM US

Thanks to the authors of the previous editions of this book:

Especially, we thank the authors of the last edition, *IBM Application Development and Problem Determination Tools V7 for System z*, on which we relied heavily for reference materials. It was published in June of 2007 under another book number, SG24-7372. The authors were: Amintore de Nardis, Mark Duckworth, Jennifer Nelson, and Adrian Simcock.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks® in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Archived

Archived



Application development tools for System z: An introduction

In this chapter we introduce the IBM tools that support application development on System z:

- ▶ Application Performance Analyzer for z/OS
- ▶ Debug Tool for z/OS and Debug Tool Utilities and Advanced Functions
- ▶ Fault Analyzer for z/OS
- ▶ File Manager for z/OS
- ▶ Optim MOVE
- ▶ Workload Simulator
- ▶ Rational® Developer for System z

These tools provide you with a comprehensive suite of products that can help improve application delivery and enhance production availability and performance.

1.1 Introduction

As businesses and the critical IT systems that support them grow increasingly complex, application developers are constantly struggling to meet the demands placed upon them. Service oriented architecture (SOA) has become the hottest topic in IT today, because it holds the promise of helping to address these demands. The ability to reuse existing assets is the cornerstone of SOA. This possibility is significant because reusing assets can be significantly less expensive than rewriting them. With the vast amount of existing applications running on the IBM System z platform, it only makes sense that System z applications can be a big part of the move to SOA.

IBM Application Performance Analyzer for z/OS, IBM Debug Tool Utilities and Advanced Functions for z/OS, IBM Fault Analyzer for z/OS and IBM File Manager for z/OS, along with Optim MOVE and Rational Developer for System z, provide a robust suite of problem determination tools that can help improve application delivery throughout the application life cycle.

You can use these tools to help increase productivity and IT effectiveness across source code generation and debugging, application abend analysis, data management and application-performance analysis. IBM problem determination tools do much more than support traditional applications. They include capabilities that enable you to build SOA applications. They are also tightly integrated with other tools in the IBM problem determination tools portfolio. All the tools also continue to support and make the most of the latest subsystem levels.

1.2 IBM Application Performance Analyzer for z/OS

Application Performance Analyzer for z/OS helps programmers and systems personnel to identify constraints and to improve the performance of their applications. It is a non-intrusive performance analyzer that helps you with design, development, production, and maintenance cycles.

The product's key function is to measure and report on how system resources are used by applications running in a z/OS address space, such as TSO and batch; and online subsystems, such as IMS™, CICS, or WebSphere Application Server; as well as DB2® stored procedures.

With a single interface, you can monitor applications in test, as well as in production, and in multiple source languages, including Assembler, C/C++, COBOL, PL/I and Java. Optimized code support for COBOL and PL/I is provided to enable you to monitor production applications.

1.2.1 Highlights

Here we summarize the main features of Application Performance Analyzer:

- ▶ Provides easy-to-use function that helps isolate application performance problems.
- ▶ Helps pinpoint performance bottlenecks affecting online transaction response times.
- ▶ Assists in reducing batch-application turnaround time.
- ▶ Supports Assembler, C/C++, CICS, COBOL, DB2, IMS, Java, PL/I, WebSphere MQ and WebSphere Application Server, including all the latest versions.
- ▶ Increases application understanding during stress and regression testing.
- ▶ Shares side files with Fault Analyzer and Debug Tool.
- ▶ Provides support for UNIX® System Services.
- ▶ Provides support for IBM Parallel Sysplex® Coupling Facility.
- ▶ Can be invoked from other programs such as IBM Tivoli® OMEGAMON® and IBM Tivoli Enterprise Portal.
- ▶ Provides the ability to compare two observation reports to see the relevant differences
- ▶ Supports threshold monitoring.
- ▶ Supports Java source mapping, along with support for a wide range of Java applications.
- ▶ Provides recommendations that can help you to improve the performance of your DB2 applications.

Using Application Performance Analyzer helps you maximize the performance of your existing hardware resources and helps you improve the performance of your applications and subsystems. It lets you evaluate application prototypes in the design phase, review the impact of increased data volume or changes in business requirements on performance, and generate historical data and reports to analyze performance trends and evaluate program changes for cost-effectiveness.

Running in a separate address space, Application Performance Analyzer non-intrusively collects resource utilization, wait time, and statistical samples from the address space being monitored. This data is then analyzed and documented. The reports generated help you to identify the key performance bottlenecks that require examination and resolution. This data is available in online and printed reports that you can choose to create as PDF files for viewing on a workstation.

1.2.2 Support for IBM subsystems

Application Performance Analyzer provides application-monitoring support for all of the major IBM subsystems, as shown in Figure 1-1.

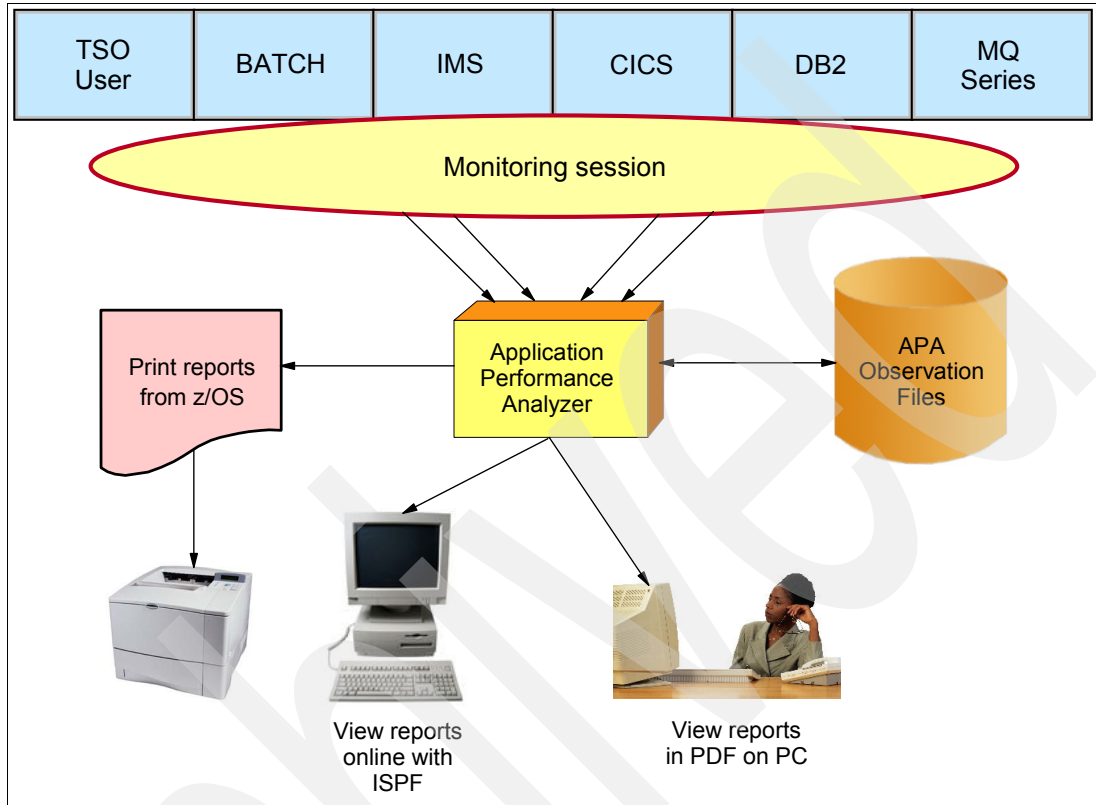


Figure 1-1 The APA environment

CICS

Support for IBM CICS Transaction Server enables you to monitor specific CICS transactions using wildcard transaction prefixes or by termID selection. With this support, you can trace transactions during critical situations, rather than waiting to review data that is collected periodically. Java application code running in the Java 2 Platform, Standard Edition (J2SE™) environment under CICS Transaction Server is also supported.

DB2

Support for DB2 delivers relevant information for performance analysis and tuning, including SQL statements and processor usage by SQL statement as well as for IBM DB2 stored procedures written in a traditional language or in Java.

IMS

Support for IMS applications means that you can have IMS application-performance data-on-call time and service-call time for DL/I. You can also choose to trace all IMS calls.

WebSphere MQ

WebSphere MQ support provides information about CPU usage by queue, by request, and by transaction. Application Performance Analyzer also provides service time by queue, request and transaction, as well as wait time, for the same categories.

1.3 IBM Debug Tool Utilities and Advanced Functions for z/OS

Debug Tool Utilities and Advanced Functions for z/OS provides debugging capability for applications running in a variety of environments, such as IBM CICS, IBM IMS, IBM DB2 stored procedures, and IBM UNIX System Services. To meet the requirements of IBM DB2 Universal Database™ for z/OS, Versions 8 and 9 and IBM CICS Transaction Server for z/OS, Version 3, Debug Tool Utilities and Advanced Functions also includes features to help you identify old OS/VS and VS COBOL II source code and to upgrade the code to IBM Enterprise COBOL.

To effectively build and service applications, you require robust, easy-to-use tools to compile, test, and debug them. IBM Debug Tool Utilities and Advanced Functions for z/OS, Version 8.1 software provides a complete solution that can help you to reduce application development cycle times. IBM Debug Tool for z/OS is included with Debug Tool Utilities and Advanced Functions for z/OS.

1.3.1 Highlights

Here we summarize the main features of IBM Debug Tool Utilities and Advanced Functions for z/OS:

- ▶ Provides a single debugging tool for batch, TSO, CICS, DB2, DB2 stored procedures, and IMS applications written in COBOL, PL/I, C/C++, Assembler and Java.
- ▶ Offers more productivity enhancements when used with IBM Rational Developer for System z. With Debug Tool Utilities and Advanced Functions, you can use Rational Developer for System z to provide a fully integrated development, test, and debugging environment for all applications running on z/OS, whether traditional or Web-based.
- ▶ Includes tools to identify OS/VS COBOL code and convert it to ANSI 85 standard.
- ▶ Supplies tools to help you determine how thoroughly your code has been tested.
- ▶ Helps you debug an application while it runs in a host environment, such as a batch application, TSO, Interactive System Productivity Facility (ISPF), CICS, IMS, or DB2 Universal Database (including IBM DB2 stored procedures) environments. Debug Tool Utilities and Advanced Functions can help you debug almost any application in almost any host environment, including COBOL, PL/I, C/C++ and Assembler applications running on z/OS systems. With Debug Tool Utilities and Advanced Functions, you can compile and link your COBOL, PL/I, C and C++, and Java programs, and assemble and link Assembler programs, as well as preprocessing and compiling your CICS and DB2 programs.
- ▶ Provides a CICS utility transaction that enables you to control debugging in the CICS environment.
- ▶ Provides an interactive, full-screen 3270 system-based terminal interface with four windows that enable single-step debugging, dynamic patching, and breakpoints:
 - A monitor window displays the status of items you select, variables and registers. You can view, monitor, and alter application variables or storage in real time.
 - A source window displays the program code, highlighting the statement being run. In the prefix area of this window, you can enter commands to set, display, and remove breakpoints.
 - A log window records and displays your interactions with Debug Tool and can show program output. The information that you see in this window is included in the log file.

- A memory window (swappable with the log window) that helps you display and scroll through sections of memory. You can update memory by typing over existing data with new data. The memory window keeps track of addresses for easier navigation.
- ▶ Using the Debug Tool Utilities and Advanced Functions source window, you can monitor application code while an application runs. You can also debug applications written in a mix of COBOL, C, C++, PL/I, or Java languages without leaving the tool. You can include Assembler programs in this mix and, using the disassembly view, you can debug programs compiled with the NOTEST compiler option or applications that include other languages. You can use commands to dynamically patch or alter the value of variables and structures and to control the flow of an application.

With Debug Tool Utilities and Advanced Functions, you can debug Enterprise COBOL and Enterprise PL/I applications that have been compiled with standard or full-optimization compiler options. You can also analyze your load modules to help you identify candidate OS/VS COBOL programs for conversion and then convert these OS/VS COBOL applications to Enterprise COBOL. These applications then can be compiled and debugged to extend the life of your existing code. Debug Tool Utilities and Advanced Functions software also provides coverage tools that enable you to conduct analysis on your test cases to determine how thoroughly they validate your programs.

1.4 IBM Fault Analyzer for z/OS

Fault Analyzer for z/OS provides the information you require to determine the cause, and assist with the resolution of application and system failures. Integrated support for Java and IBM WebSphere Application Server for z/OS gives Fault Analyzer expanded application coverage and related business value beyond traditional applications. You can use this one tool to assist in composite-application abend analysis, including 64-bit DB2 Universal Database support. It helps you repair failures quickly by gathering information about an application and its environment at the time of failure.

1.4.1 Highlights

Here we summarize the main features of Fault Analyzer for z/OS:

- ▶ Provides a detailed report about program failures to help you resolve them quickly
- ▶ Includes a fault-history file that enables you to track and manage application failures and fault reports
- ▶ Offers a view of storage contents, trace tables, and terminal screen images at the time of failure to help speed corrective action
- ▶ Provides the ability to customize message descriptions to be used in application-failure reports
- ▶ Supports all z/OS subsystems and compilers
- ▶ Provides integration with Rational Developer for System z

When an application abend occurs, Fault Analyzer captures and analyzes real-time information about the application and its environment, then generates an analysis report detailing the cause of the failure. The report describes the failure in terms of the application code, so you no longer lose time reviewing cumbersome, low-level system error messages. Fault Analyzer allows you to choose a report format to locate the information more easily.

Each application abend is recorded by Fault Analyzer in a fault-history file by job name, failure code and other details, along with the analysis report and storage pages referenced during the analysis. This information can later be retrieved to reanalyze the failure.

Through the inclusion of softcopy versions of selected manuals, Fault Analyzer can extract message and failure-code descriptions and insert them into the analysis report where applicable. You can also provide your own descriptions for messages.

You can write your own user exits. For example, you can write a user exit to access compiler listings that are compressed or available only through a proprietary access method.

Integration with Rational Developer for System z allows application developers to work with fault entries directly from their development environment, and also allows Debug Tool and Fault Analyzer to share common source files without redundancy.

1.4.2 Support for IBM subsystems

In the following sections, we discuss the support available for IBM subsystems.

CICS

Fault Analyzer functions are available from within the CICS transaction-based environment in a manner that is consistent with the ISPF usage, permitting you to review and analyze fault entries in history files without the necessity for a TSO logon.

The ability of Fault Analyzer to detect and analyze dump formatting and storage violations is particularly helpful for system programmers who have to debug CICS system problems. Options are provided to speed processing of duplicate dumps and to skip analysis of repeated abends.

DB2

Detail of the last SQL activity is provided, along with plan and security information.

IMS

Fault Analyzer provides general information about an IMS region along with detail of the last DL/I call parameter list, information for all PCBs in the order of their relative PCB number and, if available, JCB call trace information. IMS accounting information is also provided.

You can optimize the processing of dumps within the IMS environment, and speed up the processing of duplicate dumps.

WebSphere MQ

API information and return-code diagnostics are provided to assist in problem determination of WebSphere MQ applications.

WebSphere Application Server

Support for the latest versions of WebSphere Application Server and Java is provided.

1.4.3 Analysis options

Fault Analyzer provides three modes to help you better track and analyze application and system failure information:

Real-time analysis

When an application failure occurs, the Fault Analyzer exit starts real-time analysis. After failure processing, you can view the analysis report in the job output or through the Fault Analyzer ISPF or RDz interface.

Batch reanalysis

Batch reanalysis generates a new analysis report. This report is based on the dump and information gathered in real time, but with potentially different options specified, or with compiler listings or side files made available. You can submit a Fault Analyzer batch-mode job using either the Fault Analyzer ISPF or RDz interface or your own job control language.

Interactive reanalysis

Interactive reanalysis runs under ISPF and enables you to navigate through a formatted, structured view of a fully detailed reanalysis. This Fault Analyzer mode enables you to view working storage and control blocks at the time the dump was written. The interface has many point-and-click fields for easy navigation through the interactive reports.

1.5 IBM File Manager for z/OS

File Manager for z/OS offers comprehensive, user-friendly tools for working with z/OS datasets, CICS resources, DB2 data, and IMS data.

Extending the standard browse, edit, copy, and print utilities of ISPF, File Manager not only contains tools that support personnel will find useful, but also delivers enhancements that are designed to address the requirements of application developers working with files containing structured data. Also, because the basic features of the File Manager editor and the ISPF/PDF editor are deliberately almost identical, you can take advantage of extra features without having to relearn fundamental skills.

1.5.1 Highlights

Here we summarize the main features of File Manager for z/OS:

- ▶ Includes a File Manager editor that maintains the same general look and feel across both view (read-only) and edit functionality in all of the key areas, whether they are traditional z/OS datasets, CICS resources, or data stored in a DB2 or IMS database.
- ▶ Provides the ability to define record and field properties in the same manner as your application programs, that is, by using your existing COBOL and PL/I copybooks stored in partitioned data sets or library-management systems.
- ▶ Enhances usability by enabling you to customize which fields to display, copy or print.
- ▶ Enhances usability by providing the ability to define fields describing your data even when you do not have an existing copybook, by using dynamic templates.
- ▶ Supports development and production through extensive editing, browsing, and batch and interactive utility capabilities.
- ▶ Provides the ability to find, create, browse, copy, edit, print, format or reformat data files in the most-popular z/OS file formats. Not only can the same data mapping be applied as your applications use, but by the usage of File Managers templates, filtering can easily be applied to these operations based on the content of fields in your data. The template concepts of identification and selection criteria provide a simple to use, yet flexible interface for providing the simplest to the more complex of criteria.

- ▶ Provides data scrambling support to provide for certain forms of data protection.
- ▶ Provides integration with Rational Developer for System z.

1.5.2 File Manager components

In this section we describe the main components of File Manager.

Base component

The base component manages z/OS data sets, such as queued sequential access method (QSAM), VSAM, partitioned data sets (PDS and PDSE) and IBM z/OS UNIX System Services hierarchical file system (HFS) files.

The File Manager base component helps speed the application-development process by identifying the structure of your records and displaying each field in a human readable format, according to its data type.

This component supports QSAM, VSAM (including IAM - Innovation Access Method files), PDS and HFS datasets, including support for double byte character set (DBCS) data in these data sets.

You can edit entire files (regardless of size) and use a template or copybook for formatting and record selection during an edit or browse session.

DB2 component

Whether you are a DB2 DBA, application programmer, or just wanting to retrieve information stored in a DB2 database, the File Manager DB2 component provides something for you.

Database management abilities (such as create and drop objects, copying data within DB2, handling privileges and import/export) and an editor for DB2 tables and views (in either read only or update mode), encompassing all the usual Insert, Delete and Update functionality one typically requires in a database application.

Full tables, views, or the results of a limited SQL query that you have customized yourself are provided, with the ability to save the query you made, in a File Manager template.

If you are writing SQL to be used in your applications, another handy File Manager usage is the ability to be able to refine and test an SQL query by the prototyping and analysis feature (which includes an explanations feature).

Getting data out of, or back into DB2 to or from QSAM or VSAM files is also provided for by a powerful utility that allows the export of DB2 data in a number of formats including the external format used by the DB2 UNLOAD utility, the format used by the DB2 sample unload program DSNTIAUL, a tailor able File Manager export format with multiple options to control handling of NULL values and options for various column data types, and a delimited text format, such as comma-separated value output.

You can also generate batch JCL for the most commonly used DB2 utilities to save time

IMS component

With File Manager's IMS component, accessing data stored in an IMS database also becomes very easy. Although the database storage concepts are different than DB2, File Manager still delivers a similar look when viewing or editing your data and creating and saving customized queries is just as easy as the other components.

Using record structures defined in COBOL or PL/I copybooks, just like the BASE component, the File Manager IMS component enables you to view or edit IMS segments displayed with formatting according to their individual field data types. You can find and change data across an entire database hierarchy or in specified segments and fields.

Data displays for IMS always include the segment name to help you identify where you are in the database (handy in the cases where you are viewing a large result set, or an entire hierarchy).

Database navigation commands are available with flexible parameters and scoping, to allow very selective navigation or applicability, or a more generic approach, depending on your requirements.

You can use flexible criteria to select IMS database segments containing specific field values, and extract the segments into a sequential data set for later use. Or you can use the IMS component to load them into a different database.

CICS component

Those users wishing to manage or query data in CICS VSAM file resources, transient data queues and temporary data queues, have been provided for with the File Manager for CICS feature allowing access to CICS resources under a CICS transaction. The CICS resources supported for view, edit and certain File Manager utilities (such as data create, copy and compare) are VSAM files, temporary storage queues and transient data queues.

You can authorize users to view and change the status of supported local and remote CICS resources and perform File Manager base and File Manager IMS tasks from the File Manager CICS transaction.

The same display and record selection capabilities are present in the CICS environment as are in the BASE product, providing for quick easy access to CICS data. Also very useful is the ability to create full copies of open CICS files (or even TS or TD queues) to another QSAM, VSAM, HFS or even PDS file.

Another nice benefit of File Manager for CICS is that the majority of File Manager related processing happens offline to the CICS task, so it runs little risk of adversely affecting other CICS users, even when (for example) copying an entire file somewhere else.

File Manager for CICS even maintains the same look and set of editor commands that you are possible used to in an ISPF environment. Logging off of your CICS environment to log on to your ISPF is not even necessary if all you want to do is use the File Manager BASE or IMS components.

1.5.3 SOA support

File Manager enables you to generate XML data from files. A File Manager template that describes the data-record layouts is required. The XML tags are generated based on the field names from the template, and the XML content comes from the data. A number of options are provided for the handling of invalid and unprintable data.

File Manager provides a plug-in for integration with Rational Developer for System z, allowing all aspects of Web-service (and traditional application) development to be undertaken from the same developer tool.

File Manager is enhanced to provide a scrambling algorithm that modifies data in a field while maintaining its system data type. Scrambling is intended to de-identify (disguise) personal information in different ways for different data types. The goal of repeatable scrambling is that application relationships based on equality tests can be maintained, if desired, even after the data is scrambled. A number of standard algorithms are provided to give the user complete control over the type of scrambling performed.

1.6 Optim MOVE

IBM Optim MOVE for DB2 and IBM Optim for Legacy for z/OS are components of the IBM Optim solution for the z/OS environment. Optim enables you to assess, classify, subset, archive, store, and access enterprise application data. Optim uses the relationships defined in the DB2 Catalog, where available, and supplements these relationships with those defined in the Optim (OPT) Directory. Optim runs as a TSO/ISPF application and incorporates familiar ISPF commands. Optim handles any number of tables and relationships, regardless of their complexity.

Using the features in Optim, you can:

- ▶ Isolate historical data from current activity and safely remove it to a secure archive.
- ▶ Access archived historical data easily, using familiar tools and interfaces.
- ▶ Restore archived data to its original business context when it requires additional processing.
- ▶ Create realistic test data.

Optim helps you achieve these benefits with the following components: ACCESS, ARCHIVE, MOVE, and COMPARE. This section introduces MOVE and Move for Legacy. A reference to Move includes Move for Legacy.

1.6.1 Test data management

IBM Optim MOVE offers comprehensive test data management capabilities for creating realistic, right-sized test databases, protecting data privacy in vulnerable development and testing environments and validating test results. You eliminate costly cloning processes and correct defects early in the development process, where they are cheapest and easiest to fix. Most importantly, Optim scales to meet your development and testing requirements, across applications, databases, operating systems, and hardware platforms.

The Optim test data management capabilities provide an efficient alternative to database cloning, allowing you to create development and testing environments that are sized appropriately.

MOVE is a relational facility used to copy or move sets of relationally intact data from multiple source tables to corresponding destination tables. Move for Legacy allows you to perform the same processes, using data in IMS, Virtual Storage Access Method (VSAM), or sequential files.

MOVE is indispensable when:

- ▶ Creating test data that is a relationally intact subset of existing production data.
- ▶ Copying related sets of data to an “area” where any problems can be resolved. Then, the corrected data can be re-introduced to the production data.
- ▶ Migrating subsets of data that require data transformations as part of the Migration.

Programmers and database administrators (DBAs) are no longer required to spend hours writing programs to accomplish these tasks. Instead, MOVE provides an interactive utility to prompt for the necessary information and perform the task.

Because MOVE runs as a TSO/ISPF application, the Help and Tutorial facility in ISPF are also supported. Menu-driven prompt screens or panels are used to specify which data to obtain and which process to perform. Intelligent screen handling technology provides simultaneous display of multiple tables, pop-up windows, cursor-sensitive online help, and tutorials.

Tables

Throughout the Optim documentation, the term “tables” refers to not just to DB2 tables, but also to views, aliases, and synonyms, as well as to Legacy Tables that describe data in IMS, VSAM, or sequential files.

1.6.2 Processing flow

To copy a set of related data, you define the source of the data and the destination. The source is defined in the Extract Process and the destination in the Insert Process.

Extract process

An Extract is the process of copying data from one table or a set of related tables to an “Extract File.” An Extract File is a sequential file that contains the extracted data and information about that data. The supporting information defines the characteristics of the data, and optionally includes the object definitions for that data. (Object definitions are the definitions for tables, keys, indexes, and so forth.) An Extract File can be used repeatedly to create new databases or to refresh existing databases.

To perform an Extract Process, you must specify:

- ▶ The names of the tables from which data is extracted and the name of the Start Table or the table used as the starting point for the process.
- ▶ The relationships between the tables to be used for the extract. By default, all relationships are used. Note that, if an unload program is available, you also can indicate whether it is to be used to extract the data.

To identify or limit the rows to be extracted, you can use:

- ▶ Selection criteria for any table
- ▶ A list of rows in the Start Table
- ▶ A selection factor and maximum row limit

Table definitions (DB2 and Legacy) and column definitions are always extracted. You can also extract subordinate definitions as follows:

- ▶ Primary Keys • Column Field Procedure Names
- ▶ Relationships • Triggers
- ▶ Indexes • User Defined Types
- ▶ Views • User Defined Functions
- ▶ Aliases • Stored Procedures
- ▶ Synonyms

Access Definitions

The information that identifies the data for the Extract Process — table names, relationships, and selection criteria — is stored in an Access Definition, which is copied to the Extract File.

A named Access Definition is available to all components of Optim. You can also define an unnamed, temporary Access Definition to be used for a single Extract Process.

Next we describe the various Access Definitions.

Insert Process

After an Extract File is created, the data in the file can be inserted into the destination tables. These tables might or might not reside in the DB2 subsystem from which they were extracted.

Table Maps

A Table Map matches source tables to destination tables. Using a Table Map, you can exclude individual tables from processing and map tables with dissimilar names. You can use one or more existing Table Maps (APPLY command) as the basis for a new Table Map, or define the Table Map at the same time you define other specifications for the process.

When the names and attributes of columns in the destination table match those of columns in the source table, MOVE automatically inserts the data. If the columns do not match, you can use one or more Column Maps to tell MOVE how to insert the data.

Column Maps

A Column Map matches source and destination columns that have different names or attributes. You can also use a Column Map to provide default values, or specify literals, constants, expressions and exit routines used to populate destination columns. Although MOVE automatically handles many transformations, you can use custom exit routines for more complex processing.

Create Object Definitions

MOVE generates the appropriate SQL DDL statements to create destination tables that do not exist. These tables can be created as part of the Insert or in a separate step, and the created tables can be identical to the source tables and columns, or you can modify the table definitions.

Other object definitions, such as primary keys, relationships, indexes, etc., that have been extracted with the data can be created by request also.

Insert processing

You can choose the type of processing for all tables as a group or by table:

- ▶ If you select Insert processing: A row in the Extract File is inserted when it does not already exist at the destination. However, if the row exists, the row to be inserted is discarded.
- ▶ If you select Update processing: A row in the Extract File replaces or updates a matching row in the destination. However, a source row that does not match a destination row is not inserted.
- ▶ If you select both Insert and Update processing: New rows are inserted and existing rows are updated.

Other processes

In addition to Extract and Insert, MOVE includes the following other processes or facilities:

- ▶ **Load Process:** To insert large volumes of data, the Extract File can be transformed into a DB2 Load or other load format to be used with a Load Utility.
- ▶ **Create Process:** The object definitions in the Extract File can be used to create any and all objects (tables, primary keys and relationships, indexes, etc.) without inserting source data.
- ▶ **Convert Process:** The data in the Extract File can be transformed according to parameters in the Table Map and Column Maps and saved as modified. Conversion is useful for masking source data or creating consistent, refreshable test data.
- ▶ **Retry:** Retry can be executed for one or more rows in the Extract File that cannot be inserted. A row that cannot be processed is marked as discarded. The Retry Process attempts the previous operation for the discarded rows only.
- ▶ **Restart:** Restart can be executed when a process does not complete successfully. A process might terminate unexpectedly due to resource restrictions, such as space allocation or time limitations. Using Restart, the process is restarted from the last commit point. MOVE tracks the most recently committed row in the Extract File and restarts the process with the succeeding row.
- ▶ **Browse:** The Extract File can be browsed. Browsing is useful for reviewing the error status of individual rows or for verifying the data in an Extract File.

1.7 Workload Simulator

Workload Simulator (WSim) provides the ability to simulate terminals and the associated messages. It also lets the user alter message loads during a run. It can be used to generate a large volume of messages to evaluate the reliability and approximate performance characteristics of a network under expected operating conditions. Simply stated, anything that a real user can do at terminal, Workload Simulator can do faster, more reliably, and typically at less cost.

IBM Workload Simulator has these benefits and features:

- ▶ Helps prepare your networks for peak transaction volumes
- ▶ Enables testers to conduct reliable tests on stress, performance, and capacity as well as regression and function tests
- ▶ Simulates different terminals, terminal features, and terminal operator actions
- ▶ Provides support for enhanced TCP/IP support, SNA, and CPI-C (LU 6.2)
- ▶ Helps manage the test process
- ▶ Offers several options for creating scripts to use in simulations
- ▶ Provides screen images, data, and reports during simulation
- ▶ Helps testers compute and analyze test results

You can run WSim on MVS™ in batch mode, as a procedure, using the more user friendly WSim/ISPF Interface, or under TSO.

1.8 Rational Developer for System z

Rational Developer for System z (RDz) speeds the development of traditional mainframe, Web, and composite applications. Built on an Eclipse platform, it lends itself readily to integration with other products, both IBM and non-IBM.

1.8.1 Support for the IBM problem determination tools

In this section we discuss the IBM problem determination (PD) tools.

Remote compile generation

With remote editing, compiling, and debugging, you can develop or enhance many types of applications, including CICS, batch, TSO, or IMS Transaction Manager applications. These applications can access many forms of data such as DB2, VSAM, DL/I, and QSAM data. You can save time by editing, compiling, and debugging host applications remotely. When editing, compiling, and debugging host files from the workstation, you work in a cooperative Windows® and TCP/IP-based development environment, avoiding lengthy downloads and uploads unless explicitly desired.

Debug Tool Utilities and Advanced Functions (DTU&AF)

COBOL and PL/I applications, EGL applications, and Java applications can be debugged through a common interface using the local debugger. The remote debugger supports debugging of code that runs in the following z/OS environments:

- ▶ Batch
- ▶ TSO
- ▶ CICS
- ▶ IMS, both IMS Database Manager and IMS Transaction Manager, with or without Batch Terminal Simulator (BTS)
- ▶ DB2 (including stored procedures)
- ▶ WebSphere Application Server

The debugging sessions are cooperative. The remote distributed debugger resides on the workstation and interfaces with the IBM Debug Tool Utilities and Advanced Functions, which runs on the host with your application. The workstation interface communicates with the host z/OS products through TCP/IP.

File Manager

File Manager (FM) integration enables access to:

- ▶ VSAM KSDS files for browsing and updating
- ▶ template-driven display of VSAM, PDS members, and sequential file data

Fault Analyzer

Fault Analyzer (FA) integration allows users to:

- ▶ Browse FA abend reports
- ▶ View dump selections relating to abends
- ▶ Annotate reports for future reference or to share comments with other users who browse the same reports

1.9 IBM problem determination tools at a glance

IBM problem determination tools include Application Performance Analyzer, Debug Tool Utilities and Advanced Functions, Fault Analyzer and File Manager. These tools, along with Optim MOVE and Rational Developer for System z, are designed to help ease the burden of developing, testing and supporting service-oriented and composite applications across complex IBM System z environments.

By helping to improve application delivery throughout the application life cycle, these tools provide increased user productivity and IT effectiveness across source code debugging, application-abend analysis, data management and application-performance management.

System z tools, including problem determination tools, CICS tools and application development tools, support the entire enterprise-application life cycle to help you build, integrate, test and manage enterprise solutions. As a result, you can make the most of your System z platform investments and expedite your move to SOA. With these tools, you can transform your applications and optimize your IT operations to achieve greater business flexibility, without losing touch with governance and compliance.



Integration of the Problem Determination Tools

The IBM Problem Determination Tools have functionality that compliments each other as well as the other tools in the IBM Software portfolio. In this chapter, we describe these functions.

2.1 Debug Tool Integration with Rational Developer for System z

The IBM Debug tool Debug Tool Utilities and advanced functions (DTUAF) can interface with RDz to provide an interactive source debugging of z/OS applications. DTUAF helps you debug applications running in a variety of environments, such as CICS, IMS, DB2 stored procedures, and UNIX System Services.

In this section we cover the integration of DTUAF with the RDz interface, and how this integration can help ease the process of developing applications for System z. DTUAF is composed of complimentary products which enable symbolic debugging of COBOL, PL/I, C, C++, and High Level Assembler (HLASM) code. The symbolic information that is made available in the debugging perspective includes entry point names, variable names, labels, and registers including general purpose registers as well as floating point registers.

The debugger allows you to control the execution of your program by setting breakpoints, suspending launched programs, stepping through your code, and examining the contents of variables. The debugger has a client/server design so you can debug programs running remotely on other systems in the network as well as programs running locally on your workstation. The debug client runs inside the workbench on your workstation, while the debugger server runs on the same system as the program you want to debug. This could be a program launched on your workstation (local debugging) or a program started on a computer that is accessible through a network (remote debugging).

2.1.1 Debugging COBOL, PL/I, Assembler, and C++ Programs

The most dramatic difference between Debug on 3270 and Debug on RDz is the graphical user interface (GUI). While the 3270 provides access to a superset of the tool's functions, RDz provides a GUI front end to a large amount of Debug's capabilities, and has access to a 3270 emulator for those functions not currently available with the interface. In order to use an interactive debugger we have to tell the debugger where to interact with us (the end user). We could use a VTAM® screen (also known as a green screen), but this is not necessary because we can route the interactive debug session to RDz. To do this, go to the debug perspective and ensure that RDz is listening for an interactive debug session and determine the IP address of our client to direct the debug session to our RDz client. To switch to the debug perspective, use the Window pull-down (Window → Open Perspective → Other) in the Open Perspective pop-up and select Debug.

When working with RDz, you can use a z/OS-based debugging engine to debug code in an MVS subproject, or you can use a workstation-based debugging engine to debug code in a local project. For the z/OS based you must be in online state, and when working on a local project you can work remotely. The user's experience for both is very similar, and the following tasks can be executed on the source code during your debug efforts.

After launching a debug session, there are debug views available that provide access to a variety of debug tasks. Views that are available for debugging are listed in Table 2-1.

Table 2-1 Default views for the debug perspective

View	Description
Debug view	Allows you to manage program debugging
Debugger editor	Displays source for your program

View	Description
Breakpoints view	Offers a convenient location for setting and working with breakpoints.
Variables view	Contains a list of all variables in your application and which allows you to edit variables
Registers view	Displays registers in your program.
Monitors view	Provides a convenient location for working with variables, expressions, and registers that you choose to monitor.
Modules view	Displays a list of modules loaded while running your program.
Debug Console	Allows you to issue commands to the debug engine, view output from the engine, and see results of commands that you have issued.
Memory view	Allows you to view and map memory used by your application.

Call stack

A call stack is a dynamic stack data structure which stores information about the active subroutines of a computer program. A call stack is often used for several related purposes, but the main reason for having one is to keep track of the point to which each active subroutine should return control when it finishes executing. With RDz, when a program is suspended, the Call Stack is shown (Figure 2-1).

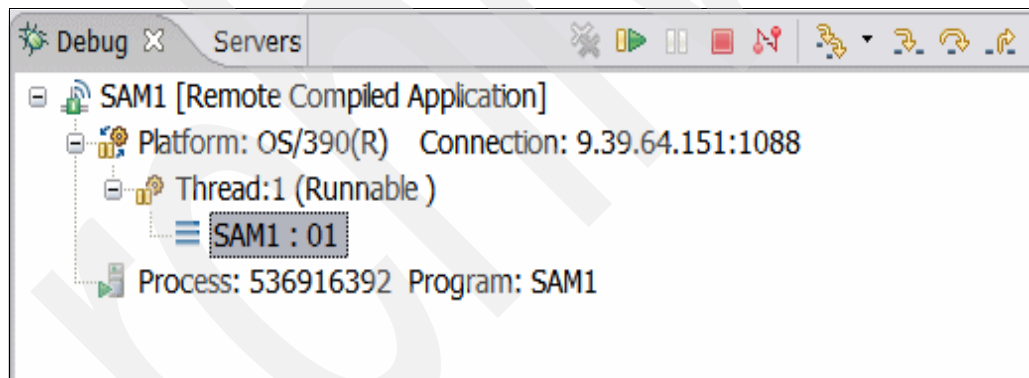


Figure 2-1 Call Stack

Setting and learning breakpoints at a specific line

Breakpoints are temporary markers that you place in your executable program to tell the debugger to stop your program at a given point. When a breakpoint is encountered, execution suspends at the breakpoint before the statement is executed, at which point you can see the stack for the thread and check the contents of variables, registers, and memory. You can then step over (execute) the statement and see what effect it has on the argument.

To access the wizards for setting breakpoints, right-click in the Breakpoints view and select Add Breakpoint from the pop-up menu. This will expand to a menu that allows you to choose the breakpoint type that you want to set. When you use the wizard to set a breakpoint, you can specify optional breakpoint parameters and set conditional breakpoints.

The breakpoint types in Table 2-2 are supported when debugging System z applications.

Table 2-2 Breakpoint descriptions

Type	Description
Statement breakpoints	Triggered when the statement they are set on is about to be executed.
Entry Breakpoints	Triggered when the entry points they apply to are entered.
Address Breakpoints	Triggered before the disassembly instruction at a particular address is executed
Watch Breakpoints	Triggered when execution changes data at a specific address.
Load Breakpoints	Triggered when a DLL or object module is loaded.
Exception Breakpoints	Triggered when an exception that is recognized by the debugger is thrown.

Exception breakpoints are set in the Breakpoints view by clicking the Manage Compiled Language Exception breakpoints push button and then selecting the exception type that you want the debugger to catch in the Manage Exception Breakpoints dialog box. The exception types that are available when debugging System z are TEST(ALL), TEST(ERROR), and TEST(NONE). You can select only one of these selection types.

Statement breakpoints can be set in the Debugger Editor by double-clicking on the ruler area to the left of a statement or by a right-click pop-up menu action - or they can be set by wizard in the Breakpoints view. Entry breakpoints can be set in the Modules view by right-clicking an entry point and selecting Set entry breakpoint from the pop-up menu — or they can be set by wizard in the Breakpoints view. In addition, you can right-click the debug target (or one of its threads or stack frames) in the Debug view and select **Options** → **Stop At All Function Entries** from the pop-up menu to stop at all entry points (this option is also available in the Breakpoints view pop-up menu). All other breakpoint types are set by wizard in the Breakpoints view.

A list of all breakpoints (for all debug sessions) is displayed in the Breakpoints view, unless you use the filter by debug target action or link the Breakpoints view to the Debug view. To filter out breakpoints that are not related to the current debug session, click the Breakpoints view Show Breakpoints Supported by Selected Target push button. To link the Breakpoints view with the Debug view click the Link with Debug View toggle - when this toggle is selected and a breakpoint suspends a debug session, that breakpoint will automatically be selected in the Breakpoints view.

The breakpoint entries in the list provide you, in brackets, with a summary of the breakpoints' properties. With pop-up menu options, you can add breakpoints, remove breakpoints, and enable or disable breakpoints. If you have Debug Tool for z/OS V6R1 or higher, you can also edit breakpoint properties with a pop-up menu option. With push buttons in the Breakpoints view, you can remove breakpoints.

Breakpoints can be enabled and disabled with pop-up menus in the Breakpoints view or the editor and by check box in the Breakpoints view. When a breakpoint is enabled, it will cause all threads to suspend whenever it is hit. When a breakpoint is disabled, it will not cause threads to suspend.

In the Breakpoints view, there are two indicators to the left of a set breakpoint. To the far left is a check box which indicates if the breakpoint is enabled (when enabled, the check box contains a check mark). To the near left, an indicator with a check mark overlay indicates a breakpoint that has been successfully installed by the debug engine (if the breakpoint is enabled, this indicator is filled — if the breakpoint is disabled, this indicator is not filled). In the editor, statement breakpoints are indicated by an indicator with a check mark overlay, indicating a breakpoint that has been successfully installed by the debug engine (if the breakpoint is enabled, this indicator is filled - if the breakpoint is disabled, this indicator is not filled).

Breakpoints must be installed before they will suspend execution. It is possible to add a breakpoint that is not valid for the current debug session. This breakpoint will not be installed until it is part of a debug session which includes a debug engine that will recognize the breakpoint.

In the editor, statement and entry breakpoint indicators are displayed in the marker bar to the left of the editor. Indicators for statement, entry, address, watch, and load breakpoints are displayed in the Breakpoints view.

While in the Breakpoints view, the source editor will open to the location of a breakpoint if you do one of the following actions:

- ▶ Double-click the breakpoint.
- ▶ Select the breakpoint and click the Go to File For Breakpoint push button.
- ▶ Right-click on the breakpoint and select Go to File from the pop-up menu.

Step procedures

After the code is suspended, you are able to use the stepping procedures, which allows you step through the execution of the program line-by-line using the step controls you. While performing a step operation, if a breakpoint or exception is encountered, execution suspends at the breakpoint or exception, and the step operation ends. You can use step commands to step through your program a single instruction or location at a time. The available step commands are listed in Table 2-3.

Table 2-3 Step actions

Command	Description
Step Over	Called functions run without stepping into it.
Step Into	Program runs to the next hook or statement.
Animated Step Into	The debugger issues a step into action repeatedly. You can control the delay between each step by selecting the Animated Step Into icon down-arrow.
Step Return	Program runs to the return point (just after the call point).

Select the **Step Into** command to step into the next statement at the currently executing line of code. As you step through the program, the line of code that you are currently executing on will be highlighted as shown in Figure 2-2. The values of the variables that are currently being used are shown and will change as the code executes.

```

252          ACCEPT CURRENT-TIME FROM TIME.
253          DISPLAY 'SAM1 STARTED DATE = ' CURRENT-MONTH '/'
254              CURRENT-DAY '/' CURRENT-YEAR ' (mm/dd/yy)'.
255          DISPLAY '          TIME = ' CURRENT-HOUR ':'
256              CURRENT-MINUTE ':' CURRENT-SECOND.
257

```

Figure 2-2 Current line of execution

Select the **Step Over** command to step over the next method call (without entering it) at the currently executing line of code. Even though the method is skipped, as far as stepping is concerned, it is still executed all the same.

Variable tools

Any function or program executes to change data in an expected way, so a key part of debugging is to locate and determine if the program is acting on the data correctly. The first step is finding a variable. Locating and determining its value in a complex program while debugging can be a tedious. RDz and debug have a couple of solutions for this. In one method, when a breakpoint is set and the program is suspended, you can use the Hover feature. This feature allows you to see a variable's current value by placing the mouse pointer over the variable name while the program is suspended.

```

DDS0200.ADLAB.SYSDEBUG(SAM1)
Line 257      Column 45      Insert      Browse
-----+-----+-----+-----+-----+-----+-----+-----+
251          ACCEPT CURRENT-DATE FROM DATE.
252          ACCEPT CURRENT-TIME FROM TIME.
253          DISPLAY 'SAM1 STARTED DATE = ' CURRENT-MONTH '/'
254              CURRENT-DAY '/' CURRENT-YEAR ' (mm/dd/yy)'.
255          DISPLAY '          TIME = ' CURRENT-HOUR ':'
256              CURRENT-MINUTE ':' CURRENT-SEC[CURRENT-HOUR = 13]
257
258          PERFORM 900-OPEN-TRAN-AND-RPT-FILES.
259          PERFORM 800-INIT-REPORT .
260
261          PERFORM 100-PROCESS-TRANSACTIONS
262              UNTIL WS-TRAN-FILE-EOF = 'Y' .

```

Figure 2-3 Variable "Hover" Feature

You can also set up a variables window that will show the variable name and its value as you step through the program. To open the variables window, you go to *Window* in the top tool bar, select **Show View**, then near the bottom you see *Variables*. You can also press ALT+SHIFT+Q, V, and a window will pop up near your *Breakpoints* window, looking somewhat like Figure 2-4.

Breakpoints Registers Monitors Modules (x)= Variables ☒	
Name	Value
● CURRENT-MONTH	05
● CURRENT-DAY	02
● CURRENT-YEAR	08

Figure 2-4 Current variable value display

In the Variables View, you can dynamically change the variable values as you step through the code. This will allow you to enter correct data into the variable for testing the rest of the program.

2.1.2 Right-clicking in the code space

During a Debug session, the “right-click” is your access to many features. Right-clicking in the code space during a debug session gives access to functionality that is useful in debugging source code.

Table 2-4 Right-click menu in the code space

Action Name	Description	Hot Keys
Find Text	Enter a string of characters to find within the code.	CTRL+F
Find Function or Entry Point	Enter the name of the function or entry point that you are searching for.	CTRL+F10
Add Watch Breakpoint	Set up a “Watch Breakpoint” with the wizard that pops up. Set up number of bites to watch and frequency.	
Monitor Expression®	Select thread that has the expression you want to monitor, and the tool places a monitor in the monitor tab	
Monitor Memory	Allows you to add a new memory monitor, which lets you view and change contents of memory or memory areas used by your program.	
Edit Source Lookup Path	Edit the path used to locate source files.	
Change Text Files	Allows you to enter the name of an overriding file for the source.	
Switch Views	Change views.	

2.2 Fault Analyzer Integration with Rational Developer for System z

The Fault Analyzer plug-in is the newest addition to the suite of Rational Developer for System z tools that help you control every part of the software design and deployment process. The plug-in simplifiesabend analysis by connecting to a remote IBM z/OS system and displaying the result of the analysis in the familiar workspace.

2.2.1 Fault Analyzer Artifacts view

The Fault Analyzer Artifacts view (Figure 2-5) enables you to register a set of history files that can be browsed to identify the cause of failures in your applications, and to monitor the contents of history files to alert you when a failure occurs in one of your applications.

History files are organized in a tree structure in which the root of the tree is FA Artifacts; which contains children that represent different mainframe systems. Each system node potentially contains two children, one for containing all history files for the system and another for containing the view information. When you double-click on a history file element, all fault entries contained in the selected history file are displayed in the Detailed View (page 5). A view is a Fault Analyzer concept that enables you to group a set of history files. When you select a view element, fault entries contained in all history files defined in the selected view are displayed in the Detailed View.

Note: A view enables you to group a set of history files, as well as to define the column layout for the Fault Entry List display in the ISPF environment. However, the column layout information defined in the view definition is ignored by the Fault Analyzer Integration feature at this stage.

Views must be defined on the host and retrieved using Retrieve View Information action from the context menu in the Fault Analyzer Artifacts view (FA Artifacts). See the section, “Setting up views,” in the *Fault Analyzer User's Guide and Reference*, SC19-1253.

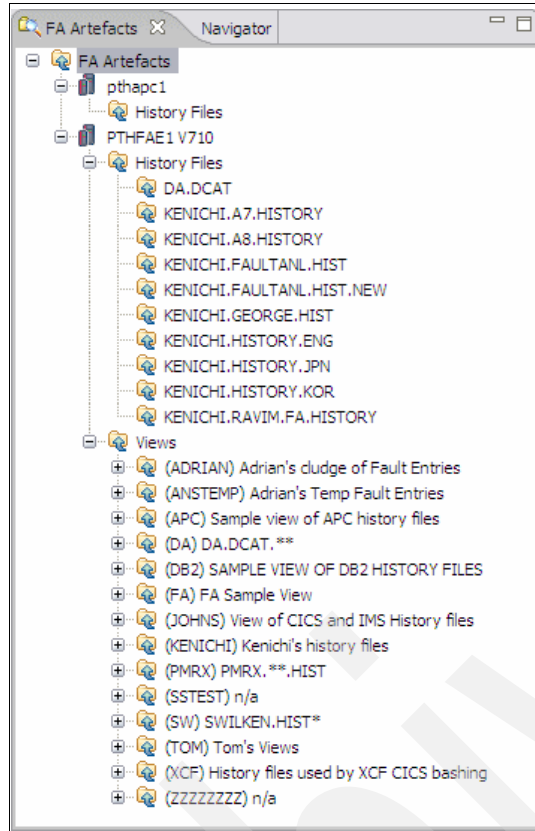


Figure 2-5 FA Artefacts view

When you invoke the Fault Analyzer ISPF interface, a set of history files that you are using is stored as part of your ISPF profile information. To retrieve the information of your last-accessed history files:

1. In the FA Artifacts view, right-click on **FA Artifacts** and select **Retrieve Last Access History Files**.
2. When the Retrieve ISPF Profile dialog displays, use the arrow for the drop-down menu to select the following information for these fields (also see 3-7):
 - Existing RSE Connection: PTHAPC1 ISPF Profile Name: USERID().APC1ISPF.ISPPROF
 - Temporary DSN: USERID().FA.COPY.ISPF.ISPPROF
 - Exec Name: IDIC1.V7R1MO.SIDIEEXEC(IDICSXDS)
 - FA Application ID: IDA

See Figure 2-6 for an example of information retrieval.

Figure 2-6 Retrieve ISPF Profile pop-up

Table 2-5 describes these fields.

Table 2-5 Retrieve ISPF Profile field descriptions

Field Names	Description
Existing RSE Connection	Name of the RSE (remote system explorer) connection profile. The ISPF profile information is retrieved from the specified host.
ISPF Profile Name	Name of a data set that contains your ISPF profile.
Temporary DSN	Name of temporary data set that is used to copy your ISPF profile. You must have permission to allocate and write to the data set.
Exec Name	Data set name that contains REXX™ (Restructured Extended Executor, or exec) to retrieve your ISPF profile information. The default location is IDI.SIDIEEXEC(IDICSXDS). Some customization is required after the host installation (refer to the exec for further information).
FA Application ID	Application ID of Fault Analyzer installation on the host. The default value is IDI (for initial domain identifier).

Table 2-6 provides a summary of all actions available in the FA Artifacts view.

Table 2-6 FA Artifacts View actions

Action Name	Description
Add new history file	Enables you to add a history file from a particular host to the view.
Retrieve last accessed history file	Enables you to retrieve the set of history files you last used on the host, based on the information stored in your ISPF profile.
Retrieve view information	Enables you to retrieve information about views, which are stored in a data set.
Delete from view	Deletes the selected item from the view.
Populate/refresh history file	Refreshes the cached information about the selected history file.
Configure history file monitor	A history file monitor can be configured to monitor the changes made to a history file on the host. Whenever changes are made to the selected history file, the user is notified.
Set encoding	Enables you to specify the encoding for the selected system. That is, the language option used when the fault entry was created.

2.2.2 Detailed view

The Detailed view (Figure 2-7) displays the summary of fault entries contained in the selected history file or view in the FA Artifacts view. This view gives you a quick summary of what is happening on your system (history file-centric view of the system). The column headings are configurable, depending on your preference and area of your interest.

The screenshot shows a software interface with a 'System Name' field set to 'PTHFAE1' and a 'Fault History File or View' field set to 'DA.DCAT'. Below these fields is a table with the following columns: Fault_ID, Job/Tran, User_ID, Sys/Job, Abend, I_Abend, and Job_ID. The table contains 20 rows of fault data.

Fault_ID	Job/Tran	User_ID	Sys/Job	Abend	I_Abend	Job_ID
BAT06033	CS65	ZFAYDI	CSCB0650	SNAP	SNAP	JOB02426
BAT06032	CS65	ZFAYDI	CSCB0650	SNAP	SNAP	JOB02426
BAT06030	JBATCH2	PSTRAND	FAE1	SOC9	SOC9	STC21519
BAT06031	JBATCH2	PSTRAND	FAE1	SOC9	SOC9	STC21519
BAT06029	SSHD5	KENICHI	FAE1	U4038	U4038	STC21529
BAT06028	SSHD9	KENICHI	FAE1	SNAP	SNAP	STC21519
BAT06027	SSHD6	KENICHI	FAE1	SNAP	SNAP	STC21520
BAT06026	SSHD6	KENICHI	FAE1	SNAP	SNAP	STC21520
BAT06025		RTURNER	FAE1	n/a	n/a	TSU21505
BAT06024	IDCB0740	ANDYMEL	FAE2	S522	S522	JOB03662
BAT06023	SDIVPD32	ANDYMEL	FAE2	SOC4	U4039	JOB02938
BAT06022	SDIVPD12	ANDYMEL	FAE2	SOC4	U4039	JOB02934
BAT06021	SDIVPD2	ANDYMEL	FAE2	SOC4	U4039	JOB02932
BAT06020	SICB0110	ANDYMEL	FAE2	SOC9	U4039	JOB02930
BAT06019	SICB0100	ANDYMEL	FAE2	SOC9	U4039	JOB02928

Figure 2-7 Detailed view of fault entry summaries

From this view, you can invoke the Report browser (see 2.2.4, “Fault Analyzer Report browser”). This will enable you to see the detailed analysis of the selected fault entry or to invoke the Dump browser (see 2.2.5, “Fault Analyzer Dump browser”) to browse a hex dump display of mini-dump pages stored in the selected fault entry.

Table 2-7 shows a summary of all actions available from this view.

Table 2-7 FA Report browser action

Action Name	Description
Browse report	Retrieves the report associated with the selected fault entry and displays the report. The report is cached locally after it is retrieved.
Browse mini-dump	Retrieves the mini-dump pages associated with the selected fault entry and displays the pages. When the pages associated with a fault entry are retrieved, they are cached locally.
Delete cached data	Deletes the cached information for the selected fault entry.
Delete from view	Deletes the selected item from the view.
Set encoding	Allows you to select an encoding for the system where the selected history file or view is located.

2.2.3 Working with older fault entries

For a Fault Analyzer Integration client to show fault entries in a history file correctly, the fault entries must be created with the correct version of the Fault Analyzer feature, with the appropriate set of options specified. In these cases, a fault entry cannot be displayed correctly by using the Fault Analyzer Integration client:

- ▶ A fault entry was created by an older version of the Fault Analyzer.
- ▶ A fault entry was created without the WdZClient option.
- ▶ A fault entry was created by using the DeferredReport option.

In such cases, the Fault Analyzer Integration client detects the invalidity of the selected fault entry and displays the Fault Entry Refresh dialog (Figure 2-8), so that you can refresh the contents of the fault entry by rerunning the analysis with the correct set of options.

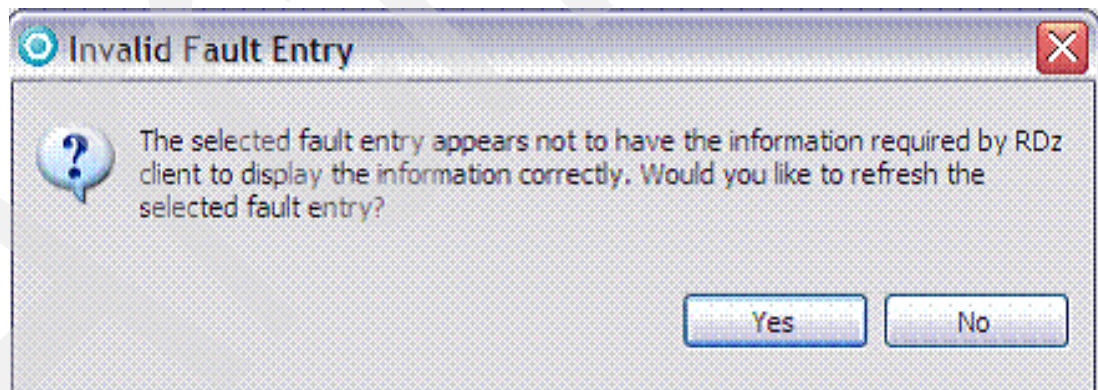


Figure 2-8 Invalid Fault Entry pop-up

2.2.4 Fault Analyzer Report browser

The Fault Analyzer Report browser (Figure 2-9) is a multi-tabbed browser that displays the report associated with the selected fault entry. The browser is typically invoked by running the Browse Report action in the detailed view.

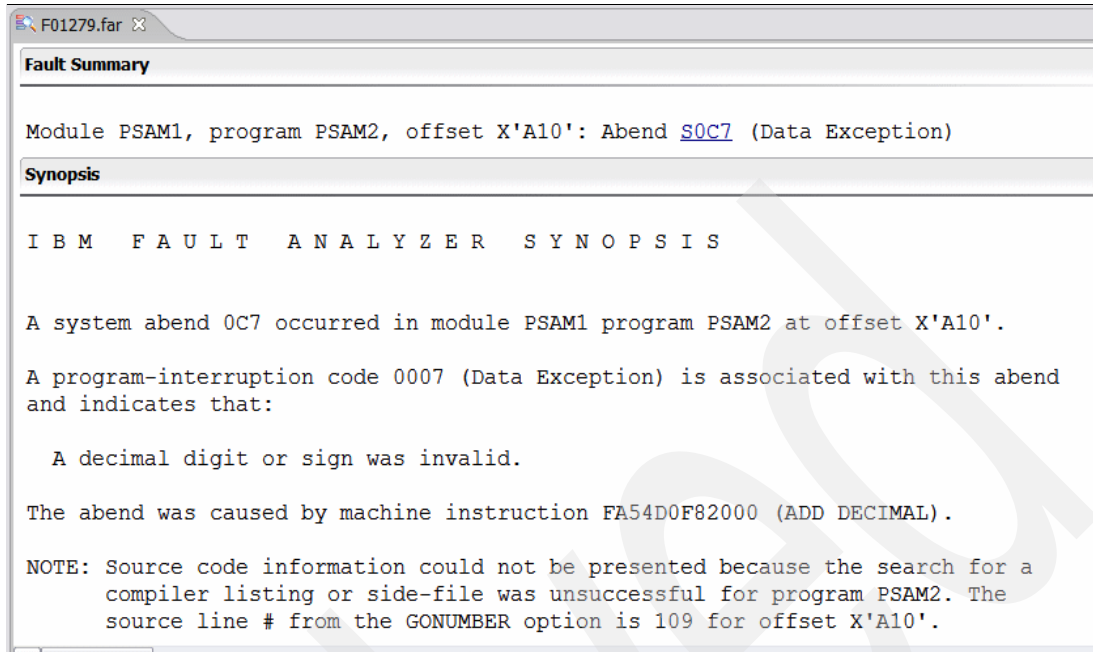


Figure 2-9 FA synopsis of aBEND

Different parts of a report are organized by using tabs for categories that show along the bottom of the screen. The tabs separate the report into these five categories and related subcategories:

► **Main Report:**

- **Fault summary section:** A brief description of the fault, which includes the hyperlinked source line number of the abending program (if possible) and the abend code. If you click on the source line number, it displays the source code of the abending program where the failure occurred (see 2.2.6, “Source code display” for further information). If you click on the abend code, it displays the description (see 2.2.7, “Lookup view”).
- **Synopsis section:** The synopsis section provides a brief description of the fault and its analysis.

► **Event Summary:**

- The Event Summary is a list of all events, in chronological order.
- The **Event details** subsection provides detailed information about each event. Additional information associated with the event, such as message description and the contents of the program's working storage, are included in this subsection. The source code information or failing machine instruction is also included here.

► **Abend Job Information:**

- This section provides information about the abending job associated with the real-time invocation of the Fault Analyzer feature, such as:
 - Abend date
 - Time
 - Job ID
 - Job name
 - Job step name
 - Execution environment
 - Language Environment® runtime options

► **System-wide Information:**

- This section contains, for example, console messages that are not identified as belonging to any specific event or CICS system-related information, such as trace data and 3270 screen buffer contents. Information about open files that could not be associated with any specific event might also be included here.
- If there is no information in this section, then it does not appear in the report.

► **Miscellaneous Information:**

- **Fault Analyzer options:** Lists of the Fault Analyzer options that were in effect at the time of the analysis.
- **Prolog section:** Information about the version, release, and modification level of the Fault Analyzer, as well as the latest APAR (authorized program analysis report) or PTF (program temporary fix) installed.
- **Epilog section:** Provides information about the invocation exists used and the approximate amount of above-the-line storage allocated during the analysis, followed by the fault ID assigned. It also includes the time and date when the report was created.

See the *Fault Analyzer User's Guide and Reference*, SC19-1253 for a comprehensive description.

2.2.5 Fault Analyzer Dump browser

Fault Analyzer Dump browser (Figure 2-10) is a multi-tabbed browser that enables you to browse the mini-dump pages stored for the selected fault entry in a history file. This browser is typically opened by selecting the **Open Mini-Dump** action against a fault entry in the summary view. **Mini-dump pages** are the virtual storage pages that are accessed during analysis and stored as part of a fault entry.

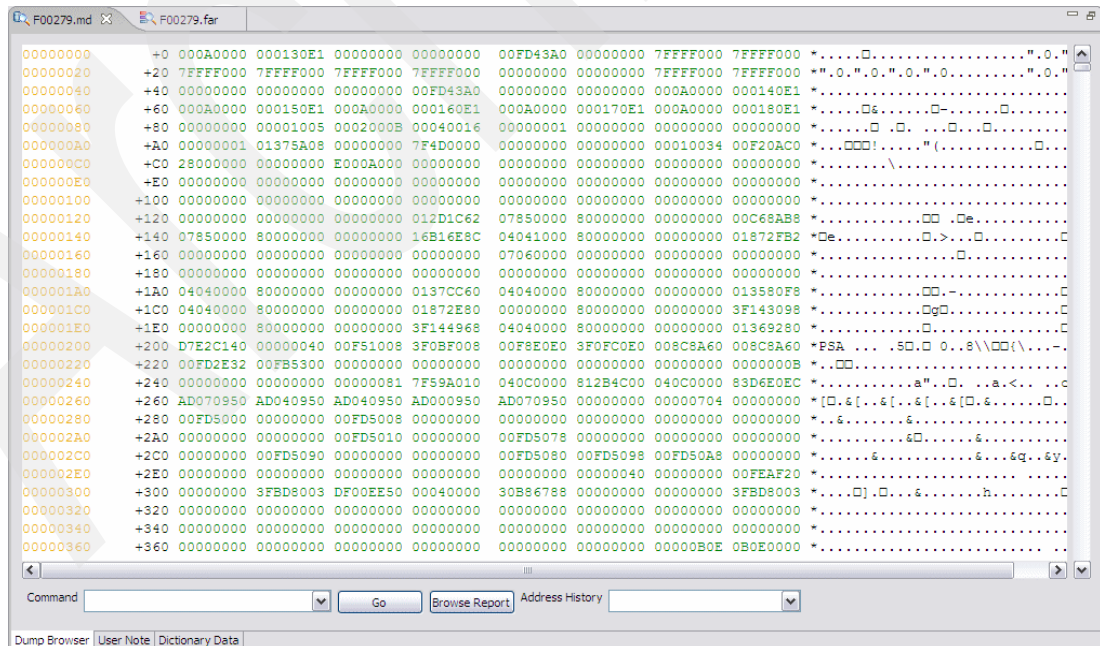


Figure 2-10 Mini-dump

Mini-dump pages are annotated with the key information that the Fault Analyzer found during the analysis (dictionary data). This dictionary data includes information such as various control blocks, modules, Multiple Virtual Storage (MVS) areas, and register values. A summary of the dictionary data associated with the selected fault entry is presented in the Outline view (Figure 2-11) or in the Dictionary Data tab within the Dump browser.

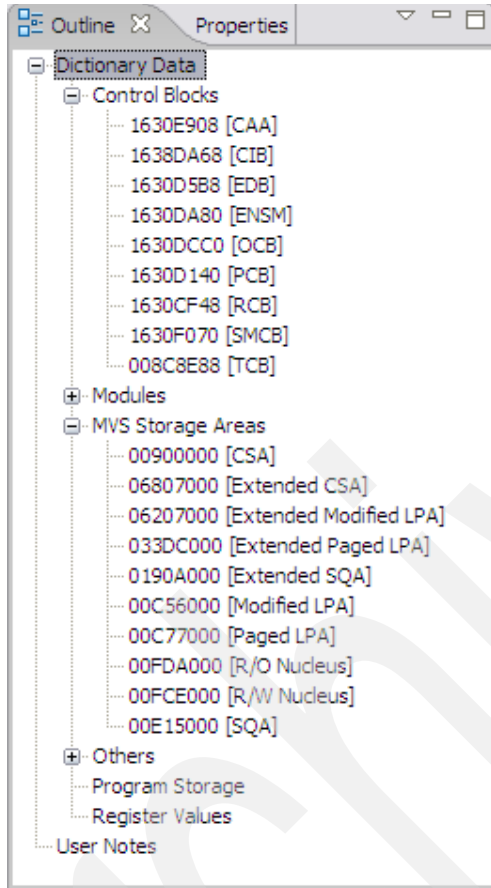


Figure 2-11 Mini-dump Outline view

2.2.6 Source code display

Fault Analyzer supports several different side-file formats to display the associated source information of an abending program. One of the supported formats is called the LANGX file, because it is produced by running IDILANGX utility program against a listing file produced by a compiler.

The Fault Analyzer Integration option supports the source line display of the abending program if the correct LANGX file was available during the analysis (see Figure 2-12). If the correct LANGX file was available during the analysis of your abending program, the source line number of the program is hyperlinked in the report. (The location of your LANGX file is specified in the IDILANGX DD card in your job step.) When you click the source line number (Line 17 in the example in Figure 2-12): PROCEDURE DIVISION), it retrieves the LANGX file from the host and generates the source line for display. The source file opens in your favorite editor in read-only mode, and the line where the abend occurred is highlighted.

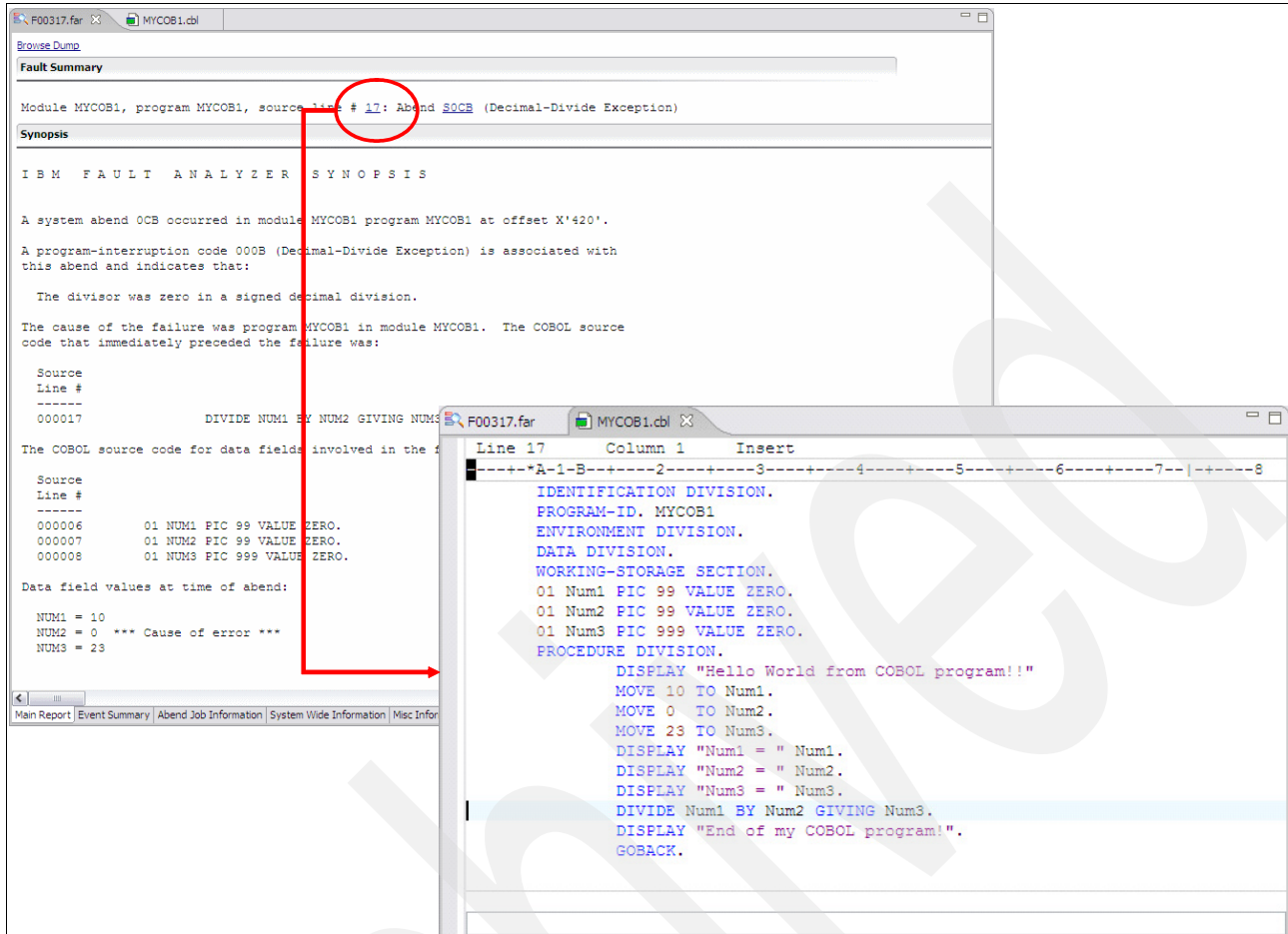


Figure 2-12 FA source code display

The location of LANGX files is specified in the IDILANGX DD statement in your job. Alternatively, you are given an opportunity to specify the data set names containing your LANGX files during the fault entry refresh process (Figure 2-13).

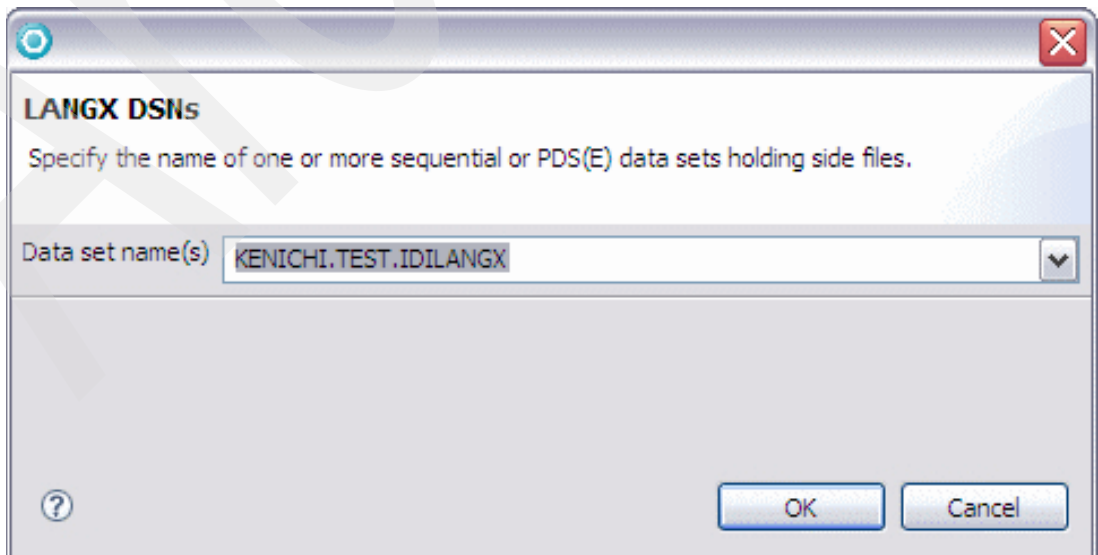


Figure 2-13 LANGX DSNs pop-up

2.2.7 Lookup view

In the Lookup view, you can browse to the description about abend codes, messages, and other miscellaneous information. Figure 2-14 shows the description of IMS user abend U0002. You can browse the information in one of three ways:

1. Find the description for a particular abend code, message, or miscellaneous information by navigating through the tree structure.
2. Type in a pattern name in the Search box. All matching abend codes, messages, and miscellaneous information are displayed in the Results tab in the view. From the list of matching results, you can browse to find the description.
3. An abend code associated with a particular fault entry is hyperlinked in the report view. Click the hyperlinked abend code to display the description.

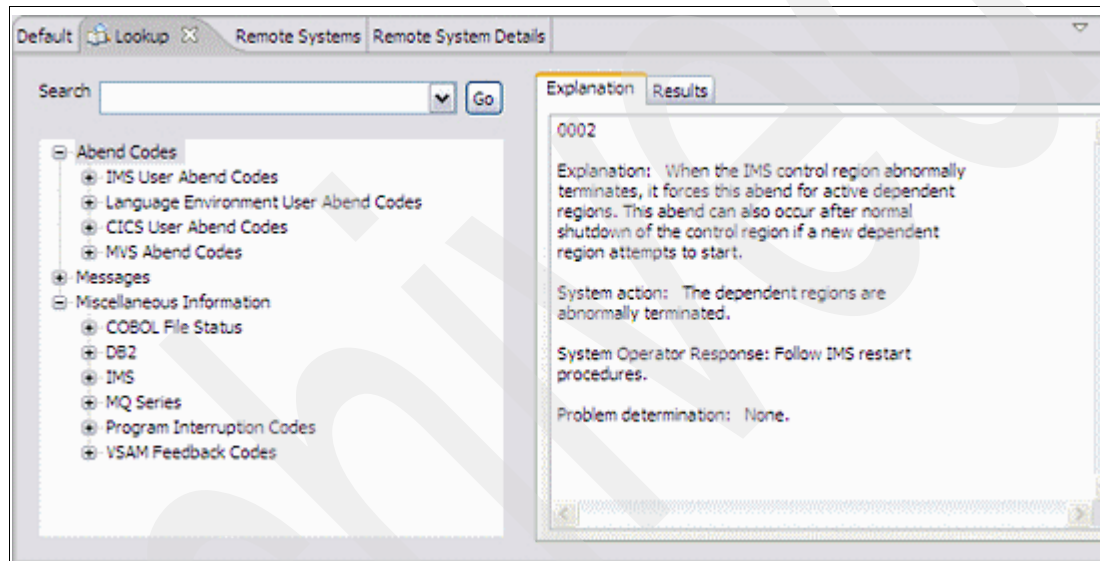


Figure 2-14 Lookup view

You can switch between English or Japanese descriptions for abend codes, messages, and miscellaneous information.

1. Select the **Language Setting** action from the drop-down menu within the view.
2. When the Lookup Language Setting dialog displays, choose your preferred language.

2.3 File Manager Integration for Rational Developer for System z

IBM File Manager for z/OS is a member of the IBM Problem Determination Tools suite, which provides comprehensive tools for working with OS/390® data sets, DB2 data, IMS data, and CICS data. These tools include the familiar browse, edit, copy, and print utilities found in ISPF, enhanced to meet the requirements of application developers.

The IBM Rational Developer for System z (RDz) version 7.1 File Manager feature integrates with IBM File Manager for z/OS (FM) to provide application developers with the ability to browse and edit sequential data sets, partitioned data set members, and VSAM data sets of type KSDS and ESDS in a workstation RDz environment. It can show records formatted according to field layouts typically provided by COBOL COPYBOOKs or PL/I include files.

In this section, we demonstrate the features present in the RDz File Manager to edit a data set. We explain the steps required to configure RDz File Manager so that it can communicate with IBM File Manager. Then we provide an example of using the RDz File Manager Formatted Data Editor to edit a key-sequenced data set (KSDS).

2.3.1 Editing data set records using RDz File Manager

In this section, we demonstrate the basic flow of how to browse and edit records stored in a data set using RDz File Manager. In particular, we demonstrate how to specify the record format of the data stored in a KSDS data set using a COBOL COPYBOOK. The basic steps we describe in this section should be the same in principal for other types of data sets such as VSAM files and when specifying the record format of the data using a PL/I include or IBM File Manager for z/OS template instead of a COBOL COPYBOOK.

Create a data set containing sample data

In this step, we create a key-sequenced data set containing the name and address information of various people living in Australia. We submitted the existing JCL in KENICHI.TEST.JCL(BLDVSAM) to create the new dataset. Figure 2-15 shows the BLDVSAM member being edited in the RDz LPEX editor. Notice that the key is defined to start at position 0 and have length 6 (KEYS(6 0)).

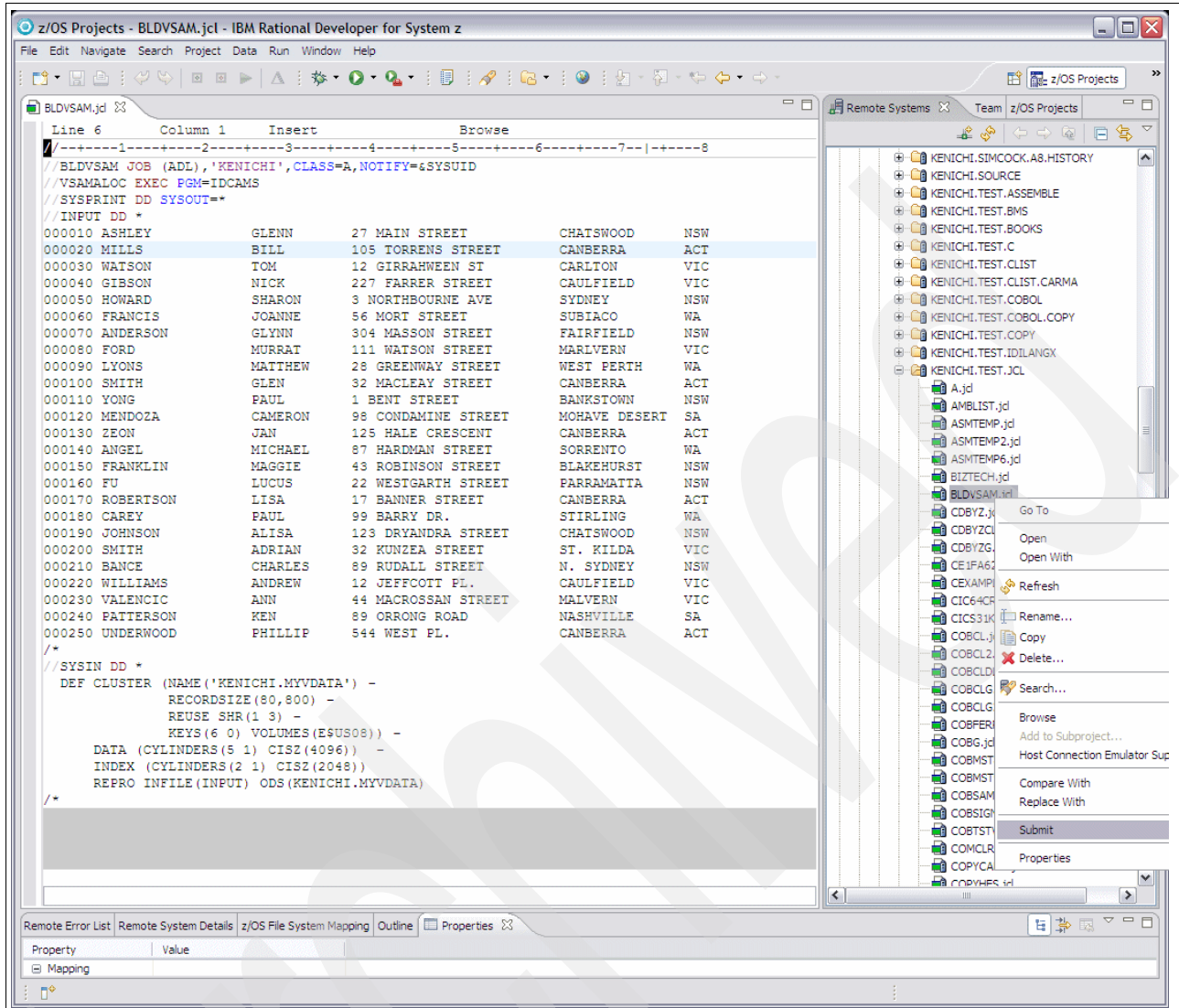


Figure 2-15 BDLMS member creation

Copybook to describe the record format

In this step, we create a COBOL COPYBOOK that describes the format of records stored in the sample data set. The sample COBOL COPYBOOK is shown in Figure 2-16 in the RDz System z LPEX Editor. The sample COPYBOOK is a member of the partitioned data set called 'KENICHI.TEST.COBOLO.COPY' named 'MYDATA'. Note that all of the fields are defined as alphanumeric (PIC X) except for the first one, which is defined as being zoned decimal (PIC 9). The first field also happens to be the key field and has length 6 to match the KSDS we created.

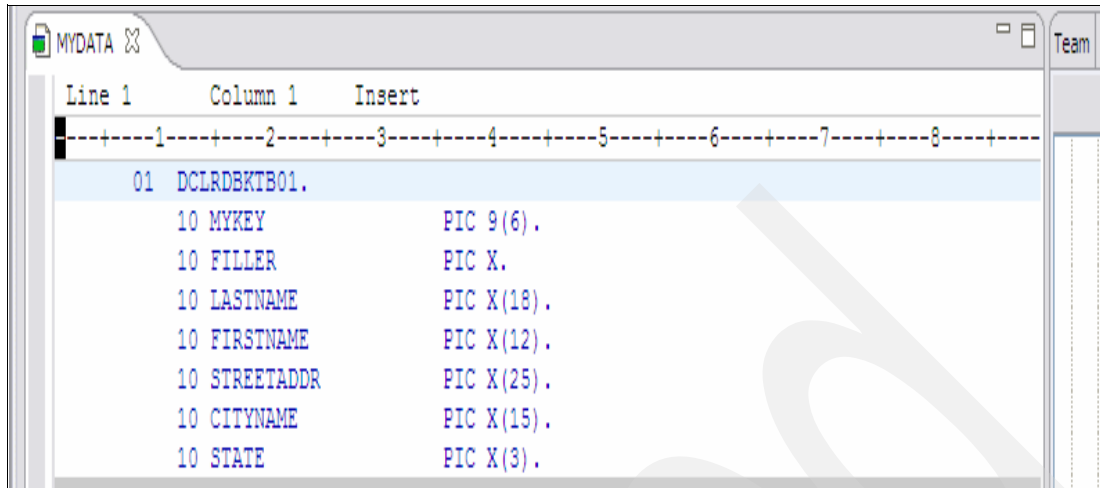


Figure 2-16 LPEX Editor

Specify z/OS file mapping for the sample data set

In the Remote Systems view under MVS files, a list of data sets are listed with different icons to indicate different types of data sets and their associated editor when you choose to browse a data set. The association of a data set name and its preferred editor are specified in the z/OS File System Mapping view (Figure 2-17).

By default, mainframe datasets matching the pattern ****DATA** will be given the **tdat** extension when being displayed in the Remote Systems view. If you browse the file associations for **tdat** files (**Window** → **Preferences** → **General** → **Editors** → **File Association**), they have Formatted Data Editor by default. The data within these files will be displayed with the RD/z File Manager Formatted Data Editor by default. In addition to editing **tdat** files, the Formatted Data Editor will also be used to edit VSAM clusters by default whether or not they have the **tdat** extension, because this is the only RDz editor currently capable of editing VSAM cluster data.

If the default mappings do not match the naming conventions of your data sets, custom mappings can be added as well. It is not necessary in this case because the data set we have created is a VSAM cluster and it matches the ****DATA** pattern, with both of these conditions causing the data set to be opened in the Formatted Data Editor, but we will add a mapping as an example.

In the z/OS File System Mapping view, right-click and select **Add Data Set Mapping**. Enter **'KENICHI.MYVDATA'** as the mapping criterion and **'tdat'** in the *Other* text box under Workstation File Extension. Notice that the sample data set ('KENICHI.MYVDATA') now has **'tdat'** as its preferred workstation file extension. If there are a few data sets that still do not have the correct workstation file extension, the file extension can be forced by editing the *Extension* field in the *Mapping* screen of the data set's properties dialog.

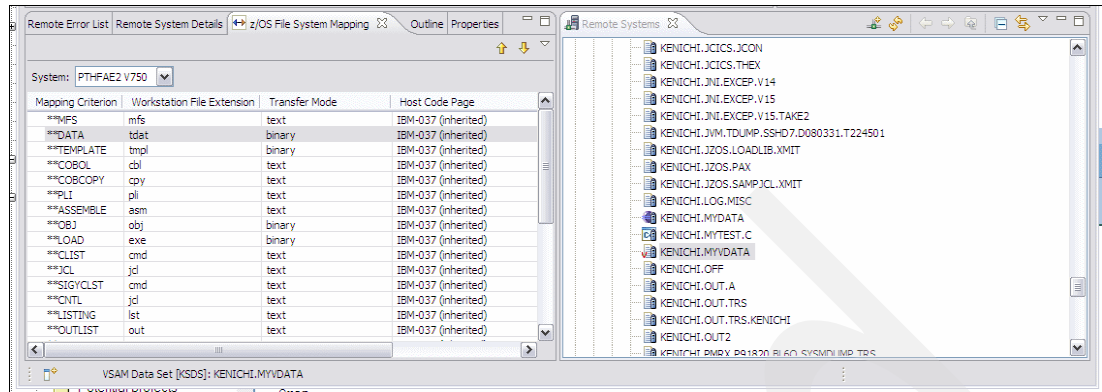


Figure 2-17 Data Set Mapping view

Specify the data set's record layout

The Formatted Data Editor has to know the format of records stored in a data set in order to display the contents in a meaningful way for application developers. For this example, we use the COBOL COPYBOOK created in the previous step to describe the data. Right-click on the KSDS containing the sample data ('KENICHI.MYVDATA'). From the context menu, select Properties. This brings up the Properties dialog for the selected data set in Figure 2-18. Select **Template Association**. In this view, you can either specify the full name of the template to use for the selected data set or you can browse through the data sets available in your Remote Systems filter for the current target system. In this case, we simply browsed and selected 'KENICHI.TEST.COBOL.COPY(MYDATA)'.

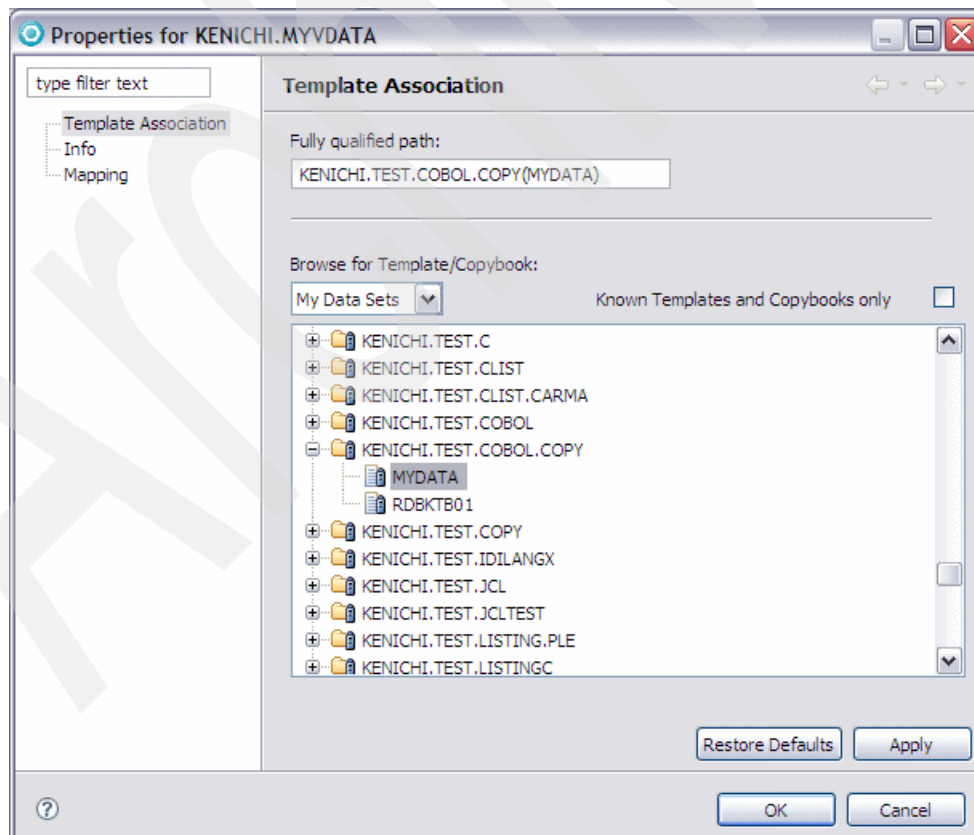


Figure 2-18 Data set layout selection

In addition to using COBOL COPYBOOKS to specifying the record layout, you can also use PL/I include files and IBM File Manager templates. File Manager templates are derived from COBOL COPYBOOKS, and PL/I includes them by compiling them under ISPF. They can contain additional information such as identification and selection criteria. While browsing for templates in the Template Association screen (Figure 2-19), the **Known Templates and Copybooks only** option can be selected to filter out all data sets that do not currently have the workstation file extensions for COBOL COPYBOOKS, PL/I includes, or IBM File Manager templates. In the RDz File Manager user interface and documentation, *template* is typically used as a generic term to refer to all three of these dataset types. Refer to the *IBM File Manager User's Guide* for further information about templates.

2.3.2 Editing the sample data set in the Formatted Data Editor

In this section we describe how to edit a data set using the Formatted Data Editor.

Opening the data set in the Formatted Data Editor

If we right-click on the sample data set ('KENICHI.MYVDATA'). Select **Open With** → **Formatted Data Editor**. A progress dialog will appear while an IBM File Manager for z/OS edit session is being started for the requested data set. After the IBM File Manager for z/OS edit session has begun and the Formatted Data Editor has retrieved the initial data for the edit session, the Formatted Data Editor will appear (Figure 2-19).

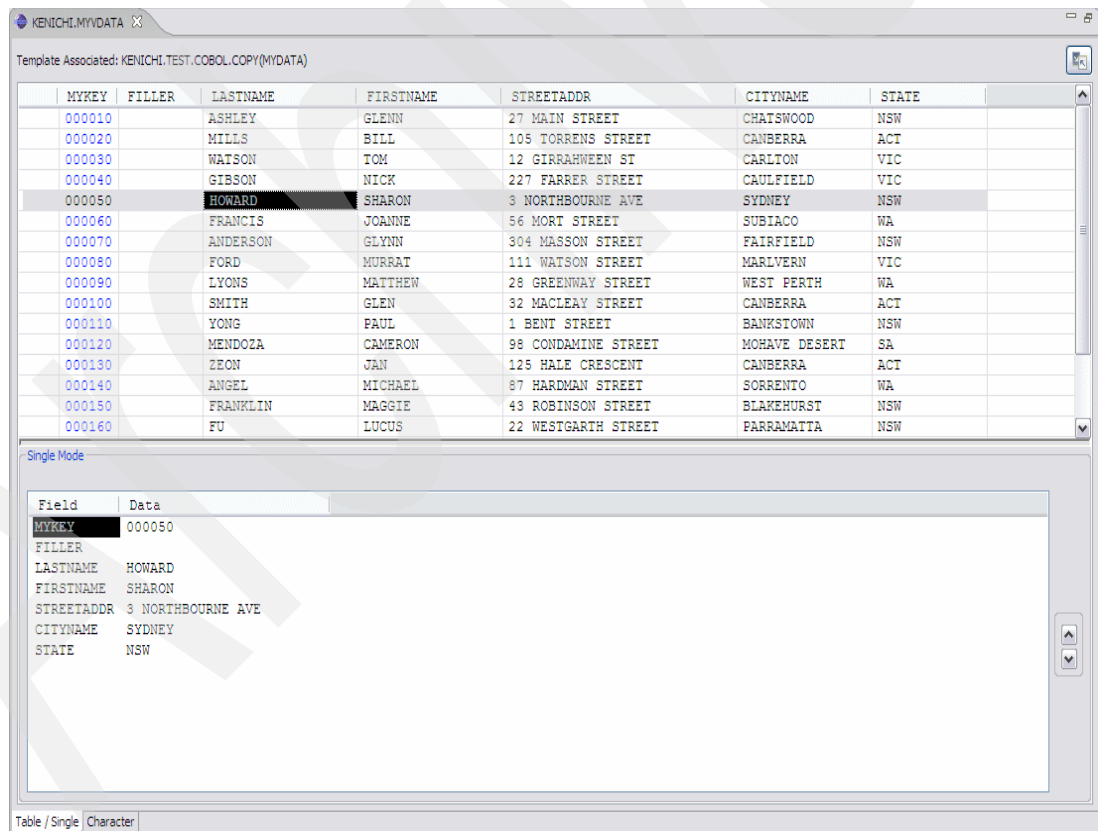


Figure 2-19 Data Editor

Table and Single Modes

Notice that, in the default Formatted Data Editor interface:

- ▶ The template that is providing the record layout for the session is listed near the top after *Template Associated:* .
- ▶ A tabular view of the data is provided below the name of the template that has been associated. This is called the Table Mode:
 - Each row in the table represents one record in the data set.
 - The fields within each record are formatted correctly in columns with the appropriate headings for each column being based on the information specified in the COBOL COPYBOOK.
 - As defined when creating the KSDS, the first field contains the KSDS key. To better highlight them, key fields are displayed in blue instead of black.
 - The first field has also been displayed in a simplified numeric format. For example, the first field actually contains 000100, but this has been simplified to 100 for display, because this is a zoned decimal field.
- ▶ A single record's data is displayed at a time in the lower part of the editor. This is called the Single Mode.
 - All modes are synchronized. Changes made in one mode will be immediately reflected in all others.
 - Single mode can be removed from view by clicking on the button that is circled in Figure 6. Its size with respect to the table mode can also be changed by dragging the divider between them up or down.
 - The up and down buttons on the right side of the single mode will move the selected record up or down. The change in selection will also be displayed in the table mode.

Character Mode

At the bottom-left of the editor, there are two tabs. By default, **Table/Single** is selected. By clicking on the tab marked **Character**, the Character Mode can be enabled. When in Character Mode (Figure 2-20), the records are treated as if they contain exclusively alphanumeric data.

The record layout described in the template is not applied to the records. If data in the records is not alphanumeric and cannot be displayed as such, a box character will appear to represent the non-displayable character. If a data set is opened without a template being associated with it, the table mode will display data in the same way as character mode does for all data sets. Because this mode is less useful for basic editing operations, we will return to the Table/Single mode tab for the rest of the example.

Associated: KENICHI.TEST.COBOL.COPY(MYDATA)					
000010	ASHLEY	GLENN	27 MAIN STREET	CHATSWOOD	NSW
000020	MILLS	BILL	105 TORRENS STREET	CANBERRA	ACT
000030	WATSON	TOM	12 GIRRAHWEEN ST	CARLTON	VIC
000040	GIBSON	NICK	227 FARRER STREET	CAULFIELD	VIC
000050	HOWARD	SHARON	3 NORTHBOURNE AVE	SYDNEY	NSW
000060	FRANCIS	JOANNE	56 MORT STREET	SUBIACO	WA
000070	ANDERSON	GLYNN	304 MASSON STREET	FAIRFIELD	NSW
000080	FORD	MURRAT	111 WATSON STREET	MARLVERN	VIC
000090	LYONS	MATTHEW	28 GREENWAY STREET	WEST PERTH	WA
000100	SMITH	GLEN	32 MACLEAY STREET	CANBERRA	ACT
000110	YONG	PAUL	1 BENT STREET	BANKSTOWN	NSW
000120	MENDOZA	CAMERON	98 CONDAMINE STREET	MOHAVE DESERT	SA
000130	ZEON	JAN	125 HALE CRESCENT	CANBERRA	ACT
000140	ANGEL	MICHAEL	87 HARDMAN STREET	SORRENTO	WA
000150	FRANKLIN	MAGGIE	43 ROBINSON STREET	BLAKEHURST	NSW
000160	FU	LUCUS	22 WESTGARTH STREET	PARRAMATTA	NSW
000170	ROBERTSON	LISA	17 BANNER STREET	CANBERRA	ACT
000180	CAREY	PAUL	99 BARRY DR.	STIRLING	WA
000190	JOHNSON	ALISA	123 DRYANDRA STREET	CHATSWOOD	NSW
000200	SMITH	ADRIAN	32 KUNZEA STREET	ST. KILDA	VIC

Figure 2-20 Character Mode

Field metadata

In Table and Character modes, the record layout field metadata will be displayed when the mouse hovers over the column header. In Figure 2-21, we see that the first field is field #2 from the associated copybook, MYKEY, that it is a zoned decimal field (ZD), that it starts at position 1 in the dataset and is 6 bytes long, and that it contains KSDS key data. Likewise, if we were to hover over LASTNAME, we would see that it is field #4 from the copybook, that it is alphanumeric (AN), that it starts at position 8, and that it is 18 bytes long.

KENICHI.MYVDATA		
Template Associated: KENICHI.TEST.COBOL.COPY(MYDATA)		
MYKEY	FILLER	LASTNAME
000		ASHLEY
000		MILLS
000		WATSON
000040		GIBSON
000050		HOWARD
000060		FRANCIS
000070		ANDERSON
000080		FORD

Figure 2-21 Field metadata

Editing a field

To edit a field, simply select it with the mouse, and modify the contents. In the current example, changes to the first field must be numeric. If invalid data is entered for a given field type, the error will be displayed near the top of the editor as shown in Figure 2-22. If there is an error in the entered data, the change to the field cannot be saved. The field contents will be reverted to their previous value if Enter is pressed while there is an error in a field.

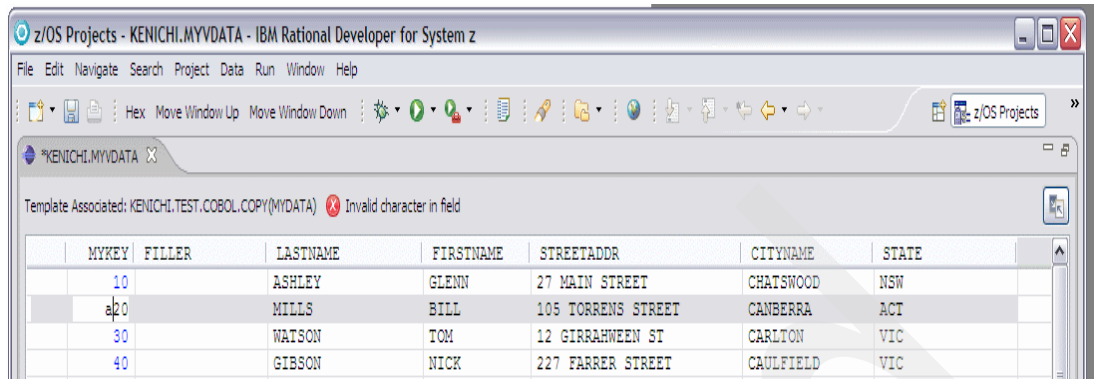


Figure 2-22 Data Error

If the new contents of the field are valid, the change will be allowed and maintained. For this example, we will now change record 2's MYKEY field from 20 to 25 (Figure 2-23). Note that the single mode has been updated to show the new values for the record as expected and that the name of the data set on the editor tab at the top left has an asterisk beside it. The asterisk denotes that the editor is "dirty." When the editor is dirty, there are pending changes that have been made to the data, but that have not been saved back to the data set itself. Until a save operation is requested (**File** → **Save**), the data set and the data in the current edit session are out of sync.

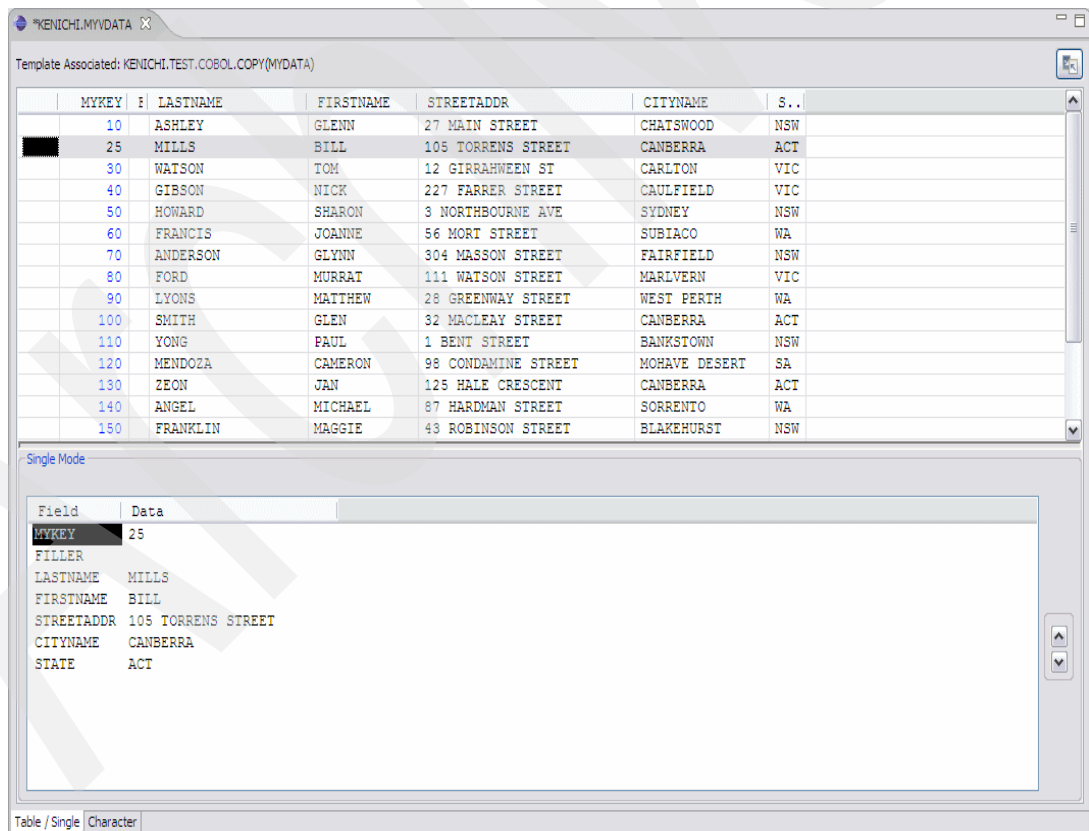


Figure 2-23 Unsaved Changes in Data Editor

Hex editing

The underlying hexadecimal data can be displayed and edited by clicking on the circled button in Figure 2-24. The hexadecimal representation of the data is displayed below each record such that the two byte nibbles are directly underneath the respective byte in the main record display. This is the same way that the hexadecimal would be displayed by default in IBM File Manager.

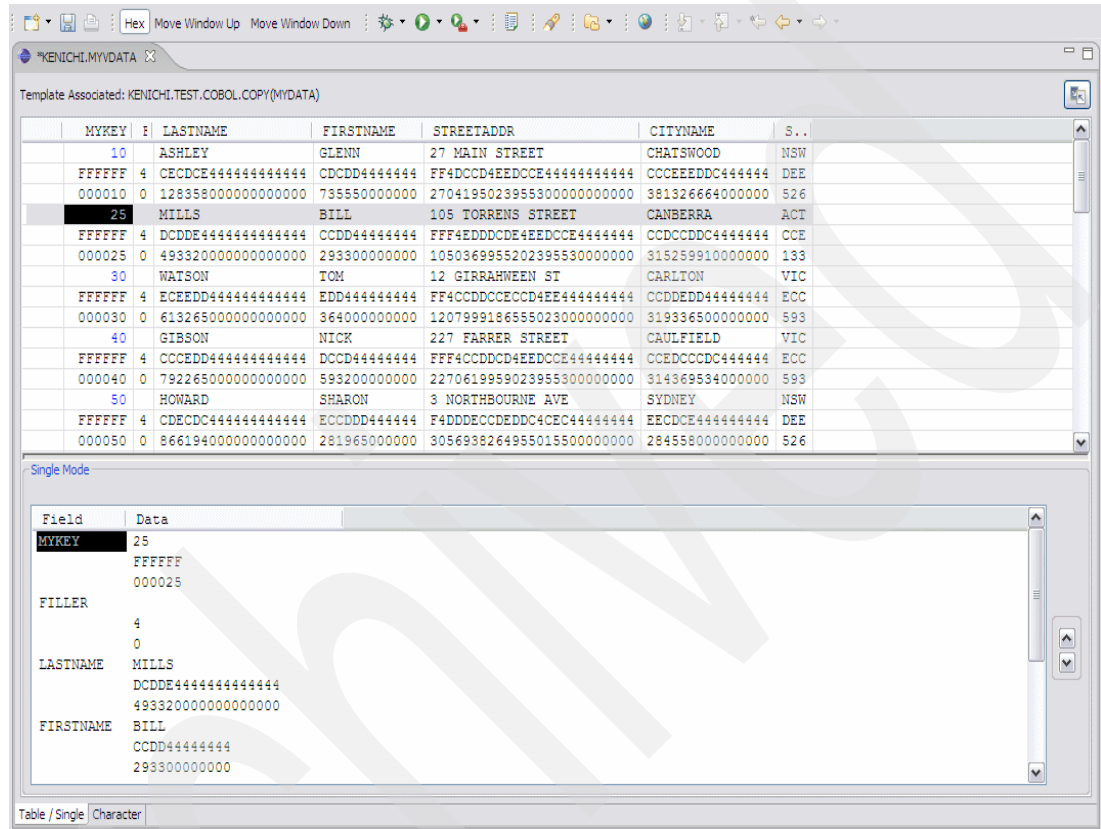


Figure 2-24 Hex Data Editor

The hexadecimal representation can be edited. Invalid values for field types (for example, an alphabetic character in our zoned decimal field) can even be forced into a field by editing the hexadecimal representation. If a field contains an invalid value for its specified type, its display representation becomes a series of asterisks as seen in Figure 2-25.

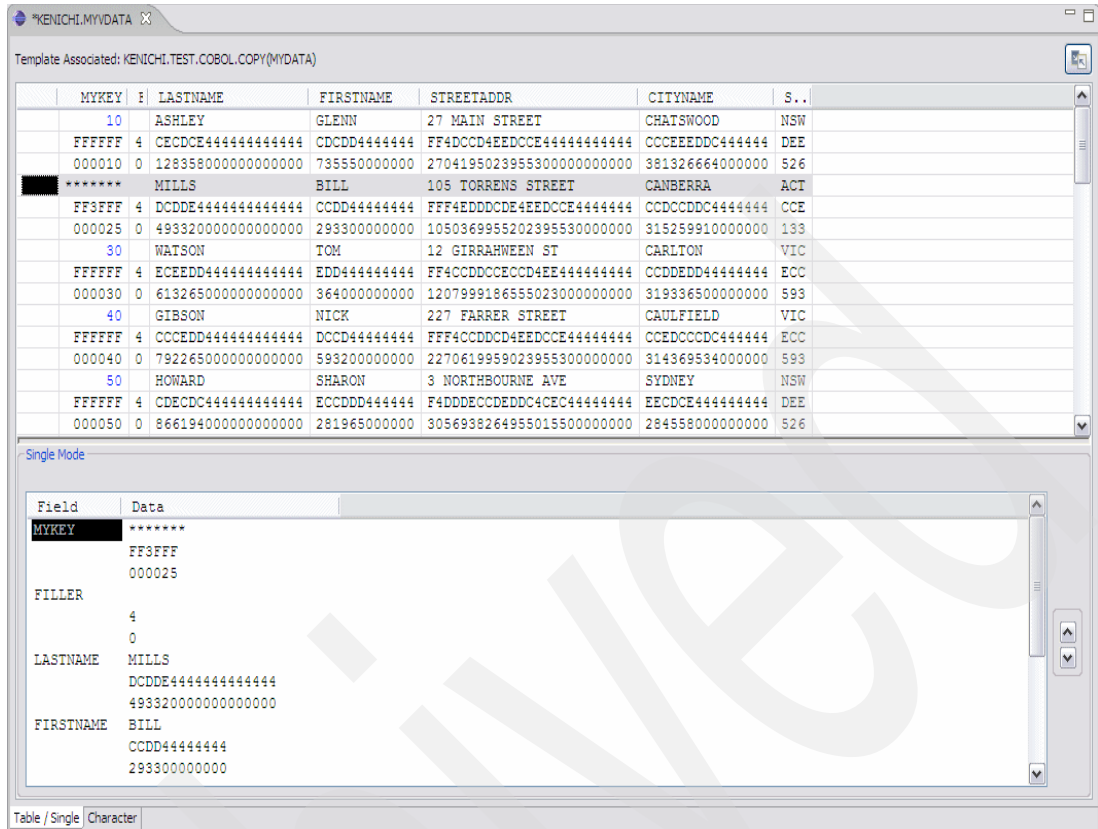


Figure 2-25 Invalid Type Entry

Very Large Data Set Support

Some data sets might be large enough that they will not fit in the workstation's available system memory. To allow editing of such large data sets, the Formatted Data Editor will pull a subset of the records in the target data set at a time. The records currently in the editor are known as the editor's window of the data set. The default window size is 200 records. Two buttons are provided at the top of the Formatted Data Editor that can be used to move the editor's window of the data set Figure 2-26. Moving the window up or down is also referred to as "stepping" through the data set. By default, the window step size is 50 records.

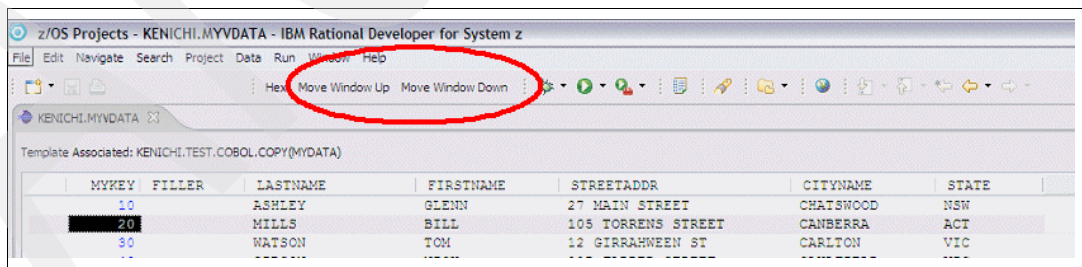


Figure 2-26 Move Window Up/Down in Formatted Data Editor

The current position in the dataset is shown at the bottom of the editor. The format of the display is:

Records <top window record number> - <bottom record number> (<total number of records>).

In Figure 2-27, records 21-40 out of 774 are currently being displayed.

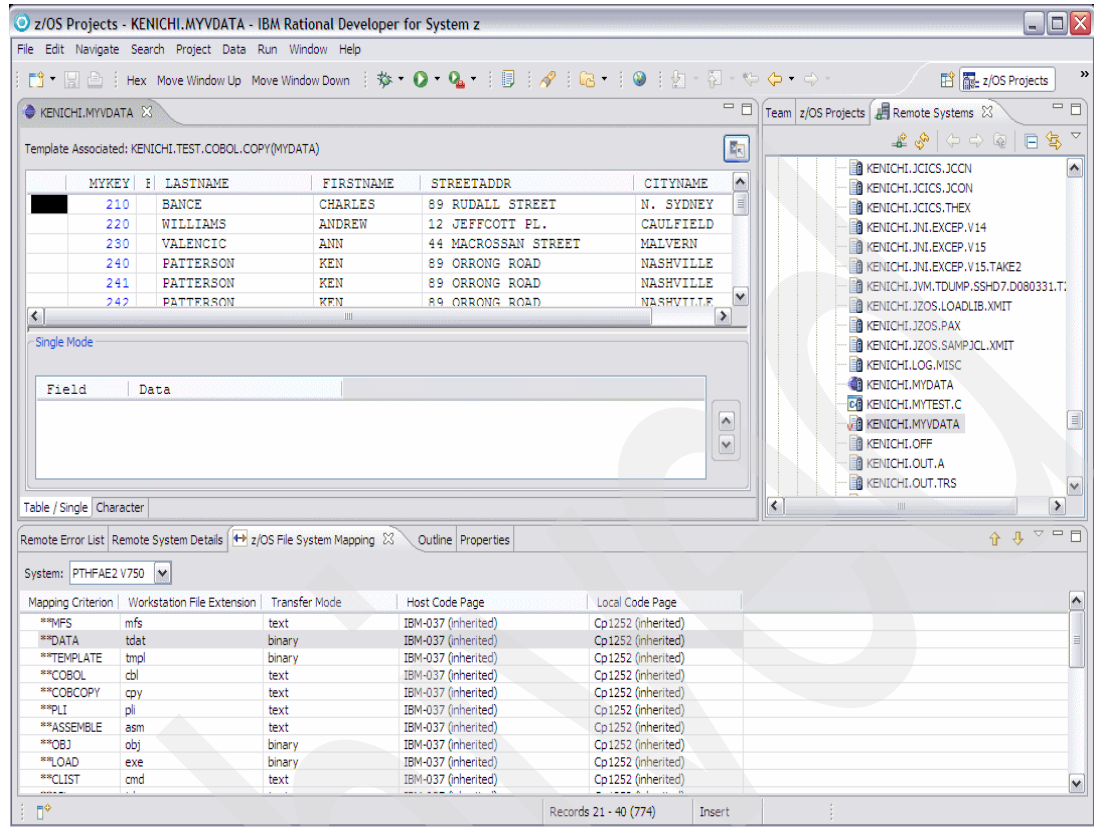


Figure 2-27 Display of data

2.4 Across platform utilization

The three Problem Determination Tools that are integrated with RDz compliment each other by providing key ingredients to coordinating a problem solution effort. After a job is run and an abend has been detected, the Fault Analyzer provides key information about the abend in the “Main Report” of the Fault Summary. The Fault Analyzer synopsis gives an explanation of what happened to cause the abend (Figure 2-28).

```

Module SAM2, CSECT SAM2, offset X'39E': Abend S0C7 (Data Exception)
Synopsis
-----
I B M   F A U L T   A N A L Y Z E R   S Y N O P S I S

A system abend 0C7 occurred in module SAM2 CSECT SAM2 at offset X'39E'.

A program-interruption code 0007 (Data Exception) is associated with this abend
and indicates that:

    A decimal digit or sign was invalid.

The abend was caused by machine instruction FA442005301E (ADD DECIMAL) .

```

Figure 2-28 FA Synopsis

This information can be used in the Debug perspective to locate the variable that is causing issues. Set up breakpoints and step through the program in the debug perspective. When you reach the area that is causing problems, Debug will inform you that an event occurred (Figure 2-29).

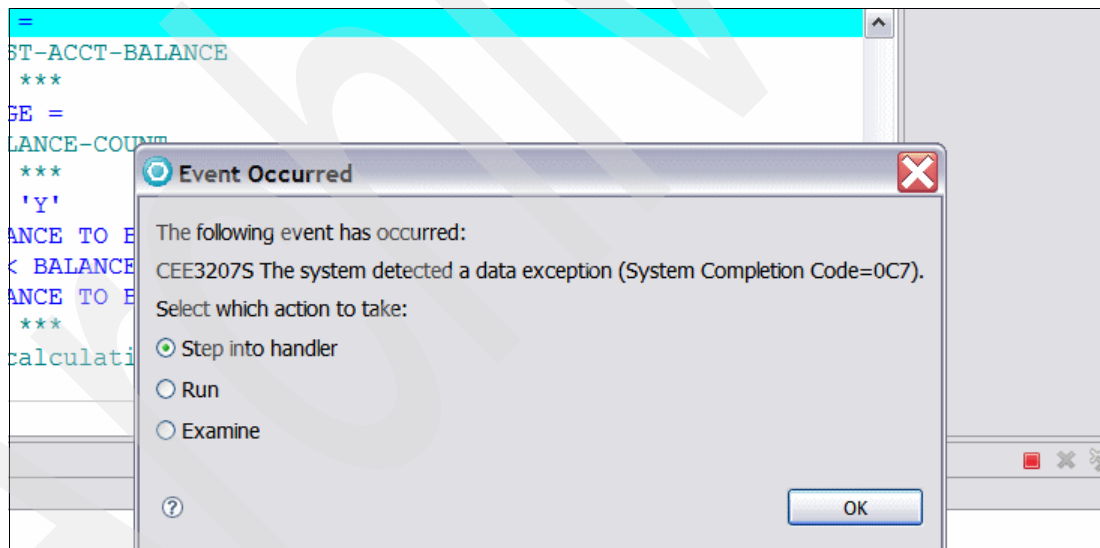


Figure 2-29 Event Notification

When the variable name is found, we can begin looking at the data, by going back to the z/OS perspective and selecting the dataset that holds the data. You can use File Manager to take a look at the data. After applying the correct copybook, we can open the dataset in File Manager. Double-clicking on the tab for the editor of the file, we can see our file in full screen (see picture below). This allows us to see a lot of data. For those familiar with the green screen version of File Manager, you will notice that we can see both the Table Mode and Single Mode on the same screen. Selecting any record in the table section activates the Single record section and synchronizes the record. It is easy to see that the record containing the name of Turner, Paige has bad data. This is denoted by the red asterisks in the account balance column. To fix the data, simply highlight the column and over-type the asterisks with a valid number (Figure 2-30).

Template Associated: DDS0200.ADLAB.COPYLIB(CUSTMAST)							
C...	F	NAME	ACCT-BA...	ORDERS...	ADDR	CITY	S. CC
03115	A	Graham, Holly	254.53	1	3100 Oaktree Ct	Raleigh	NC US
05580	A	Moore, Adeline	498.95	3	4700 S. Syracuse	Denver	CO US
06075	A	Dubree, Dustin	192.98	1	9229 Delegate's Row	Indianapolis	IN US
06927	A	Buchs, Jillian	99.99	0	41 Avendale Drive	Carlisle	PA US
07008	A	Houston, Roger	296.97	1	4111 Northside PkWay	Atlanta	GA US
07025	A	Marx, Audrey	450.51	2	90 South Cascade	Boulder	CO US
11204	A	Ness, Luke	513.06	3	5166 Sprinkle Road	Portage	MI US
11544	A	Graf, Polly	244.42	1	1551 S. Washington	Dowagiac	MI US
12689	A	Boyd, Luke A.	86.88	0	221 Yale Rd	South Perth	Au
13062	A	Turner, Paige	*****	4	3039 Cornwallis Road	Boone	NC US
13295	A	Well, Alice	731.26	4	125 Nicholson Rd	North Perth	Au

Figure 2-30 Data Error

2.5 APA Integration with OMEGAMON

In an increasingly complex and competitive business environment, IT departments are under pressure to operate systems at peak levels. The increasing complexity of business critical applications heightens the risk of missing service level commitments. IBM Application Performance Analyzer for z/OS provides a solution that can mitigate these risks with broad and deep application performance measurement information finely tune applications.

Analyzing and tuning applications for maximum performance can give IT technicians a necessary edge in maintaining service level commitments and customer satisfaction. Many organizations are turning to application performance measurement software to help fine-tune applications running on IBM System z processors for greater resource use and efficiency.

A performance analyzer for application programs is a tool that shows resources used by an application so you can identify the cause of performance problems. This tool delivers information about an application, answering; what programs are running, what percentage of time is spent in each program, which lines of code are using the most time, what files and databases are being used, and why is the application waiting. Application Performance Analyzer (APA) is geared to the application developer who requires a powerful and flexible tool to get to the source — and source code — of an application delay. It helps you evaluate application prototypes prior to deployment in production, uncover underlying problems in applications currently in use, and track application performance trends. APA can help you isolate performance problems in both batch and on-line applications and pinpoint the code that impacts application response times with bottlenecks.

APA provides performance analysis for Assembler, COBOL, C/C++, Java, and PL/I programs. Environment specific statistics for CICS, DB2, IMS, Java, Sysplex, and WebSphere MQ environments. The user sets the parameters, observation duration, and sampling rate. Sometimes a user will want to monitor an application at a current time and sometimes it would be better to wait until later. Application Performance Analyzer is flexible, and conforms to your schedule. You can check on one or more active jobs to monitor with online analysis. Alternately, Application Performance Analyzer can begin monitoring a job or program — or all DB2, IMS, WebSphere MQ, or CICS applications you choose — for a specified duration or number of samples when the job or program becomes active.

Other IT team members, such as system programmers and operations staff, will also find the tool valuable for identifying performance degradation issues prior to referring the problem to the application development team for further investigation and resolution. Application Performance Analyzer functionality is both broad and deep. With support of all major IBM subsystems and support for both z/OS “system” and “user” address spaces, Application Performance Analyzer provides comprehensive analysis and detailed drill-down to source code as required.

Use it to collect information so you can:

- ▶ Improve response time in online applications
- ▶ Reduce batch run time
- ▶ Isolate performance problems in applications
- ▶ Identify excessive I/O activity
- ▶ Identify excessive CPU usage
- ▶ Determine the performance effects of increased workload
- ▶ Monitor virtually any address space

Omegamon XE is a suite of products that monitor and manage system and network applications on a variety of platforms. It helps keep track of the availability and performance of all parts of your enterprise from one or more designated workstations, Omegamon provides reports that you can use to track trends and troubleshoot problems.

Use it to:

- ▶ Display the status of various aspects of your system, helping you to identify bottlenecks and evaluate tuning decisions
- ▶ Simultaneously monitor multiple instances of MVS
- ▶ Define situations based on user-specified thresholds or on the status of a given resource to raise different types of alerts.

Application Performance Analyzer gathers information by measuring and reporting on system resources used by an application running in virtually any system-owned or user address space, such as a step in a batch region, transactions executing under an online subsystem (that is, IMS or CICS transactions) or a TSO region.

Application Performance Analyzer starts when you initiate a session either manually or scheduled. Based on user input, the tool collects samples from the monitored address space and writes the information to an observation file. The data collected can then be analyzed online via ISPF or a PDF file created for viewing with a PDF reader. Examples of information that can be collected include:

- ▶ Control section (CSECT) usage within each load module
- ▶ Instructions or statement usage within each CSECT
- ▶ Assembler, COBOL and PL/I statement usage within each module
- ▶ IBM WebSphere MQ queue information
- ▶ Direct access storage device statistics
- ▶ Processor usage

Spans all major IBM subsystems

Application Performance Analyzer offers powerful breadth in a single tool, providing application monitoring support for all major IBM subsystems:

- ▶ IBM CICS Transaction Server support monitors specific CICS transactions and transaction prefixes with wildcards to help you trace transactions during critical situations, rather than waiting to review periodically collected data. (For post-event performance analysis, you can use CICS Performance Analyzer; see 2.6, “APA and CICS PA in tandem” on page 51.)

- ▶ Support for IMS applications reveals on-call time and service-call time for DL/I and allows you to trace all IMS calls.
- ▶ WebSphere MQ support provides CPU usage by queue, by request, and by transaction in WebSphere MQ. Application Performance Analyzer also provides service time by queue, request, and transaction, and wait time for the same categories.
- ▶ IBM Parallel Sysplex coupling facility support lets you monitor jobs and transactions on any logical-partition (LPAR) image in a Parallel Sysplex.
- ▶ Batch support helps fine-tune efficiency to shrink the time for batch application turnaround.
- ▶ Application Performance Analyzer helps you monitor applications in both test and production, and in multiple source languages, including Assembler, COBOL, and PL/I. Optimized code support for COBOL and PL/I enables monitoring of production applications.

2.5.1 Tool integration

APA shares the same application side file as IBM Fault Analyzer and IBM Debug Tool Utilities and Advanced Functions, which reduces the number of files required across source code debugging, application abend analysis, and performance analysis. In addition, you can use IBM Tivoli OMEGAMON XE to launch Application Performance Analyzer when conditions indicate a requirement to see how an application is performing.

The IBM Tivoli OMEGAMON family of products provides a comprehensive performance and availability solution to proactively analyze and manage operating systems, databases, or other environments for optimal performance. It helps you detect bottlenecks and other potential performance problems from multiple vantage points, and quickly isolates and takes action automatically to resolve these issues.

2.5.2 Using OMEGAMON with APA

The OMEGAMON integration with APA requires Application Performance Analyzer V7R1 or later, and OMEGAMON Omegamon XE or later with TEP (Tivoli Enterprise Portal) configured (Tivoli Web Interface), and TEMS (Tivoli Enterprise Management System) started (Started Task where run the Situations).

From OMEGAMON Workspace, the user creates a situation. A situation is a logical expression involving one or more system conditions, they are used to monitor the condition of systems in your network. You can manage situations from the TEPS by using the Situation editor. From the situation editor, you can create, copy, edit, or delete a situation. TEPS is also where situations are displayed, started, and stopped.

To show the integration between OMEGAMON and APA, in the following sections we explain how to set up an OMEGAMON situation that invokes APA. Our example observes the following situation parameters:

Selection criteria:

- ▶ For any CICS
- ▶ For a Non-CICS Transaction
- ▶ If the Elapsed Time is greater than 1 sec.

Action:

- ▶ Start an APA monitoring session
- ▶ Turn CICS, DB2, DB2+ and DB2V extractors ON
- ▶ Use a default damping rate

Response requested:

- ▶ Transaction Code
- ▶ CICS Job name
- ▶ z/OS System name

Defining a situation with OMEGAMON

After setting the up the initial parameters for the OMEGAMON situation, the formula view is where you define the conditions that will trigger monitoring of a resource or program. When a *Formula* of the selection criteria has been selected, the user would then define the actions for OMEGAMON invoke with the *Action* tab (Figure 2-31).

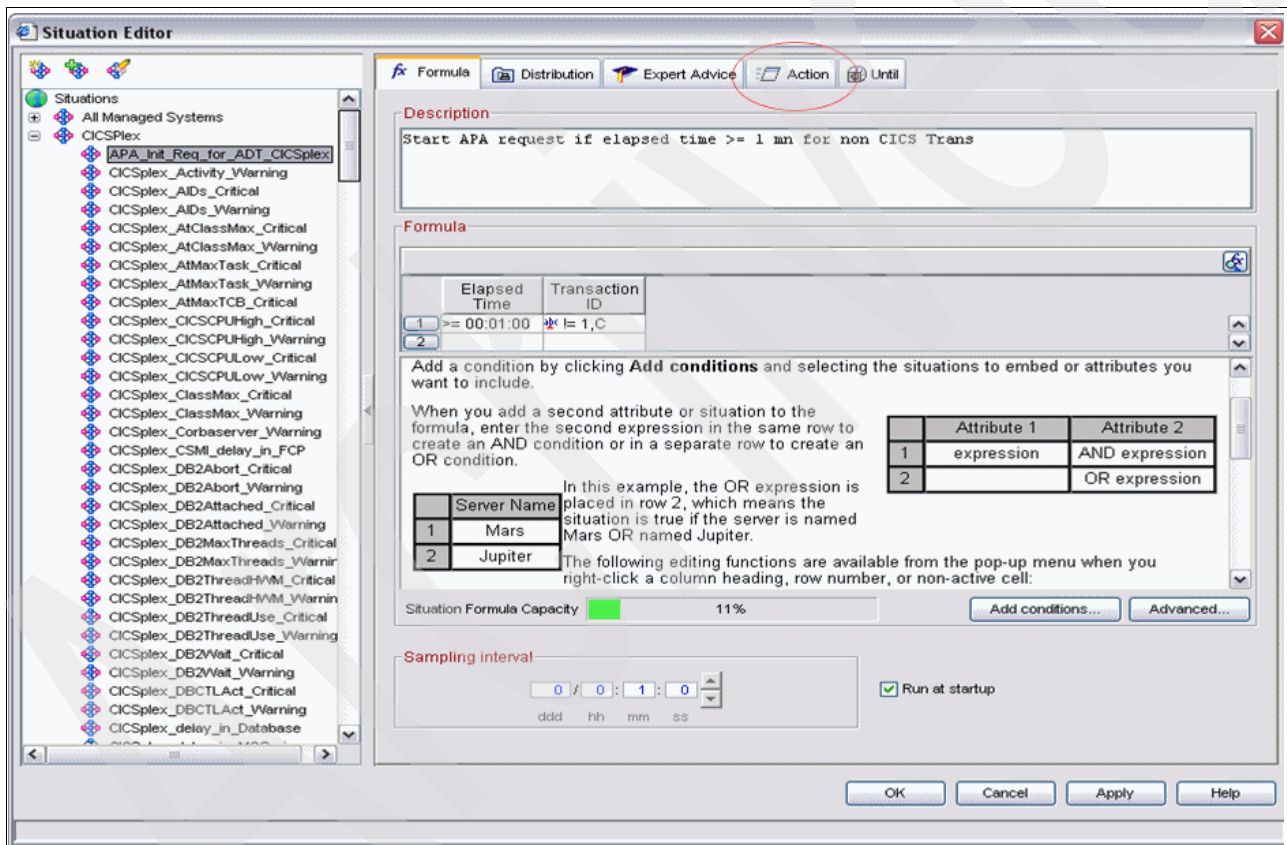


Figure 2-31 OMEGAMON Formula view

From the **Action** view, select the **Attribute Substitution** button (Figure 2-32). This pop-up allows you to select which resource information extractors you would like to turn on (Figure 2-33).

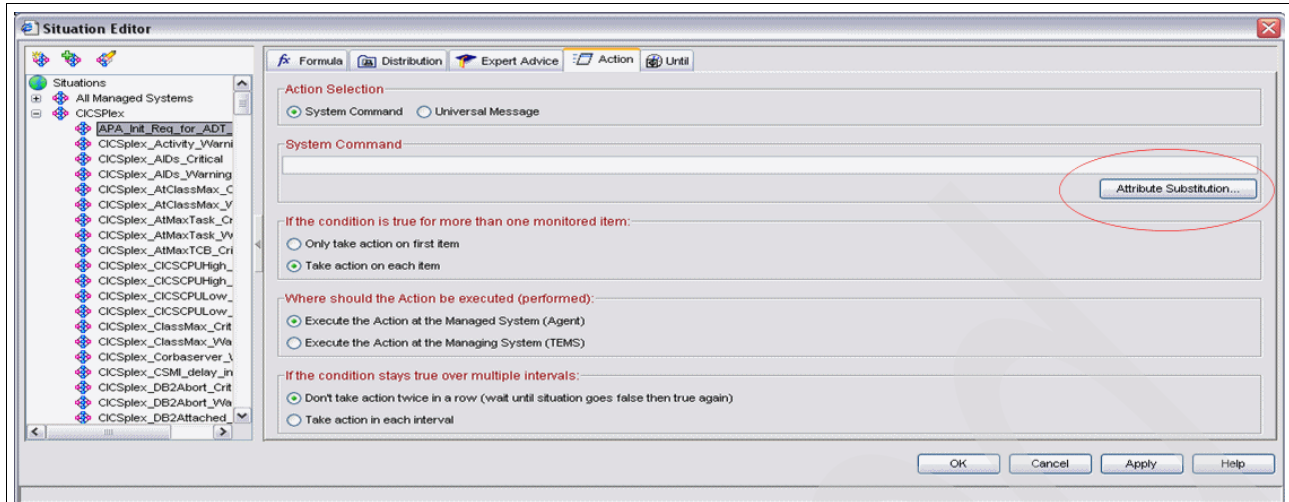


Figure 2-32 OMEGAMON Action View

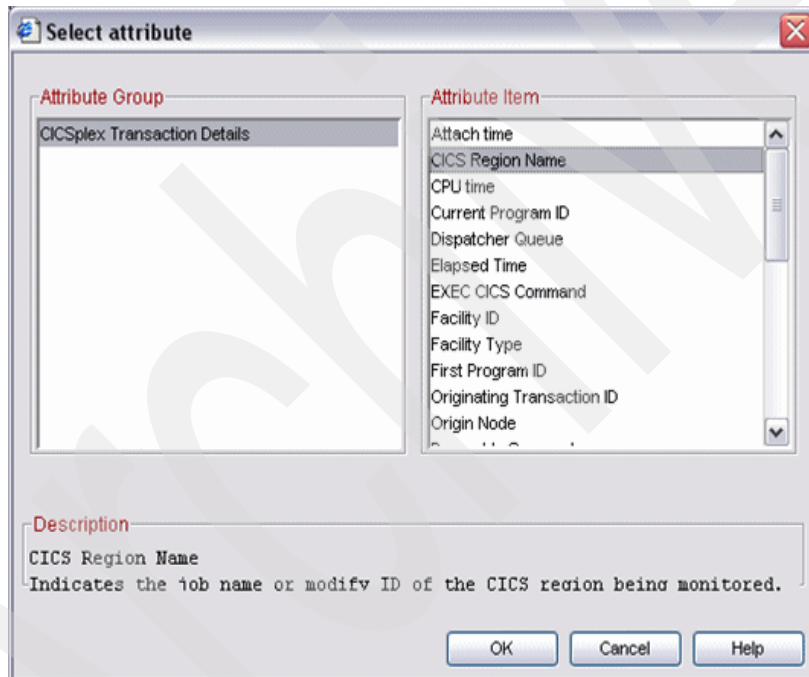


Figure 2-33 OMEGAMON Select Attribute Window

Finally, fill in the **System Command** line. This line specifies the commands to be executed in the event of the situation parameters being met (Figure 2-34).

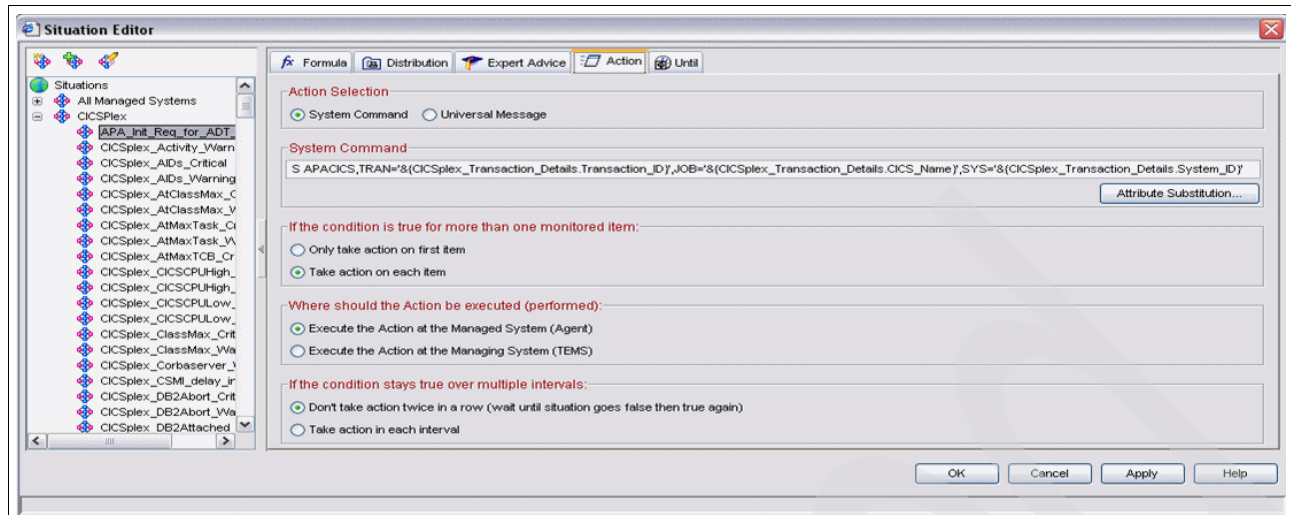


Figure 2-34 System Command Line in Action View

In our case, the System Command line reads:

```
S APACICS,TRAN='&{CICSplex_Transaction_Details.Transaction_ID} ',JOB='&
{CICSplex_Transaction_Details.CICS_Name}'
```

Translation: S APACICS - This starts Application Performance Analyzer for CICS. The remainder of the coding contains the parameters for Transaction Code, CICS Region Name, and System Name that you want to use to direct APA to the correct monitoring location based on the information produced by OMEGAMON.

Application Performance Analyzer digs deep

When an event takes place, and APA is called upon to start monitoring at the location of the detected problem. APA returns an abundant amount of information about the problem area.

2.6 APA and CICS PA in tandem

CICS Performance Analyzer for z/OS (CICS PA) uses performance, statistics, and accounting data written to MVS System Management Facility (SMF) data sets. This includes the data collected by the CICS monitoring facility, DB2 Universal Database for z/OS and WebSphere MQ accounting, the MVS System Logger, IBM Tivoli OMEGAMON XE for CICS (containing transaction data for Adabas, CA-Datcom, CA-IDMS, and Supra database).

You can produce all the CICS PA reports and extracts by simply defining your CICS Systems (Applids), MVS Images, DB2 Subsystems, MVS System Logger and their associated unloaded SMF data sets. Other CICS PA data sets include Report Sets, Report Forms and Object Lists. Report Sets define your report and data extract requests. Report Forms enable you to tailor your reports and extracts to include the information that you want to see. Object Lists enable you to group objects for reporting purposes, for example, to analyze the resource usage of a particular group of transactions or users.

CICS PA reports are designed to enable comprehensive analysis of your entire CICS system — with a detailed overview of transaction volume and performance — so you can evaluate resource usage and predict future usage quickly.

As mentioned before, APA provides a deep dive from the system level to the application level of a problem area. Because APA can provide such a large amount of data, it is important to understand what to look for and where to look for it. CICS PA provides reports that analyze all facets of your CICS systems. including:

- ▶ CICS Application Performance
- ▶ CICS System Resource Usage
- ▶ Cross-System Performance, including Multi-Region Operation (MRO), Advanced Program-to-Program Communication (APPC) and DB2 subsystems
- ▶ Transaction Resource Usage
- ▶ File and TSQueue Resource Usage
- ▶ External Subsystems used by your CICS applications, including WebSphere MQ, DB2, and IMS (DBCTL)
- ▶ MVS Workload Manager (WLM)
- ▶ CICS Business Transaction Services (BTS)
- ▶ Transaction Groups
- ▶ CICS Web Support, IIOp, ECI over TCP/IP, and so on.
- ▶ Exception events that cause performance degradation

CICS reports

The reports generated are also flexible to meet your requirements. Some examples of reports you can produce with CICS PA are:

- ▶ Transaction reports detailing all transactions processed, with the information presented in the format and order that you want
- ▶ Transaction activity reports that can be tailored to produce a detailed analysis and comprehensive overview of transaction performance
- ▶ Resource usage reports that show the performance and utilization of your CICS system resources, including:
 - Transactions and programs
 - Files
 - Temporary storage
 - Transient data
 - Journals and logstreams
 - Virtual storage
 - Terminals
 - BMS maps
 - Secure Sockets Layer (SSL) and CICS Web support
 - Java Virtual Machine (JVM™) support
 - SOAP for CICS
- ▶ Cross-system work reports that combine the CMF data from all of your CICS systems to evaluate the performance of Multi-Region Operation(MRO) and Advanced Program-to-Program Communication (APPC) transactions
- ▶ Transaction resource usage reports showing, at a transaction-level, the individual file and temporary storage queue resources accessed by a transaction
- ▶ Workload activity reports that show, at a glance, if the MVS WLM goals that you have set for your CICS transactions are being met
- ▶ Transaction group reports that combine the performance data of all transactions in your Transaction Groups to produce a consolidated report

- ▶ CICS Business Transaction Services (BTS) Reports that evaluate your BTS application performance
- ▶ External subsystem reports that show the performance and utilization of DB2 and IMS Data Base Control Subsystems

CICS PA offers a comprehensive extract capability that allows you to export transaction performance data to DB2 or your PC workstation for further analysis and graphical representation. Exception reports list and summarize CICS Exception Events that impact the performance of your CICS systems. Performance wait analysis reports are provided to help identify performance bottlenecks. Once the bottlenecks are identified, an APA session can be initiated at the correct location to drill down to the application level.

The CICS reports analyze all facets of your CICS systems, including these:

- ▶ The report formatting capability of CICS PA enables you to tailor the content of your reports to meet your individual requirements
- ▶ A powerful record selection and sorting capability enables you to request the information that you want, in the order that you want it.
- ▶ To help you get started quickly, CICS Performance Analyzer comes with more than 100 standard reports.

You can also select a particular subset of SMF data and use it as input to CICS Performance Analyzer for faster processing.

Historical Database

The Historical Database (HDB) provides a flexible and easy-to-use facility for collecting and managing historical performance data for your CICS systems. This can be very valuable in detecting problem areas to drill down on with APA that might not be apparent except when looking at the performance over time. The CICS PA HDB function provides:

- ▶ Short-term history data detailing individual transaction performance for use in performance problem analysis
- ▶ Long-term history data summarized over time can be used for trend analysis and capacity planning
- ▶ A powerful and flexible definition facility for the historical data repositories
- ▶ Comprehensive reporting facilities
- ▶ A facility to optionally load history data into DB2 for further analysis and reporting using DB2 reporting tools such as Query Management Facility (QMF™)
- ▶ Definition and management of the HDBs from the CICS PA ISPF dialog

Application Grouping

An update to CICS PA V2.1 (described by APAR PK63876) introduced Application Grouping, which enables you to create reports that relate data to logical business units or functions known as Applications. In this context, an Application is a name that you associate with a set of performance data field values. For example, you can associate the Application name *Accounting* with the CICS transaction ID (TRAN) field values DEPT, WDRW, and ACC* (that is, matching any transaction ID beginning with the characters ACC). Then you can create reports or extracts that refer to the performance data for those transaction IDs as belonging to the *Accounting* Application.

The Application Grouping feature of CICS PA allows closer integration with Application Performance Analyzer, enabling you to group and sort data at the application level, while retaining the ability to drill-down to the specific source of a problem. The CICS PA report in Figure 2-35 summarizes performance at an application level (under Application names such as *Accounting* in the BUSFUNC, or *business function*, Group column heading), while also showing performance data for each transaction identifier (under the Tran column heading).

BUSFUNC Group	Tran	#Tasks	Avg Response Time	Max Response Time	Avg Dispatch Time	Avg User CPU Time	Avg Suspend Time	Max Suspend Time	Avg DispWait Time	Avg FC Wait Time
Accounting	ADJQ	9	.3912	.8376	.0030	.0022	.3881	.8312	.0034	.0000
Accounting	ADUS	9	.3343	.7729	.2517	.0485	.0825	.1783	.0243	.0000
Accounting	AEVS	39	.0281	.3362	.0015	.0012	.0266	.3344	.0010	.0000
...										
Accounting		76	1.2924	74.6819	.3350	.0808	.9574	64.9890	.0382	.0000
CICS-supplied transactions	CSKP	16	.0090	.0232	.0016	.0011	.0074	.0213	.0017	.0000
CICS-supplied transactions	CSMI	34193	.2999	31.9736	.0013	.0009	.2986	31.9695	.0005	.0000
.0041										
CICS-supplied transactions	CSM2	2	.0010	.0015	.0004	.0004	.0005	.0008	.0000	.0000
...										
CICS-supplied transactions		34340	.3002	31.9736	.0026	.0009	.2976	31.9695	.0007	.0041
Delivery	DBEC	99	.0196	.1998	.0015	.0012	.0180	.1981	.0031	.0175
Delivery	DBUS	23	.6747	2.5609	.5730	.1066	.1017	.2060	.0220	.0000
Delivery	DI12	148	.0284	.1486	.0012	.0009	.0272	.1467	.0017	.0000
...										
Delivery		326	.2626	2.5609	.0473	.0114	.2152	2.3320	.0047	.0053
Finance	FJD3	46	.0024	.0185	.0019	.0015	.0005	.0165	.0000	.0000
Finance	FTB2	2	9.4656	10.4636	.0059	.0045	9.4597	10.4562	.0018	.0000
Finance	FTB3	8	2.0977	2.1966	.0019	.0014	2.0958	2.1951	.0030	.0000
...										
Finance		102	1.3056	10.4636	.0019	.0015	1.3036	10.4562	.0006	.0000
Statistics collection	SSFR	9	.0264	.0359	.0221	.0054	.0043	.0096	.0012	.0000
Statistics collection	#BEK	3	.0020	.0022	.0008	.0007	.0012	.0013	.0002	.0000
Statistics collection	#DDS	927	4.9054	20.0135	2.4500	.0376	2.4554	15.8274	.0254	.0000
...										
Statistics collection		3497	1.8609	24.1543	.8081	.0335	1.0528	24.0906	.0146	.0000
Unassigned transactions	IFB4	4	.8400	2.7034	.6737	.0057	.1663	.2678	.0067	.0000
Unassigned transactions	MD15	1	.0199	.0199	.0016	.0012	.0182	.0182	.0002	.0000
Unassigned transactions	MD16	6	.0003	.0005	.0002	.0002	.0001	.0003	.0000	.0000
...										
Unassigned transactions		21599	.2013	8.2288	.1256	.0239	.0756	6.6346	.0235	.0000
Total		59940	.3584	74.6819	.0946	.0113	.2638	64.9890	.0098	.0024

Figure 2-35 CICS PA Performance Summary Report: summarizing by Application, and then by transaction identifiers within each Application

2.7 Workload Simulator with Rational Test Manager

Workload Simulator (WSim) for z/OS and OS/390 is a z/OS-based, terminal and network simulation tool. WSim TM is a z/OS-based usability enhancement to Workload Simulator that guides the user through the test process. WSim TM creates projects that contain schedules for simulation runs.

Rational TestManager is a Windows console for test activity management, execution, and reporting. This tool is an open and extensible framework that unites all of the tools, assets, and data related to the testing effort. Rational TestManager is the central console for test activity management, execution and reporting. Built for extensibility, it supports everything from pure manual test approaches to various automated paradigms including unit testing, functional regression testing, and performance testing. Rational TestManager is meant to be accessed by all members of a project team, ensuring the high visibility of test coverage information, defect trends, and application readiness.

The WSim Adapters extend Rational TestManager for Windows 2000 and Windows XP. There are two sets of WSim Adapters in all, one for running Workload Simulator Test Manager (WSim TM) schedules and one for running custom JCL scripts. The WSim Adapters enable software testers to use Rational TestManager to run WSim TM schedules and custom JCL scripts remotely and in batch mode.

Working with Workload Simulator Test Manager schedules

After installing the WSim Adapters for running WSim TM schedules, you have to register them as a new test script type. You can then use them to retrieve and display WSim TM projects and schedules in Rational TestManager, run the schedules, and see the pass/fail results and job output of the run.

Remember that to run WSim TM schedules from Rational TestManager, the schedules must already exist in WSim TM on the host system. In addition, you require the following IDs and passwords to run WSim TM schedules:

- ▶ TSO FTP User Name and password
- ▶ TSO WSim User Name. This is the TSO user name that created the projects and schedules through WSim TM. This can be the same ID as the TSO FTP User Name.

Registering the WSim Adapters for running WSim TM schedules

The first step is to register the WSim Adapters for running WSim TM schedules. This registration is required for each Rational TestManager project used to run WSim TM schedules. You must have administrator privileges to register the WSim Adapters. You can register the WSim Adapters two ways: automatically, by using the WSim Registration Tool, or manually.

Creating a new test script source

The next step in the process is to create a local test script source for the schedules. This local source corresponds to the schedules located on the host system. Think of the local source as corresponding to a host account.

Retrieving WSim TM projects and schedules

The WSim Adapters retrieve all project names assigned to a WSim user ID and all schedule names for each of those projects. The WSim TM project and schedule names and associated information required to run the schedules are also retrieved. It should be noted that you cannot view or edit the content of WSim TM schedules in Rational TestManager. You can view and edit schedules only on the host system.

Working with custom JCL scripts

After installing the WSim Adapters for running custom JCL scripts, you have to register them as a new test script type. You can then use them to run custom JCL scripts and see the pass/fail results and job output of the run.

Registering the WSim Adapters for running custom JCL scripts

The first step is to register the WSim Adapters for running custom JCL scripts. You can register the WSim Adapters automatically by using the WSim Registration Tool, or you can register them manually.

Archived



Program preparation

In this chapter we summarize the options that you have available in preparing your source code for use by the IBM Problem Determination Tools. We also provide recommended approaches.

3.1 Introduction

The IBM PD Tools products are designed to use modules and source information files produced by IBM compilers in conjunction with utilities supplied with the IBM PD Tools. Each compiler can generate different kinds of source information files, and each PD Tool product can support one or more different file formats. You should select the best formats for your combination of compilers and PD Tools products — perhaps it might be optimal to store different file formats for different compilers. The source information file formats, and the compilers that can produce them, are listed in Table 3-1.

Table 3-1 Source information file formats

Source information file type	Can be generated by:
SYSDEBUG	All LE COBOL, Enterprise PL/I V3.5 or later
Compiler listing	All compilers
LANGX	All compilers
Expanded source	Enterprise PL/I, C, C++
DWARF	C, C++
SYSADATA	Assembler

Figure 3-1 shows a diagram of the source information file preparation options.

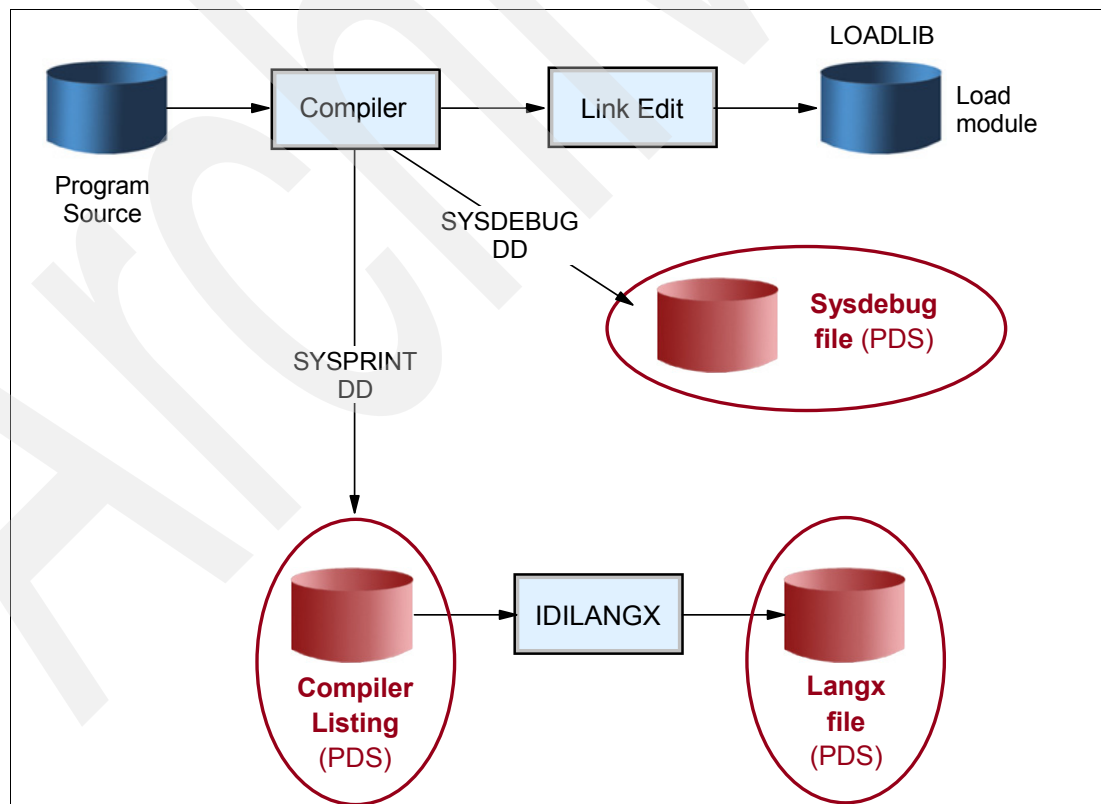


Figure 3-1 Source information file preparation options

When your programs are compiled for the PD Tools, and you have the right source information files, then proceed as follows:

- ▶ When using IBM Debug Tool and Utilities and Advanced Functions (DTUAF), you can perform functions such as stepping through program source statements, setting breakpoints, and viewing and changing variables.
- ▶ When using IBM Fault Analyzer (FA), you can pinpoint the abending statement, and show other statements and variable values associated with theabend.
- ▶ When using IBM Application Performance Analyzer (APA), you can highlight the program statements using the most CPU resource, or being the cause of the most wait time.

Ultimately, you can decide on an approach that is most appropriate for your organization's current processes and environment, along with your plans for usage of the tools. We believe that it is a best practice to save source information, in some form, for all of your production programs.

For each compiler you use, research the options and methods available, and decide:

- ▶ Which formats are best to store source information files
- ▶ Which compiler options are required to support this

When this has been decided:

- ▶ Compile processes will have to be changed to specify the appropriate compiler options and to save source information files.
- ▶ Program promotion processes must be changed to promote source information files along with load modules. The vehicle for promotion can be either a copy, move or recompile.

Our recommendation is that you promote the source information, in whichever format you choose, along with the load module as it moves through the application development cycle and into production. See the diagram in Figure 3-2.

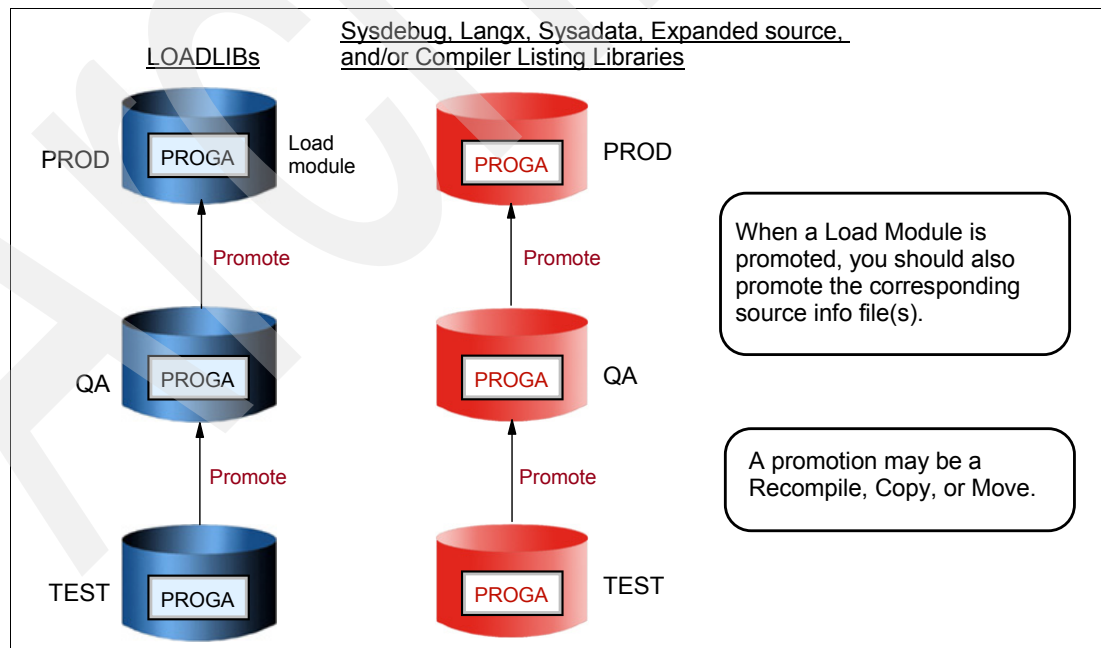


Figure 3-2 Code promotion

For some compilers/environments, you might want to have Debug Tool start-up automatically when the application load module is accessed.

3.1.1 SYSDEBUG

A SYSDEBUG file is a special file format that contains program source and symbolics information. It is a format available only with LE COBOL compilers and the v3.5 Enterprise PLI compiler or later. The compiler generates a SYSDEBUG file when you specify the TEST compiler option with the SEPARATE parameter. Examples:

```
COBOL: TEST(NOHOOK,SEPARATE,EJPD),OPTIMIZE
PLI:    TEST(ALL,NOHOOK,SYM,SEPARATE)
```

Note: NOTEST is the default compiler listing. Programs or subprograms compiled with the NOTEST option are executed but ignored by Debug Tool.

The generated module is stored in a PDSE or PDS data set, where the member name equals the program name, with LRECL=(80 to 1024), RECFM=FB, and BLKSIZE=(any multiple of lrecl).

If required, you can produce a listing, similar to a compile listing, using either the supplied EQALANGP, IDILANGP, or CAZLANGP utility.

The general recommendation for both test and production is to create and save, where possible, a SYSDEBUG file. These are some benefits of this approach:

- ▶ SYSDEBUG files are smaller than compiler listings (less disk space).
- ▶ The SYSDEBUG file can be used by all of the PD Tools products.
- ▶ The load module is production-ready — it can be copied directly into production without performance concerns.
- ▶ The SYSDEBUG file can be automatically located by some tools — the compiler stores the data set name of the debug file in the load module.

If you will not be using the load module in production, then consider using the compiler listing rather than a SYSDEBUG file. You can then recompile with different options when moving into production.

You would use this method instead of the general recommendation if the Debug Tool Dynamic Debug facility is not installed, or if you have to use Debug Tool (DT) with programs in z/OS protected-storage. For example, DB2 stored procedures can run protected.

3.1.2 Compiler listing

A compiler listing is the report produced by a compiler. For use by the PD tools, instead of writing it directly to SYSOUT, you have to save it in a file.

In many cases, if you still have the original compiler listing for an existing load module, you can use it for FA and APA, provided they were produced with the necessary compiler options. This might not be true for DT, because it requires the TEST compiler option.

For FA and APA, you can re-create a compiler listing for an existing load module by recompiling the program, provided that neither the source code nor the compiler has changed since the load module was created. This will not work for DT, because it requires an exact time stamp match between the listing and the load module.

To use compiler listings with PD Tools, you store them in a PDS or PDSE where the member name equals the program name:

- ▶ COBOL (except OS/VS COBOL) SYSPRINT:
 - RECFM=FBA,LRECL=133
- ▶ OS/VS COBOL SYSPRINT:
 - RECFM=FBA,LRECL=121
- ▶ PLI SYSPRINT:
 - RECFM=VBA,LRECL=125 minimum
- ▶ Enterprise PLI SYSPRINT:
 - RECFM=VBA,LRECL=137 minimum
- ▶ C or C++ SYSPRINT
 - RECFM=VB or VBA,LRECL=137 minimum
 - RECFM=FB or FBA,LRECL=133

3.1.3 LANGX

A LANGX file, also called a *Side File*, is a special file format that is produced by a utility program that comes with each of the PD Tools. It is an alternative to storing compiler listings.

The relevant utility for each of the PD Tools is:

- ▶ FA: IDILANGX
- ▶ DTU&AF: EQALANGX
- ▶ APA: CAZLANGX
- ▶ Assembler: EQALANGX

The utility reads a compiler listing or an assembler SYSADATA file and produces a LANGX file. This can then be stored in a PDS or PDSE data set with LRECL=1562,RECFM=VB,BLKSIZE=(1566 to 32K).

For some languages, LANGX files are supported by more of the PD Tools products than other file types. In these cases, if you store a LANGX file, you might not have to store other types of source information files.

Note: A LANGX file can only be used by Debug Tool when generated from OS/VS COBOL or Assembler source code.

LANGX files are smaller than compiler listings, comparable in size to SYSDEBUG files.

While LANGX files are not directly “human readable”, you can produce a listing similar to a compiler listing from a LANGX file with the FA IDILANGP utility or the APA CAZLANGP utility.

If required, you can produce a listing, similar to a compile listing, using either the supplied EQALANGP, IDILANGP, or CAZLANGP utility.

Expanded program source file: This is a program source code that has been expanded, or preprocessed, by the compiler. It contains statements generated from PLI %include, C #include, in-line functions, or macro expansions. You should use it with PLI, C, and C++ compilers when you have to see complete source information in the PD Tools. It is stored in a PDS, PDSE or sequential data set with any format that is supported by the compiler.

3.1.4 SYSADATA

The SYSADATA file is the file format produced by Assembler. It can be processed directly by both Fault Analyzer and APA, or used as input to the EQALANGX utility to create a LANGX file. It is generated when you specify the ADATA assembler option and stored in a PDSE or PDS data set, where the member name equals the CSECT or entry point name, with LRECL=(8188 to 32K-4),RECFM=VB,BLKSIZE=(lrecl+4 to 32K).

3.1.5 DWARF

A DWARF file is a special file format, available with C and C++ V1.5 and later compilers, that contains debug information. During a debug session, Debug Tool obtains information from both this file and the source file.

C and C++ programs compiled with the FORMAT(DWARF) option create a load module smaller in size than that created with the TEST option.

3.2 COBOL

For general information on the COBOL compilers, refer to:

<http://www.ibm.com/software/awdtools/cobol/>

Release history of the various COBOL compilers can be found at:

<http://www.ibm.com/servers/eserver/zseries/zos/1e/history/history.html>

A COBOL code conversion utility is supplied with Debug Tool Utilities and Advanced Functions. It can be used to migrate old COBOL source, both batch and CICS, up to the current level. Further migration information can be found in the Migration Guide, available at:

<http://www.ibm.com/software/awdtools/cobol/zos/library/>

3.2.1 Enterprise COBOL V4.1 (5655-S71)

This topic describes how to prepare programs written in Enterprise COBOL v4.1.

We recommend that you use the TEST(NOHOOK,SEPARATE,EJPD),OPTIMIZE compiler option. This will not place hooks into the program and hence there is no run-time overhead when not debugging. It does require that you have dynamic debug active. The sub-option of the TEST compiler option, EJPD, enables the Debug Tool commands JUMPTO and GOTO for production debugging.

You can optionally use compiler listings instead of SYSDEBUG files. However, the options LIST,MAP,SOURCE,XREF are required, resulting in a larger load module. To use a compiler listing with APA, you must also use the NONUMBER compiler option. See Table 3-2.

Table 3-2 Enterprise COBOL V4.1 compiler options

Source file	Compiler options	Production ready	DTU&AF	FA	APA
SYSDEBUG	TEST(NOHOOK,SEPARATE,EJPD),OPTIMIZE	Yes	Yes(1)	Yes(1)	Yes(1)

Source file	Compiler options	Production ready	DTU&AF	FA	APA
SYSDEBUG	TEST(HOOK,SEPARATE,EJPD),OPTIMIZE		Yes (2)	Yes (2)	Yes (2)
Compiler listing	TEST(xxx,NOSEP,EJPD),LIST,MAP,SOURCE,XREF,NONUMBER		Yes	Yes	Yes
Compiler listing	TEST(xxx,NOSEP,EJPD),LIST,MAP,SOURCE,XREF,NUMBER		Yes	Yes	
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF,NONUM	Yes		Yes	Yes
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF,NUMBER	Yes		Yes	
LANGX	LIST,MAP,SOURCE,XREF	Yes		Yes	Yes

Notes:

1. Recommended for production and test
2. Good option for test only

3.2.2 Enterprise COBOL V3 (5655-G53) and COBOL for OS/390 and VM (5648-A25)

This topic describes how to prepare programs written in Enterprise COBOL prior to V4.1.

We recommend that you use the TEST(NONE,SYM,SEPARATE),NOOPTIMIZE compiler options. This will not place hooks into the program and hence there is no run-time overheads when not debugging. It does require that you have dynamic debug active.

You must use the TEST compiler option to use Debug Tool. The NOOPTIMIZE option is also recommended. Debug Tool has some restrictions with OPTIMIZE:

- ▶ The GOTO and JUMPTO commands are disabled.
- ▶ The SET WARN OFF setting is required before you can change variable values.

For DB2 stored procedures only, you must specify TEST(ALL,SYM,SEPARATE).

You can optionally use compiler listings instead of SYSDEBUG files. However, the options LIST,MAP,SOURCE,XREF are required, resulting in a larger load module. To use a compiler listing with APA, you must also use the NONUMBER compiler option. See Table 3-3.

Table 3-3 Enterprise COBOL V3.n compiler options

Source file	Compiler options	Production ready	DTU&AF	FA	APA
SYSDEBUG	TEST(NONE,SYM,SEPARATE)	Yes	Yes(1)	Yes(1)	Yes(1)
SYSDEBUG	TEST(ALL,SYM,SEP),NOOPT		Yes (2)	Yes (2)	Yes (2)
Compiler listing	TEST(xxx,xxx,NOSEP),LIST,MAP,SOURCE,XREF,NONUMBER		Yes	Yes	Yes

Source file	Compiler options	Production ready	DTU&A F	FA	APA
Compiler listing	TEST(xxx,xxx,NOSEP),LIST,MAP,SOURCE,XREF,NUMBER		Yes	Yes	
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NONUMBER	Yes		Yes	Yes
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NUMBER	Yes		Yes	
LANGX	LIST,MAP,SOURCE,XREF	Yes		Yes	Yes

Notes:

1. Recommended for production and test
2. Good option for test only

3.2.3 COBOL for MVS and VM (5688-197)

This topic describes how to prepare programs written for COBOL for MVS and VM.

Note: Our recommendation is that you migrate all your back level COBOL to Enterprise COBOL. Debug Tool Utilities and Advanced Functions has function that assist with this.

We recommend that you use the TEST(ALL,SYM),NOOPTIMIZE,LIST,MAP,XREF,SOURCE) compiler options.You could recompile the program with different options for production if performance is a concern.

You can use the generated compiler listing with all of the PD Tools products.

A VS COBOL II module generated with the TEST option has run-time overhead because the compiler places debug hooks at statements and path points, and stores symbol tables in the module. See Table 3-4.

To use a compiler listing with APA, you must use the NONUMBER compiler option.

Table 3-4 COBOL for MVS and VM compiler options

Source file	Compiler options	Production ready	DTU&A F	FA	APA
Compiler listing	TEST(ALL,SYM),LIST,MAP,NOOPT, SOURCE,XREF,NONUMBER		Yes	Yes	Yes
Compiler listing	TEST(ALL,SYM),LIST,MAP,NOOPT, SOURCE,XREF,NUMBER		Yes	Yes	
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NONUMBER	Yes		Yes	Yes
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NUMBER	Yes		Yes	
LANGX	LIST,MAP,SOURCE,XREF	Yes		Yes	Yes

3.2.4 VS COBOL II (5668-958)

This topic describes how to prepare programs written for VS COBOL II. The options vary depending upon whether or not you are running with LE run-time services.

Note: Our recommendation is that you migrate all your back level COBOL to Enterprise COBOL. Debug Tool Utilities and Advanced Functions has function that assist with this.

Module is linked with LE run-time services

For test environments only, we recommend that you create and save a compiler listing using the compiler options TEST,LIST,MAP,SOURCE,XREF,NONUMBER. Remember that TEST is mandatory for use with Debug Tool. You could recompile the program with different options for production if performance is a concern.

You can use the generated compiler listing with all of the PD Tools products.

A VS COBOL II module generated with the TEST option has run-time overhead because the compiler places debug hooks at statements and path points, and stores symbol tables in the module.

To use a compiler listing with APA, you must use the NONUMBER compiler option.

For a production environment, you should create and save a compiler listing using the compiler options NOTEST,LIST,MAP,SOURCE,XREF,NONUMBER. While the compiler listing can be used by APA and FA, DT cannot be used with a module compiled with NOTEST. However, if you do not use DT, you can use this configuration for both test and production. If you want to reduce the amount of disk space required to save compiler listings and you do not plan to use DT, then consider creating LANGX files. See Table 3-5.

Table 3-5 VS COBOL II compiler options

Source file	Compiler options	Production ready	DTU&A F	FA	APA
Compiler listing	TEST,LIST,MAP,SOURCE,XREF, NONUMBER		Yes(2)	Yes(2)	Yes(2)
Compiler listing	TEST,LIST,MAP,SOURCE,XREF, NONUMBER		Yes	Yes	
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NONUMBER	Yes		Yes(1)	Yes(1)
Compiler listing	NOTEST,LIST,MAP,SOURCE,XREF, NUMBER	Yes		Yes	
LANGX	NOTEST,LIST,MAP,SOURCE,XREF (either NUM or NONUM)	Yes		Yes	Yes

Notes:

1. Recommended for production and test
2. Good option for test only

Module is NOT linked with LE run-time services

We recommend that you create and save a compiler listing using the compiler options NOTEST,LIST,MAP,SOURCE,XREF,NONUMBER. The compiled module will be production ready.

Note: PTF UK32141 must be applied to allow Debug Tool to work in this environment.

Write the compiler listing to a file by modifying the SYSPRINT DD statement in the compiler step. It can be a temporary file if you do not want to save the compiler listing permanently.

Add another step after the compiler step to run the LANGX utility program. This utility reads the compiler listing and produces a LANGX file. The LANGX file is the source information file that can be used by the tools. Code the JCL so that the file is written to the LANGX library, and specify a member name that matches the program name or entry point name. The LANGX utility program is delivered with Debug Tool as EQALANGX, in Fault Analyzer as IDILANGX, and in APA as CAZLANGX. Any of the three can be used, they are equivalent. See Table 3-6.

Table 3-6 VS COBOL II compiler options

Source file	Compiler options	Production ready	DTU&A F	FA	APA
LANGX	NOTEST,LIST,MAP,SOURCE,XREF (either NUM or NONUM)	Yes	Yes(1)	Yes(1)	Yes(1)

Notes:

1. Recommended for production and test

3.2.5 OS/VS COBOL (5740-CB1)

This topic describes how to prepare programs written for OS/VS COBOL.

Note: Our recommendation is that you migrate all your back level COBOL to Enterprise COBOL. Debug Tool Utilities and Advanced Functions has function that assists with this.

For both test and production we recommend that you create and save a LANGX file using the compiler options NOBATCH, NOCLIST, NOCOUNT, DMAP, NOLST, PMAP, SOURCE, NOSYMDMP, NOTEST, VERB, XREF.

The generated LANGX file can be used by all of the PD Tools products and the load module is production-ready.

The NOOPTIMIZE option is recommended for Debug Tool. In some cases, it can be confusing if this is not used, because there might not be a 1-to-1 correlation between source statements and object code. See Table 3-7.

Note: For Debug Tool, you must link-edit the module using the LE run-time library, not the VS COBOL II run-time library.

Table 3-7 OS/VS COBOL compiler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
Compiler listing	DMAP,NOCLIST,NOLIST,PMAP,SOURCE,XREF,VERB	Yes		Yes	
Compiler listing	(LIST,NOPMAP) or (CLIST,NOPMAP) or (CLIST,PMAP)	Yes			Yes
LANGX	NOBATCH,NOCLIST,NOCOUNT,DMAP,NOLST,PMAP,SOURCE,NOSYMDMP,NOTEST,VERB,XREF	Yes	Yes(1)	Yes(1)	Yes(1)

Notes:

1. Recommended for production and test

3.3 PLI

For general information on the PLI compilers, refer to:

<http://www.ibm.com/software/awdtools/pli/>

Release history of the various PLI compilers can be found at:

<http://www.ibm.com/servers/eserver/zseries/zos/1e/history/plimvs.html>

3.3.1 Enterprise PLI V3.7 and later (5655-H31)

If you are using Enterprise PL/I for z/OS, Version 3.7 or later and you want to get the most DT functionality and a small program size, we recommend that you use the compiler options TEST(ALL,NOHOOK,SYM,SEPARATE,SOURCE). You must have the Dynamic Debug facility installed, and you might have to install the Authorized Debug facility.

To get all Debug Tool functionality but have a larger program size and do not have the debug information in a separate file, use TEST(ALL,HOOK,SYM,NOSEPARATE). See Table 3-8.

Table 3-8 Enterprise PL/I V3.7 and later compiler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
SYSDEBUG	TEST(ALL,NOHOOK,SYM,SEPARATE,SOURCE)	Yes(1)	Yes(1)	Yes(1)	
SYSDEBUG	TEST(ALL,HOOK,SYM,NOSEP)		Yes (2)	Yes (2)	
Compiler listing	TEST(ALL,HOOK,SYM,NOSEP),LIST,MAP,SOURCE,XREF,NONUMBER			Yes	Yes
Compiler listin	TEST(ALL,HOOK,SYM,NOSEP),LIST,MAP,SOURCE,XREF,NUMBER			Yes	
Compiler listin	NOTEST,LIST,MAP,SOURCE,XREF,NONUMBER	Yes		Yes	Yes
Compiler listin	NOTEST,LIST,MAP,SOURCE,XREF,NUMBER	Yes		Yes	
LANGX	LIST,MAP,SOURCE,XREF	Yes		Yes	Yes(1)

Notes:

1. Recommended for production and test
2. Good option for test only

3.3.2 Enterprise PLI V3.5 and V3.6 (5655-H31)

This topic describes how to prepare programs written using V3.5 and later versions of the Enterprise PLI compiler.

For test environments, and optionally production, we recommend that you perform a 2-stage compile. In the first compile step, %INCLUDEs and Macros are expanded. In the second compile step, either:

1. Create and save a SYSDEBUG file for use with DT using the compiler options TEST(ALL,SYM,NOHOOK,SEP),NOOPTIMIZE. The TEST option makes the load module larger, a possible concern when you consider the move to production. This is a difference between COBOL and Enterprise PLI.
2. Create and save a LANGX file for use with FA and APA using the compiler options AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OPTIONS, SOURCE, STMT, XREF(FULL).

If the load module will not be used in production, then consider changing the TEST option to TEST(ALL,SYM,HOOK,SEP). This will also be necessary if the Debug Tool Dynamic Debug facility is not installed or if you have to use DT with programs in z/OS protected storage.

We recommend for production environments, and optionally test environments if you do not use DT, that you create and save a LANGX file using the compiler options NOTEST, AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OPTIONS, SOURCE, STMT, XREF(FULL). This file can be used both by APA and FA.

If you will be using both Enterprise PLI 3.5 (or later), and also a previous version (3.4 or earlier), then consider using the recommendations in the next topic (3.3.3, "Enterprise PLI V3.4 and earlier (5655-H31)" on page 69). This will allow source information files to be being handled the same way for all versions of your Enterprise PLI compilers. See Table 3-9.

Table 3-9 Enterprise PLI V3.5 and V3.6 compiler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
SYSDEBUG	Second stage of compile (3) TEST(ALL,SYM,NOHOOK,SEPARATE),NOOPT	Yes (1) (4)	Yes (2)	Yes	
SYSDEBUG	Second stage of compile TEST(ALL,SYM,OHOOK,SEPARATE),NOOPT		Yes	Yes	
Expanded source file	Second stage of compile TEST(ALL,SYM,OHOOK,SEPARATE),NOOPT		Yes		
Compiler listin	AGGREGATE,ATTRIBUTES(FULL),NOBLKOFF,LIST,MAP,NEST,NOTEST, NONUMBER,OFFSET,OPTIONS,SOURCE,STMT,XREF(FULL) (3)	Yes		Yes	
LANGX		Yes		Yes(1)	Yes(1)

Notes:

1. Recommended for production and test
2. Good option for test (and optionally production)
3. These can be combined (use TEST)
4. Module is larger than with NOTEST

3.3.3 Enterprise PLI V3.4 and earlier (5655-H31)

This topic describes how to prepare programs written for Enterprise PLI v3.4 and earlier.

For test environments, we recommend that you perform a 2-stage compile. In the first compile step, INCLUDEs and Macros are expanded. In the second compile step, either:

1. Create and save an expanded file, for use with Debug Tool using the compiler options TEST(ALL,SYM),NOOPTIMIZE, or
2. Create and save a LANGX file for use with FA and APA using the compiler options AGGREGATE, ATTRIBUTES(FULL), NOBLKOFF, LIST, MAP, NEST, NONUMBER, OPTIONS, SOURCE, STMT, XREF(FULL).

For production environments, we recommend that you create and save a LANGX file, in a PDSE or PDS data set where the member name equals the program name, using the compiler options and parameters NOTEST, AGGREGATE, ATTRIBUTES(FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL).

If you do not use Debug Tool, you can use this configuration for both test and production. See Table 3-10.

Table 3-10 Enterprise PLI V3.4 and earlier compiler options

Source file	Compiler options	Production ready	DTU&A F	FA	APA
Expanded source file	Second stage of compile (3) TEST(ALL),NOOPT		Yes (2)		
Compiler listing	AGGREGATE,ATTRIBUTES(FULL), NOBLKOFF,LIST,MAP,NEST,NOTEST, NONUMBER,OFFSET,OPTIONS, SOURCE,STMT,XREF(FULL) (3)	Yes		Yes	
LANGX		Yes		Yes(1)	Yes(1)

Notes:

1. Recommended for production and test
2. Good option for test only
3. These can be combined (use TEST)

3.3.4 PLI for MVS and VM (5688-235), and OS/PLI (5668-909 and 5734-PL1)

This topic describes how to prepare programs written for PLI for MVS and VM and OS PLI for MVS and VM.

We recommend for test environments that you either:

1. Create and save an expanded source file for DT using the compiler options TEST(ALL),NOOPTIMIZE. This is considered a “testing-only” configuration because a module generated with the test option has run-time overhead:
 - The compiler places “debug hooks” at statements and path points.
 - The tables are stored in the module, which makes it much larger.
2. Create and save a LANGX file for FA and APA using the compiler options AGGREGATE, ATTRIBUTES(FULL), ESD, LIST, MAP, NEST, OPTIONS, SOURCE, STMT, XREF(FULL).

This is the recommendation for production environments, and optionally test if you do not use DT.

For CICS applications only, if you use the DTCN transaction to start DT, you will have to link-edit the Debug Tool CICS startup exit module, EQADCCXT, into your application load module. This exit can be found in the Debug Tool load library, SEQAMOD.

This is not required if you are using the CADP transaction. See Table 3-11.

Table 3-11 PL/I for MVS and VM and OS/PLI compiler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
Compiler listing	TEST(ALL),AGGREGATE, ATTRIBUTES(FULL),ESD,LIST,MAP, NEST,OPTIONS,SOURCE,STMT, XREF(FULL),NOOPT		Yes (2)	Yes	
LANGX				Yes	Yes
Compiler listing	TEST(ALL)AGGREGATE, ATTRIBUTES(FULL),ESD,LIST,MAP, NEST,OPTIONS,SOURCE,STMT, XREF(FULL)	Yes		Yes	
LANGX		Yes		Yes(1)	Yes(1)

Notes:

1. Recommended for production and test
2. Good option for test only

3.4 Assembler

This topic describes how to prepare programs written for High Level Assembler. For general information about High Level Assembler, refer to the High Level Assembler (HLASM) and Toolkit Feature at:

<http://www.ibm.com/software/awdtools/hlasm>

We recommend that for both test and production, you create and save a LANGX file. When you do this, you must use the assembler ADATA option. If this listing will be used with Application Performance Analyzer, then you also have to save the SYSADATA file.

The LANGX file can be used by DT and FA and the ADATA file can be used by FA and APA.

The load module is production-ready.

A SYSADATA file contains source and symbolics information about the program. See Table 3-12.

Table 3-12 HLASM assembler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
SYSADATA	ADATA	Yes		Yes	Yes(1)
LANGX	ADATA	Yes	Yes (1)	Yes (1)	

Notes:

1. Recommended for production and test

3.5 C and C++ (5694-A01) and later

This topic describes how to prepare programs written for z/OS XL C / C++. For general information, refer to:

<http://www.ibm.com/software/awdtools/czos>

We recommend for test environments, and optionally for production environments, that you perform a 2-stage compile. In the first compile step, INCLUDEs and macros are expanded, creating an Expanded Source file using the options and parameters TEST(ALL),NOOPT.

In the second compile step, create and save a compiler listing, and optionally a LANGX file, using the options and parameters NOTEST,AGGREGATE,NOIPA, LIST, NOOFFSET, SOURCE, STMT, XREF(FULL).

Alternatively, with C / C++ V1.5 or later, you can create and save a load module using the FORMAT(DWARF) compiler option. This module is then production ready yet can be used, in conjunction with the source listing, by Debug Tool.

The Expanded Source file can be used by Debug Tool and the compiler listing and LANGX file can be used by FA and APA.

This is considered a “testing-only” configuration because the load module generated with the TEST option has run-time overhead. The compiler places debug hooks at statements and path points and symbol tables are stored in the module, both of which make it much larger.

For production environments, and optionally test environments if you do not use Debug Tool), we recommend that you create and save a LANGX file. See Table 3-13.

Table 3-13 C and C++ compiler options

Source file	Compiler options	Production ready	DTU& AF	FA	APA
Expanded Source file	TEST(ALL)		Yes (2)		
DWARF	DEBUG(FORMAT(DWARF)), HOOK(LINE,NOBLOCK,PATH), SYMBOL,FILE(location)	Yes	Yes(1)		
Compiler listing	TEST(ALL),AGGREGATE,ATTR IBUTES(FULL),ESD,LIST,MAP, NEST,OPTIONS,SOURCE,STM T,XREF(FULL)			Yes	Yes
LANGX				Yes	Yes
Compiler listing	TEST(ALL),AGGREGATE,ATTR IBUTES(FULL),ESD,LIST,MAP, NEST,OPTIONS,SOURCE,STM T,XREF(FULL)	Yes		Yes (1)	Yes (1)
LANGX		Yes		Yes	Yes

Notes:

1. Recommended for production and test
2. Good option for test only

3.6 Starting Debug Tool automatically

- ▶ For some programs, you might want to start Debug Tool automatically when the program starts. This approach is strongly recommended for online IMS programs and DB2 stored procedures. It cannot be applied to OS/VS COBOL or Assembler programs, and should not be used for CICS applications.

You implement this functionality by link-editing a Debug Tool LE startup exit module into your application load module:

- ▶ For batch applications: EQADBCXT
- ▶ For online IMS applications: EQADICXT
- ▶ For DB2 stored procedures: EQADDCXT

The exits are in stored in the Debug Tool SEQAMOD load library. Because the exit adds a small amount of overhead when the program starts, you might or might not consider this to be production-ready.

The exit can optionally be link-edited into a copy of LE module CEEBINIT (for batch and IMS) or CEEPIPI (for DB2 Stored Procedures) to avoid linking the exit into your application module. Then you can STEPLIB or JOBLIB to access the LE module with the exit.



Using Debug Tool on CICS programs

The scenario described in this chapter provides an example of Debug Tool being used to find and fix a problem in a CICS transaction that had abnormally terminated. The programs associated with the failing transaction are coded in PL/I.

4.1 What happened?

A CICS transaction had abnormally terminated with an ABM0 abend code, as shown in Figure 4-1.

```
DISPLAY PATIENT HISTORY

ENTER PATIENT ID
999999999

DFHAC2206 17:53:15 CICSACB5 Transaction HCHI failed with abend ABM0. Updates
to local recoverable resources backed out.
```

Figure 4-1 Transaction abend

We used Debug Tool to determine the cause of the abend. If Fault Analyzer had been available, it would have activated and produced a real time analysis report showing the line of code that caused the abend.

Because it is a CICS transaction, a debugging profile had to be created. We did this using the CADP transaction. ENTERing CADP in a blank CICS screen returned this screen (Figure 4-2).

```

CADP      -      CICS Application Debugging Profile Manager      -      CICSACB5

All Debugging Profiles      (A=Activate, I=Inactivate, D=Delete, C=Copy)

  Owner   Profile  S Tran Program  Compile Unit  Applid  Userid  Term  Type
  -----  -----  - ----  -
- DNET161  ADC1ALL  I ADC1 *      *      CICSACB5 DNET161 *    Comp
- ADPOT10  ADPOT10  I *    EPSL02 *      *      *      *    *    Comp
- DNET100  APA2     I APA2 *      *      *      *    *    Comp
- CICSUSER APINER   I HCHI HCHIST *      *      CICSACB5 CICSUSER 0055 Comp
- DNET331  ASSEM    I ABND *      *      CICS*    DNET331 *    Comp
- DNET039  A1       A EPS* *      *      CICSACB5 DNET039 *    Comp
- DNET514  BSC      I *    EPSL02 *      *      *      *    *    Comp
- DNET514  BSC2     I *    EPSL03 *      *      *      *    *    Comp
- SYS029  CCCAECC0 I ECC0 *      *      CICSACB5 SYS029 *    Comp
- ADPOT23  CTGTEST  I *    EPSL03 *      *      *      DNET100 *    Comp
- ADPOT12  DAYSOLD  I *    EPSL02 *      *      *      ADPOT12 *    Comp
- ADPOT14  DAYSOLD  I *    EPSL02 *      *      *      ADPOT14 *    Comp
- ADPOT16  DAYSOLD  I *    EPSL02 *      *      *      ADPOT16 *    Comp
- DBA022  DBA022   I HCHI *      *      CICSACB5 DBA022 0019 Comp
- DBA022  DBA022A I CDAT *      *      CICSACB5 DBA022 0019 Comp
- DBA022  DBA022H I HCHI *      *      CICSACB5 DBA022 0019 Comp

96 profile(s). All profiles shown
Enter=Process PF1=Help 2=Filter 3=Exit 4=View 5=Create Comp 6=Create Java
8=Forward 9=Set display device 10=Edit 11=Sort
  
```

Figure 4-2 Existing CICS debug profiles

This was a list of all the CICS debugging profiles currently in the system. We had to create a new Compiled Debugging Profile to allow us to trap the offending program with Debug Tool. To do this, we pressed PF5 and then:

- ▶ We typed a unique profile name (HCHI71 in our example).
- ▶ We ENTERed the ID of the failing transaction (HCHI).
- ▶ We ENTERed * into the Program and Compile Unit fields.
- ▶ We changed the Test Level parameter to Error.
- ▶ We did not change any other fields.
- ▶ We pressed the Enter key.

The Debugging profile created message appeared in the bottom left-hand corner of the screen (Figure 4-3).

```
CADP      -      CICS Application Debugging Profile Manager      -      CICSACB5

Create Compiled Debugging Profile ==> HCHI71      for DBA071

CICS Resources To Debug (use * to specify generic values e.g. *, A*, AB*, etc.)
Transaction      ==> HCHI                               Applid  ==> CICSACB5
Program          ==> *                                 Userid  ==> DBA071
Compile Unit     ==> *                                 Termid  ==> 0012
                                                         Netname ==> TCP00012

Debug Tool Language Environment Options
Test Level       ==> Error                               (All,Error,None)
Command File     ==>
Prompt Level     ==> PROMPT
Preference File  ==>

Other Language Environment Options
==>
==>
==>
==>

Duplicate - use replace or change name
Enter=Create PF1=Help 2=Save options as defaults 3=Exit 10=Replace 12=Return
```

Figure 4-3 CADP screen

Pressing PF12 returned us to an updated list of all the debugging profiles, including the one we had just created.

We located this new profile and typed an A line command adjacent to it. We refer to commands entered on the command line as *primary* commands and to those entered in the left side column as *line* commands (Figure 4-4).

```

CADP          -      CICS Application Debugging Profile Manager      -      CICSACB5

All Debugging Profiles          (A=Activate, I=Inactivate, D=Delete, C=Copy)

  Owner   Profile  S Tran Program  Compile Unit  Applid  Userid  Term  Type
  _ ADPOT02 HCHI    I HCHI HCHIST  *          CICSACB5 ADPOT02 Z040 Comp
  _ DBA287  HCHI    I HCHI *        *          *        DBA287  *    Comp
  _ DNET070 HCHI    I HCHI *        *          CICSACB5 DNET070 Z042 Comp
  _ DNET100 HCHISTO I *    HCHIST  *          CICSACB5 DNET100 *    Comp
  _ DNET049 HCHI049 I HCHI *        *          CICSACB5 DNET049 Z040 Comp
  _ DNET422 HCHI22  I HCHI *        *          CICSACB5 DNET422 0082 Comp
  _ DNET100 HCHI24  I HCHI *        *          CICSACB5 DNET100 *    Comp
  _ SYS029  HCHI29  I HCHI HCHIST  *          *        SYS029  *    Comp
  _ DNET504 HCHI504 I HCHI HCHIST  *          CICSACB5 DNET504 Z042 Comp
  _ DNET620 HCHI620 I HCHI HCHIST  *          CICSACB5 DNET620 Z046 Comp
  A DBA071  HCHI71  I HCHI *        *          CICSACB5 DBA071 0012 Comp
  _ DNET481 HCHI81  I HCHI HCHIST  *          CICSACB5 DNET481 0083 Comp
  _ ADPOT23 HEALTH  I HCHI *        *          CICSACB5 ADPOT23 0036 Comp
  _ DNET331 HEALTHC I HCHI *        *          CICSACB5 DNET331 *    Comp
  _ DNET032 JAJ2    I CDAT *        *          CICSACB5 DNET032 Z002 Comp
  _ $EXAMPLE JAVA    I TR* *          *          PENFOLD*   Java
96 profile(s)
Enter=Process PF1=Help 2=Filter 3=Exit 4=View 5=Create Comp 6=Create Java
7=Back 8=Forward 9=Set display device 10=Edit 11=Sort

```

Figure 4-4 Updated CICS debug profiles

The profile was then activated by pressing the Enter key.

The Profile Manager displayed the following screen, providing information that identified the display device where Debug Tool would run (Figure 4-5).

```
CADP      -      CICS Application Debugging Profile Manager      -      CICSACB5

Set Compiled Debugging Display Device (checked at PROFILE activation time)

Debugging Display Device
Session Type          ==>  3270                      (3270,TCP)
3270 Display Terminal ==>  0012

TCP/IP Name Or Address
==>
==>
==>
==>
Port                  ==>  08000

Type of socket communication ==> Single              (Single, Multiple)

Display this panel on comp profile activation ==> YES

Enter=Save and return PF1=Help 3=Exit 12=Cancel
```

Figure 4-5 Debug display device detail

Pressing the Enter key then completed the profile activation. We then exited the Profile Manager by pressing PF3.

We then reran the failing transaction. It was now under the control of Debug Tool and stopped executing at the entry point to the first program invoked (Figure 4-6).

```

PL/I      LOCATION: HCHIST initialization
Command ==>
Scroll ==> PAGE
MONITOR  -+-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE:  HCHIST  ---1---+---2---+---3---+---4---+---5---+ LINE: 1 OF 246
***** TOP OF SOURCE *****
1  HCHIST: PROC (CA_PTR) OPTIONS(MAIN,REENTRANT) REORDER;
2  /*****
3  /* MAIN ROUTINE TO DISPLAY A PATIENT'S HISTORY */
4  /* - THIS ROUTINE CALL A ROUTINE WRITTEN IN C LANGUAGE */
5  /* WHICH WILL CALL A COBOL PROGRAM. */

LOG 0-----1-----2-----3-----4-----5-----6- LINE: 2 OF 5
0002 05/08/2008 6:09:14 PM
0003 5655-S17 and 5655-S16: Copyright IBM Corp. 1992, 2007
0004 EQA1872E An error occurred while opening file: DEVELP.TEST.GLBLPREF. The
0005 file may not exist, or is not accessible.
PF 1:?      2:STEP      3:QUIT      4:LIST      5:FIND      6:AT/CLEAR
PF 7:UP     8:DOWN     9:GO      10:ZOOM     11:ZOOM LOG 12:RETRIEVE

```

Figure 4-6 Debug session starts

The default debug screen shows three windows:

- ▶ The Monitor window displays current variable values. Just which variables are displayed is determined by commands that we issue to Debug Tool.
- ▶ The Source window displays program statements, highlighting the next statement that will be executed.
- ▶ The Log window displays a list of user commands, responses to user commands, and any messages for the user.

Note that in most instances, the error message in this screen does not indicate an error. You do not always require a global preferences file. By issuing a ZOOM command, or by positioning the cursor in any window and pressing PF10, we can toggle in and out of any window.

A fourth window, the Memory window, can be swapped in to replace the Log window. It is used to display the contents of storage.

We then issued a series of GO commands by repeatedly pressing PF9 to move through the executing transaction until we reached the point of failure. Debug Tool stops at the start of each program it encounters. Debug Tool stopped at entry to program HCBLDDL multiple times. We used the NAMES EXCLUDE command to advise Debug Tool not to stop when it encountered an HCBLDDL program (Figure 4-7).

```

COBOL   LOCATION: HCBLDDL ENTRY
Command ==>                               Scroll ==> PAGE
MONITOR --1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCBLDDL --1-----2-----3-----4-----5-----6- LINE: 2 OF 49
2      PROGRAM-ID. HCBLDDL.
3      ENVIRONMENT DIVISION.
4      CONFIGURATION SECTION.
5      SOURCE-COMPUTER. IBM-370.
6      OBJECT-COMPUTER. IBM-370.
7      INPUT-OUTPUT SECTION.

LOG 0--1-----2-----3-----4-----5-----6 LINE: 8 OF 11
0008  GO ;
0009  GO ;
0010  GO ;
0011  NAMES EXCLUDE CU HCBLDDL ;

PF 1:?      2:STEP      3:QUIT      4:LIST      5:FIND      6:AT/CLEAR
PF 7:UP     8:DOWN     9:GO      10:ZOOM     11:ZOOM LOG  12:RETRIEVE

```

Figure 4-7 Stepping through the debug session

Debug Tool eventually stopped at an EXEC CICS SEND MAP statement. The error message for the ABMO abend appeared in the LOG window (Figure 4-8).

```

PL/I      LOCATION: HCHIST :> 214
Command ==>
                                                    Scroll ==> PAGE
MONITOR  -+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 214 OF 246
214 EXEC CICS
215 SEND MAP (HCHIST_DETAIL_MAP) MAPSET('HCMAPS')
216 FROM(HIDETLO) ERASE CURSOR;
217 W_MAP_TO_RECV = HCHIST_DETAIL_MAP;
218 RETURN;
219 END SEND_PATIENT_HISTORY_DETAIL_MAP;

LOG 0-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+----- LINE: 15 OF 18
0015 CEE35I is a severity or class 4 condition.
0016 The operating system has generated the following message:
0017 CEE3250C The system or user abend ABMO was issued.
0018 The current location is HCHIST :> HCHIST :> 214.
PF 1:?          2:STEP          3:QUIT          4:LIST          5:FIND          6:AT/CLEAR
PF 7:UP         8:DOWN          9:GO           10:ZOOM         11:ZOOM LOG    12:RETRIEVE

```

Figure 4-8 Debug session halted after abend

The name of the map is stored in the HCHIST_DETAIL_MAP variable.

To see the contents of this variable, we issued the LI HCHIST_DETAIL_MAP command. The result was displayed in the LOG window (Figure 4-9).

```

PL/I      LOCATION: HCHIST :> 214
Command ==>
                                                    Scroll ==> PAGE
MONITOR  +-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 214 OF 246
214 EXEC CICS
215     SEND MAP (HCHIST_DETAIL_MAP) MAPSET('HCMAPS')
216     FROM (HIDETLO) ERASE CURSOR;
217     W_MAP_TO_RECV = HCHIST_DETAIL_MAP;
218     RETURN;
219 END SEND_PATIENT_HISTORY_DETAIL_MAP;

LOG 0-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+----- LINE: 17 OF 20
0017 CEE3250C The system or user abend ABM0 was issued.
0018 The current location is HCHIST :> HCHIST :> 214.
0019 LIST HCHIST_DETAIL_MAP ;
0020 HCHIST_DETAIL_MAP = '00 '
PF 1:?          2:STEP          3:QUIT          4:LIST          5:FIND          6:AT/CLEAR
PF 7:UP         8:DOWN         9:GO           10:ZOOM        11:ZOOM LOG    12:RETRIEVE

```

Figure 4-9 Checking variable content

The current value of the HCDIST_DETAIL_MAP variable was not a valid value.

HCHIST is the name of the program that failed. To start our debugging, we checked all the statements that referenced the HCHIST_DETAIL_MAP variable to determine if any of them modified it.

We issued the ZOOM SOURCE command, followed by the F HCHIST_DETAIL_MAP FIRST command to find the first reference to this variable. We found that it was initialized with the value HIDETL when it was defined. This is the name of the BMS map that the program should display (Figure 4-10).

```

PL/I    LOCATION: HCHIST :> 214
Command ==>                               Scroll ==> PAGE
SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 50 OF 246
50      05 HCHIST_DETAIL_MAP PIC '(7)X' INIT('HIDETL');
51      05 HCHIST_DONE_MAP  PIC '(7)X' INIT('HISTDON');
52

```

Figure 4-10 Locating variable reference

We repeated the FIND command using PF5 multiple times to locate other references to the variable. Three were found, in statement lines 106, 215, and 217 (see Figure 4-11 and Figure 4-12).

```

PL/I    LOCATION: HCHIST :> 214
Command ==>                               Scroll ==> PAGE
SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 106 OF 246
106     WHEN (W_MAP_TO_RECV = HCHIST_DETAIL_MAP) DO;
107     CALL SEND_PATIENT_HISTORY_DETAIL_MAP;
108     END;

```

Figure 4-11 Locating variable reference

```

PL/I    LOCATION: HCHIST :> 214
Command ==>                               Scroll ==> PAGE
SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 215 OF 246
215     SEND MAP (HCHIST_DETAIL_MAP) MAPSET('HCMAPS')
216     FROM(HIDETLO) ERASE CURSOR;
217     W_MAP_TO_RECV = HCHIST_DETAIL_MAP;

```

Figure 4-12 Locating variable reference

None of these statements appeared to modify the variable — additional investigation was required. We issued the GO command, causing CICS to handle the abend and the Debug Tool session to end.

We knew that the abend was occurring in program HIHIST. We could therefore concentrate our debugging here. To do this, we re-ENTERed the CADP transaction and pressed PF10 to update the profile and change the value of the Program field from * to HCHIST. Pressing PF10 again caused the profile manager to save the updated profile (Figure 4-13).

```

CADP      -      CICS Application Debugging Profile Manager      -      CICSACB5

Create Compiled Debugging Profile ==> HCHI71      for DBA071
activated by DBA071

to 0012
CICS Resources To Debug (use * to specify generic values e.g. *, A*, AB*, etc.)
Transaction      ==> HCHI      Applid      ==> CICSACB5
Program          ==> HCHIST    Userid     ==> DBA071
Compile Unit     ==> *        Termid    ==> 0012
Netname         ==> TCP00012

Debug Tool Language Environment Options
Test Level       ==> Error      (All,Error,None)
Command File     ==>
Prompt Level     ==> PROMPT
Preference File  ==>

Other Language Environment Options
==>
==>
==>
==>
Debugging profile replaced
Enter=Create PF1=Help 2=Save options as defaults 3=Exit 10=Replace 12=Return

```

Figure 4-13 Re-specifying the debug profile

We exited the profile manager by pressing PF3.

We had learned that the cause of the transaction failure was an invalid value in the HCHIST_DETAIL_MAP variable. No statements in the program appeared to change the content of this variable from its initialized value of "HIDETL". Using breakpoints, we can cause Debug Tool to halt its processing when specific events occur. We set a breakpoint to occur when the value of the HCHIST_DETAIL_MAP variable changed.

We ran the failing transaction again. Debug Tool stopped it at the entry point of program HCHIST. To start program execution, we issued the STEP command. This brought us into the program and did the required initialization (Figure 4-14).

```

PL/I      LOCATION: HCHIST :> HCHIST ENTRY
Command ==>
Scroll ==> PAGE
MONITOR  -+----1-+----2-+----3-+----4-+----5-+----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5---+ LINE: 1 OF 246
***** TOP OF SOURCE *****
1  HCHIST: PROC (CA_PTR) OPTIONS(MAIN,REENTRANT) REORDER;
2  /*****
3  /* MAIN ROUTINE TO DISPLAY A PATIENT'S HISTORY */
4  /* - THIS ROUTINE CALL A ROUTINE WRITTEN IN C LANGUAGE */
5  /* WHICH WILL CALL A COBOL PROGRAM. */

LOG 0-----1-----2-----3-----4-----5-----6- LINE: 3 OF 6
0003 5655-S17 and 5655-S16: Copyright IBM Corp. 1992, 2007
0004 EQA1872E An error occurred while opening file: DEVELP.TEST.GLBLPREF. The
0005      file may not exist, or is not accessible.
0006 STEP ;
PF 1:?      2:STEP      3:QUIT      4:LIST      5:FIND      6:AT/CLEAR
PF 7:UP     8:DOWN     9:GO      10:ZOOM     11:ZOOM LOG  12:RETRIEVE

```

Figure 4-14 Debug session halted at entry point

We could then set a breakpoint.

We issued the AT CHANGE HCHIST_DETAIL_MAP command. This would cause program execution to stop when the value of the HCHIST_DETAIL_MAP variable changed. The GO command was then issued (Figure 4-15).

```

PL/I LOCATION: HCHIST :> 196
Command ==>
MONITOR -----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST -----1-----2-----3-----4-----5- LINE: 196 OF 246
196 END;
197 IF (HISTORY_DETAIL_RC = 0) THEN DO;
198 IF (HISTORY_DETAIL_LINE_COUNT = 0) THEN DO;
199 OUT_MSG1 = 'PATIENT HAS NO HISTORY.';
200 OUT_MSG2 = ' ';
201 END;
LOG 0-----1-----2-----3-----4-----5-----6 LINE: 8 OF 11
0008 file may not exist, or is not accessible.
0009
0010 STEP ;
0011 AT CHANGE HCHIST_DETAIL_MAP ;
0012 GO ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: AT/CLEAR
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: RETRIEVE

```

Figure 4-15 Stopping program execution

Debug Tool stopped program execution at an END statement (line 196). We issued an UP 3 command to view the statements immediately preceding the END.

The program code showed that we were in a DO loop. The value of HCHIST_DETAIL_MAP had changed in the previous statement (line 194). See Figure 4-16.

```

PL/I      LOCATION: HCHIST :> 196
Command ==>
                                                    Scroll ==> PAGE
MONITOR  -+-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST ---1-----2-----3-----4-----5--- LINE: 193 OF 246
193      DO DETAIL_LINE_INDEX=1 TO HISTORY_DETAIL_LINE_COUNT;
194          PATHIST_DETAIL_LINE_TEXT (DETAIL_LINE_INDEX) =
195              HISTORY_DETAIL_MAP_LINE (DETAIL_LINE_INDEX);
196      END;
197      IF (HISTORY_DETAIL_RC = 0) THEN DO;
198          IF (HISTORY_DETAIL_LINE_COUNT = 0) THEN DO;
LOG 0-----1-----2-----3-----4-----5-----6 LINE: 8 OF 11
0008      file may not exist, or is not accessible.
0009      STEP ;
0010      AT CHANGE HCHIST_DETAIL_MAP ;
0011      GO ;
PF 1:?          2:STEP          3:QUIT          4:LIST          5:FIND          6:AT/CLEAR
PF 7:UP         8:DOWN         9:GO           10:ZOOM         11:ZOOM LOG    12:RETRIEVE
MA a                                                    02/015

```

Figure 4-16 Debug session stopped when value of variable changed

At this point we checked the value of the `DETAIL_LINE_INDEX` variable by issuing an `LI DETAIL_LINE_INDEX` command. This told us that the change to the contents of `HCHIST_DETAIL_MAP` occurred when the value of `DETAIL_LINE_INDEX` equaled 18 (Figure 4-17).

```

PL/I      LOCATION: HCHIST :> 196
Command ==>
                                           Scroll ==> PAGE
MONITOR  -+-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE:  HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 193 OF 246
193      DO DETAIL_LINE_INDEX=1 TO HISTORY_DETAIL_LINE_COUNT;
194          PATHIST_DETAIL_LINE_TEXT (DETAIL_LINE_INDEX) =
195              HISTORY_DETAIL_MAP_LINE (DETAIL_LINE_INDEX);
196      END;
197      IF (HISTORY_DETAIL_RC = 0) THEN DO;
198          IF (HISTORY_DETAIL_LINE_COUNT = 0) THEN DO;
LOG 0---+---1---+---2---+---3---+---4---+---5---+---6 LINE: 7 OF 10
0007 AT CHANGE HCHIST_DETAIL_MAP ;
0008 GO ;
0009 LIST DETAIL_LINE_INDEX ;
0010 DETAIL_LINE_INDEX =      18
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP      8: DOWN      9: GO       10: ZOOM     11: ZOOM LOG  12: RETRIEVE

```

Figure 4-17 Checking value of `DETAIL_LINE_INDEX`

We ended this debug session by issuing a `QUIT ABEND` command. We had to determine when and how the value of `DETAIL_LINE_INDEX` was set to 18.

Again we started the failing transaction, HCHI. When Debug Tool gained control at the start of program HCHIST we STEPPed into the executing code. We then set a breakpoint to come into effect when the value of `DETAIL_LINE_INDEX` changed, knowing that we were not really interested until it approaches a value of 18.

We issued the command `AT CHANGE DETAIL_LINE_INDEX` and pressed the Enter key. Debug Tool responded with the message, `Current command is incomplete, pending more input`. We then ENTERed the remainder of the command, `IF DETAIL_LINE_INDEX <= 16 THEN GO`.

It was necessary to ENTER the breakpoint command in two pieces because it was too long to fit on the command line. The hyphen at the end of the first command ensured that the second portion was a continuation of the first.

We then issued the GO command.

Debug Tool stopped processing at source code line 194, indicating that the breakpoint had been triggered by the previous statement (Figure 4-18).

```
PL/I      LOCATION: HCHIST :> 194
Command ==>                               Scroll ==> PAGE
MONITOR  +-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE:  HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 194 OF 246
194     PATHIST_DETAIL_LINE_TEXT (DETAIL_LINE_INDEX) =
195     HISTORY_DETAIL_MAP_LINE (DETAIL_LINE_INDEX);
196     END;
197     IF (HISTORY_DETAIL_RC = 0) THEN DO;
198     IF (HISTORY_DETAIL_LINE_COUNT = 0) THEN DO;
199     OUT_MSG1 = 'PATIENT HAS NO HISTORY.'

LOG 0-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+----- LINE: 11 OF 14
0011  AT CHANGE_DETAIL_LINE_INDEX
0012  IF DETAIL_LINE_INDEX <= 16 THEN
0013  GO ;
0014  GO ;

PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG  12: RETRIEVE
```

Figure 4-18 Debug session stops when `DETAIL_LINE_INDEX > 18`

Issuing an UP 3 command repositioned the information in the source window. We then issued the following commands to allow close monitoring of the two variables involved:

```
MON LI DETAIL_LINE_INDEX
MON LI HCHIST_DETAIL_MAP
```

This caused current values of these two variables to be displayed in the MONITOR window. At this time we saw that both values were valid (Figure 4-19).

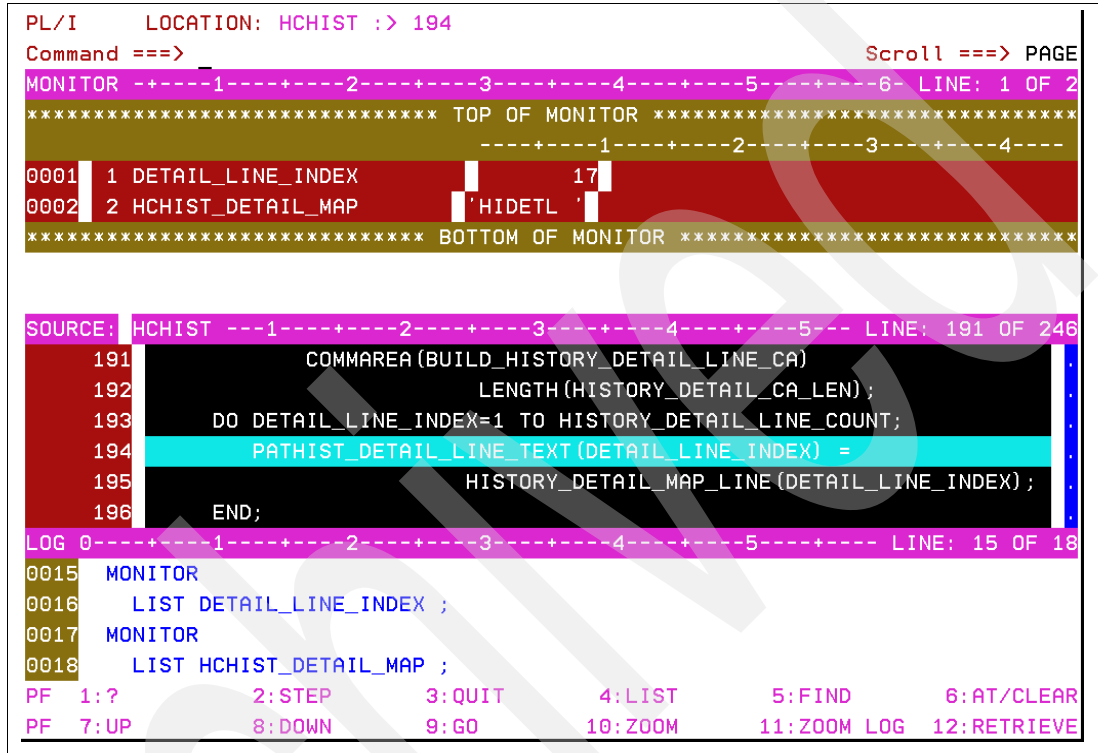


Figure 4-19 Checking variable contents

We were now able to STEP through the code, monitoring changes to the variables as we did so. When saw that the value of `DETAIL_LINE_INDEX` reached 18 the content of `HCHIST_DETAIL_MAP` became invalid. This suggested we could have a storage overlay problem, possibly caused by stepping outside the bounds of an array (Figure 4-20).

```

PL/I      LOCATION: HCHIST :> 196
Command ==>
Scroll ==> PAGE
MONITOR  -+-----1-----2-----3-----4-----5-----6- LINE: 1 OF 2
***** TOP OF MONITOR *****
-----+-----1-----2-----3-----4-----
0001  1  DETAIL_LINE_INDEX          18
0002  2  HCHIST_DETAIL_MAP          '00
***** BOTTOM OF MONITOR *****

SOURCE:  HCHIST ---1-----2-----3-----4-----5--- LINE: 191 OF 246
191      COMMAREA (BUILD_HISTORY_DETAIL_LINE_CA)
192          LENGTH (HISTORY_DETAIL_CA_LEN);
193      DO DETAIL_LINE_INDEX=1 TO HISTORY_DETAIL_LINE_COUNT;
194          PATHIST_DETAIL_LINE_TEXT (DETAIL_LINE_INDEX) =
195              HISTORY_DETAIL_MAP_LINE (DETAIL_LINE_INDEX);
196      END;

LOG 0-----1-----2-----3-----4-----5----- LINE: 18 OF 21
0018  LIST HCHIST_DETAIL_MAP ;
0019  STEP ;
0020  STEP ;
0021  STEP ;

PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN     9: GO       10: ZOOM     11: ZOOM LOG  12: RETRIEVE

```

Figure 4-20 Variable content changes to invalid value

We next examined the definition of the PATHIST_DETAIL_LINE_TEXT variable by issuing a FIND PATHIST_DETAIL_LINE_TEXT, then scrolling back by issuing an UP 4 command. This showed us that this variable was a sub-element in an array of 15 elements (Figure 4-21).

```

PL/I      LOCATION: HCHIST :> 196
Command ==>                               Scroll ==> PAGE
MONITOR  +-----1-----2-----3-----4-----5-----6- LINE: 1 OF 2
***** TOP OF MONITOR *****
-----1-----2-----3-----4-----
0001 1  DETAIL_LINE_INDEX          18
0002 2  HCHIST_DETAIL_MAP          '00
***** BOTTOM OF MONITOR *****

SOURCE:  HCHIST ---1-----2-----3-----4-----5--- LINE: 42 OF 246
42      05  PATHIST_DETAIL_LINE_OF_MAP (15),
43      15  PATHIST_DETAIL_LINE
44      25  PATHIST_DETAIL_LINE_LEN FIXED BIN(15),
45      25  PATHIST_DETAIL_LINE_ATTR PIC '(01)X',
46      25  PATHIST_DETAIL_LINE_TEXT PIC '(78)X';
47

LOG 0-----1-----2-----3-----4-----5----- LINE: 18 OF 21
0018  LIST HCHIST_DETAIL_MAP ;
0019  STEP ;
0020  STEP ;
0021  STEP ;

PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN     9: GO       10: ZOOM     11: ZOOM LOG  12: RETRIEVE

```

Figure 4-21 Checking variable definition

Previously we had learned that the end of the DO loop was coded to occur when the index value equalled the value of HISTORY_DETAIL_LINE_COUNT. We checked the definition of this variable to understand what it had been initialized with, using the command, F HISTORY_DETAIL_LINE_COUNT FIRST. There is no definition of this variable, so it has been defined automatically and never initialized.

By issuing a MON LI HISTORY_DETAIL_LINE_COUNT, we checked its value (Figure 4-22).

```

PL/I      LOCATION: HCHIST :> 59
Command ==>
Scroll ==> PAGE
MONITOR  -+----1-+----2-+----3-+----4-+----5-+----6- LINE: 1 OF 1
***** TOP OF MONITOR *****
-----1-----2-----3-----4-----
0001 1 HISTORY_DETAIL_LINE_COUNT 16448
***** BOTTOM OF MONITOR *****

SOURCE: HCHIST ---1---+---2---+---3---+---4---+---5--- LINE: 193 OF 246
193 DO DETAIL_LINE_INDEX=1 TO HISTORY_DETAIL_LINE_COUNT;
194     PATHIST_DETAIL_LINE_TEXT (DETAIL_LINE_INDEX) =
195     HISTORY_DETAIL_MAP_LINE (DETAIL_LINE_INDEX);
196 END;
197 IF (HISTORY_DETAIL_RC = 0) THEN DO;
198     IF (HISTORY_DETAIL_LINE_COUNT = 0) THEN DO;
LOG 0-----1-----2-----3-----4-----5-----6- LINE: 6 OF 9
0006 STEP ;
0007 STEP ;
0008 MONITOR
0009 LIST HISTORY_DETAIL_LINE_COUNT ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG  12: RETRIEVE

```

Figure 4-22 Checking variable content

The value of the HISTORY_DETAIL_LINE_COUNT was 16448, well outside of the number of elements in the array. No wonder we had a problem!

Correct definition and initialization of the HISTORY_DETAIL_LINE_COUNT variable was the resolution of our problem.

Scenario for PD Tools with RDz

For the purposes of this chapter, we assume that the PD Tools and RDz have already been installed and customized.

In this scenario, we are using a subset of the PD Tools to demonstrate how the tools work together to quickly and efficiently detect an error, correct it, and test it. We do that with Fault Analyzer, Debug Tool, and File Manager, all under the umbrella of RDz. The same scenario can be accomplished in batch (green screen) without the use of RDz.

We are not providing source code, data, or any other resources for the reader to exactly duplicate this scenario. However, the steps taken and screen captures provided should be sufficient when used as a guide for performing similar tasks in your environment. We are also assuming that you have completed the setup procedures to establish connection to the mainframe, and gained access to the data sets required.

5.1 Submitting the job

The first step in this scenario, as well as most processes, is to submit your JCL. We access the JCL through the Remote Systems Connection, in our case DEMOMVS, and locate the JCL in the MVS files (Figure 5-1).

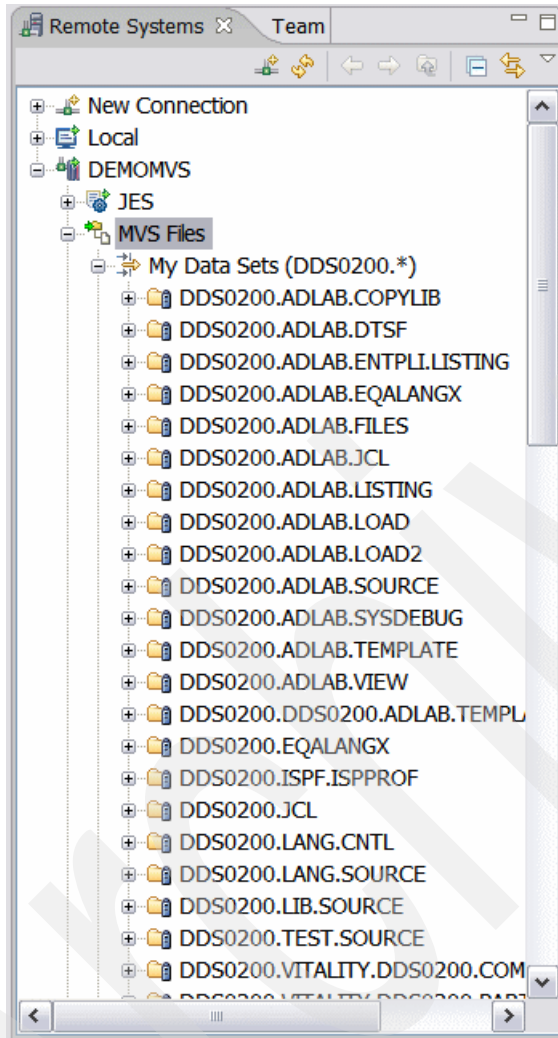


Figure 5-1 Remote Systems view

After the JCL is located, we can submit the Job, Right-clicking on the JCL brings up the menu for actions that can be taken on the JCL. Select **Submit** (Figure 5-2).

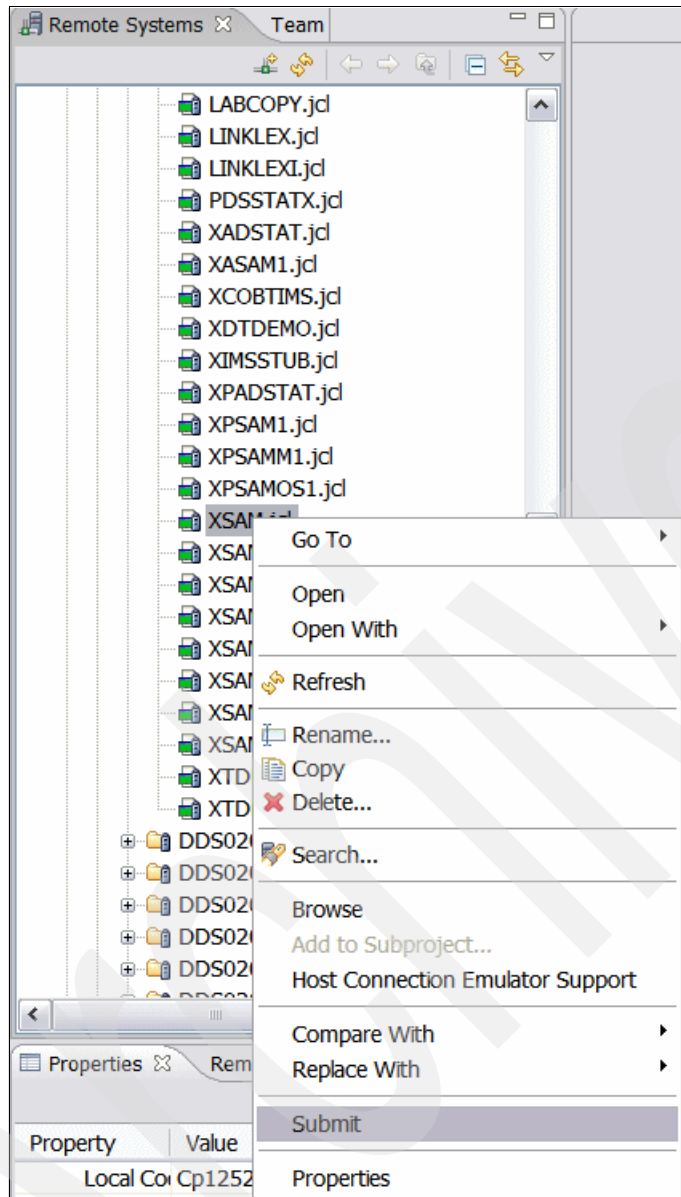


Figure 5-2 Submit JCL from Remote Systems view

Validating your job submission return code

By scrolling to the top of the box under the **Remote System** tab, locate the + next to the item labeled JES. Click on the + next to JES, then click on the + next to the filter that contains your submitted jobs, in our case, the **My Jobs** filter. There you should be able to find the job that was submitted; if not, you might have to right-click on the JES and select **Refresh**.

Properties window

As you use RDz, notice that as you highlight an item, the box with the Properties tab in the lower left hand corner will automatically populate with information for whatever item is highlighted. If we highlight our batch job output, we can see the properties for this item. You might have to scroll down the Properties box to find the Return Code for this job. You should expect to see a return code of 0C7 (Figure 5-3).

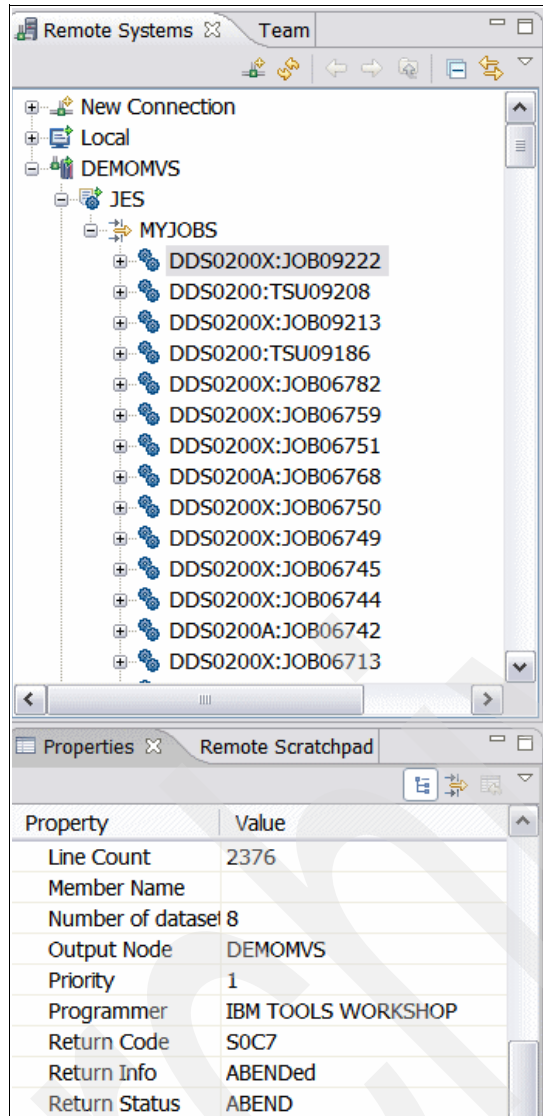


Figure 5-3 Properties window

Now that you know you have an abend, you are ready to begin analyzing the Fault. Double-click on the JES output, and the results will appear in the box next to the Remote System Explorer, under the tab labeled **jobname.out**, where jobname is the label name of the JES output. Look for the line that looks similar to +IDI0003I Fault ID F01385 assigned in history file FAULTANL.V8R1.HIST (Figure 5-4).

```

JOB09222 +IDI0002I Module SAM2, program SAM2, source line # 89: Abend S0C7 (Dat
JOB09222 +IDI0003I Fault ID F01405 assigned in history file FAULTANL.V8R1.HIST
JOB09222 IEF450I DDS0200X RUNSAM1 ABEND=S0C7 U0000 REASON=00000007 805
          TIME=14.47.13

JOB09222 -
JOB09222 -JOBNAME  STEPNAM  PROCSTEP   RC   EXCP   CPU   SRB   CLOCK  SERV
JOB09222 -DDS0200X          RUNSAM1  *S0C7   1130   .00   .00   3.13   9325
JOB09222 IEF404I DDS0200X - ENDED - TIME=14.47.13
JOB09222 -DDS0200X ENDED.  NAME-IBM TOOLS WORKSHOP  TOTAL CPU TIME= .00 TOT
JOB09222 $HASP395 DDS0200X ENDED

```

Figure 5-4 JES output view

Using the CTRL-F key combination brings up a find menu that can be used to find this line by finding the string “hist”. If you highlight the history file name (ADPOT00.HIST) in this example, we can use this in specifying the history file we will be using in the next section.

Analyzing 0C7 with Fault Analyzer

Now we will start to use the Fault Analyzer perspective. Switch perspectives to **Fault Analyzer**, and right-click on **FA Artefacts** in the top left box and select the **Add New History File** option from the context menu. A pop-up box will appear; in the input line labeled **History File Name**, enter the name of the history file found in the job output. When the history file is specified, click the **OK** button (Figure 5-5).

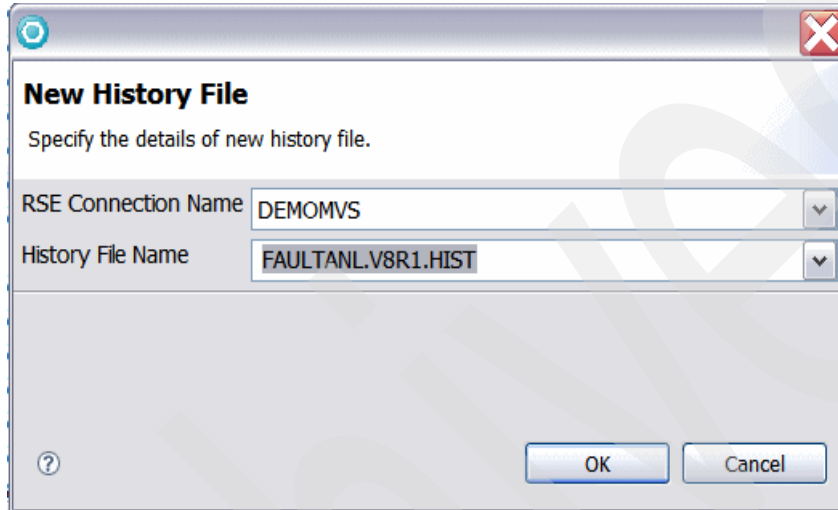


Figure 5-5 New History File pop-up

This will add the history file name under the FA Artefacts label. Right-click on the history file and select **Populate/Refresh history file** from the context menu. In the bottom right of the screen, you will see a matrix of the most recent abends. One of them should be the 0C7 just produced from our batch job. To start analyzing the problem, right-click on our abend and select **Browse Report** from the context menu (Figure 5-6).

Fault_ID	Job/Tran	User_ID	Sys/Job	Abend	I_Abend	Job_ID	Jobname
F01405	DDS0200X	DDS0200	DEMOMVS	S0C7	S0C7	JOB09222	DDS0200X
F01404	DNET424R	DNET424	DEMOMVS	S0C4	S0C4	OB09098	DNET424R
F01403	FM0024B	DNET047	DEMOMVS	S52	S52	OB08837	FM0024B
F01402	FM0024B	DNET047	DEMOMVS	S52	S52	OB08778	FM0024B
F01401	DNET775Q	DNET775	DEMOMVS	S0C4	S0C4	OB08345	DNET775Q
F01400	DNET775Q	DNET775	DEMOMVS	S0C4	S0C4	OB08342	DNET775Q
F01399	CARENVEX	DNET152	DEMOMVS	S0C4	S0C4	OB07771	CARENVEX
F01398	CARENVEX	DNET152	DEMOMVS	S0C4	S0C4	JOB07749	CARENVEX
F01397	CARENVEX	DNET152	DEMOMVS	S0C4	S0C4	JOB07653	CARENVEX

Figure 5-6 Contents of Fault History File

When the report comes up, you can see all the tools that are available in the Fault Analyzer perspective. The FA Artefact in the top left allows us access one to many history files from one or many mainframes. Below this box is the outline that can make navigating the history file quick and easy. The bottom section shows the list of faults and details about the faults in the columns. These columns can be customized and sorted to make finding a specific abend quick and easy. Finally, the FA report is in the main part of the screen that contains the actual Fault History details.

The Fault History details are presented in five tabs:

- ▶ The Initial tab gives a summary of the problem (Figure 5-7).
- ▶ The Event Summary tab shows all the events (the call path) leading up to the problem and indicates in which event the original problem occurs.
- ▶ The Abend Job Information tab contains all the information about this specific abend (similar to the information shown in the abend matrix in the bottom section). In addition, it has the runtime environment settings and link edit information for the problem program.
- ▶ The System-Wide Information tab has control blocks that pertain to the problem. This is specifically important when the abends occur in a subsystem such as CICS, IMS or DB2, because this section will display values for the control blocks for the specific subsystems.
- ▶ Finally, the Misc. Information tab contains the specifics about how Fault Analyzer is invoked and some of the customizations used.

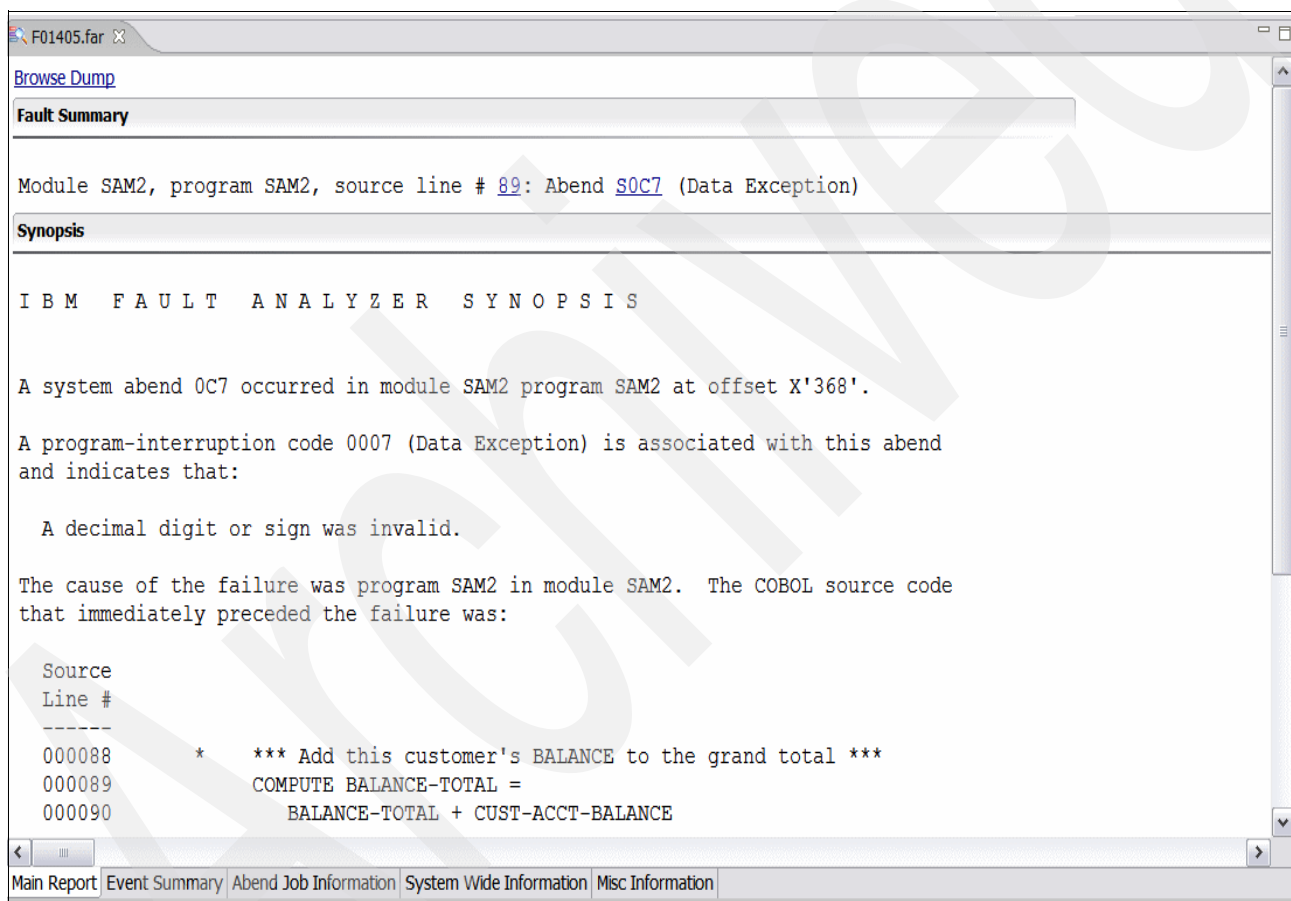


Figure 5-7 FA Fault Summary

To start analyzing the abend, we use the Main Report tab. All items underlined and in blue are links to other areas of the report. Clicking on the **SOC7** abend invokes the Lookup part of the Fault Analyzer tool. While the summary page gives a brief explanation of the abend code, Lookup gives a full explanation. Lookup can be used independent of the rest of the fault analyzer tool.

In Figure 5-7 on page 100, the Fault Summary page shows the COMPUTE statement on line 89. Further down, the variables used on line 89 and the definitions within the program are shown, and at the bottom, the values of those variables are shown. Notice that the value of CUST-ACCTBALANCE is shown in hex because the data is not a valid packed number. This is also flagged with “*** Invalid Numeric Data ***” text denoting that this value caused the problem (Figure 5-8).

```

The COBOL source code for data fields involved in the failure:

Source
Line #
-----
000059          05  CUST-ACCT-BALANCE          PIC S9(7)V99  COMP-3.
000066          05  BALANCE-TOTAL              PIC S9(7)V99  COMP-3.

Data field values at time of abend:

BALANCE-TOTAL      = 10948.44
CUST-ACCT-BALANCE = X'7C7B5B6C50' *** Invalid numeric data ***

```

[Main Report](#) | [Event Summary](#) | [Abend Job Information](#) | [System Wide Information](#) | [Misc Information](#)

Figure 5-8 FA summary data fields involved in abend

Clicking on the **Event Summary** tab, we can see the call path. This tells us that the problem occurred in a program named SAM2, and the program SAM2 is called by SAM1. Scrolling down in this part of the history report, we would see all the events within the report as well as that the Event containing the problem is already expanded and highlighted. Scrolling down to view event 3, you can see the same type of information as in the Summary tab, but in a lot more detail.

At this point, we know enough to start a debug session to continue our discovery. We know that the problem occurs in program SAM2 with a variable named CUST-ACCT-BALANCE, and the value in this variable is X'7C7B5B6C50' (Figure 5-9).

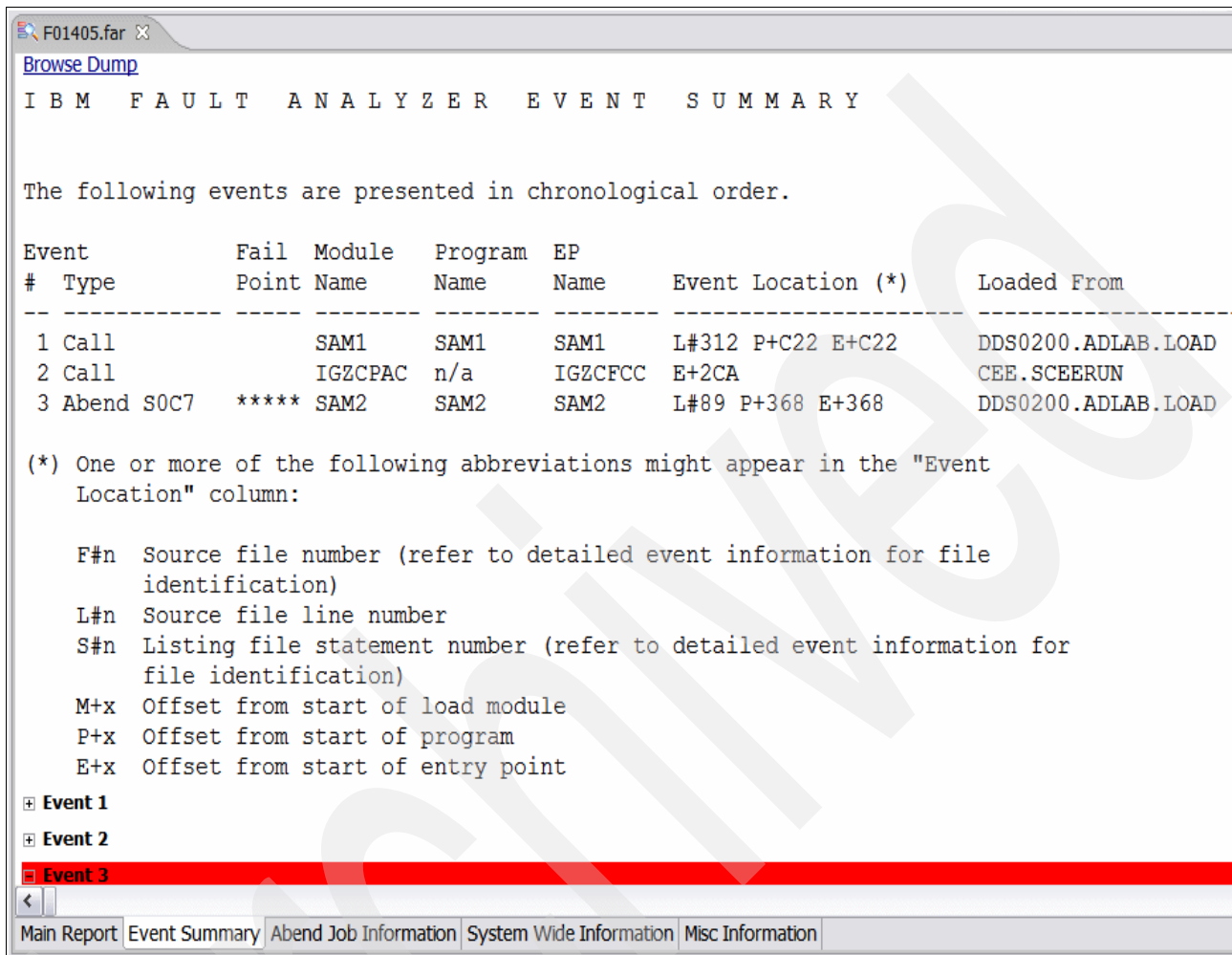


Figure 5-9 Fault Analyzer Event Summary

Debugging the program with OC7 With Debug Tool

In order to use an interactive debugger, we have to tell the debugger where to interact with us (the end user). We could use a VTAM screen (also known as a green screen) but this is not necessary because we can route the interactive debug session to RDz. To do this, you would go to the debug perspective and ensure that RDz is listening for an interactive debug session and determine the IP address of our client to direct the debug session to our RDz client.

To switch to the debug perspective, use the Window pull-down (**Window** → **Open Perspective** → **Other** → **Debug**).

In the Debug Perspective, the top left screen will have an icon that looks like an inverted tee. If the RDz is listening the ICON will be green, if not, it will be red as displayed (Figure 5-10).

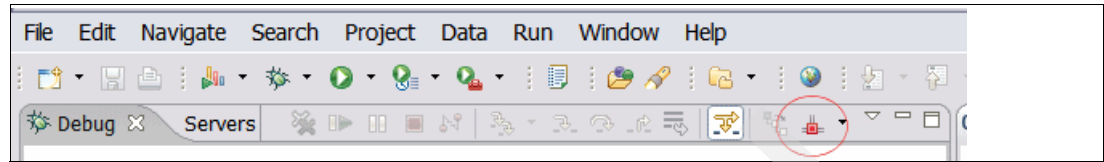


Figure 5-10 Debug Listening Icon

If RDz is not listening (as shown) use the pull-down and select **Start listening on port: ####**, where #### is the port number, usually 8001.

To have the debug session come to RDz, we have to tell the running job to have the interactive debug session come to our client IP address. To determine the client the IP address pull down the same menu and select **Get Workstation IP...** To make it easy, you can highlight the IP address and copy for later use (Figure 5-11).

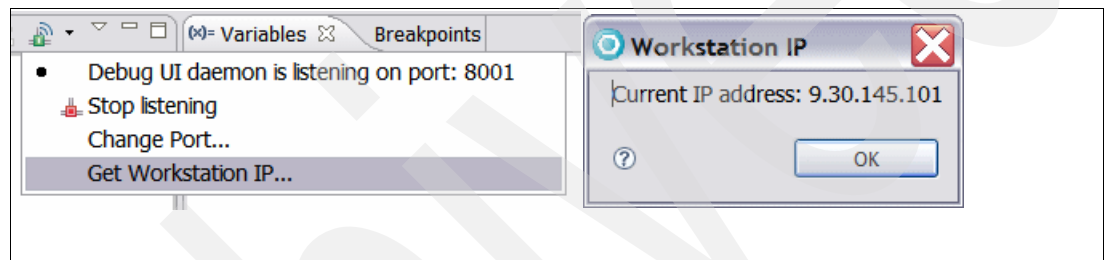


Figure 5-11 Get Workstation IP

To start the debug session, we want to modify the JCL to tell the runtime to use the debugger. We go back to the Remote System Explorer, and navigate back the XSAM.jcl member of the PDS "userid. ADLAB.JCL". We double-click on the XSAM.jcl member so we can edit this member. You must prepare your JCL with debug test parameters. Add the TCPIP parameter and IP address to reflect the IP number discovered by getting the Workstation IP address above (Figure 5-12).

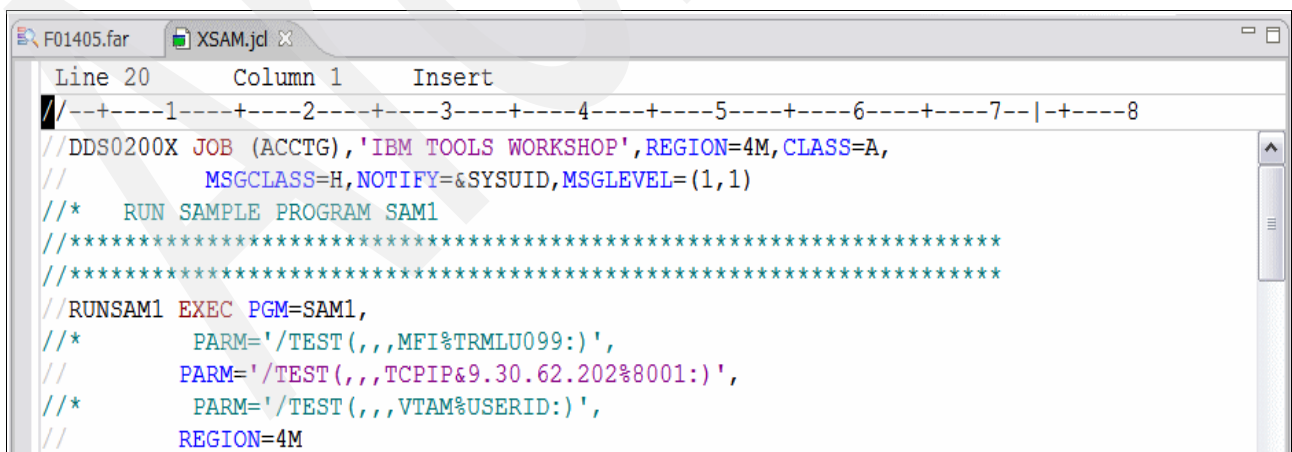


Figure 5-12 JCL modification

After the JCL change, save the source by right-clicking within the JCL source and selecting **Save** from the context menu. After saving, right-click within the source once more and this time select **Submit** within the context menu (Figure 5-13).

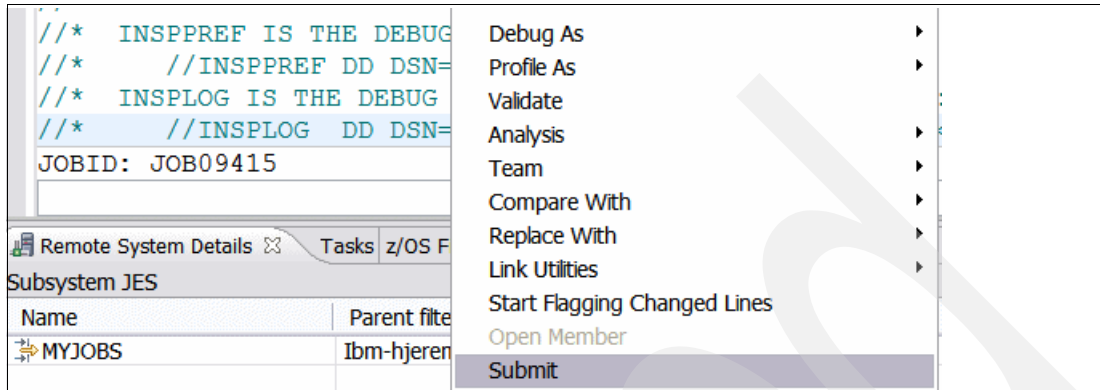


Figure 5-13 Submit JCL from editor

When the debug session starts, a pop-up menu is presented, titled Confirm Perspective Switch, to validate that you want to change to the debug perspective. Click the **Yes** button and you will enter the Debug Perspective.

From the Fault Analyzer report, we know the problem occurs in SAM2, so we should navigate to this program. In this example, we will set a deferred breakpoint for the load module of SAM2 with the entry point of SAM2. In the top right screen, select the **Breakpoints** tab. Within the Breakpoints screen, right-click and select **Add Breakpoint** → **Entry** (Figure 5-14).

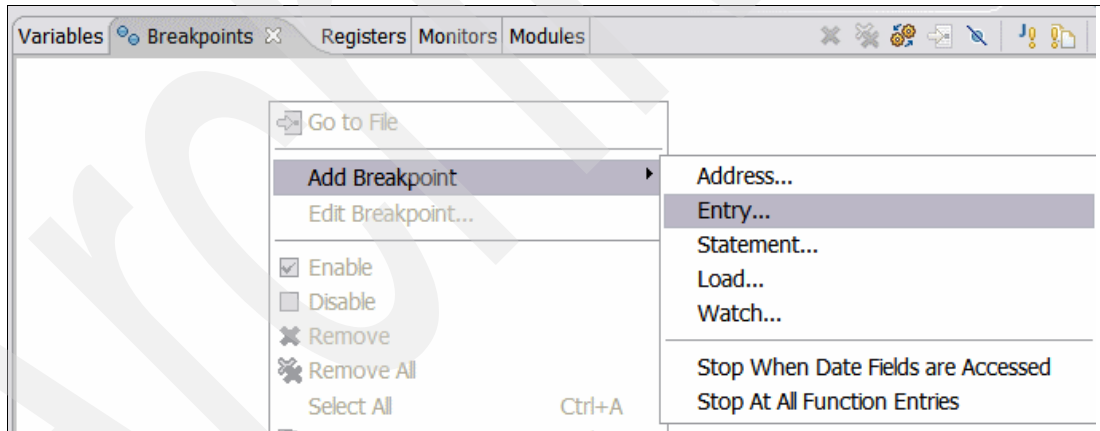


Figure 5-14 Adding a breakpoint

Specify SAM2 in all the text boxes and make sure that the box for **Defer breakpoint until executable is loaded** is checked. Then click the **Finish** button (Figure 5-15).

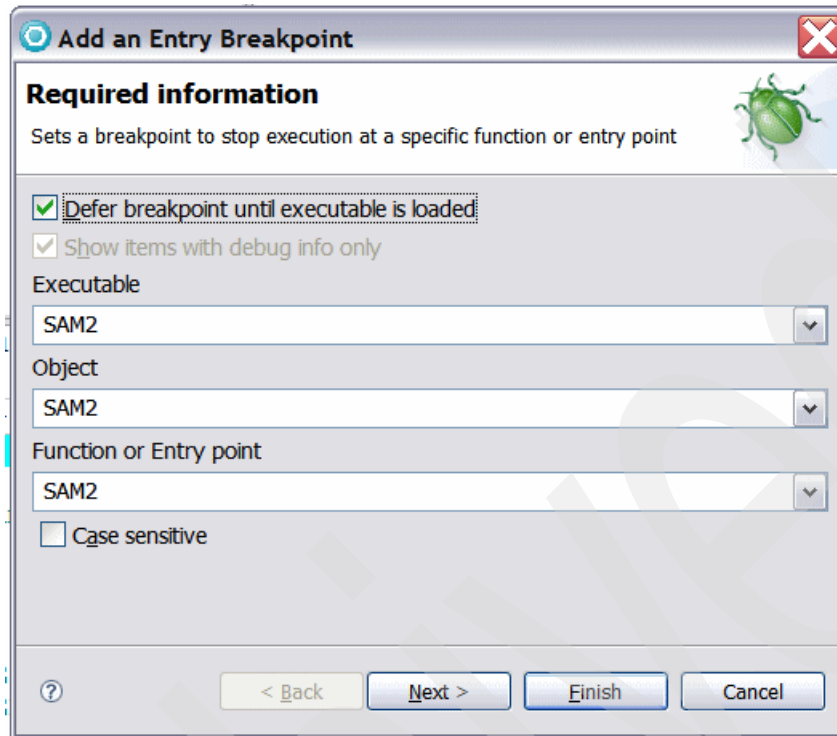


Figure 5-15 Specifying the function for a breakpoint

With the breakpoint set, use the **resume** button to run into the SAM2 program. The resume button is a green arrow located in the step tool bar (Figure 5-16).

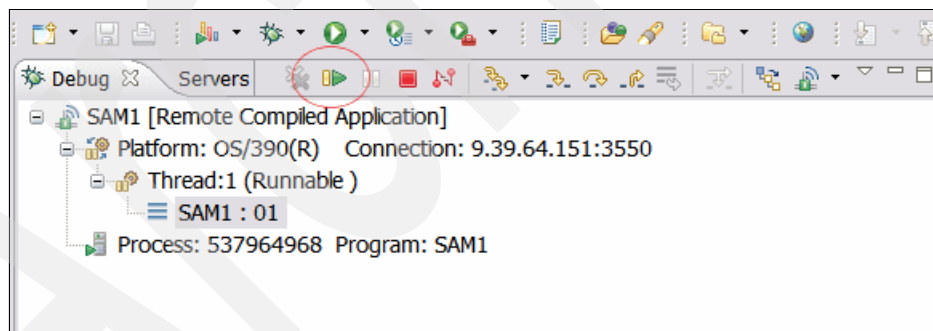


Figure 5-16 Debug Toolbar - resume button

Referencing the Fault analyzer report, we see that we want to watch the problem variable CUST-ACCT-BALANCE. To work with variables, we have to navigate into SAM2. To do this, we can use the Step icon to step over the working storage initialization and into the procedure division (Figure 5-17).

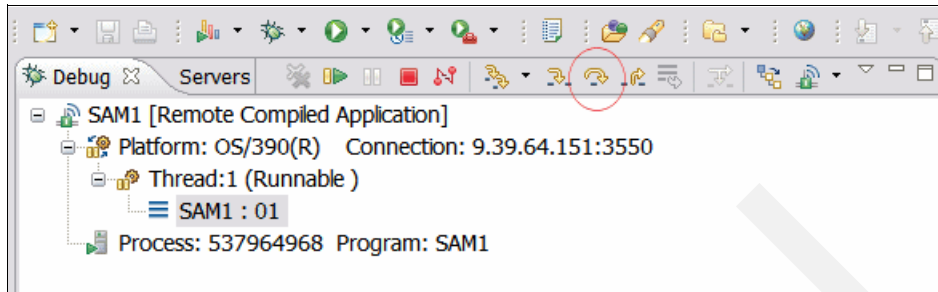


Figure 5-17 Debug Toolbar - Step Function Button

Now we can set a watch breakpoint and a monitor on CUST-ACCT-BALANCE. Once again, right-click in the breakpoint area, but this time select **Add Breakpoint** → **Watch**. In the field labeled *Address or expression*, type in CUST-ACCT-BALANCE, and then click the **Finish** button (Figure 5-18).

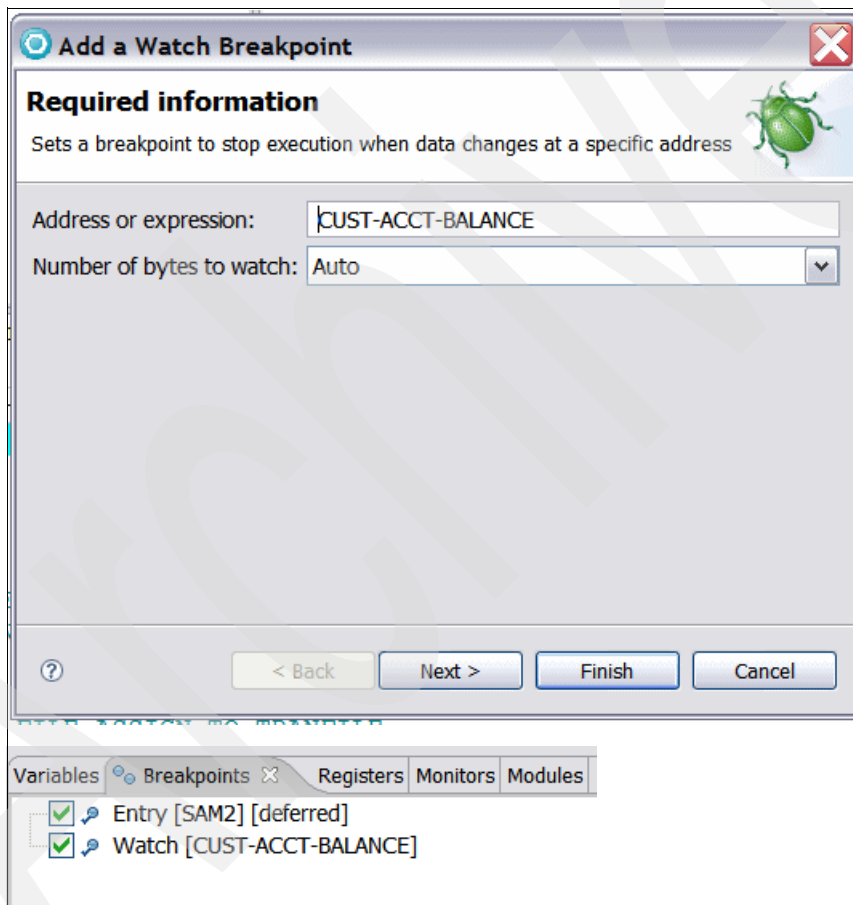


Figure 5-18 Add Watch Breakpoint

At this time we should have two entries in the Breakpoints screen. Now select the **Monitors** tab within this same window. Right-click in the monitor box and select **Monitor Expression** from the context menu, then specify CUST-ACCT-BALANCE in the text box. The name and value should appear within the monitor box (Figure 5-19).

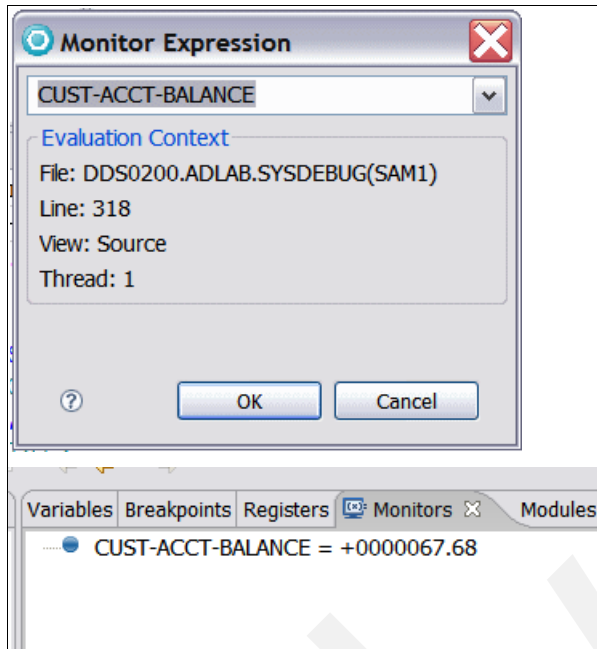


Figure 5-19 Expression Monitoring

We are ready to figure out when the value of CUST-ACCT-BALANCE contains an invalid value. Click the **Resume** button until CUST-ACCT-BALANCE contains an invalid value. Optionally we can change the representation of the value to show the value matches the invalid value in the FA report (Figure 5-20).

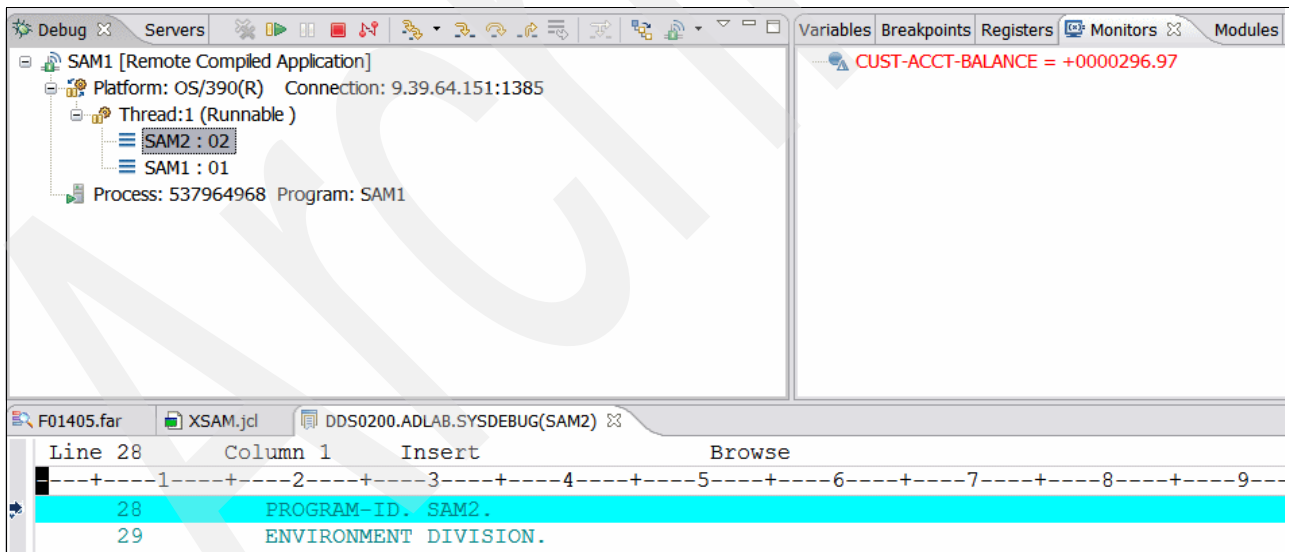


Figure 5-20 Call Stack and Expression Monitoring

Notice that we are stopped at the entry of SAM2. Because we are at the beginning of SAM2, we can infer that the bad data came from SAM1. In the top left box we can see the call path. In the Fault Analyzer Event report we saw that CUST-ACCTBALANCE is part of CUST-REC, which is the first structure passed from SAM1 to SAM2.

If we select SAM1, we can view the call to SAM2 (Figure 5-21).

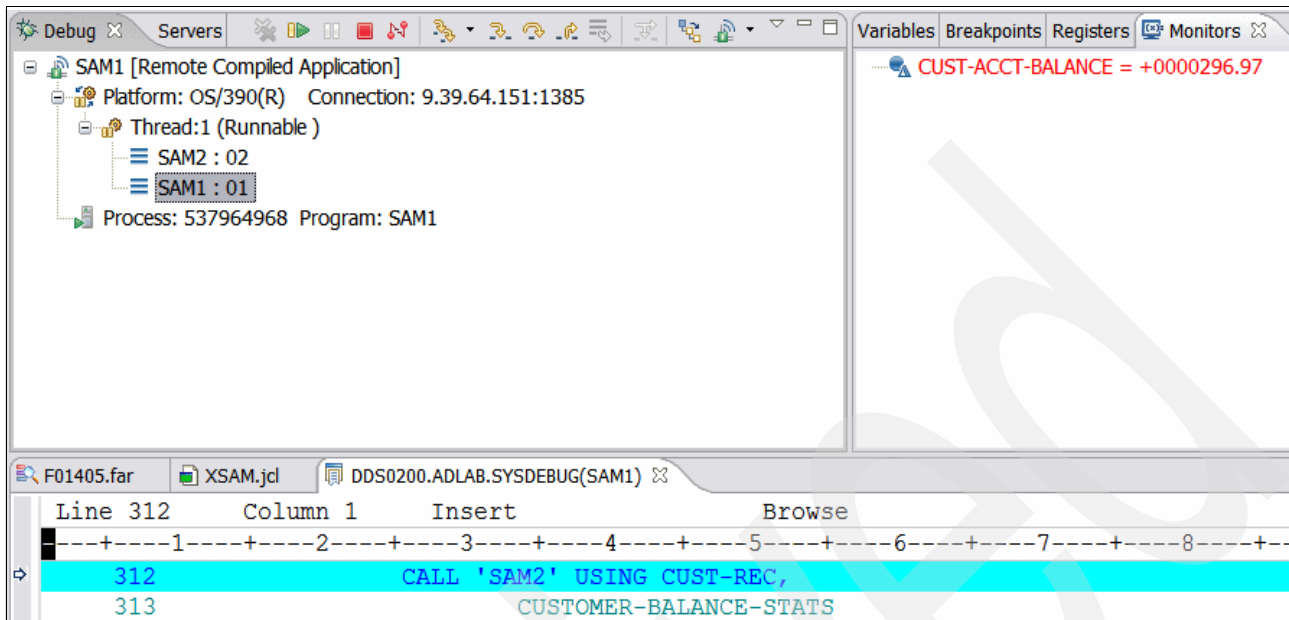


Figure 5-21 Call Stack -Entry Point to SAM2

Scrolling to the top of SAM1 and using the **Find** command, we can see that CUSTREC is part of the CUSTMER-FILE FD, and scrolling up further we can see that CUSTOMER-FILE is associated with a CUSTFILE DD statement within the JCL. This confirms that the bad data comes from a dataset allocated to the CUSTFILE DD statement. Selecting SAM2 in the call chain, we can temporarily fix the variable data, disable the breakpoints, and continue with our test.

To temporarily fix the variable, double-click the value of CUST-ACCT-BALANCE within the monitor window. Type in a valid number (we use 10 in this example) and then press Enter (Figure 5-22).

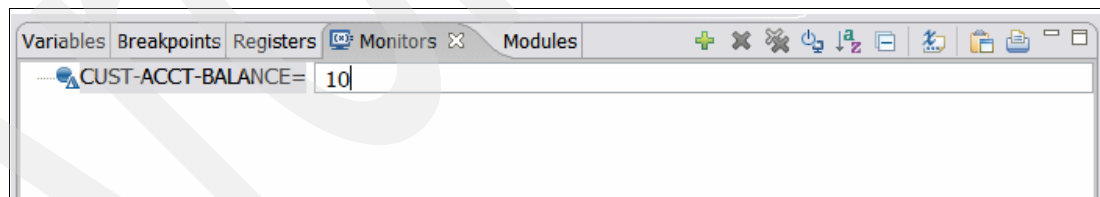


Figure 5-22 Changing a Variable Dynamically

Now select the **Breakpoints** tab and disable both breakpoints by clicking in the square box next to each breakpoint (Figure 5-23).

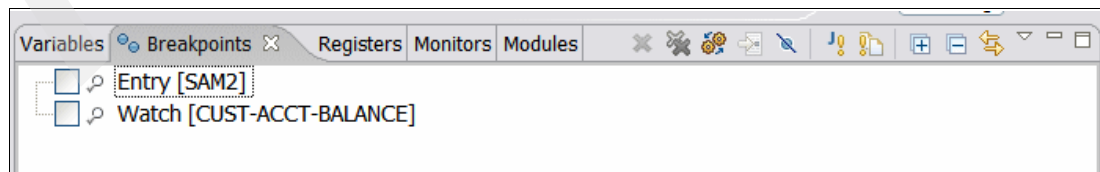


Figure 5-23 Turning off breakpoints

Using the resume button, we can let the application run to termination. We can check the return code by switching back to the Remote System perspective. In the Remote Systems tab on the top left of this perspective, we can expand JES and right-click on **My Jobs** and select **Show in Table**. At the bottom of the screen we can now see all jobs run and a return code of zero of the debug session we just ran (Figure 5-24).

Name	Job ID	Job Name	Job Ow...	Job Ent...	Return ...	Return I...	System ...	User ret...	Return ...	System ...
DDS0200X:JOB11...	JOB11707	DDS0200X	DDS0200	2008/05...	U0000	NORMAL		000	COMPLE...	
DDS0200X:JOB09...	JOB09432	DDS0200X	DDS0200	2008/05...	S0C7	ABENDed			ABEND	
DDS0200X:JOB09...	JOB09419	DDS0200X	DDS0200	2008/05...	S0C7	ABENDed			ABEND	

Figure 5-24 Remote System Details

The last step is to update erroneous data using File Manager.

Fixing a data problem with File Manager

We can figure out what file we want to edit and fix our JCL at the same time. In the Remote Systems view, select the **JCL** tab in the top center window where we edit files, and comment the TEST parameter line at the same time. Note that the CUSTFILE DD is using the dataset named userid.ADLAB.SAMFILE1 (Figure 5-25).

```

Line 28      Column 42      Insert
//--+---1---+---2---+---3---+---4---+---5---+---6---+---7---|+---
/**  INSPLOG IS THE DEBUG TOOL LOG FILE.  IT IS OPTIONAL:
/**      //INSPLOG  DD DSN=&SYSUID..ADLAB.DTLOG,DISP=SHR <== DT LOG FILE
//INSPLOG  DD SYSOUT=*      <== DT LOG TO SYSOUT
/**
/**  THESE DD STATEMENTS ARE NEEDED BY APPLICATION PROGRAM SAM1:
//STEPLIB  DD  DISP=SHR,DSN=&SYSUID..ADLAB.LOAD
/****      DD  DISP=SHR,DSN=DEBUG.DTXLOAD          (UNCOMMENT IF NEEDED)
//          DD  DISP=SHR,DSN=DEBUG.V8R1.SEQAMOD    (UNCOMMENT IF NEEDED)
/****      DD  DISP=SHR,DSN=CEE.SCEERUN          (UNCOMMENT IF NEEDED)
//CUSTFILE DD  DSN=&SYSUID..ADLAB.SAMFILE1,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//CUSTRPT  DD  SYSOUT=*
//CUSTOUT  DD  SYSOUT=*
//TRANFILE DD  *
*TRAN      (*  IN COL 1 IS A COMMENT)
*

```

Figure 5-25 JCL File Viewer

Using Remote Systems on the left, we can scroll down and locate our file. Note that datasets are sorted in two lists in alphabetic order. The first list is all PDS datasets followed by all other datasets. RDz uses two criteria to determine the contents of a dataset in order to open an appropriate editor. The primary criteria to determine a file's content is the naming convention of the datasets.

For example, any dataset ending in.JCL will be interpreted as JCL files. The other criteria to determine a file's content is the dataset organization. If the dataset is VSAM, we would automatically know it is a data file. Because our file is sequential and data files do not normally have a naming convention, we can identify our dataset as data and at the same time identify the copybook to be used to map the data. Right-click on userid.ADLAB.SAMFILE1 and select **Properties** from the context menu (Figure 5-26).

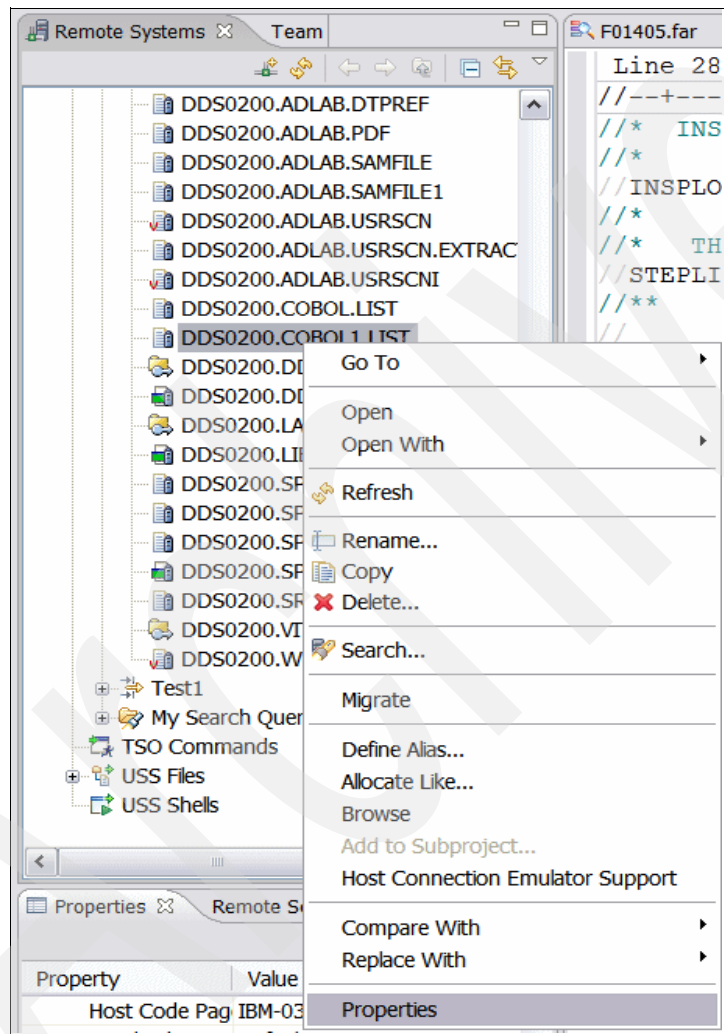


Figure 5-26 Remote Systems Data Set Properties

In order to tell RDz that the content of the dataset is data, select **Mapping**, and in the Extension section, select the **Other** radio button and specify `tdat` in the text box (Figure 5-27).

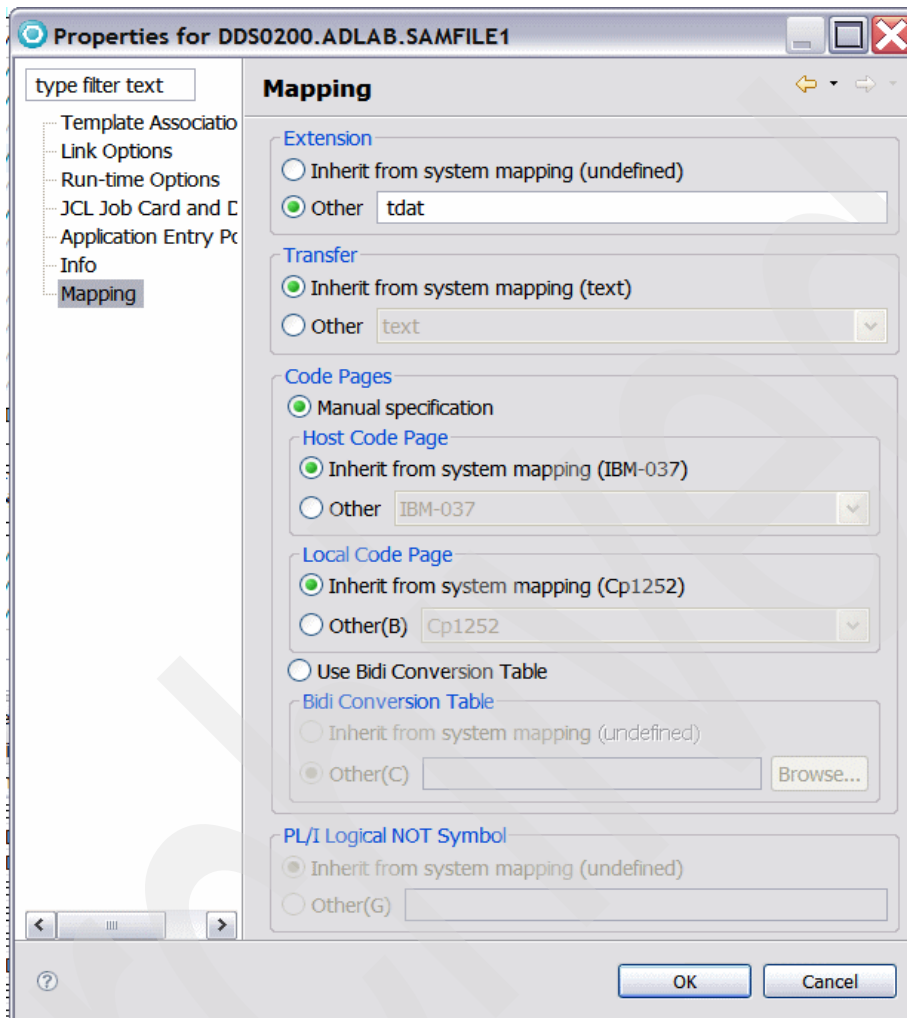


Figure 5-27 Properties -Mapping

To associate the copybook to be mapped with the data, click on **Template Association** and use the pull-down box under the title of Browse for Template/Copybook. Expand the first dataset `user.id.ADLAB.COPYLIB` and the member of `CUSTMAST.cpy`. Click the **Apply** button and then click the **OK** button (Figure 5-28).

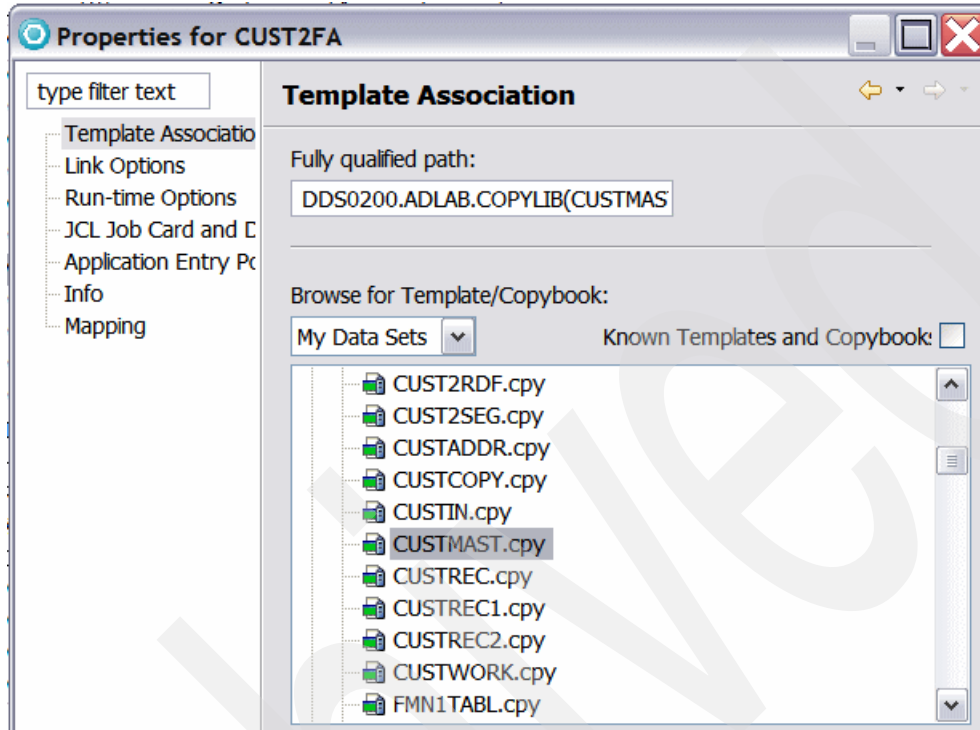


Figure 5-28 Properties - Template Association

This will cause the ICON next to our file to change. We can double-click this file to edit the file with File Manager (Figure 5-29).



Figure 5-29 Data File with Template Icon

By double-clicking on the tab for the editor of the file, we can see our file in full screen. This allows us to see a lot of data. For those familiar with the green screen version of File Manager, you will notice that we can see both the table mode and single mode on the same screen. Selecting any record in the table section activates the Single record section and synchronizes the record. It is easy to see that the record containing the name of “Turner, Page” has bad data. This is denoted by the red asterisks in the account balance column (see Figure 5-30 and Figure 5-31).

Template Associated: DDS0200.ADLAB.COPYLIB(CUSTMAST)

C...	F	NAME	ACCT-BA...	ORDERS...	ADDR	CITY
06927	A	Buchs, Jillian	99.99	0	41 Avendale Drive	Carlisle
07008	A	Houston, Roger	296.97	1	4111 Northside PkWay	Atlanta
07025	A	Marx, Audrey	450.51	2	90 South Cascade	Boulder
11204	A	Ness, Luke	513.06	3	5166 Sprinkle Road	Portage
11544	A	Graf, Polly	244.42	1	1551 S. Washington	Dowagiac
12689	A	Boyd, Luke A.	86.88	0	221 Yale Rd	South Perth
13062	A	Turner, Paige	*****	4	3039 Cornwallis Road	Boone
13295	A	Well, Alice	731.26	4	125 Nicholson Rd	North Perth
13943	A	Sawyer, Dinah	622.16	4	498 Oak Grove Rd	Springfield
14207	A	Auerbach, Hans	11.11	0	Hahnstrasse 46	Frankfurt
21508	A	Dante, Al	166.65	1	89140 Rosten 77	Trondheim
23084	A	Furst, Hugo	233.31	1	155 Bailey Avenue	Eagan
23307	A	Lapp, Victor E.	377.74	2	Rua Tutoia 2258	Sao Paulo
24390	A	Leavitt, Kent B.	399.96	2	12 Bannerghatta Road	Bangalore
25301	A	Cooke, Amelia	144.43	0	5305 Valley Park	Roanoke
26130	A	Kleiner, Rick	344.41	2	Av. Hidalgo 3500	Tampico

Figure 5-30 Invalid Data Notification Using File Manager

Single Mode	
Field	Data
CUST-ID	13062
REC-TYPE	A
NAME	Turner, Paige
ACCT-BALANCE	*****
ORDERS-YTD	4
ADDR	3039 Cornwallis Road
CITY	Boone
STATE	NC
COUNTRY	USA
MONTH (1)	2.42
MONTH (2)	14.52
MONTH (3)	0
MONTH (4)	0
MONTH (5)	0
MONTH (6)	2.42
MONTH (7)	10.89
MONTH (8)	0
MONTH (9)	71.39
MONTH (10)	71.39
MONTH (11)	71.39
MONTH (12)	7.26
OCCUPATION	Author
NOTES	le notes. More notes.
LAB-DATA-1	
LAB-DATA-2	

Figure 5-31 Single Data Entry using File Manager

To fix the data, simply highlight the column and over-type the asterisks with a valid number. In this case, we are using 10 (Figure 5-32).

Template Associated: DDS0200.ADLAB.COPYLIB(CUSTMAST)

C...	F	NAME	ACCT-BA...	ORDERS...	ADDR	CITY
06927	A	Buchs, Jillian	99.99	0	41 Avendale Drive	Carlisle
07008	A	Houston, Roger	296.97	1	4111 Northside PkWay	Atlanta
07025	A	Marx, Audrey	450.51	2	90 South Cascade	Boulder
11204	A	Ness, Luke	513.06	3	5166 Sprinkle Road	Portage
11544	A	Graf, Polly	244.42	1	1551 S. Washington	Dowagiac
12689	A	Boyd, Luke A.	86.88	0	221 Yale Rd	South Perth
13062	A	Turner, Paige	10	4	3039 Cornwallis Road	Boone
13295	A	Well, Alice	731.26	4	125 Nicholson Rd	North Perth
13943	A	Sawyer, Dinah	622.16	4	498 Oak Grove Rd	Springfield
14207	A	Auerbach, Hans	11.11	0	Hahnstrasse 46	Frankfurt
21508	A	Dante, Al	166.65	1	89140 Rosten 77	Trondheim
23084	A	Furst, Hugo	233.31	1	155 Bailey Avenue	Eagan
23307	A	Lapp, Victor E.	377.74	2	Rua Tutoia 2258	Sao Paulo
24390	A	Leavitt, Kent B.	399.96	2	12 Bannerghatta Road	Bangalore
25301	A	Cooke, Amelia	144.43	0	5305 Valley Park	Roanoke
26130	A	Kleiner, Rick	344.41	2	Av. Hidalgo 3500	Tampico

Figure 5-32 Changing Data using File Manager and RDz

When you press Enter, the number will be formatted. When you close the file, you will be prompted to save the file if you have not already done so.

Verifying results

To prove that we have fixed the problem, we can go back to our JCL and submit the job (do not forget to save the change to the PARM if you have not done so already). Check your JES output and verify that you get a return code of zero. Remember that you have to refresh the output display by clicking on the **Refresh** button (Figure 5-33).

Remote System Details | Tasks | z/OS File System Mapping

Job Filter MYJOBS

Name	Job ID	Job Name	Job Ow...	Job Ent...	Return ...	Return I..	System ...	User return code	Return Status
FEK007A9:JOB05...	JOB05089	FEK007A9	DDS0200	2008/05...	U0000	NORMAL		000	COMPLETION
DDS0200:TSU050...	TSU05081	DDS0200	DDS0200	2008/05...	S622	ABENDed			ABEND

Figure 5-33 Return Code "0"

Archived



Using APA to analyze CICS DB2 performance

The scenario described in this chapter provides an example of Application Performance Analyzer being used to analyze the performance of a CICS DB2 transaction.

6.1 Getting started

We suspected that transaction APA1 had a performance problem. We knew that this transaction could asynchronously start four other transactions:

- ▶ APA2: Searches a large VSAM file looking for records containing a specific social security number
- ▶ APA3: Accepts a date and calculates the number of days between it and the current date
- ▶ APA4: Counts the number of rows related to a set of customer numbers using an efficient SQL statement and a well indexed DB2 table
- ▶ APA5: Performs the same function as the APA4 transaction except that it uses an inefficient SQL statement a poorly indexed DB2 table

6.1.1 Starting APA

The systems programmers have set the environment such that APA can be invoked from the ISPF menu. Having selected APA from the menu, this screen was presented (Figure 6-1).

<u>ReqNum</u>	<u>Owned By</u>	<u>Description</u>	<u>Job Name</u>	<u>Date/Time</u>	<u>Samples</u>	<u>Status</u>
<u>0533</u>	DBA071	APA batch	DBA071R	May-8 13:03	100,000	Ended
<u>0530</u>	SYS029	KJC - TSFAFE on	CICSA0R4	May-2 12:50	80,000	Ended
<u>0521</u> +	DNET603	Test Job	DNET603W	Apr-29 11:37	10,000	STEPS
<u>0520</u>	DNET187		CICSA0R1	Apr-29 11:02	1,000	Ended
<u>0519</u>	DNET187		MYJOB	Apr-29 10:28	1,000	STEPS
<u>0518</u>	DNET187	my test	CICSS0A1	Apr-29 10:22	1,000	Ended
<u>0513</u>	DNET069	Second Test	DNET069G	Apr-24 3:12	21,922	Ended
<u>0512</u>	DNET424		DNET424A	Apr-23 7:01	780	Ended
<u>0503</u> +	DNET047	Belron/Safelite	DNET047X	Apr-18 13:15	3,000	STEPS
<u>0493</u>	DNET331	Second Test	DNET331G	Apr-18 4:10	44,809	Ended
<u>0492</u>	DNET069	Demo	CICSA0R6	Apr-16 12:54	10,000	Ended

Welcome to IBM APA for z/OS ISPF Version 8.100C. You are currently connected to measurement task id CAZ0. Enter CONNECT for an alternate connection, VERSION for version information, NEW to start a measurement.

F9=Swap F10=Left F11=Right F12=Cancel

Figure 6-1 Existing observation sessions

We had to build a new observation session. We started by issuing the **new** command on the command line in the foregoing screen. We refer to commands entered here as *primary* commands. Commands entered in the left side column are referred to as *line* commands.

6.1.2 Building the observation session

To capture activity associated with program execution, we had to build an observation session.

This had to specify:

1. The name of the CICS region, CICSOR5, that the APA1 transaction will run in
2. The amount of time we wanted to monitor
3. The number of samples to be captured in this time

We labeled this observation session APA CICS DB2 (Figure 6-2).

```
File View Navigate Help
-----
R03: Schedule New Measurement                               Row 00001 of 00013
Command ==> _____ Scroll ==> CSR
-----
● 1. Job Information      3. Multi Steps      5. CICS Options      7. Schedule
● 2. Options              4. Active Jobs      6. Sysplex           8. Sched Options
-----
Panel 1. Job Information                                     Input more data or ENTER to submit

Job Name/Pattern . . . CICSOR5      System Name . . . DEMOMVS
                      (Active)

Step Specification
  Step No. . . . . _____ Specify step number, program name,
  Program Name . . . _____ step name or step name + Proc step
  Step Name . . . . _____ name. Use panel 3 to specify more
  ProcStepName . . . _____ than one step.

Description . . . . . APA CICS DB2
Number of Samples . . 50000      Measure to step end . . . N
Duration (min:sec) . . 0:50      Delay by (secs) . . . . _____
Notify TSO User . . . DBA071     Retain file for (days) . 0
F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel
```

Figure 6-2 Specify job details

After pressing the Enter key, the message Input more data or ENTER to submit was returned. The next requirement was to select option 5 so that we could identify to Application Performance Analyzer the detail of the transaction to be monitored.

After typing 5 on the command line and pressing the Enter key, the CICS Transactions and Terminals screen was returned. Although APA1 is the specific transaction we wanted to monitor, we knew that it in turn started four other asynchronous transactions, all beginning with the letters APA. To monitor all transactions starting with the letters APA, we entered APA* into field 01 on this screen and pressed the Enter key. Alternatively, we could have entered the five individual transaction codes into fields 01 through 05 (Figure 6-3).

```

File View Navigate Help
-----
R03: Schedule New Measurement                               Row 00001 of 00015
Command ==> _____ Scroll ==> CSR

● 1. Job Information   3. Multi Steps   5. CICS Options   7. Schedule
● 2. Options          4. Active Jobs  6. Sysplex       8. Sched Options

Panel 5. CICS Transactions and Terminals

Specify up to 16 CICS trancodes for which measurement data is to be recorded.

01 APA* 02 ____ 03 ____ 04 ____ 05 ____ 06 ____ 07 ____ 08 ____
09 ____ 10 ____ 11 ____ 12 ____ 13 ____ 14 ____ 15 ____ 16 ____

Include CICS system transactions in measurement (Y/N): N

Wildcard character '*' can be specified at the end of a partial name.
'*' by itself specifies all transactions or terminals.

Specify up to 8 CICS terminal ids for which measurement data is to be recorded.

F1=Help  F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right F12=Cancel

```

Figure 6-3 Specify CICS options

APA does not capture specific subsystem data unless it is requested. To do this, we selected option 2 and pressed the Enter key, causing the Measurement Options screen to be displayed. Here we selected the CICS and DB2 Data Extractors. An explanation of any of the 8 options, or indeed any field in the APA screens, can always be obtained using the PF1 key (Figure 6-4).

```

File  View  Navigate  Help
-----
R03: Schedule New Measurement          Row 00001 of 00024
Command ==> _____          Scroll ==> CSR

● 1. Job Information      3. Multi Steps      ● 5. CICS Options      7. Schedule
● 2. Options              ● 4. Active Jobs    ● 6. Sysplex          8. Sched Options

Panel 2. Measurement Options          Input more data or ENTER to submit

Data Extractors. '/' to select extended measurement options:
 / CICS  CICS information
 / DB2   SQL call information
- DB2+  SQL service/CPU time/counts
- DB2V  SQL Variables
= IMS   DLI call information
- IMS+  DLI service/CPU time/counts
- MQ    MQSeries call information
- Java  Java information

Specify up to 10 load libraries to be searched by IBM APA for z/OS for
external symbol information. These are applicable only when sampled modules
F1=Help  F2=Split  F3=End   F4=Jump   F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right F12=Cancel

```

Figure 6-4 Specify subsystem options

By pressing the Enter key twice, we returned to the APA Observation List. The observation session we just created was now capturing information about activity in the CICS AOR5 region.

We then started the APA1 transaction. By pressing the Enter key again in our Observation Session its status changed to ACTIVE with a count showing the of the number of samples so far collected. As we continued to press the Enter key, the number of samples continued to increment on its way to 50,000.

We obtained a snapshot of progress by keying an `r` line command on the line for our Observation Session and pressing Enter. At this time we saw that 35,640 samples had been collected and that for the majority of these samples the system was in a WAIT state (Figure 6-5).

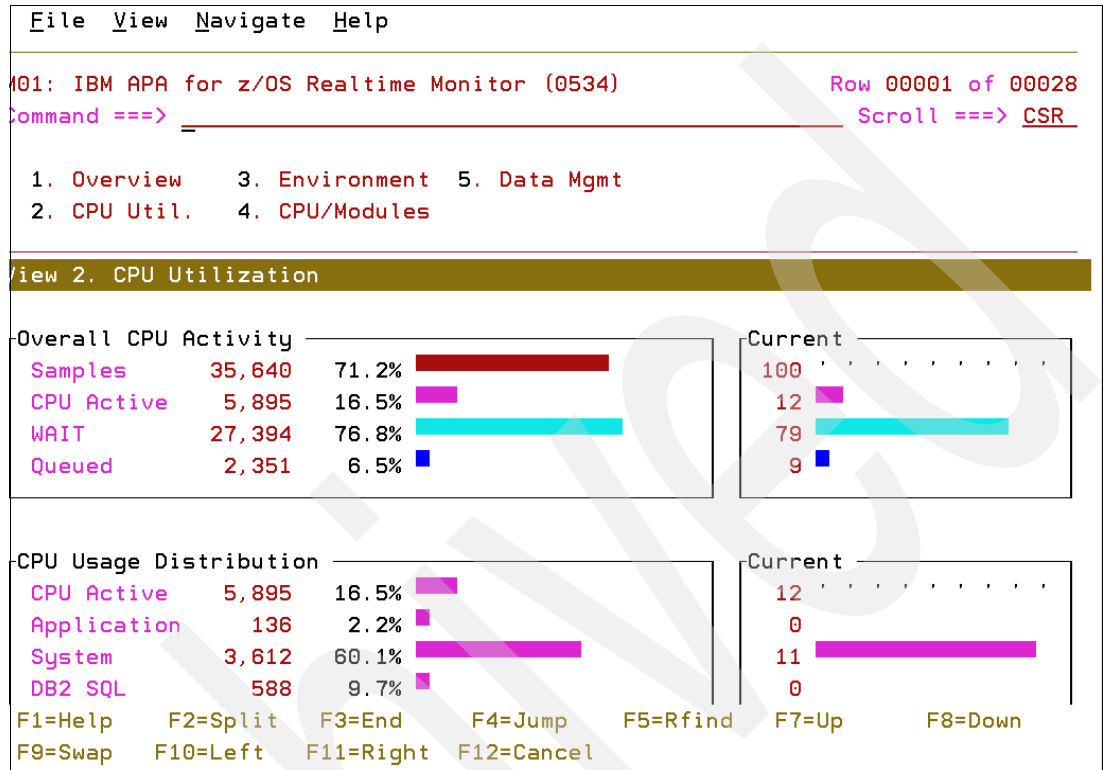


Figure 6-5 Real time observation

Pressing PF3 returned us to the Observation List screen.

Our task was to use the data collected in the APA observation session to identify potential problems and also to create an Adobe® PDF report that could be transferred to a work station for further review. The observation session we created was still at the head of the Observation List. Had it not been, we could have searched for it using an F 'APA CICS DB2' command (Figure 6-6).

File View Navigate Help							
R02: IBM APA for z/OS Observation List (CAZ0)						Row 00001 of 00181	
Command ==>						Scroll ==> CSR	
ReqNum	Owned By	Description	Job Name	Date/Time	Samples	Status	
<u>0534</u>	DBA071	APA CICS DB2	CICSA0R5	May-8 15:27	50,000	Ended	
<u>0533</u>	DBA071	APA batch	DBA071R	May-8 13:03	100,000	Ended	
<u>0530</u>	SYS029	KJC - TSFAFE on	CICSA0R4	May-2 12:50	80,000	Ended	
<u>0521</u> +	DNET603	Test Job	DNET603W	Apr-29 11:37	10,000	STEPS	
<u>0520</u>	DNET187		CICSA0R1	Apr-29 11:02	1,000	Ended	
<u>0519</u>	DNET187		MYJOB	Apr-29 10:28	1,000	STEPS	
<u>0518</u>	DNET187	my test	CICSS0A1	Apr-29 10:22	1,000	Ended	
<u>0513</u>	DNET069	Second Test	DNET069G	Apr-24 3:12	21,922	Ended	
<u>0512</u>	DNET424		DNET424A	Apr-23 7:01	780	Ended	
<u>0503</u> +	DNET047	Belron/Safelite	DNET047X	Apr-18 13:15	3,000	STEPS	
<u>0493</u>	DNET331	Second Test	DNET331G	Apr-18 4:10	44,809	Ended	
<u>0492</u>	DNET069	Demo	CICSA0R6	Apr-16 12:54	10,000	Ended	
<u>0491</u>	DNET069	First Test	DNET069G	Apr-16 12:29	2,840	Ended	
<u>0487</u>	DNET331	Demo	CICSA0R5	Apr-16 5:11	50,000	Ended	
<u>0486</u>	DNET331	Demo	CICSA0R4	Apr-16 5:08	50,000	Ended	
F1=Help	F2=Split	F3=End	F4=Jump	F5=Rfind	F7=Up	F8=Down	
F9=Swap	F10=Left	F11=Right	F12=Cancel				

Figure 6-6 Observation List, showing our job completed

To obtain static detail about our observation session, we issued the ++ line command on our observation (number 534). See Figure 6-7.

```
File View Navigate Help
More: +
-
General
Request Number      0534
Request Description  APA CICS DB2
Request Status      Ended
Owner Id            DBA071
Time of Request     Thursday May 8 2008 15:26:16.68
Session Start Time  Thursday May 8 2008 15:26:17.19
Session End Time    Thursday May 8 2008 15:27:07.19
Session Duration    0 minutes, 49.99 seconds
Session Delete Date Do not Delete

Measurement Criteria
Select by Job Name  CICSA0R5
Select by Sys Name  DEMOMVS
Sample Interval     1000 microseconds
F1=Help    F2=Split  F3=End    F4=Jump    F5=Rfind  F7=Up
F8=Down    F9=Swap   F10=Left  F11=Right  F12=Cancel

0486   DNET331  Demo                CICSA0R4  Apr-16  5:08   50,000  Ended
F1=Help  F2=Split  F3=End    F4=Jump    F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right F12=Cancel
```

Figure 6-7 Observation session metrics

We used PF3 to return to our Observation List screen and selected our session by using an S line command and pressing the Enter key.

The top half of this screen lists all the APA report categories. The highlighting indicates those categories with/without reports. In this example there was no data collected for HFS, IMS, MQ, and Java, and there were no Variance Reports. Detail in the bottom section of the screen was determined by the selection made in the top section (Figure 6-8).

```

File View Navigate Help
-----
R01: IBM APA for z/OS Performance Reports (0534)          Row 00001 of 00007
Command ==> _____ Scroll ==> CSR

Select a category from the list to the right to view the available reports in the selection list below.

_ A Admin/Miscellaneous      _ I IMS Measurement
_ S Statistics/Storage      _ E CICS Measurement
_ C CPU Usage Analysis      _ F DB2 Measurement
_ D DASD I/O Analysis      _ Q MQ Measurement
_ W CPU WAIT Analysis      _ G Coupling Facility
_ H HFS Analysis            _ J Java Measurement
_ V Variance Reports

More: +

Enter S to make a selection or enter the report code on the command line

_ C01 CPU Usage by Category      _ C07 CPU Usage by Procedure
_ C02 CPU Usage by Module        _ C08 CPU Referred Attribution
_ C03 CPU Usage by Code Slice    _ C09 CPU Usage by PSW/ObjCode
_ C04 CPU Usage Timeline
_ C05 CPU Usage Task/Category
_ C06 CPU Usage Task/Module

F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left F11=Right F12=Cancel

```

Figure 6-8 Ro01 report

We had selected Statistics and Storage category for our initial analysis activity. Pressing the Enter key changed the bottom half of the screen to now display the names of all the reports available in the Statistics/Storage category (Figure 6-9).

```

File View Navigate Help
-----
R01: IBM APA for z/OS Performance Reports (0534)          Row 00001 of 00007
Command ==> _____ Scroll ==> CSR

Select a category from the list to the right to view the available reports in the selection list below.
_ A Admin/Miscellaneous      _ I IMS Measurement
_ S Statistics/Storage      _ E CICS Measurement
_ C CPU Usage Analysis       _ F DB2 Measurement
_ D DASD I/O Analysis        _ Q MQ Measurement
_ W CPU WAIT Analysis        _ G Coupling Facility
_ H HFS Analysis             _ J Java Measurement
_ V Variance Reports

More: +

Enter S to make a selection or enter the report code on the command line

_ S01 Measurement Profile          = S07 TCB Execution Summary
_ S02 Load Module Attributes       _ S08 Processor Utilization Summary
_ S03 Load Module Summary         _ S09 Measurement Analysis
_ S04 TCB Summary
_ S05 Memory Usage Timeline
_ S06 Data Space Usage Timeline

F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left  F11=Right  F12=Cancel

```

Figure 6-9 Selecting the Measurement Profile report

From the available reports for Statistics/Storage, we selected **S01 Measurement Profile**.

This report told us that a high percentage of the samples were taken while the system was waiting, possibly due to an unavailable resource (Figure 6-10).

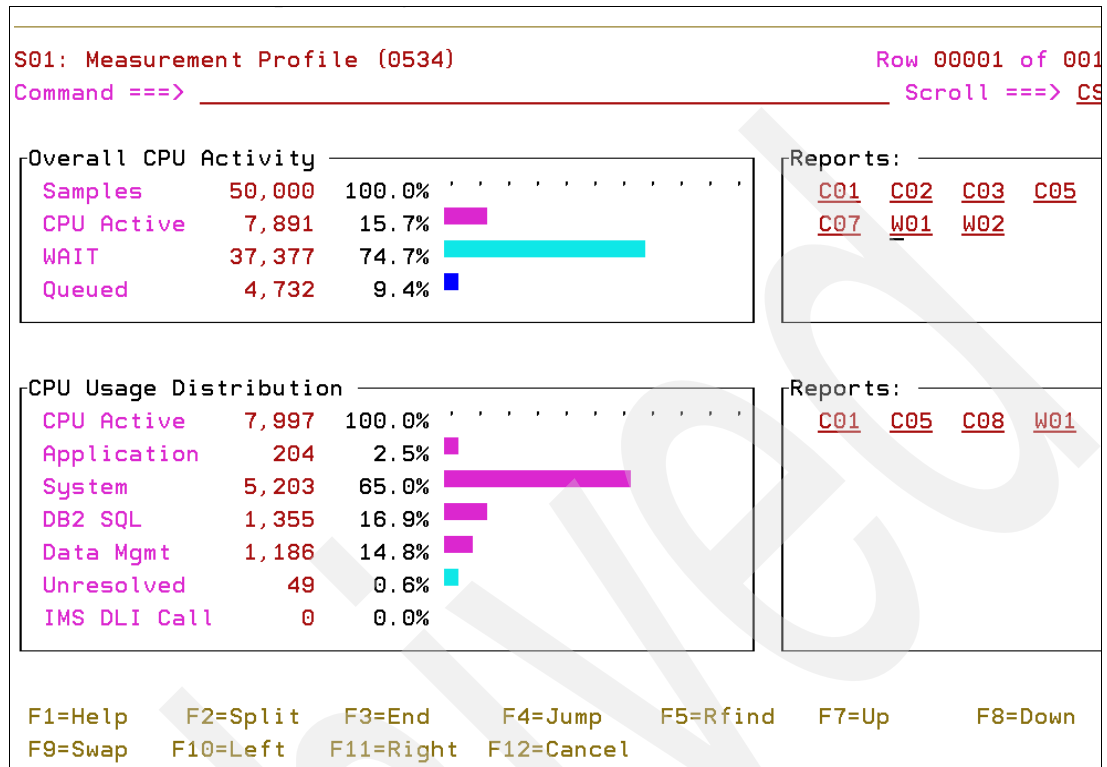


Figure 6-10 S01 report

To obtain additional details about the wait time, we looked at the Wait Time by Task/Module report by positioning the cursor on the W02 link to the right of the screen and pressing the Enter key.

We noted that the largest wait time was in the DFHSIP module, the CICS system initialization program (Figure 6-11).

File View Navigate Help			
W01: WAIT Time by Task/Category (0534)		Row 00001 of 00044	
Command ==>		Scroll ==> CSR	
Name	Description	Percent of Time in WAIT * 10.00%	±0.4%
+3HSIP-002	TCB=006FF028	100.00	[Progress bar]
DFHKETCB-001	TCB=006E5610	20.60	[Progress bar]
DFHKETCB-007	TCB=006E5C50	0.13	
DFHKETCB-008	TCB=006E5930	0.06	
DFHKETCB-037	TCB=006955E8	0.05	
DFHKETCB-043	TCB=00695E88	0.04	
DFHKETCB-035	TCB=00698238	0.03	
DFHKETCB-040	TCB=00695200	0.03	
DFHKETCB-039	TCB=00695438	0.03	
DFHKETCB-044	TCB=00695CD8	0.03	
DFHKETCB-042	TCB=00692048	0.03	
DFHKETCB-038	TCB=006983E8	0.03	
DFHKETCB-041	TCB=00695050	0.03	
EZACIC03-036	TCB=00695808	0.00	
DFHKETCB-014	TCB=006AA2A8	0.00	

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
F9=Swap F10=Left F11=Right F12=Cancel

Figure 6-11 W01 report

To obtain further detail, we expanded this item by using a +3 line command as shown.

From this screen, we could see that all of the wait time was attributed to CICS Services. This is an indication that the system was lightly loaded at the time we collected our samples (Figure 6-12).

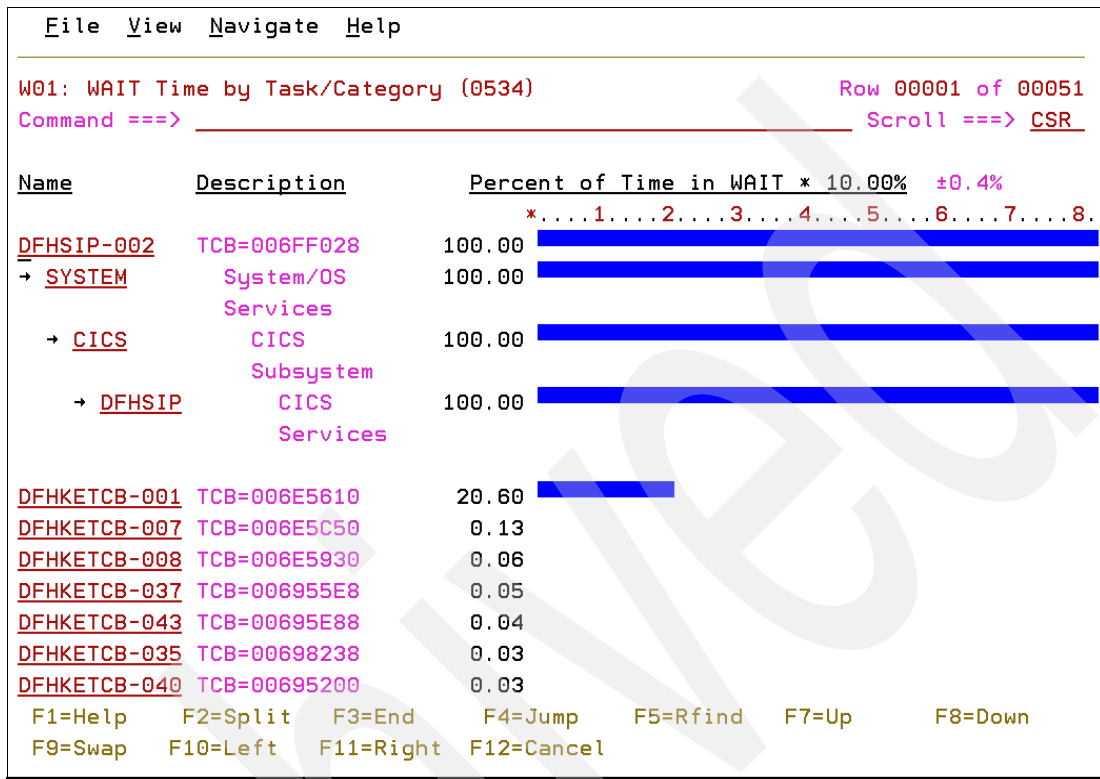


Figure 6-12 W01 report, expanded

We returned to the Measurement Profile report by pressing PF3.

The box labeled CPU Usage Distribution contains statistics related to CPU resources used by the CICS region. The statistics showed that System activities consumed the largest part of the region's CPU usage; Data Management and DB2 SQL also showed high percentages of CPU usage (Figure 6-13).

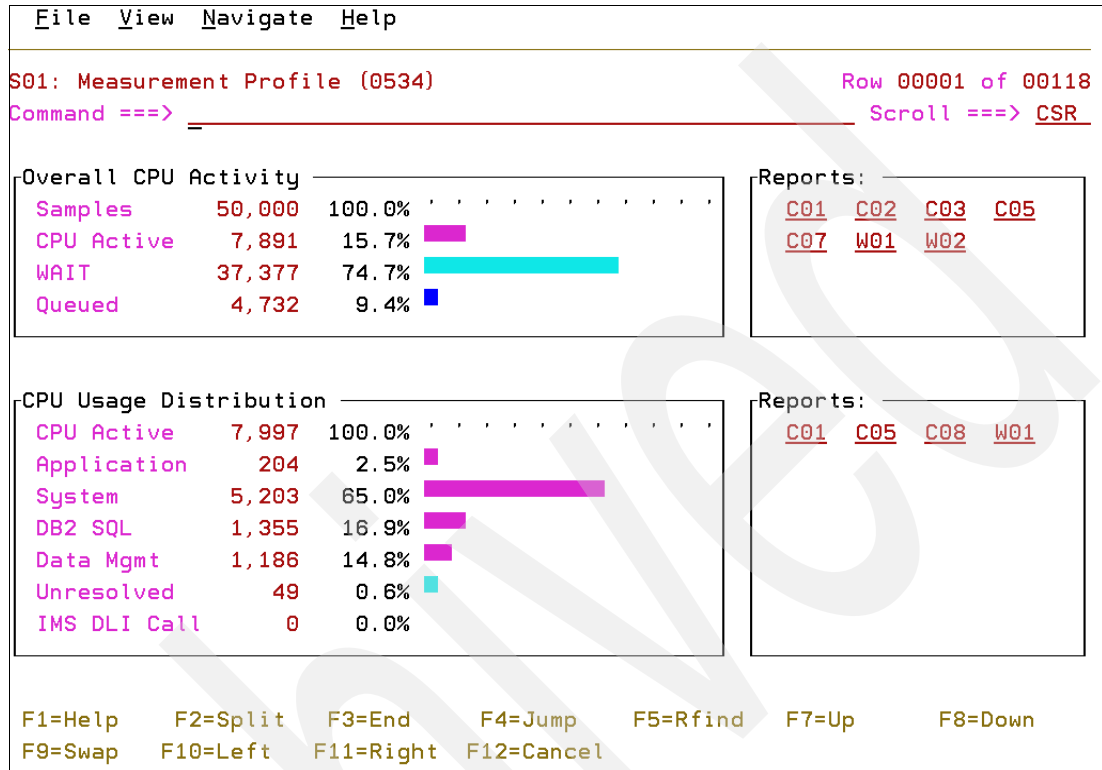
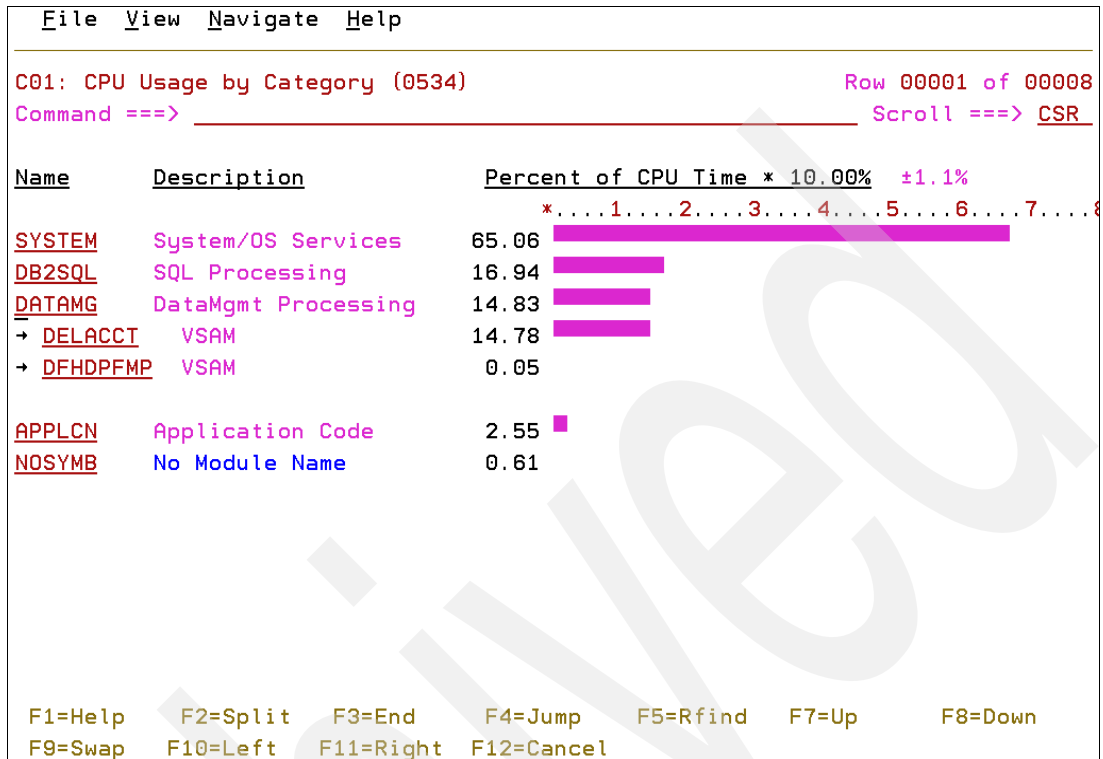


Figure 6-13 S01 report

CPU usage details are available in the CPU usage by Category report. To display this report, we positioned the cursor on C01 and pressed Enter (Figure 6-14).



We expanded the DATAMG category by using a + line command.

The expanded detail showed us that most of the data management CPU time was associated with the VSAM file DELACCT. Using a further + command, we saw that most of the activity was associated with GET requests to this file (Figure 6-15).

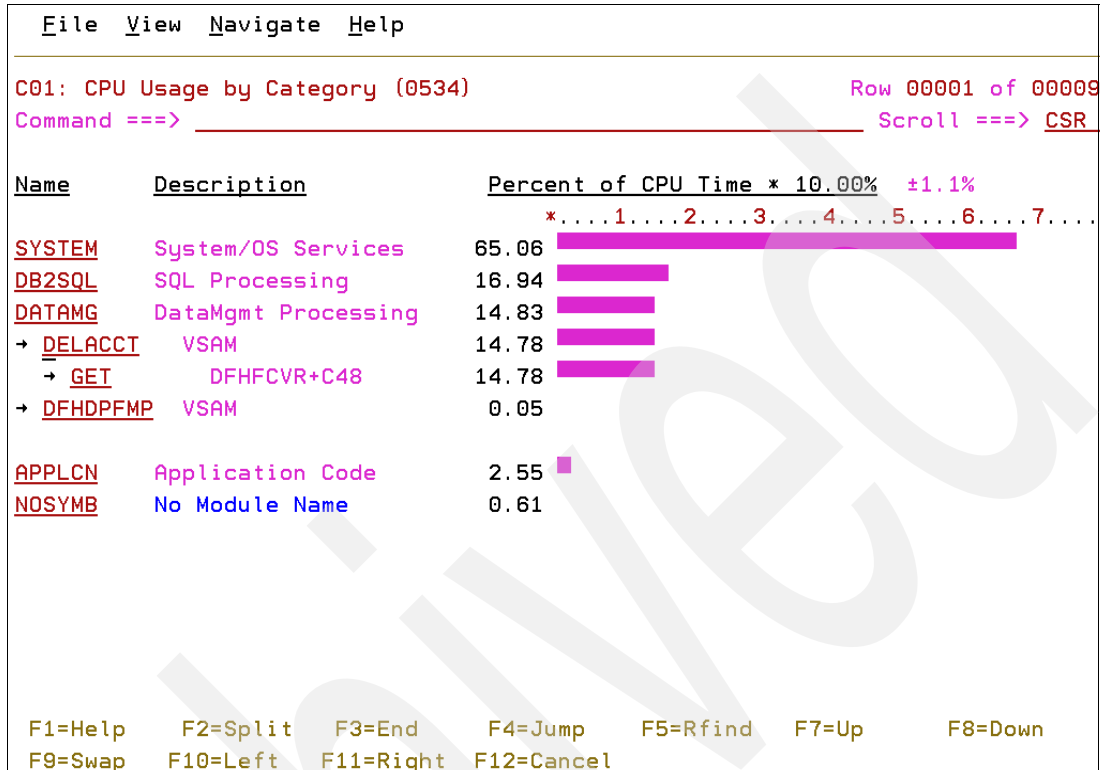


Figure 6-15 C01 report, showing second level of expansion

We then turned our attention to the CICS details and the APA* transactions. We used the E03 report, obtained by entering E03 on the command line, to see details of CPU time by transaction.

From this report, we learned that transaction APA2 was using the majority of the CPU time (Figure 6-16).

File View Navigate Help			
E03: CICS CPU Usage by Transaction (0078/CICSA0R5)		Row 00001 of 00005	
Command ==>		Scroll ==> CSR	
Name	NTxns/Description	Percent of CPU Time * 10.00%	±1.1%
<u>+3A2</u>	8	79.08	
<u>APA5</u>	1	16.01	
<u>APA1</u>	1	0.08	
<u>APA4</u>	8	0.04	
<u>APA3</u>	3	0.02	

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
 F9=Swap F10=Left F11=Right F12=Cancel

Figure 6-16 E03 report

We expanded this entry to get additional details by using a +3 line command.

At this level of detail, we saw that program CCIDDELA was using most of the CPU time with a READNEXT request to file DELACCT (Figure 6-17).

File View Navigate Help		
E03: CICS CPU Usage by Transaction (0534)		Row 00001 of 00030
Command ==>		Scroll ==> CSR
Name	NTxns/Description	Percent of CPU Time * 10.00% ±1.1%
		*.....1.....2.....3.....4.....5.....6.....7
<u>APA2</u>	8	77.30
→ <u>CCIDDELA</u>	EXEC CICS	67.53
→ <u>+0614</u>	READNEXT FILE (DELACCT)	67.38
→ <u>+058C</u>	STARTBR FILE (DELACCT)	0.15
→ <u>CCIDDELA</u>	CICS Program	9.69
→ <u>DFHAIP</u>	CICS Services	4.57
→ <u>DFHSIP</u>	CICS Services	3.07
→ <u>CCIDDELA</u>		1.43
→ <u>CICS</u>	System Services	0.60
→ <u>CICS</u>	System Services	0.03
→ <u>IDA019L1</u>	Virtual I/O (VIO) and VSAM	0.02
→ <u>DFHSIP</u>	CICS Services	0.01
F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down		
F9=Swap F10=Left F11=Right F12=Cancel		

Figure 6-17 E03 report, detail expanded

It was now time to look at the source code. We did this by using a **p** line command for the READNEXT entry (Figure 6-18).

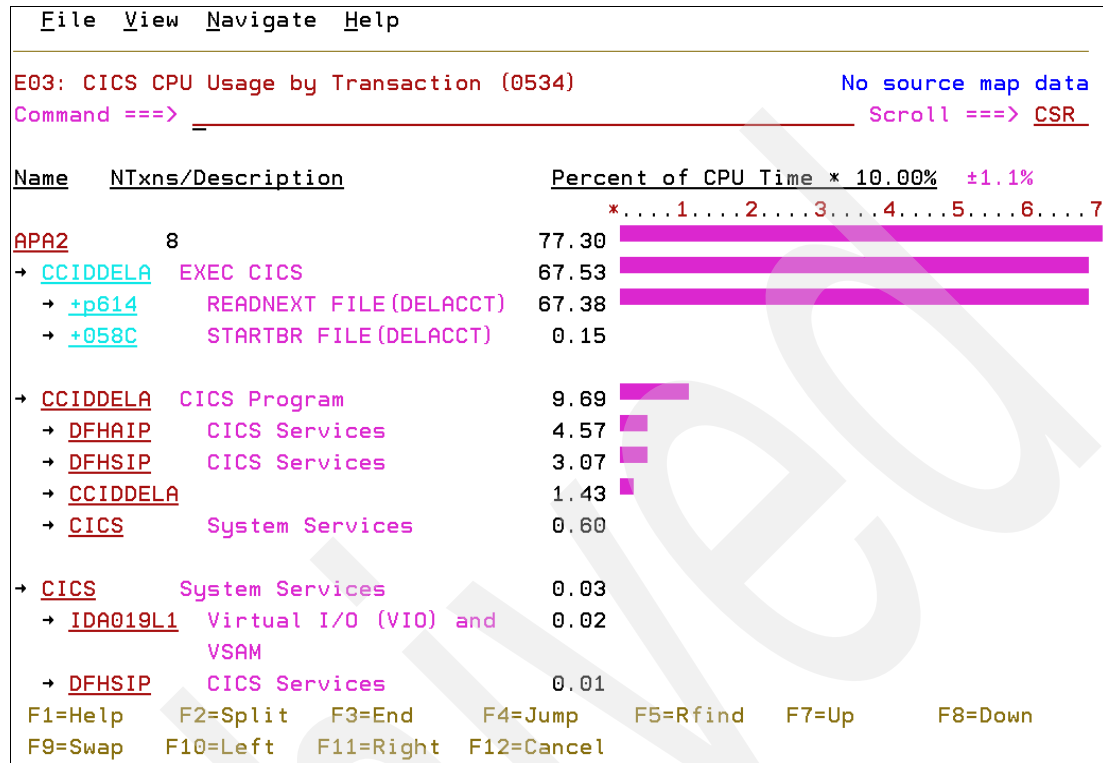


Figure 6-18 E03 report, showing request for source code

In response we received a No source map data message. This indicated that we have not provided APA with the required source data. We knew that a SYSDEBUG file had been created for program CCIDDELA. To find it, we issued the A01 command to take us to the Source Program Mapping screen.

We scanned the list of source members to find detail for CCIDDELA (Figure 6-19).

```

File View Navigate Help
-----
A01 - Source Program Mapping (0078/CICSA0R5) Row 00009 of 00015
Command ==> _____ Scroll ==> CSR

Enter the following information to specify a source mapping file to be
used in the analysis of this measurement information.

File type . . . . _ (L=listing, A=ADATA, S=LANGX SideFile, D=SYSDEBUG)
Dataset name . . _____
                    (Leave blank to search A04 dataset list)
Member name . . . _____ Match on Compile Date & Time N

Seqn  ID-ReqNum  Load  Type/Status  Lang  Member  DSN
-----
0009 CAZ0-0330  NO    L-Inact     COB   SAM2V   DBA071.ADLAB.LISTING
0010 CAZ0-0330  NO    L-Inact     COB   SAM1V   DBA071.ADLAB.LISTING
0011 CAZ0-0332  NO    L-Inact     COB   SAM2V   DBA071.ADLAB.LISTING
0012 CAZ0-0332  NO    L-Inact     COB   SAM1V   DBA071.ADLAB.LISTING
0013 CAZ0-0333  NO    L-Inact     COB   SAM2V   DBA071.ADLAB.LISTING
0014 CAZ0-0333  NO    L-Inact     COB   SAM1V   DBA071.ADLAB.LISTING
L015 CAZ0-0534  NO    D-Inact     COB   CCIDDELA  SYS030.CCDEMO.SYSDEBUG

F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 6-19 A01 - Source Program Mapping

In this example the source code was available as a SYSDEBUG file. We loaded it using a 'L' line command.

Had this not already been in our list, we could have made it available by entering the File type and Dataset name in the areas provided.

The SYSDEBUG file was now loaded and available to APA (Figure 6-20).

```

File  View  Navigate  Help
-----
A01 - Source Program Mapping (0534)                               Source map data loaded
Command ==> _____ Scroll ==> CSR

Enter the following information to specify a source mapping file to be
used in the analysis of this measurement information.

File type . . . . _      (L=listing, A=ADATA, S=LANGX SideFile, D=SYSDEBUG)
Dataset name . . . _____
                        (Leave blank to search A04 dataset list)
Member name . . . _____ Match on Compile Date & Time  N

Seqn  ID-ReqNum  Load  Type/Status  Lang  Member  DSN
-----
0001  CAZ0-0534  NO    D-Loaded    COB   CCIDDELA  SYS030.CCDEMO.SYSDEBUG
0002  CAZ0-0034  NO    L-Inact     COB   SAM1V     DBA071.ADLAB.LISTING
0003  CAZ0-0034  NO    L-Inact     COB   SAM2V     DBA071.ADLAB.LISTING
0004  CAZ0-0040  NO    L-Inact     COB   SAM1V     DBA071.ADLAB.LISTING
0005  CAZ0-0040  NO    L-Inact     COB   SAM2V     DBA071.ADLAB.LISTING
0006  CAZ0-0042  NO    L-Inact     COB   SAM1V     DBA071.ADLAB.LISTING
0007  CAZ0-0042  NO    L-Inact     COB   SAM2V     DBA071.ADLAB.LISTING
0008  CAZ0-0329  NO    L-Inact     COB   SAM2V     DBA071.ADLAB.LISTING

F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left  F11=Right  F12=Cancel

```

Figure 6-20 A01 - Confirmation that source code is loaded

PF3 returned us to the E03 screen where we re-issued the **p** line command.

The source code was now displayed with the EXEC CICS READNEXT instruction highlighted. Note that APA took a large number of samples (5389 of the 7891 taken in the application) while this instruction was being executed (Figure 6-21).

```

File View Navigate Help
-----
P01: Source Program Attribution (0534)                               Row 00001 of 00022
Command ==> _____ Scroll ==> CSR

LineNo Offset Count Source Statement
-----
000320 0004F8                MOVE DFHCOMMAREA TO WK-ID-DELETED-ACCT-AREA.
000321 00050C                MOVE ZEROES TO ID-DELETED-ACCT-COUNT.
000322 000516                MOVE ZEROES TO ID-DELETED-ACCT-DATE1.
000323 000520                MOVE ZEROES TO ID-DELETED-ACCT-DATE2.
000324 00052A                MOVE ZEROES TO WK-DELACCT-FILE-KEY.
000325 000538                EXEC CICS
000326                        STARTBR DATASET(WK-DELACCT-FILE-NAME)
000327                        RIDFLD(WK-DELACCT-FILE-KEY) GTEQ
000328                END-EXEC.
000329 000598                PERFORM UNTIL END-LOOP = 'Y'
000330 0005B0 5389          EXEC CICS
000331                        READNEXT DATASET(WK-DELACCT-FILE-NAME)
000332                        INTO(WK-DELACCT-FILE-REC)
000333                        LENGTH(WK-DELACCT-FILE-LENGTH)
000334                        RIDFLD(WK-DELACCT-FILE-KEY)
F1=Help   F2=Split   F3=End     F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 6-21 P01 report

An analysis of the code showed us that the program was using skip sequential processing in a keyed file.

We confirmed our observation by issuing the D06 primary command to view the DASD VSAM statistics (Figure 6-22).

File View Navigate Help									
D06: DASD VSAM Statistics (0534)							Row 00001 of 00012		
Command ==> _____							Scroll ==> CSR		
DDNAME	Logical Records During Interval						+/- During Inte		
	Retrvd	Added	Insrtd	Deletd	Updatd	EXCPs	FreeSpc	CISplits	C
DELACCT	1181K	+0	0	0	0	32881	+0	+0	
DFHDPFMP	312	+0	0	0	0	0	+0	+0	
DFHDPFMB	312	+0	0	0	0	0	+0	+0	
DFHGCD	0	+2	2	0	0	6	+0	+0	
HCREGID	0	+0	0	0	0	0	+0	+0	
HCVISIT	0	+0	0	0	0	0	+0	+0	
FMNPROF	0	+0	0	0	0	0	+0	+0	
ADCUST1	0	+0	0	0	0	0	+0	+0	
DFHLCD	0	+0	0	0	0	1	+0	+0	
DFHTEMP	0	+0	0	0	0	0	+0	+0	
DFHINTRA	0	+0	0	0	0	0	+0	+0	
DFHLRQ	0	+0	0	0	0	0	+0	+0	

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
 F9=Swap F10=Left F11=Right F12=Cancel

Figure 6-22 D06 report

The report showed us that a large number of records were read (Retrvd) from file DELACCT. We knew that we could find more detail about this activity from the D05 report.

This report showed us that the read activity was split almost evenly between retrieving data records and retrieving index records (Figure 6-23).

File View Navigate Help						
D05: DASD EXCP Summary (0534)					Row 00001 of 00078	
Command ==> _____					Scroll ==> CSR	
DDNAME	Type	Concat	Number of EXCPs			
			At Start	At End	During Measurement	
DELACCT	VSAM-DATA		3113669	3130078	16409	
	VSAM_INDEX		3125835	3142307	16472	
DFHRPL	Non-VSAM	+1	36	50	14	
DFHGCD	VSAM-DATA		1510	1516	6	
	VSAM_INDEX		479	479	0	
STEPLIB	Non-VSAM	+10	56	60	4	
STEPLIB	Non-VSAM	+1	70	73	3	
STEPLIB	Non-VSAM	+4	56	59	3	
DFHLCD	VSAM-DATA		1584	1585	1	
	VSAM_INDEX		29	29	0	
HCREGID	VSAM-DATA		99	99	0	
	VSAM_INDEX		77	77	0	
HCVISIT	VSAM-DATA		108369	108369	0	
	VSAM_INDEX		88782	88782	0	
F1=Help	F2=Split	F3=End	F4=Jump	F5=Rfind	F7=Up	F8=Down
F9=Swap	F10=Left	F11=Right	F12=Cancel			

Figure 6-23 D05 report

The D04 report, Dataset Attributes, would give us specific details relating to file DELACCT.

The D04 line command returned details for all files in the application. We used a FIND command to get to the details for DELACCT (Figure 6-24).

```

File View Navigate Help
-----
D04: Dataset Attributes (0534)                               Row 00145 of 00717
Command ==> _____ Scroll ==> CSR

VSAM file DELACCT OPENed at 15:26:20.77 Thursday May 8 2008

DDNAME                DELACCT
Open Intent            KEY, DSN, DIR, SEQ, SKP, OUT, NLW, LSR SHRPOOL=1
Dataset Name           SYS030.DELACCT.FILE.DATA
Management Class      USRMGMT
Storage Class          USBASE
Device Type            3390
% Free Bytes in CI    0%
Volume Serial          DMPU01
CI Size                2,048
Record Size (LRECL)   255
Number of Extents     3
SHAREOPTIONS           (4 3)
Organization           KSDS
CIs per CA             315
Free CIs per CA       0
F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right  F12=Cancel

          Initial      Last
CI Splits      0          0
CA Splits      0          0
Logical Records 150,000    150,000
Deleted Records 0          0
Insrted Records 0          0
Retrvd Records 223,270,395 224,451,507
Updated Records 0          0
Bytes Free Space 892,928    892,928

```

Figure 6-24 D04 report

This report confirmed that most of the activity happened during the read (Retrieved Records) operations.

We then turned our attention to the DB2 activity. Issuing the C01 primary command took us back to the CPU Usage by Category report. This time we used the + line command to expand the DB2SQL entry. This showed the activity attributed to DB2 SQL statements.

This report told us that the bulk of the processing in DB2SQL occurred as a result of statement 39 in the READBIGT module (Figure 6-25).

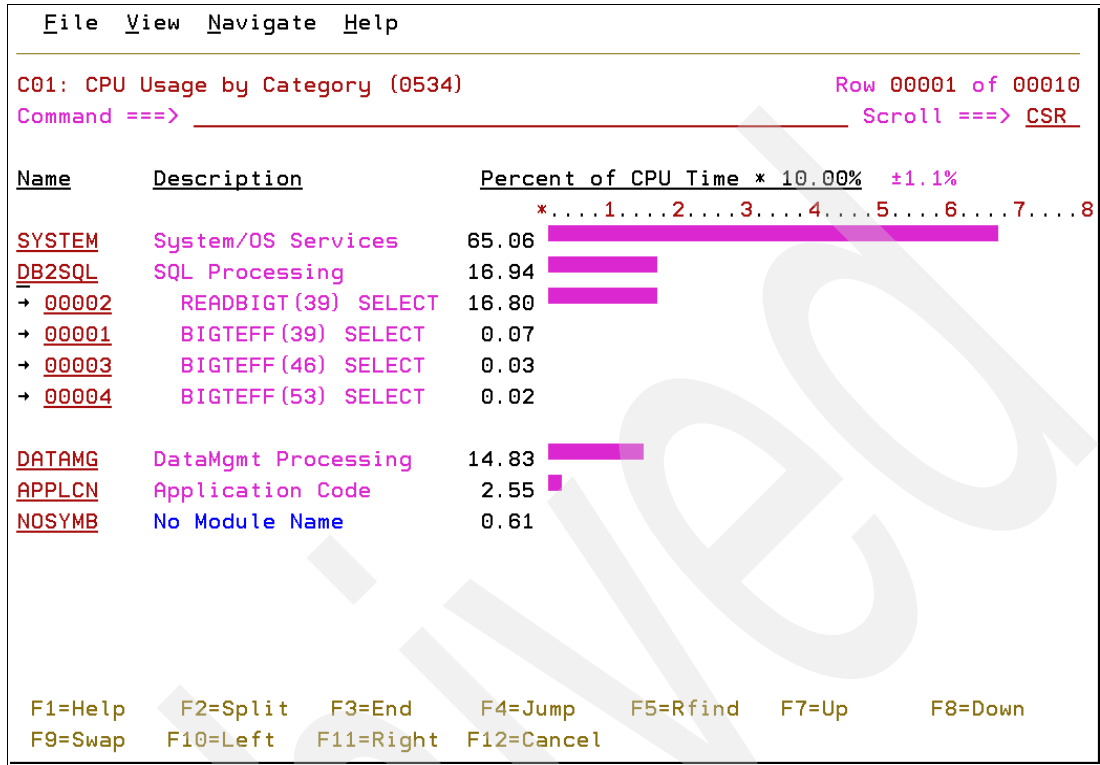


Figure 6-25 C01 report - SQL activity expanded

Suspecting that this meant there was a problem in this program, we issued the F01 primary command to view the DB2 Measurement Profile report.

This report provides general DB2 statistics. It confirmed that READBIGT is the cause of almost all of the DB2SQL processing (Figure 6-26).

File View Navigate Help			
F01: DB2 Measurement Profile (0534)		Row 00001 of 00031	
Command ==> _____		Scroll ==> CSR	
Most Active DB2 Plans			Reports:
Samples	50,000	100.0%
APADEMO	1,993	3.9%	■
Most Active Package/DBRMs			Reports:
Samples	50,000	100.0%
READBIGT	1,976	3.9%	■
BIGTEFF	26	0.0%	■
Most Active SQL Statements			Reports:
Samples	50,000	100.0%
READBIGT:00039 SELECT	1,976	3.9%	■
BIGTEFF:00039 SELECT	20	0.0%	■
BIGTEFF:00053 SELECT	4	0.0%	
BIGTEFF:00046 SELECT	2	0.0%	
F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down			
F9=Swap F10=Left F11=Right F12=Cancel			

Figure 6-26 F01 report

Additional detail about the DB2 activity can be found in the F04 SQL Activity by Statement report. We issued the F04 line command to get to the report and then used a + line command to expand the entry.

This showed the SQL statement being executed (Figure 6-27).

```

File View Navigate Help
-----
F04: SQL Activity by Statement (0534)                               Row 00001 of 00008
Command ==> _____ Scroll ==> CSR

Segno Program Stmt# SQL Function Percent of Time * 10.00% ±0.4%
*.....1.....2.....3.....4.....5.....6.....7...
S0002 READBIGT 39 SELECT 3.95 ■
> SELECT COUNT ( * ) INTO : H FROM SYS030 . BIGTABLE
> WHERE CAT2_CLIENT_NBR IN ( SELECT DISTINCT
> KEY_CUST_NUM FROM SYS030 . KEY_CUST_NUM )

S0001 BIGTEFF 39 SELECT 0.04
S0004 BIGTEFF 53 SELECT 0.00
S0003 BIGTEFF 46 SELECT 0.00

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 6-27 F04 report

Now it was time to talk to our DBA. The WHEN clause is valid though poorly constructed. The use of a table “join” could improve performance. Our DBA also suggested that the addition of table indices and running the RUNSTATS utility might also improve performance.

APA has a facility that allows the comparison of before and after reports. Once we had made the changes identified so far, we reran the observation session, making sure that none of the parameters or options had changed. A comparison of the two reports highlighted the change in performance.

Lastly, we created an Adobe PDF report for distribution and later review. Using the A02 primary command we displayed the Request Printed Reports screen (Figure 6-28).

```
File View Navigate Help
A02: Request Printed Reports (0534) Row 00001 of 00062
Command ==> _____ Scroll ==> CSR

Enter / to include a section in the report, blank to exclude the
section, S to include the section and set formatting options. Use
UP/DOWN (PF7/PF8) to scroll the list of report sections. After
entering your selections, press ENTER to generate the report JCL.

  Select   Report Section

      /      S01 Measurement Profile
      -      S02 Load Module Attributes
      -      S03 Load Module Summary
      -      S04 TCB Summary
      -      S05 Memory Usage Timeline
      -      S06 Data Space Usage Timeline
      -      S07 TCB Execution Summary
      -      S08 Processor Utilization Summary
      -      S09 Measurement Analysis

F1=Help  F2=Split  F3=End    F4=Jump  F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left   F11=Right F12=Cancel
```

Figure 6-28 A02 - report request screen

To reduce the size of the report, yet still have the relevant data, we selected only those reports we had worked with online. Once selected, we pressed the Enter key.

This positioned us at the screen used to create the JCL for the report creation job. Note that the data set for the PDF file had to have been previously created (Figure 6-29).

```
A02: Report JCL Submission (0534)
Command ==> _____ Scroll ==> CSR

Specify the following and press ENTER to either SUBMIT the print JCL
or to launch EDIT for the generated JCL.

Enter "/" to select options
  / produce PDF (Portable Document Format) file
  / generate JES-managed report file (SYSOUT=*)
  _ EDIT the generated JCL member, otherwise SUBMIT

Job Statement - edit if necessary
==> //DBA071P JOB (), 'CAZRPT01 ', CLASS=A, MSGCLASS=T, NOTIFY=&SYSUID
==> //*
==> //*

PDF File DSN (if applicable) must be cataloged FB 80
==> 'DBA071.PDF'

Location where generated JCL is to be saved
F1=Help      F2=Split    F3=End      F4=Jump     F5=Rfind    F7=Up
F8=Down      F9=Swap     F10=Left   F11=Right   F12=Cancel
```

Figure 6-29 A02 - report JCL creation

Pressing Enter submitted the job. Once the job completed, we were able to download and distribute the output PDF file. It had to be downloaded in binary form using a suitable FTP program.



Using APA to analyze batch performance

The scenario described in this chapter provides an example of Application Performance Analyzer being used to locate the statements that are the main contributors to the poor performance of a program.

7.1 Setting the scene

The operations team have noticed that a batch job, recently migrated to production, takes much longer to complete than was expected or predicted during development and test.

7.1.1 Starting APA

The systems programmers have set up the environment so that APA can be invoked from the ISPF menu. Having selected APA from the menu, the screen in Figure 7-1 was presented.

ReqNum	Owned By	Description	Job Name	Date/Time	Samples	Status
<u>0079</u>	DNET461	'for prueba de	CICSA0R5	May-12 15:13	50,000	Ended
<u>0078</u>	DBA071	APA CICS DB2	CICSA0R5	May-12 12:55	50,000	Ended
<u>0077</u>	DNET461	For apa demo	CICSA0R5	May-12 9:40	50,000	Ended
<u>0076</u>	DNET328		CICSA0R5	May-12 5:34	60,000	Ended
<u>0074</u>	SYS029	KJC - APA Measu	CICSA0R5	Jun-19 14:55	50,000	Ended
<u>0072</u>	DNET047	Demo	DNET047X	Jun-19 13:02	30,000	Failed
<u>0071</u>	DNET070	CICS	DNET070	Jun-19 8:55	1	Ended
<u>0070</u>	DNET070	DB2	DNET070	Jun-19 8:54	1	Ended
<u>0069</u>	DNET070	BATCH I/O	DNET070	Jun-19 8:54	1	Ended
<u>0067</u>	DNET070	BATCH	DNET070	Jun-19 8:51	1	Ended
<u>0065</u>	DNET187	batch example s	DNET187A	Jun-18 13:16	5,000	Ended

Welcome to IBM APA for z/OS ISPF Version 8.100E. You are currently connected to measurement task id CAZ0. Enter CONNECT for an alternate connection, VERSION for version information, NEW to start a measurement.

F9=Swap F10=Left F11=Right F12=Cancel

Figure 7-1 Existing observation sessions

7.1.2 Building the observation session

To capture activity associated with program execution, we built an observation session to specify:

1. The amount of time we wish to monitor
2. How many samples to take in this time
3. The name of the job or job step for which data was to be captured

In this example, we specified a jobname that started with our userid and asked for 100,000 samples in two minutes.

An observation session is created by issuing the `new` command. We refer to commands entered on the command line as *primary* commands and those entered in the left side column as *line* commands. In this example, we used a primary command. Doing this, the next screen you see is the Schedule New Measurement screen, which had been initialized with the name of the MVS system we were running in, and our userid (Figure 7-2).

```

File View Navigate Help
-----
R03: Schedule New Measurement                               Row 00001 of 00012
Command ==> _____ Scroll ==> CSR

1. Job Information      3. Multi Steps        5. CICS/IMS          7. Schedule
2. Options              4. Active Jobs        6. Sysplex           8. Sched Options

Panel 1. Job Information

Job Name/Pattern . . . _____ System Name . . . DEMOMVS
                               (Inactive)

Step Specification
Step No. . . . . _____ Specify step number, program name,
Program Name . . . _____ step name or step name + Proc step
Step Name . . . _____ name. Use panel 3 to specify more
ProcStepName . . . _____ than one step.

Description . . . _____

Number of Samples . . . _____ Measure to step end . . . N
Duration (min:sec) . . . _____ Delay by (secs) . . . _____
Notify TSO User . . . DBA071 Retain file for (days) . . . _____

F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 7-2 R03 - specify job detail

We then entered the required values (Figure 7-3).

```

File  View  Navigate  Help
-----
R03: Schedule New Measurement                               Row 00001 of 00012
Command ==> _____ Scroll ==> CSR

1. Job Information      3. Multi Steps      5. CICS/IMS          7. Schedule
2. Options             4. Active Jobs     6. Sysplex           8. Sched Options

Panel 1. Job Information

Job Name/Pattern . . . DBA071*      System Name . . . DEMOMVS
                        (Inactive)

Step Specification
  Step No. . . . . 2_____ Specify step number, program name,
  Program Name . . . _____ step name or step name + Proc step
  Step Name . . . . _____ name. Use panel 3 to specify more
  ProcStepName . . . _____ than one step.

Description . . . . APA batch
Number of Samples . 100000 Measure to step end . . . N
Duration (min:sec) . 2:00_____ Delay by (secs) . . . . _____
Notify TSO User . . . DBA071_____ Retain file for (days) . _____

F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left  F11=Right  F12=Cancel
  
```

Figure 7-3 R03 - job detail entered

Active Jobs screen: We then submitted the job that we believed had a performance problem. After again hitting the Enter key, we saw that our job was active.

This particular batch application that we were analyzing only accesses VSAM data sets, and executes solely in step two of the job stream. We were not in a sysplex environment and we wanted to do the analysis immediately. Hence there was no requirement for any of the other measurement options to be used (Figure 7-4).

```

_ File View Navigate Help
-----
R03: Schedule New Measurement                               Row 00001 of 00003
Command ==> _____ Scroll ==> CSR

  1. Job Information   3. Multi Steps     5. CICS Options   7. Schedule
  2. Options          4. Active Jobs    6. Sysplex       8. Sched Options

Panel 4. Active Jobs
-----
Enter S to select an active job step to be measured.   Prefix . . DBA071*

  JobName  Type  JobId   StepName  ProcStep  ASIDX  System   CPU%   SIO
  -  DBA071  TSO   TSU13212 SPIFFY           0114  DEMOMVS   0.00   0.00
  -  DBA071R JOB   JOB13213 RUNSAM1           0036  DEMOMVS  47.66  12.00

F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up    F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 7-4 R03 - select active jobs

We selected our job by using an `S` line command against it and pressing the Enter key twice. This returned us to the initial APA screen, the Observations List. We saw that our job was being monitored in real time — pressing the Enter key caused the number of samples to change, eventually reaching the 100,000 that we requested.

While the observation session was still in progress, we selected it using an `r` line command.

We caught the session after 3,530 samples of the required 100,000 had been taken (Figure 7-5).

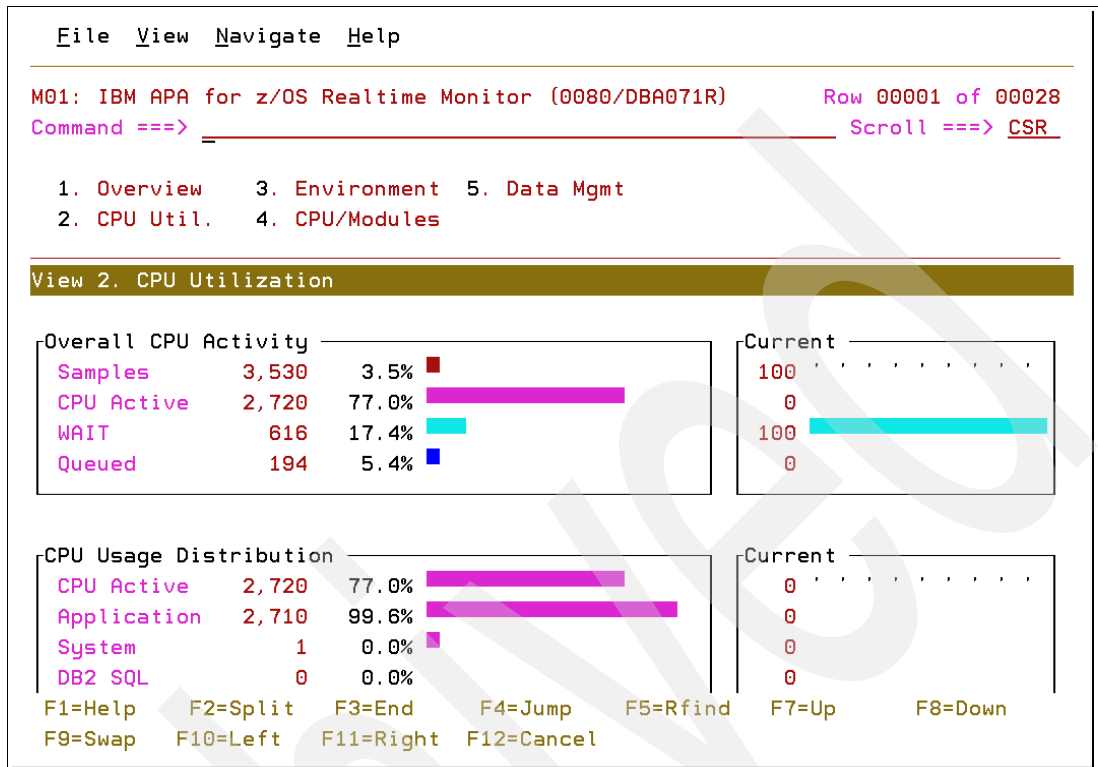


Figure 7-5 M01 report - real time monitoring

When the observation session was completed, the reports were available for analysis. Again, by using an `r` line command and pressing Enter, we saw the master report menu. We used the `S` line command to open the observation and then selected the **S01 Measurement Profile** report.

This is only a portion of the S01 report. We used PF7 and PF8 to scroll through it (Figure 7-6).

File View Navigate Help			
S01: Measurement Profile (0080/DBA071R)		Row 00083 of 00099	
Command ==>		Scroll ==> CSR	
Total samples	100,000	Duration	1 min 59.99 sec
Sampling rate	833.40 per sec	Sample file size	13.97MB
CPU/WAIT samples	98,867	Meas significance	98.86%
TCB samples	100,000	CPU queued samples	1,133
Overall CPU	14.13%		
Pages in	0	EXCPs	21,721
Pages out	0		
CPU consumption			
CPU active samples	38,107	CPU time TCB	43.99 sec
CPU active time	38.10%	CPU time SRB	1.33 sec
CPU WAIT samples	60,760	Service Units	1,158,338
CPU WAIT time	60.76%	Measurement SRB	1.49 sec
F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down			
F9=Swap F10=Left F11=Right F12=Cancel			

Figure 7-6 S01 report

We have noted that CPU appears to be the biggest portion of the active processing. The C02 report can provide more detail.

We used PF3 to return to the Performance Reports screen, selected **C** for **CPU Usage Analysis** and then selected the **C02 CPU Usage by Module** report. See Figure 7-7 and Figure 7-8.

```

File View Navigate Help
-----
R01: IBM APA for z/OS Performance Reports (0533)           Row 00001 of 00007
Command ==> _____ Scroll ==> CSR

Select a category from the list to the right to view the available reports in the selection list below.
_ A Admin/Miscellaneous      _ I IMS Measurement
_ S Statistics/Storage      _ E CICS Measurement
_ C CPU Usage Analysis       _ F DB2 Measurement
_ D DASD I/O Analysis       _ Q MQ Measurement
_ W CPU WAIT Analysis       _ G Coupling Facility
_ H HFS Analysis            _ J Java Measurement
_ V Variance Reports

More: +
Enter S to make a selection or enter the report code on the command line

_ C01 CPU Usage by Category      _ C07 CPU Usage by Procedure
_ C02 CPU Usage by Module        _ C08 CPU Referred Attribution
_ C03 CPU Usage by Code Slice    _ C09 CPU Usage by PSW/ObjCode
_ C04 CPU Usage Timeline
_ C05 CPU Usage Task/Category
_ C06 CPU Usage Task/Module
F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left F11=Right F12=Cancel

```

Figure 7-7 Selecting C02 - Usage by Module report

```

File View Navigate Help
-----
C02: CPU Usage by Module (0080/DBA071R)           Row 00001 of 00035
Command ==> _____ Scroll ==> CSR

Name      Description      Percent of CPU Time * 10.00% ±0.5%
*...1...2...3...4...5...6...7...8...9
+AM2V     Application Program      98.40
IDA019L1  Virtual I/O (VIO) and VSAM  0.90
IEAVELK   Supervisor Control       0.34
IAXPQ     Real storage manager (RSM)  0.09
ICYSTOR   VSAM/Media manager        0.07
IGZCPCO   COBPACK                   0.05
IEAVETCL  System trace               0.02
IDAEFSIO  Virtual I/O (VIO) and VSAM  0.01
ICYRDWR   VSAM/Media                 0.01
F1=Help  F2=Split  F3=End  F4=Jump  F5=Rfind  F7=Up  F8=Down
F9=Swap  F10=Left F11=Right F12=Cancel

```

Figure 7-8 C02 report

It was obvious from this report that SAM2V was using most of the CPU. We used the + line command to get to the SAM2V CSECT (Figure 7-9).

File View Navigate Help			
C02: CPU Usage by Module (0080/DBA071R)			Row 00001 of 00037
Command ==>			Scroll ==> CSR
Name	Description	Percent of CPU Time * 10.00%	±0.5%
		*...1...2...3...4...5...6...7...8...9	
<u>SAM2V</u>	Application Program	98.40	
→ <u>SAM2V</u>	CSECT in SAM2V	98.40	
<u>IDA019L1</u>	Virtual I/O (VIO) and VSAM	0.90	
<u>IEAVELK</u>	Supervisor Control	0.34	
<u>IAXPQ</u>	Real storage manager (RSM)	0.09	
<u>ICYSTOR</u>	VSAM/Media manager	0.07	
<u>IGZPCCO</u>	COBPACK	0.05	
<u>IEAVETCL</u>	System trace	0.02	
<u>IDAEFISIO</u>	Virtual I/O	0.01	
F1=Help	F2=Split	F3=End	F4=Jump
F9=Swap	F10=Left	F11=Right	F12=Cancel
		F5=Rfind	F7=Up
			F8=Down

Figure 7-9 C02 report - module expanded

At this time we did not have the program source code loaded.

Using the A01 primary command, we specified where the source listing could be found. Using the L line command caused the listings to be loaded where they could be accessed by APA (Figure 7-10).

```

File View Navigate Help
-----
A01 - Source Program Mapping (0533)                               Row 00001 of 00014
Command ==> _____ Scroll ==> CSR

Enter the following information to specify a source mapping file to be
used in the analysis of this measurement information.

File type . . . . _ (L=listing, A=ADATA, S=LANGX SideFile, D=SYSDEBUG)
Dataset name . . . _____
                    (Leave blank to search A04 dataset list)
Member name . . . _____ Match on Compile Date & Time  N

Seqn  ID-ReqNum  Load  Type/Status  Lang  Member  DSN
0001  CAZ0-0034  NO    L-Loaded    COB   SAM1V   DBA071.ADLAB.LISTING
0002  CAZ0-0034  NO    L-Loaded    COB   SAM2V   DBA071.ADLAB.LISTING
0003  CAZ0-0040  NO    L-Loaded    COB   SAM1V   DBA071.ADLAB.LISTING
0004  CAZ0-0040  NO    L-Loaded    COB   SAM2V   DBA071.ADLAB.LISTING
0005  CAZ0-0042  NO    L-Loaded    COB   SAM1V   DBA071.ADLAB.LISTING
0006  CAZ0-0042  NO    L-Loaded    COB   SAM2V   DBA071.ADLAB.LISTING
0007  CAZ0-0329  NO    L-Loaded    COB   SAM2V   DBA071.ADLAB.LISTING
0008  CAZ0-0329  NO    L-Loaded    COB   SAM1V   DBA071.ADLAB.LISTING

F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 7-10 A01 screen - load source data

PF3 took us back to the C02 report.

Using the P line command on the line listing, the SAM2V CSECT allowed us to view the program source code. Scrolling down, we discovered several source statements that were responsible for most of the CPU usage (Figure 7-11).

```

File View Navigate Help
-----
P01: Source Program Attribution (0533)                               Row 00216 of 00232
Command ==> _____ Scroll ==> CSR

LineNo Offset Count Source Statement
-----
000215                                300-PROCESS-CPU-CRUNCH.
000216 000864                                MOVE 0 TO LOOP-COUNT.
000217 00086A 409 310-CRUNCH-LOOP
000218                                UNTIL LOOP-COUNT > CRUNCH-CPU-LOOPS .
000219
000220                                310-CRUNCH-LOOP.
000221                                * CALUCLUATE AVERAGE OF ALL MONTHS
000222 0008AA 15 MOVE 0 TO WORK-SUM.
000223 0008B0 34 MOVE 1 TO MONTH-SUB.
000224 0008B6 190 PERFORM VARYING MONTH-SUB FROM 1 BY 1 UNTIL (MON
000225 0008C0 9999+ IF CUST-MONTH(MONTH-SUB) IS NOT NUMERIC
000226 00090A                                MOVE 0 TO CUST-MONTH(MONTH-SUB)
000227                                END-IF
000228 000942 9999+ COMPUTE WORK-SUM = WORK-SUM + CUST-MONTH(MONTH
000229 000994 9999+ END-PERFORM .
000230 0009C2 706 COMPUTE MONTH-AVERAGE = WORK-SUM / 12 .

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 7-11 P01 report

The application seems to be spending most of its time in the code between offsets 8C0 and 9C2.

Additional detail about CPU usage can be found in the C09 report, CPU Usage by PSW/Object Code.

Using a C09 primary command took us to the required report screen (Figure 7-12).

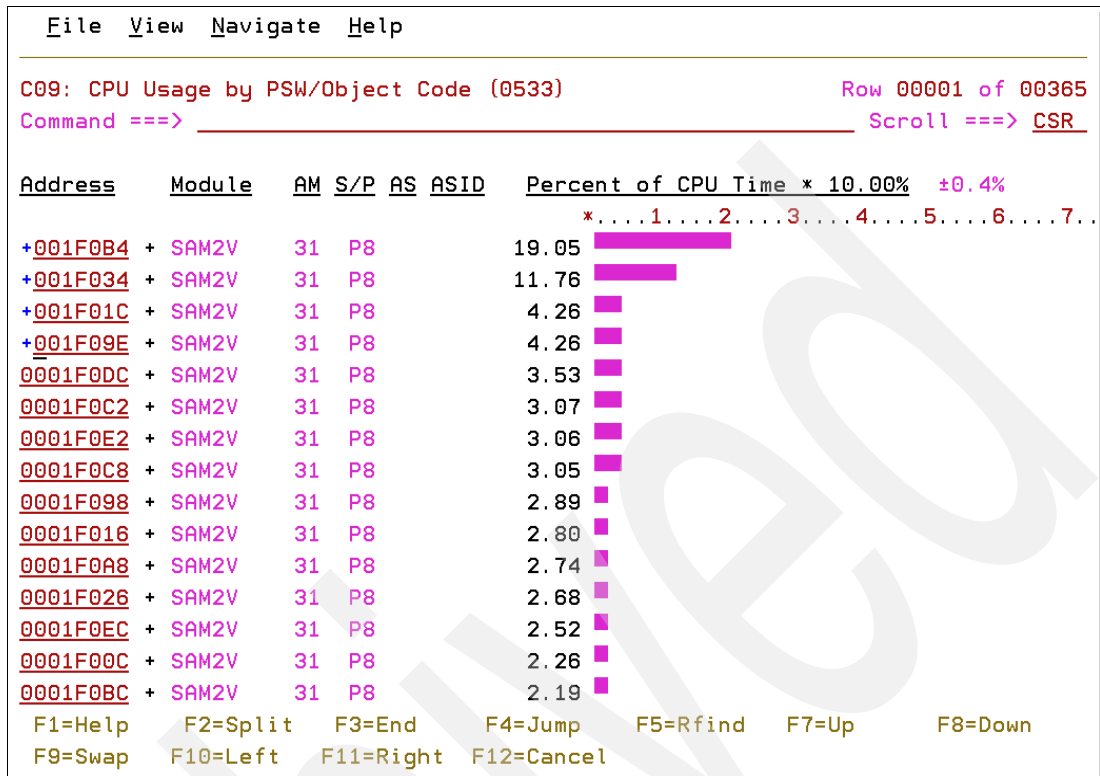


Figure 7-12 C09 report

To obtain more detail of the identified high usage code sections we entered a '+' line command adjacent to each and pressed the Enter key (Figure 7-13).

File View Navigate Help									
C09: CPU Usage by PSW/Object Code (0533)							Row 00001 of 00382		
Command ==> _____							Scroll ==> CSR		
Address	Module	AM	S/P	AS	ASID	Percent of CPU Time * 10.00%	±0.4%		
							* 1 2 3 4 5 6 7		
0001F0B4	- SAM2V	31	P8			19.05	██████████		
→	SAM2V+0976	F090	D118	0002	SRP	280 (10, R13), 2 (R0), 0			
→	SAM2V+097C	960F	D121		OI	289 (R13), 15			
0001F034	- SAM2V	31	P8			11.76	██████████		
→	SAM2V+08F6	DD04	304A	A0C6	TRT	74 (5, R3), 198 (R10)			
→	SAM2V+08FC	1832			LR	R3, R2			
→	SAM2V+08FE	47D0	B3C2		BC	13, 962 (, R11)			
0001F01C	- SAM2V	31	P8			4.26	██████		
→	SAM2V+08DE	FCB2	D138	D121	MP	312 (12, R13), 289 (3, R13)			
→	SAM2V+08E4	D202	D13C	C02A	MVC	316 (3, R13), 42 (R12)			
F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down									
F9=Swap F10=Left F11=Right F12=Cancel									

Figure 7-13 C09 report - modules expanded

The report verified that these areas of high usage were in the same area as the statements identified as causing the highest usage.

We noted that the assembler instructions involved were SRP (Shift and Round Decimal), TRT (Translate and Test) and MP (Multiply Decimal). These are all expensive assembler instructions, indicating that data has to be converted before it can be used in an arithmetic computation. This suggests that we should review the data definitions used in the program.

We navigate back to our program listing report by selecting the **Navigate** option at the top of the screen.

We then selected option 1 (Figure 7-14).

```

File View Navigate Help
-----
C09: CPU Us  1 1. Select Window (WIN)
Command ===  2. Toggle Window (JUMP/PF4)
              3. Close and End (END/PF3)
Address      4. Retain and End (CANCEL/PF12)
              5. Jump to Menu (MENU)
0001F0B4 -   2.....3.....4.....5.....6.....7..
                                     Time * 10.00% ±0.4%
                                     █

→ SAM2V+0976   F090 D118 0002   SRP   280 (10, R13) , 2 (R0) , 0
→ SAM2V+097C   960F D121         OI    289 (R13) , 15

0001F034 - SAM2V   31 P8           11.76 █

→ SAM2V+08F6   DD04 304A A0C6   TRT   74 (5, R3) , 198 (R10)
→ SAM2V+08FC   1832           LR    R3, R2
→ SAM2V+08FE   47D0 B3C2       BC    13, 962 (, R11)

0001F01C - SAM2V   31 P8           4.26 █

→ SAM2V+08DE   FCB2 D138 D121   MP    312 (12, R13) , 289 (3, R13)
→ SAM2V+08E4   D202 D13C C02A   MVC   316 (3, R13) , 42 (R12)
F1=Help  F2=Split  F3=End   F4=Jump  F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right F12=Cancel

```

Figure 7-14 Windows navigation

Pressing the Enter key took us to the list of open APA reports.

From here we select the P01 entry (Figure 7-15).

```
File View Navigate Help
-
C
C
Enter S to Jump to Window:
A  _ R02: IBM APA for z/OS Observation List (CAZ0)
O  _ R01: IBM APA for z/OS Performance Reports (0533)
   _ C02: CPU Usage by Module (0533)
     _ P01: Source Program Attribution (0533)
     = C09: CPU Usage by PSW/Object Code (0533)
+
+
O
+
+
F1=Help  F2=Split  F3=End   F4=Jump  F5=Rfind  F7=Up
F8=Down  F9=Swap   F10=Left F11=Right

0001F01C - SAM2V    31 P8          4.26 █
→ SAM2V+08DE      FCB2 D138 D121  MP    312(12,R13),289(3,R13)
→ SAM2V+08E4      D202 D13C C02A  MVC    316(3,R13),42(R12)
F1=Help  F2=Split  F3=End   F4=Jump  F5=Rfind  F7=Up    F8=Down
F9=Swap  F10=Left  F11=Right F12=Cancel
```

Figure 7-15 Selecting a previous window

This took us to the screen shown earlier in Figure 7-11 on page 157.

At this point we could not see the data declarations in the program. We selected the **View** option from the top of the screen and used option **1, Report Setup**.

Here we selected **Display ALL statements of the source program**.

From the P01 report (Figure 7-11 on page 157) we knew that the data variables MONTH-AVERAGE and MONTH-SUB are involved. Using the FIND MONTH FIRST command, we located the lines where these were declared. See Figure 7-16 and Figure 7-17.

```

File View Navigate Help
-----
P01: Source Program Attribution (0533)                               Row 00036 of 00232
Command ==> _____ Scroll ==> CSR

LineNo Offset Count Source Statement
000035          05 WORK-SUM          PIC 9(16)          V
000036          05 MONTH-AVERAGE    PIC 9(16)          V
000037          05 MONTH-SUB          PIC 9(16)          V
000038
000039          *****
000040          LINKAGE SECTION.
000041
000042          COPY CUSTCOPY REPLACING ==:TAG:== BY ==CUST==.
000043          *** *****
000044          * SAMPLE COBOL COPYBOOK FOR IBM PD TOOLS WORKSHOPS
000045          *
000046          * THE SAMPLE DATA DESCRIBED BY THIS COPY BOOK
000047          * IS <USERID>.ADLAB.CUSTFILE
000048          *
000049          * The following File Manager OPTION:
000050          * 6. COBOL compiler specifications
F1=Help   F2=Split  F3=End    F4=Jump   F5=Rfind  F7=Up     F8=Down
F9=Swap   F10=Left  F11=Right F12=Cancel

```

Figure 7-16 Program source


```

File View Navigate Help

P01: Source Program Attribution (0533) Row 00070 of 00232
Command ==> _____ Scroll ==> CSR

LineNo Offset Count Source Statement
000069          05 CUST-COUNTRY PIC X(11).
000070          05 CUST-MONTH PIC S9(7)V99 COMP-3
000071          05 CUST-OCCUPATION PIC X(30).
000072          05 CUST-NOTES PIC X(120).
000073          05 CUST-DATA-1 PIC X(05).
000074          05 CUST-DATA-2 PIC X(40).
000075          01 CUST-CONTACT-REC.
000076          05 CUST-CONTACT-KEY.
000077          10 CUST-CONTACT-ID PIC X(5).
000078          10 CUST-CONTACT-REC-TYPE PIC X.
000079          05 CUST-CONTACT-NAME PIC X(17).
000080          05 CUST-DESCRIPTION PIC X(10).
000081          05 CUST-CONTACT-INFO PIC X(20).
000082          * 05 :TAG:-DATA-3 PIC X(05).
000083          05 CUST-DATA-3 PIC 99999 BINARY.
000084          * 05 :TAG:-DATA-4 PIC X(05).

F1=Help F2=Split F3=End F4=Jump F5=Rfind F7=Up F8=Down
F9=Swap F10=Left F11=Right F12=Cancel

```

Figure 7-17 Program source

From here we deduced that making these fields binary (COMP) rather than PIC would reduce the high resource utilization currently being experienced.

7.2 Summary

This scenario highlights the benefits of using APA at all points in the application development life cycle. Had it been used in an earlier phase, the performance hit could have been fixed before the program went into production.

A useful exercise in a scenario such as this is to make the identified changes and then re-do the analysis. APA has functionality that allows the comparison of “before” and “after” reports, highlighting the differences.

Reports of poor performance can be subjective. A tool such as Omegamon can introduce objectivity into performance observations by alerting you when a threshold is reached. This alert can then be used to dynamically trigger an APA observation session to collect performance data for later analysis.

Archived



Optim MOVE test data capture

The scenario described in this chapter provides an example of Optim MOVE for DB2 being used to capture test data from multiple production data sources.

8.1 Introduction

Application testing is becoming a high priority because resolving problems early in the development stages can significantly reduce costs, but without the appropriate solutions, developers often have to write complex extract programs just to create the test data. IBM Optim MOVE for DB2 is a tool used to extract data quickly and easily from production files and databases to create subsets of that data for use in test environments.

Optim MOVE supports customers' governance and compliance requirements by masking sensitive fields like customer name, or social security number while retaining essential structural features such as credit card check digits, and referential integrity, even across different data sources including DB2, VSAM, IMS, and z/OS files.

Optim MOVE for DB2 is available as part of IBM's Problem Determination Tools family. For more information on Optim MOVE, please visit:

<http://ibm.com/software/awdtools/optimmove>

The following diagrams (Figure 8-1, Figure 8-2, and Figure 8-3) show the production data relationships from which we extracted our test data.

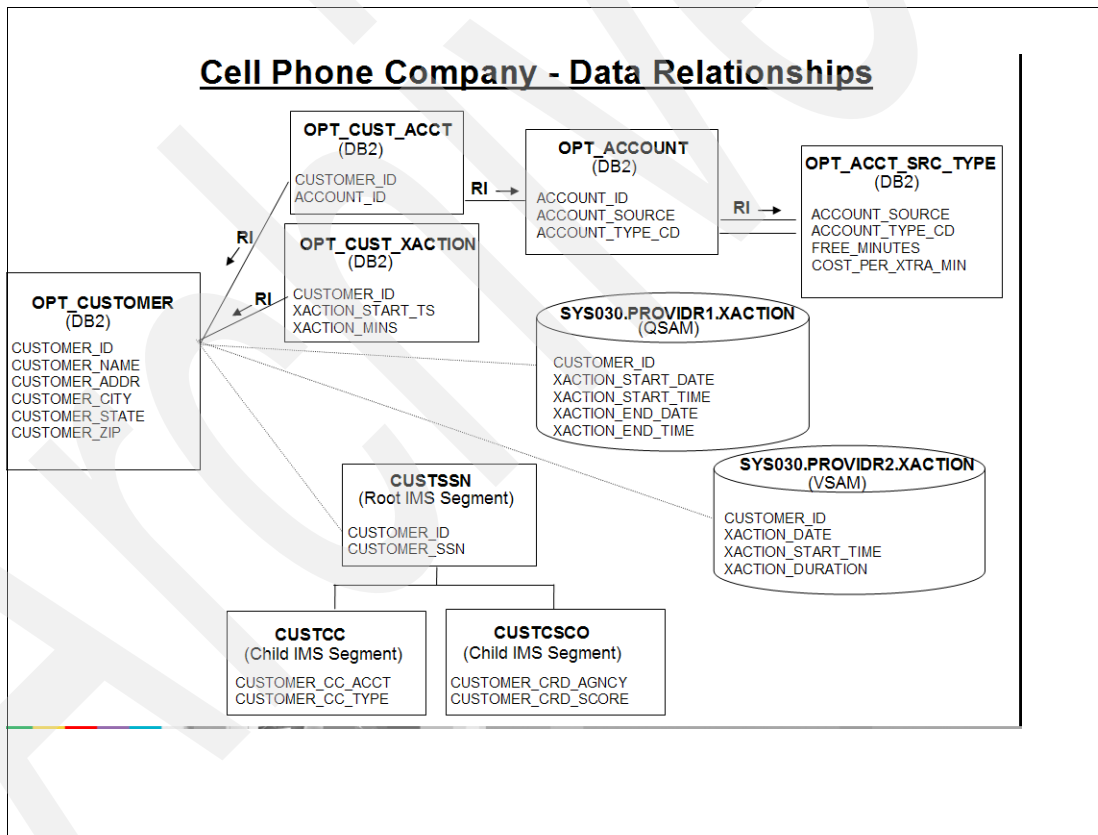


Figure 8-1 Data relationships

Note: RI = Referential Integrity

Cell Phone Company - Data Locations

➤ **DB2 Tables on Subsystem DSNC**

- ❖ SYS030.OPT_CUSTOMER
- ❖ SYS030.OPT_CUST_ACCT
- ❖ SYS030.OPT_ACCOUNT
- ❖ SYS030.OPT_ACCT_SRC_TYPE
- ❖ SYS030.OPT_CUST_XACTION

➤ **QSAM File**

- ❖ Data: SYS030.PROVIDR1.XACTION
- ❖ Copybook: SYS030.CELLPHON.COPYBOOK(PROVIDR1)

➤ **VSAM File**

- ❖ Data: SYS030.PROVIDR2.XACTION
- ❖ Copybook: SYS030.CELLPHON.COPYBOOK(PROVIDR2)

Figure 8-2 Data locations

Cell Phone Company - Data Locations

➤ **IMS Database**

- ❖ DBD
 - Source: SYS030.CELLPHON.DBDSRC(CSTPRV)
SYS030.CELLPHON.DBDSRC(CSRPRVI)
 - Assembled: SYS030.CELLPHON.DBDLIB(CSTPRV)
SYS030.CELLPHON.DBDLIB(CSTPRVI)
- ❖ Data Set for CSTPRV: SYS030.CELLPHON.CSTPRV
- ❖ Data Set for CSTPRVI: SYS030.CELLPHON.CSTPRVI
- ❖ Read Only PSB
 - Source: SYS030.CELLPHON.PSBSRC(CSTPRVG)
 - Assembled: SYS030.CELLPHON.PSBLIB(CSTPRVG)
- ❖ Read / Write PSB
 - Source: SYS030.CELLPHON.PSBSRC(CSTPRVA)
 - Assembled: SYS030.CELLPHON.PSBLIB(CSTPRVA)
- ❖ Segments and Copybooks
 - CUSTSSN - SYS030.CELLPHON.COPYBOOK(ICUSTSSN)
 - CUSTCC - SYS030.CELLPHON.COPYBOOK(ICUSTCC)
 - CUSTCSCO - SYS030.CELLPHON.COPYBOOK(ICUSTCS)

Figure 8-3 Data locations (continued)

The objects that we had to define were:

- ▶ Access Definition: This is the description of the data to be extracted.
- ▶ Relationships: These determine how tables are related. A relationship can be defined in the DB2 catalog, in IMS, or in the OPT directory.
- ▶ Legacy Table: This is an Optim object that describes the legacy data and maps it to a pseudo DB2 table format. Optim MOVE for Legacy, a companion product to Optim MOVE for DB2, uses Legacy Tables to incorporate legacy data from IMS, VSAM, or sequential data into Optim MOVE processes.
- ▶ Environment Definition: This identifies the IMS program libraries, the DBD and PSB libraries, the DFSVSAMP data set and member name, as well as the IMS system ID and AGN used with the IMS Legacy Tables.

The steps in the extract process were as follows:

1. Define the IMS environment.
2. Define IMS segments as legacy tables.
3. Define VSAM and QSAM files as legacy tables.
4. Define parameters for the data extract:
 - a. Data sources and relationships
 - b. Selection criteria
 - c. IMS parameters
 - i. PSB for the extract process
 - ii. Dataset names for the IMS data base
5. Extract selected data into a sequential file.
6. View the extract report.
7. Convert the extracted data.
8. Delete previously defined objects (if required).

We start our scenario description from the Optim MOVE master menu screen (Figure 8-4).

```
----- IBM's Optim -----
0  OPTIONS          - Site and User Options          SQLID ==> DNET999
*  BROWSE TABLE    - Browse a DB2 Table          SUBSYS ==> DSNC
*  EDIT TABLE      - Edit a DB2 Table
*  BROWSE USING AD  - Browse DB2 Tables Using Access Definition
*  EDIT USING AD    - Edit DB2 Tables Using Access Definition
5  ADS              - Create or Modify Access Definitions
6  DEFINITIONS      - Maintain Optim Definitions (Keys, Maps, ...)
7  MIGRATION        - Data Migration - Extract, Insert, Update, ...
*  COMPARE          - Compare Two Sets of Data
*  ARCHIVE          - Archive and Restore Data

T  TUTORIAL         - Information About IBM's Optim
C  CHANGES         - Changes from Prior Release(s)
X  EXIT            - Terminate Product Use

5655-U32 (C) Copyright IBM Corporation 1989, 2007.
All rights reserved. Licensed materials - property of IBM.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP schedule contract with IBM Corp.
Optim 5.5.2 Build: 276

OPTION ==>
```

Figure 8-4 Main menu screen

From here we selected option 6.

Notes:

- ▶ The Optim menu items tagged with an asterisk are not activated for this installation of Optim MOVE for DB2.
- ▶ Throughout this chapter, the term *tables* refers to DB2 tables, views, aliases, synonyms, as well as Legacy Tables.
- ▶ DBA071 was the userid we used for this scenario.

8.2 Provide IMS environment parameters

Through an IMS Environment Definition, Optim provides a consolidated way to define the information required to access IMS data. The Environment Definition simplified the definition of IMS objects by allowing us to specify the names of the DBD, PSB, and IMS Program Libraries to be used during processing.

In the Definition Option menu, we selected option 7 (Figure 8-5).

```
----- Choose a Definition Option -----
OPTION ==> 7_
                                SQLID ==> "DBA071"
1 PRIMARY KEYS   - Maintain Primary Keys    SUBSYS ==> DSNC
2 RELATIONSHIPS - Maintain Relationships
3 COLUMN MAPS   - Maintain Column Maps
4 TABLE MAPS   - Maintain Table Maps
5 ADS           - Maintain Access Definitions
6 LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7 IMS ENVIRONMENT - Maintain IMS Environment Definitions
8 IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E EXPORT        - Export Optim Object Definitions
I IMPORT        - Import Optim Object Definitions
```

Figure 8-5 Choose definition option - IMS

Pressing Enter took us to the IMS Environment definition screen.

Here we entered the name of our IMS environment (Figure 8-6).

```
----- Choose an IMS Environment -----
Command ==>
                                SUBSYS: DSNC

IMS Environment:
  Environment Name ==> DBA071_

Use '_' for DB2 LIKE character ==> YES  Y-Yes, N-No
```

Figure 8-6 Environment definition selection

We entered an explicit Environment name, in this example our userid. By leaving the name blank or keying a %, a display of all existing environments would have been displayed. Because the name keyed did not exist, an empty Define IMS Environment screen was returned (Figure 8-7).


```

----- Define IMS Environment DBA071 -----
Command ==>
                                                    Scroll ==> CSR
                                                    SUBSYS: DSNC

IMS Program Libraries   ==> 'IMS.V9R1.SDFSRESL'
                        ==>
                        ==>

IMS DBD and PSB Libraries ==> 'SYS030.CELLPHON.DBDLIB'
                        ==> 'SYS030.CELLPHON.PSBLIB' _
                        ==>
                        ==>

DFSVSAMP DSN/Member Name ==> 'IMS.V9R1.PROCLIB(DFSVSMOA)'

If data is online to IMS, enter the IMS System ID ==>      AGN ==>

```

Figure 8-7 Define IMS environment

Into this screen we keyed the names of the IMS program libraries, the names of the DBD and PSB libraries, and the name of the data set that contained the IMS buffer parameters. The Define IMS Environment screen provides the detail with which the IMS Legacy Tables will be associated — a Legacy Table must always be associated with a data source.

Pressing PF3 saved the definition we had just created. Hitting PF3 again returned us to the Definition Option screen.

8.3 Define IMS segments as Legacy Tables

Having successfully defined the IMS environment, we then had to define the Legacy Tables to represent the IMS segments containing the data we had to extract. This was done by selecting option 6 (Figure 8-8).

```

----- Choose a Definition Option -----
OPTION ==> 6_
                                                    SQLID ==> "DBA071"
                                                    SUBSYS ==> DSNC

1 PRIMARY KEYS - Maintain Primary Keys
2 RELATIONSHIPS - Maintain Relationships
3 COLUMN MAPS - Maintain Column Maps
4 TABLE MAPS - Maintain Table Maps
5 ADS - Maintain Access Definitions
6 LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7 IMS ENVIRONMENT - Maintain IMS Environment Definitions
8 IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E EXPORT - Export Optim Object Definitions
I IMPORT - Import Optim Object Definitions

```

Figure 8-8 Choose definition option - Legacy Tables

Each IMS Legacy Table must have a matching Environment Definition. That is, the Environment Definition and Creator ID names must be the same, in our case DBA071 (Figure 8-9).

```

----- Choose a Legacy Table -----
Command ==>
SUBSYS: DSNC

Legacy Table:
  Creator ID ==> DBA071
  Table Name ==> ICUSTSSN_

Use '_' for DB2 LIKE character ==> YES  Y-Yes, N-No

```

Figure 8-9 Choose a legacy table

The Table Name we chose, ICUSTSSN, did not exist. We pressed Enter, and Optim MOVE then prompted us for the information required to create a new table (Figure 8-10).

```

----- Specify Copybooks for Legacy Table: DBA071.ICUSTSSN -----
Command ==>
SUBSYS: DSNC
SCROLL ==> CSR

INSTRUCTIONS: Specify the name(s) of the Copybook members containing the record
description to be used for this Legacy Table. All members will
be concatenated before being parsed.

Legacy Table Type ==> I  I-IMS  F-File (VSAM or Sequential)

Copybook Language ==> C  C-COBOL, P-PL/I

Copybook(s) ==> 'SYS030.CELLPHON.COPYBOOK(ICUSTSSN)' _
==>
==>
==>
==>

```

Figure 8-10 Specify copybooks for legacy table

The copybook used to define our table was ICUSTSSN. It is written in COBOL ('C') and the data we were working with was in an IMS ('I') database. Pressing Enter caused Optim to parse the copybook information to populate the Legacy Table editor, as shown in the following screen (Figure 8-11).

```

----- Define Legacy Table: DBA071.ICUSTSSN -----
Command ==>
Scroll ==> CSR

Associated IMS DBD Name ==> *          Segment ==> * _
Row 1 of

Cmd Level/Field Name          Type Len  Occur Column Name
-----
*** ***** TOP *****
_  1 IMS-CUSTOMER-SSN-SEG          19      IMS_CUSTOMER_SSN_SEG
_  5 IMS-CUSTOMER-ID              CHR  10      IMS_CUSTOMER_ID
_  5 IMS-CUSTOMER-SSN              CHR   9      IMS_CUSTOMER_SSN
*** ***** BOTTOM *****

```

Figure 8-11 Define legacy table

Specifying an asterisk for both the IMS DBD name and the Segment name returned us a list of candidate values for each field. The list of potential IMS DBD names is composed of all the members in the IMS DBD and PSB libraries, identified when we defined the IMS environment (Figure 8-12).

```

----- Define Legacy Table: DBA071.ICUSTSSN -----
Command ==>
                                                    SCROLL ==> CSR
                                                    SUBSYS: DSNC
Associated IMS DBD Name ==> *           Segment ==> *
                                                    Row 1 of 3
Cmd Level/Field Name           Type Len Occur Column Name
-----
*** *****
___ 1 IMS-CUSTOM +---Member Selection List for Environment DBA071---+
___ 5 IMS-CUSTO | Cmd Member      1 OF 4
___ 5 IMS-CUSTO | -----
*** ***** | ***** TOP *****
                | S_  CSTPRV
                | _  CSTPRVA
                | _  CSTPRVG
                | _  CSTPRVI
                | ***** BOTTOM *****
                +-----+

```

Figure 8-12 Define legacy table

From the DBD list, we selected CSTPRV and pressed Enter, returning us a list of the segments associated with this DBD. We had to define a Legacy Table for each of these (Figure 8-13).

```

----- Define Legacy Table: DBA071.ICUSTSSN -----
Command ==>
                                                    SCROLL ==> CSR
                                                    SUBSYS: DSNC
Associated IMS DBD Name ==> *           Segment ==> *
                                                    Row 1 of
Cmd Level/Field Name           Type Len Occur Column Name
-----
*** *****
___ 1 IMS-CUSTOMER-S +---Segment Selection List for DBD CSTPRV---+ EG
___ 5 IMS-CUSTOMER- | Cmd Segment Name  1 OF 3
___ 5 IMS-CUSTOMER- | -----
*** ***** | ***** TOP *****
                | S_  CUSTSSN
                | _  CUSTCC
                | _  CUSTCSCO
                | ***** BOTTOM *****
                +-----+

```

Figure 8-13 Define legacy table

From this, we selected the root IMS segment, CUSTSSN. Pressing Enter saved the data and then PF3 returned us to the Choose a Legacy Table screen.

Repeating these steps, we then created Legacy Tables for the remaining two child segments, CUSTCC and CUSTCS, of CUSTSSN. Note that the IMS DBD and segment names had to be provided for each table definition. See Figure 8-14 through Figure 8-19.

```
----- Choose a Legacy Table ----- DEFINE CANCELLED
Command ==> _
SUBSYS: DSNCL

Legacy Table:
Creator ID ==> DBA071
Table Name ==> ICUSTCC

Use '_' for DB2 LIKE character ==> YES Y-Yes, N-No
```

Figure 8-14 Choose a legacy table

```
----- Specify Copybooks for Legacy Table: DBA071.ICUSTCC -----
Command ==>
SUBSYS: DSNCL
SCROLL ==> CSR

INSTRUCTIONS: Specify the name(s) of the Copybook members containing the record
description to be used for this Legacy Table. All members will
be concatenated before being parsed.

Legacy Table Type ==> I I-IMS F-File (VSAM or Sequential)
Copybook Language ==> C C-COBOL, P-PL/I
Copybook(s) ==> 'SYS030.CELLPHON.COPYBOOK(ICUSTCC)_'
==>
==>
==>
==>
```

Figure 8-15 Specify copybooks for legacy table

```
----- Define Legacy Table: DBA071.ICUSTCC ----- DBD NOT FOUND
Command ==>
Scroll ==> CSR

Associated IMS DBD Name ==> CSTRPV Segment ==> CUSTCC
Row 1 of 3

Cmd Level/Field Name Type Len Occur Column Name
-----
*** ***** TOP *****
___ 1 IMS-CUSTOMER-CRED-CARD-SEG 20 IMS_CUSTOMER_CRED_CARD_SEG
___ 5 IMS-CUSTOMER-CC-ACCT CHR 16 IMS_CUSTOMER_CC_ACCT
___ 5 IMS-CUSTOMER-CC-TYPE CHR 4 IMS_CUSTOMER_CC_TYPE
*** ***** BOTTOM *****
```

Figure 8-16 Define legacy table

```

----- Choose a Legacy Table ----- DEFINE CANCELLED
Command ==>
                                                    SUBSYS: DSNCL
Legacy Table:
  Creator ID ==> DBA071
  Table Name ==> ICUSTCS
Use '_' for DB2 LIKE character ==> YES  Y-Yes, N-No

```

Figure 8-17 Choose a legacy table

```

----- Specify Copybooks for Legacy Table: DBA071.ICUSTCS -----
Command ==>
                                                    SCROLL ==> CSR
                                                    SUBSYS: DSNCL
INSTRUCTIONS: Specify the name(s) of the Copybook members containing the record
              description to be used for this Legacy Table. All members will
              be concatenated before being parsed.

Legacy Table Type ==> I  I-IMS  F-File (VSAM or Sequential)
Copybook Language ==> C  C-COBOL, P-PL/I
Copybook(s) ==> 'SYS030.CELLPHON.COPYBOOK(ICUSTCS)_'
               ==>
               ==>
               ==>
               ==>

```

Figure 8-18 Specify copybooks for legacy table

```

----- Modify Legacy Table: DBA071.ICUSTCS -----
Command ==>
                                                    Scroll ==> CSR
Associated IMS DBD Name ==> _CSTPRV      Segment ==> CUSTCSCO
                                                    Row 1 of 3
Cmd Level/Field Name          Type  Len  Occur Column Name
-----
*** ***** TOP *****
___  1 IMS-CUSTOMER-CREDIT-SCORE-+      8      IMS_CUSTOMER_CREDIT_SCORE_SE
___  5 IMS-CUSTOMER-CREDIT-AGENCY CHR   4      IMS_CUSTOMER_CREDIT_AGENCY
___  5 IMS-CUSTOMER-CREDIT-SCORE  CHR   4      IMS_CUSTOMER_CREDIT_SCORE
*** ***** BOTTOM *****

```

Figure 8-19 Modify Legacy Table

This completed the definition of the IMS Legacy Tables. Our next step was to build Legacy Tables for the VSAM and QSAM datasets. PF3 returned us to the Legacy Table Selection screen.

8.4 Define VSAM and QSAM files as Legacy Tables

Here we entered the Creator ID and the name for our new Legacy Table (Figure 8-20).

```
----- Choose a Legacy Table -----
Command ==>
                                                    SUBSYS: DSNC
Legacy Table:
  Creator ID ==> DBA071
  Table Name ==> SRC@PROVIDER1XACTN _
Use '_' for DB2 LIKE character ==> YES   Y-Yes, N-No
```

Figure 8-20 Choose a legacy table

Pressing Enter took us to the Specify Copybooks screen. Note that the Legacy Table Type changes to F. We first defined a Legacy Table for our QSAM dataset (Figure 8-21).

```
----- Specify Copybooks for Legacy Table: DNET328.SRC@PROVIDER1XACTN -----
Command ==>
                                                    SCROLL ==> PAG
                                                    SUBSYS: DSNC
INSTRUCTIONS: Specify the name(s) of the Copybook members containing the reco
               description to be used for this Legacy Table. All members will
               be concatenated before being parsed.

Legacy Table Type ==> F   I-IMS  F-File (VSAM or Sequential)
Copybook Language ==> C   C-COBOL, P-PL/I
Copybook (s) ==> 'SYS030.CELLPHON.COPYBOOK (PROVIDR1) ' _
==>
==>
==>
==>

PF 1=HELP      2=SPLIT      3=END        4=RETURN     5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP      10=LEFT     11=RIGHT     12=RETRIEVE
```

Figure 8-21 Specify copybooks for legacy table

Pressing Enter took us to the Define Legacy Table screen showing the copybook layout (Figure 8-22).

```

----- Define Legacy Table: DBA071.SRC@PROVIDERIXACTN -----
Command ==> DS_ Scroll ==> CSR
Row 1 of 18

Cmd Level/Field Name          Type Len Occur Column Name
-----
*** ***** TOP *****
___ 1 PROVIDER1-XACTION-REC          38 PROVIDER1_XACTION_REC
___ 5 P1-CUSTOMER-ID                CHR 10 P1_CUSTOMER_ID
___ 5 P1-START-DATE                  8 P1_START_DATE
___ 10 P1-START-DATE-YYYY            NUM 4 P1_START_DATE_YYYY
___ 10 P1-START-DATE-MM              NUM 2 P1_START_DATE_MM
___ 10 P1-START-DATE-DD              NUM 2 P1_START_DATE_DD
___ 5 P1-XACTION-START-TIME          6 P1_XACTION_START_TIME
___ 10 P1-XACTION-START-HH           NUM 2 P1_XACTION_START_HH
___ 10 P1-XACTION-START-MM           NUM 2 P1_XACTION_START_MM
___ 10 P1-XACTION-START-SS           NUM 2 P1_XACTION_START_SS
___ 5 P1-END-DATE                    8 P1_END_DATE
___ 10 P1-END-DATE-YYYY              NUM 4 P1_END_DATE_YYYY
___ 10 P1-END-DATE-MM                NUM 2 P1_END_DATE_MM
___ 10 P1-END-DATE-DD                NUM 2 P1_END_DATE_DD
___ 5 P1-XACTION-END-TIME            6 P1_XACTION_END_TIME
PF 1=HELP    2=SPLIT    3=END      4=RETURN    5=RFIND    6=RCHANGE
PF 7=UP      8=DOWN     9=SWAP    10=LEFT     11=RIGHT   12=RETRIEVE

```

Figure 8-22 Define legacy table

As with IMS, a Legacy Table for VSAM or QSAM dataset must also be associated with a data source. To specify the data source information, we issued the DATASOURCE (DS) primary command. In the next screen we provided the Data Set Name of the QSAM data source (Figure 8-23).

```

Command ==> SUBSYS: DSNC Scroll ==> CSR
SUBSYS: DSNC
+-----Specify Data Source Information-----+
|
| Please specify the default Data Source information for this object.
| Enter END when complete.
|
| Data Source Type : Sequential or VSAM Dataset
| Data Source DSN ==> 'SYS030.PROVIDR1.XACTION'
|
+-----+
___ 10 P1-XACTION-START-HH          NUM 2 P1_XACTION_START_HH
___ 10 P1-XACTION-START-MM          NUM 2 P1_XACTION_START_MM
___ 10 P1-XACTION-START-SS          NUM 2 P1_XACTION_START_SS
___ 5 P1-END-DATE                    8 P1_END_DATE
___ 10 P1-END-DATE-YYYY              NUM 4 P1_END_DATE_YYYY
___ 10 P1-END-DATE-MM                NUM 2 P1_END_DATE_MM
___ 10 P1-END-DATE-DD                NUM 2 P1_END_DATE_DD
___ 5 P1-XACTION-END-TIME            6 P1_XACTION_END_TIME
PF 1=HELP    2=SPLIT    3=END      4=RETURN    5=RFIND    6=RCHANGE
PF 7=UP      8=DOWN     9=SWAP    10=LEFT     11=RIGHT   12=RETRIEVE

```

Figure 8-23 Define legacy table

Using PF3 twice returned us to the Choose a Legacy Table screen from where we repeated the process of defining a data source, this time for the VSAM data set. Note the confirmation message, LEGACY TABLE DEFINED, in the top right side corner (Figure 8-24 through Figure 8-27).

```
----- Choose a Legacy Table ----- LEGACY TABLE DEFINED
Command ==>
                                                    SUBSYS: DSN0

Legacy Table:
  Creator ID ==> DBA071
  Table Name ==> SRC@PROVIDER2XACTN
                    _

Use '_' for DB2 LIKE character ==> YES   Y-Yes, N-No
```

Figure 8-24 Choose a legacy table

```
----- Specify Copybooks for Legacy Table: DBA071.SRC@PROVIDER2XACTN -----
Command ==>
                                                    SCROLL ==> CSR
                                                    SUBSYS: DSN0

INSTRUCTIONS: Specify the name(s) of the Copybook members containing the record
              description to be used for this Legacy Table. All members will
              be concatenated before being parsed.

Legacy Table Type ==> F   I-IMS F-File (VSAM or Sequential)

Copybook Language ==> C   C-COBOL, P-PL/I

Copybook(s) ==> 'SYS030.CELLPHON.COPYBOOK(PROVIDR2)_ '
               ==>
               ==>
               ==>
               ==>
```

Figure 8-25 Specify copybooks for legacy table


```

----- Define Legacy Table: DBA071.SRC@PROVIDER2XACTN -----
Command ==> DS_                               Scroll ==> CSR
                                                    Row 1 of 14
Cmd Level/Field Name                               Type Len Occur Column Name
-----
*** ***** TOP *****
___ 1 PROVIDER2-XACTION-REC                          30      PROVIDER2_XACTION_REC
___ 5 P2-CUSTOMER-ID                                CHR  10      P2_CUSTOMER_ID
___ 5 P2-XACTION-DATE                                8        P2_XACTION_DATE
___ 10 P2-XACTION-DATE-YYYY                          NUM   4      P2_XACTION_DATE_YYYY
___ 10 P2-XACTION-DATE-MM                          NUM   2      P2_XACTION_DATE_MM
___ 10 P2-XACTION-DATE-DD                          NUM   2      P2_XACTION_DATE_DD
___ 5 P2-XACTION-START-TIME                          6        P2_XACTION_START_TIME
___ 10 P2-XACTION-START-HH                          NUM   2      P2_XACTION_START_HH
___ 10 P2-XACTION-START-MM                          NUM   2      P2_XACTION_START_MM
___ 10 P2-XACTION-START-SS                          NUM   2      P2_XACTION_START_SS
___ 5 P2-XACTION-DURATION                            6        P2_XACTION_DURATION
___ 10 P2-XACTION-DUR-HH                            NUM   2      P2_XACTION_DUR_HH
___ 10 P2-XACTION-DUR-MM                            NUM   2      P2_XACTION_DUR_MM
___ 10 P2-XACTION-DUR-SS                            NUM   2      P2_XACTION_DUR_SS
*** ***** BOTTOM *****
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-26 Define legacy table

```

----- Define Legacy Table: DBA071.SRC@PROVIDER2XACTN -----
mmand ==>                               Scroll ==> CSR
                                                    SUBSYS: DSNC
                                                    1
+-----Specify Data Source Information-----+
|
| Please specify the default Data Source information for this object.
| Enter END when complete.
|
| Data Source Type   : Sequential or VSAM Dataset
| Data Source DSN ==> 'SYS030.PROVIDR2.XACTION' _
|
+-----+
___ 10 P2-XACTION-START-HH                          NUM   2      P2_XACTION_START_HH
___ 10 P2-XACTION-START-MM                          NUM   2      P2_XACTION_START_MM
___ 10 P2-XACTION-START-SS                          NUM   2      P2_XACTION_START_SS
___ 5 P2-XACTION-DURATION                            6        P2_XACTION_DURATION
___ 10 P2-XACTION-DUR-HH                            NUM   2      P2_XACTION_DUR_HH
___ 10 P2-XACTION-DUR-MM                            NUM   2      P2_XACTION_DUR_MM
___ 10 P2-XACTION-DUR-SS                            NUM   2      P2_XACTION_DUR_SS
* ***** BOTTOM *****
1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-27 Define legacy table

We now had all the required Legacy Tables defined.

8.5 Identify extract data sources and data source relationships

Repeating PF3 three times returned us to the Optim MOVE for DB2 menu from where we took option 7 to bring us to the Data Migration screen (Figure 8-28).

```
----- Data Migration -----
OPTION  ===> 1_
                                           SQLID  ===> "DBA071"
                                           SUBSYS  ===> DSNC

1  EXTRACT  - Extract Data from Source Tables
2  INSERT   - Insert Data into Destination Tables
3  LOAD     - Create Load Files and Perform Load
4  DELETE   - Delete Data from Tables
5  CREATE   - Create Tables and Related Object Definitions
6  CONVERT  - Convert Extract File using Table and Column Maps

R  RETRY/RESTART  - Retry/Restart an Insert or Delete Process
B  BROWSE         - Browse Content of Extract or Control File
```

Figure 8-28 Data migration

Keying option 1 and pressing Enter in this screen brought us to the EXTRACT Process screen. The EXTRACT process creates an EXTRACT file that contains the selected set of related rows from one or more tables and, if requested, object definitions for those tables. The EXTRACT File is used by Optim MOVE as input to insert, load, create and convert process. Here we selected option 1 to go and define the specifications of the tables to be involved in the EXTRACT process. We also specified the name of the permanent Access Definition we were going to create (Figure 8-29).

```
----- EXTRACT Process -----
OPTION  ===> 1_
                                           SCROLL  ===> CSR
                                           SUBSYS: DSNC

1  TABLES  - Specify Set of Tables and Selection Criteria
2  PATHS    - Specify Traversal Paths via Relationship List
3  OBJECTS  - Specify Object Definitions to Extract
4  PERFORM  - Specify EXTRACT Parameters and Perform EXTRACT

Type of Access Definition to Use for EXTRACT ===> P (P-Perm, T-Temp)

If Permanent, Specify New or Existing Access Definition Name
Group  ===> EPSDEMO
User   ===> DBA071
Name   ===> FULLCELLPHON

Use '_' for DB2 LIKE Character ===> N (Y-Yes, N-No)
```

Figure 8-29 EXTRACT process

Pressing Enter brought us to the Select Tables screen. The Default Creator ID, SYS030, that we entered had been set up prior to starting our Data Extract process. The Start Table that we used was OPT_CUSTOMER, the DB2 table (Figure 8-30).

```
-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==> _                               Scroll ==> CSR
                                           SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line   : COL,SEL,SQL,ALL,GR (A) ,GP (A) ,GC (A) ,DR (A) ,PR (A) ,DP (A) ,PP (A) ,
        DC (A) ,PC (A) ,EXP,STA
                                           Table 1 of 1 <<MORE
Default Creator ID ==> SYS030                >>
Start Table      ==> OPT_CUSTOMER            >>
Start Table Options : None

                                Ref --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name  Tbl EveryNth RowLimit  Type
----->>----->>----->>----->>----->>
*** ***** TOP *****
OPT_CUSTOMER
*** ***** BOTTOM *****
```

Figure 8-30 Select Tables/Views

The GRA line command was used to show us all the tables that had been created and saved previously, including the starting table OPT_CUSTOMER (Figure 8-31).

```
-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==>                               Scroll ==> CS
                                           SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line   : COL,SEL,SQL,ALL,GR (A) ,GP (A) ,GC (A) ,DR (A) ,PR (A) ,DP (A) ,PP (A) ,
        DC (A) ,PC (A) ,EXP,STA
                                           Table 1 of 1 <<MORE
Default Creator ID ==> SYS030                >>
Start Table      ==> OPT_CUSTOMER            >>
Start Table Options : None

                                Ref --Extract Parms--
Cmd   Status      (CreatorID.)Table/View Name  Tbl EveryNth RowLimit  Type
----->>----->>----->>----->>----->>
*** ***** TOP *****
GRA   OPT_CUSTOMER
*** ***** BOTTOM *****
```

Figure 8-31 Select Tables/Views

These tables defined the data for our source DB2 dataset. The Legacy tables we had just defined were not included in the list, so we had to add them (Figure 8-32).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==>
                                Scroll ==> CSR
                                SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line   : COL,SEL,SQL,ALL,GR (A) ,GP (A) ,GC (A) ,DR (A) ,PR (A) ,DP (A) ,PP (A) ,
        DC (A) ,PC (A) ,EXP,STA
                                Table 1 of 5 <<MORE
Default Creator ID ==> SYS030
Start Table      ==> OPT_CUSTOMER
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name  Ref --Extract Parms--
-----
*** ***** TOP *****
OPT_CUSTOMER          TABLE
OPT_CUST_ACCT         N          TABLE
OPT_CUST_XACTION     N          TABLE
OPT_ACCOUNT          N          TABLE
I_ OPT_ACCT_SRC_TYPE   N          TABLE
*** ***** BOTTOM *****

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-32 Select Tables/Views

The **I** line command inserted another line into which we keyed the required IMS parent Table Name, ICUSTSSN. We now had to create a relationship for this table. To do this, we used the CRE REL (CREATE RELATIONSHIP) primary command (Figure 8-33).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==> CRE REL_
                                Scroll ==> CSR
                                SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line   : COL,SEL,SQL,ALL,GR (A) ,GP (A) ,GC (A) ,DR (A) ,PR (A) ,DP (A) ,PP (A) ,
        DC (A) ,PC (A) ,EXP,STA
                                Table 1 of 6 <<MORE
Default Creator ID ==> SYS030
Start Table      ==> OPT_CUSTOMER
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name  Ref --Extract Parms--
-----
*** ***** TOP *****
OPT_CUSTOMER          TABLE
OPT_CUST_ACCT         N          TABLE
OPT_CUST_XACTION     N          TABLE
OPT_ACCOUNT          N          TABLE
OPT_ACCT_SRC_TYPE    N          TABLE
... DBA071.ICUSTSSN   N          TABLE
*** ***** BOTTOM *****

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-33 Select Tables/Views

Pressing Enter took us to the Choose a Relationship screen (Figure 8-34).

```

----- Choose a Relationship -----
OPTION ==> 1_                               SCROLL ==> CSR
                                           SUBSYS: DSNCR
 1 CREATE - Create a Relationship for Specified Parent or Child Table
 2 MODIFY - Modify a Relationship for Specified Child Table
 3 LIST   - List All Relationships for Specified Table

Specify Table Name (Child for OPTION 2, Parent or Child for OPTIONS 1 and 3)
Creator ID ==> SYS030                        >>
Table Name ==> OPT_CUSTOMER                  >>

Specify Relationship Name (OPTIONS 1 and 2)
Relationship Name ==> CUST2IMSCUSTSSN      >>

Specify Relationship Type (OPTIONS 2 and 3)
Relationship Type ==> ALL                   (P|O-OPT, D-DB2, I-IMS, A-All)

Use '_' for DB2 LIKE character ==> YES     (Y-Yes, N-No)

```

Figure 8-34 Choose a relationship

From here we created, using option 1, a relationship called CUST2IMSCUSTSSN (Figure 8-35).

```

----- Choose a Relationship -----
OPTION ==>                                   SCROLL ==> CSR
                                           SUBSYS: DSNCR
 1 CREATE - Create a Relationship for Specified Parent or Child Table

-----Create a New Relationship-----
|
| Specified Table   : SYS030.OPT_CUSTOMER
| Table Type       ==> P                       (P-Parent, C-Child)
|
| Leave blank or include wild cards for Table Selection List
|
| Other Table:
|   Creator ID ==> DBA071                      >>
|   Table Name ==> ICUSTSSN_                  >>
|
| Relationship Name ==> CUST2IMSCUSTSSN      >>
|
-----

```

Figure 8-35 Choose a relationship

In the Create screen returned after we entered the Relationship Name, we associated our table ICUSTSSN with the relationship.

Pressing Enter saved the data, completing the build of the CUST2IMSCUSTSSN relationship definition. Then pressing PF3 twice, returning us to the Select Tables screen (Figure 8-36).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==>                                     Scroll ==> CSR
                                           SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
      DC(A),PC(A),EXP,STA
                                           Table 1 of 6 <<MORE
Default Creator ID ==> SYS030                                     >>
Start Table      ==> OPT_CUSTOMER                               >>
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name      Ref --Extract Parms--
-----
*** ***** TOP *****
OPT_CUSTOMER          TABLE
OPT_CUST_ACCT        N          TABLE
OPT_CUST_XACTION     N          TABLE
OPT_ACCOUNT          N          TABLE
OPT_ACCT_SRC_TYPE   N          TABLE
GRA                   DBA071. ICUSTSSN  N          LEGACY
*** ***** BOTTOM *****

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-36 Select Tables/Views

Using a GRA line command, we expanded the table list to show the Legacy tables representing the child segments associated with ICUSTSSN. We still had to add in the tables for our QSAM and VSAM data. An I2 line command inserted two empty rows into the table list (Figure 8-37).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==>                                     Scroll ==> CSR
                                           SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
      DC(A),PC(A),EXP,STA
                                           Table 1 of 8 <<MORE
Default Creator ID ==> SYS030                                     >>
Start Table      ==> OPT_CUSTOMER                               >>
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name      Ref --Extract Parms--
-----
*** ***** TOP *****
OPT_CUSTOMER          TABLE
OPT_CUST_ACCT        N          TABLE
OPT_CUST_XACTION     N          TABLE
OPT_ACCOUNT          N          TABLE
OPT_ACCT_SRC_TYPE   N          TABLE
DBA071. ICUSTSSN    N          LEGACY
DBA071. ICUSTCC     N          LEGACY
I2_                   DBA071. ICUSTCS  N          LEGACY
*** ***** BOTTOM *****

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-37 Select Tables/Views

We entered the names of the remaining two Legacy Tables. Then we entered CRE REL to create the required relationship definitions (Figure 8-38).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==> CRE REL_                               Scroll ==> CSR
                                                    SUBSYS: DSNC
Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
Line : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
      DC(A),PC(A),EXP,STA
                                                    Table 3 of 10 <<MORE
Default Creator ID ==> SYS030                       >>
Start Table      ==> OPT_CUSTOMER                    >>
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name  Ref --Extract Parms--
-----
----->>-----
      OPT_CUST_XACTION          N  _____  TABLE
      OPT_ACCOUNT              N  _____  TABLE
      OPT_ACCT_SRC_TYPE        N  _____  TABLE
      DBA071.ICUSTSSN          N  _____  LEGACY
      DBA071.ICUSTCC           N  _____  LEGACY
      DBA071.ICUSTCS           N  _____  LEGACY
      DBA071.SRC@PROVIDER1XACTN N  _____  LEGACY
      DBA071.SRC@PROVIDER2XACTN N  _____  LEGACY
*** ***** BOTTOM *****
PF 1=HELP    2=SPLIT    3=END      4=RETURN   5=RFIND    6=RCHANGE
PF 7=UP      8=DOWN     9=SWAP    10=LEFT    11=RIGHT   12=RETRIEVE

```

Figure 8-38 Select Tables/Views

The next relationship we built was CUST2PROVIDER1XACTN for our QSAM data (Figure 8-39).

```

----- Choose a Relationship -----
OPTION ==> 1                               SCROLL ==> CSR
                                                    SUBSYS: DSNC
1 CREATE - Create a Relationship for Specified Parent or Child Table
2 MODIFY - Modify a Relationship for Specified Child Table
3 LIST - List All Relationships for Specified Table

Specify Table Name (Child for OPTION 2, Parent or Child for OPTIONS 1 and 3)
Creator ID ==> SYS030                       >>
Table Name ==> OPT_CUSTOMER                    >>

Specify Relationship Name (OPTIONS 1 and 2)
Relationship Name ==> CUST2PROVIDER1XACTN_    >>

Specify Relationship Type (OPTIONS 2 and 3)
Relationship Type ==> ALL                    (P|O-OPT, D-DB2, I-IMS, A-ALL)

Use '_' for DB2 LIKE character ==> YES        (Y-Yes, N-No)

```

Figure 8-39 Choose a relationship

The table SRC@PROVIDER1XACTN was then associated with this relationship (Figure 8-40).

```

----- Choose a Relationship -----
OPTION ==>
                                SCROLL ==> CSR
                                SUBSYS: DSNC

  1 CREATE  - Create a Relationship for Specified Parent or Child Table

+-----Create a New Relationship-----+
|
| Specified Table   : SYS030.OPT_CUSTOMER
| Table Type       ==> P                (P-Parent, C-Child)
|
| Leave blank or include wild cards for Table Selection List
|
| Other Table:
|   Creator ID ==> DBA071                >>
|   Table Name ==> SRC@PROVIDER1XACTN_   >>
|
| Relationship Name ==> CUST2PROVIDER1XACTN >>
|
+-----+

```

Figure 8-40 Choose a relationship

Similarly, we associated table SRC@PROVIDER2XACTN with the relationship CUST2PROVIDER2XACTN (Figure 8-41 and Figure 8-42).

```

----- Choose a Relationship ----- RELATIONSHIP SAVED
OPTION ==> 1
                                SCROLL ==> CSR
                                SUBSYS: DSNC

  1 CREATE  - Create a Relationship for Specified Parent or Child Table
  2 MODIFY  - Modify a Relationship for Specified Child Table
  3 LIST    - List All Relationships for Specified Table

Specify Table Name (Child for OPTION 2, Parent or Child for OPTIONS 1 and 3)
  Creator ID ==> SYS030                >>
  Table Name ==> OPT_CUSTOMER          >>

Specify Relationship Name (OPTIONS 1 and 2)
  Relationship Name ==> CUST2PROVIDER2XACTN >>

Specify Relationship Type (OPTIONS 2 and 3)
  Relationship Type ==> ALL              (P|0-OPT, D-DB2, I-IMS, A-All)

Use '_' for DB2 LIKE character ==> YES  (Y-Yes, N-No)

```

Figure 8-41 Choose a relationship


```

----- Choose a Relationship -----
OPTION ==>
                                SCROLL ==> CSR
                                SUBSYS: DSNC
  1 CREATE  - Create a Relationship for Specified Parent or Child Table

+-----Create a New Relationship-----+
Specified Table   : SYS030.OPT_CUSTOMER
Table Type       ==> P                (P-Parent, C-Child)

Leave blank or include wild cards for Table Selection List

Other Table:
  Creator ID ==> DBA071                >>
  Table Name ==> SRC@PROVIDER2XACTN_   >>

Relationship Name ==> CUST2PROVIDER2XACTN >>

```

Figure 8-42 Choose a relationship

This completed the identification of the EXTRACT data sources and their associated relationships. PF3 saved the data. Using PF3 again brought us back to the Select Tables screen.

8.5.1 Define extract selection criteria

We then had to define the selection criteria for the extract. To do this, we used an SQL line command (Figure 8-43).

```

-- Select Tables/Views for AD: EPSDEMO.DBA071.FULLCELLPHON -----
Command ==>
                                Scroll ==> CSR
                                SUBSYS: DSNC
  Primary : COL,SEL,SQL,REL,POINT,GROUP,GET TABLES RELATED,INDENT,LIST SUBS
  Line    : COL,SEL,SQL,ALL,GR(A),GP(A),GC(A),DR(A),PR(A),DP(A),PP(A),
           DC(A),PC(A),EXP,STA
                                           Table 1 of 10 <<MORE
Default Creator ID ==> SYS030                >>
Start Table       ==> OPT_CUSTOMER           >>
Start Table Options : None

Cmd  Status      (CreatorID.)Table/View Name      Ref --Extract Parms--
----->>----->>----->>----->>----->>----->>
*** ***** TOP *****
SQL      OPT_CUSTOMER
_____ OPT_CUST_ACCT                N
_____ OPT_CUST_XACTION              N
_____ OPT_ACCOUNT                   N
_____ OPT_ACCT_SRC_TYPE              N
_____ DBA071.ICUSTSSN                N
_____ DBA071.ICUSTCC                 N
_____ DBA071.ICUSTCS                 N
PF 1=HELP  2=SPLIT  3=END      4=RETURN  5=RFIND  6=RCHANGE
PF 7=UP    8=DOWN  9=SWAP    10=LEFT   11=RIGHT  12=RETRIEVE

```

Figure 8-43 Select Tables/Views

A SQL WHERE clause was typed in the empty screen returned (Figure 8-44).

```

----- Enter an SQL WHERE Clause for a Table or View -----
Command ==>                                     Scroll ==> CSR
                                                SUBSYS: DSN
SELECT ... FROM SYS030.OPT_CUST_XACTION
Cmd      Correlation Name ==>                WHERE                1 of 8
-----
*** ***** TOP *****
___ (CUSTOMER_STATE = 'AR' AND CUSTOMER_CITY <> 'MOUNTAIN HOME')
___ OR CUSTOMER_STATE = 'NY'
___
___
___
___
___

Line Commands: (I)nsert, (D)elete, (R)epeat, (M)ove, (C)opy
Use the LIST COLUMNS command to add column names, if needed
Use the SQLEdit command to invoke the ISPF editor with all of its facilities
An optional correlation name can be entered to refer to the base table

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN        9=SWAP     10=LEFT       11=RIGHT     12=RETRIEVE

```

Figure 8-44 Enter SQL WHERE clause

Pressing Enter saved the selection criteria.

8.5.2 Specify parameters for Legacy Tables

Next step was to specify the Legacy Table parameters for the data extract. PF3 brought us to the screen where we could associate the Legacy Tables with the Data Sources (Figure 8-45).

```

----- Associate Legacy Tables with Data Sources -----
Command ==>                                     Scroll ==> CSR
                                                SUBSYS: DSN
Overriding Dataset Prefix ==>                1 of 5

Legacy Table      File-Dataset Name
                  IMS--Segment  DBD      PSB      PCB  IMSID  DBRC  LOG
-----
***** TOP *****
DBA071.SRC@PROVIDER1XACTN  File 'SYS030.PROVIDR1.XACTION'
DBA071.SRC@PROVIDER2XACTN  File 'SYS030.PROVIDR2.XACTION'
DBA071.ICUSTSSN           IMS CUSTSSN  CSTPRV  CSTPRVG  1_      N      N
DBA071.ICUSTCC            IMS CUSTCC
DBA071.ICUSTCS            IMS CUSTCSCO
***** BOTTOM *****

```

Figure 8-45 Associate legacy tables with data sources

Pressing Enter brought us to the following screen.

Here we completed the parameter specification process by entering the IMS data set names (Figure 8-46).

```

----- Associate IMS Segments With IMS Databas  DATA SET NOT SPECIFIED
Command ==>                                     Scroll ==> CSR
                                                SUBSYS: DSNC
Overriding Dataset Prefix ==>                                     1 of 4

  DBD      Segment  DD Name  IMS Database Dataset Name
-----
***** TOP *****
CSTPRV  CUSTSSN  CSTPRV  'SYS030.CELLPHON.CSTPRV'
        CUSTCC
        CUSTCSCO
        -INDEX  CSTPRVI  'SYS030.CELLPHON.CSTPRVI'
***** BOTTOM *****

```

Figure 8-46 Associate IMS segments with IMS database

We were now ready to extract the data.

8.6 Extract the data

PF3 returned us to the EXTRACT Process screen. From here we selected option 4, PERFORM (Figure 8-47).

```

----- EXTRACT Process -----
OPTION ==> 4_                                     SCROLL ==> CSR
                                                SUBSYS: DSNC

 1 TABLES          - Specify Set of Tables and Selection Criteria
 2 PATHS            - Specify Traversal Paths via Relationship List
 3 OBJECTS          - Specify Object Definitions to Extract
 4 PERFORM          - Specify EXTRACT Parameters and Perform EXTRACT

Type of Access Definition to Use for EXTRACT ==> P (P-Perm, T-Temp)

If Permanent, Specify New or Existing Access Definition Name
Group ==> EPSDEMO
User  ==> DBA071
Name  ==> FULLCELLPHON

Use '_' for DB2 LIKE Character ==> N (Y-Yes, N-No)

```

Figure 8-47 Extract process

In the Specify Extract Parameters screen we keyed the name of the dataset to which the data and object definitions would be extracted, limiting the number of rows to be extracted to 99,999 (Figure 8-48).

```

----- Specify EXTRACT Parameters and Execute -----
Command ==>
                                                    SUBSYS: DSNC

Current AD Name      : EPSDEMO.DBA071.FULLCELLPHON
Extract File DSN ==> 'DBA071.ALLDATA.EXTRACT'
Extract              ==> B                (D-Data,
                                           O-Object Definitions,
                                           B-Both)

If Extracting Data:
  Limit Number of Extract Rows ==> 99999  (1-9999999, Blank-Site Limit)
  Extract Data using           ==> D        (D-DB2, I-IBM High Perf. Unload)

Perform Convert with Extract ==> N        (Y-Yes, N-No)

Run Process in Batch or Online ==> O      (B-Batch, O-Online)
  If Batch, Review or Save JCL ==> R      (N-No, R-Review, S-Save)

Process Report Type           ==> D        (D-Detailed, S-Summary)

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE
  
```

Figure 8-48 Specify EXTRACT parameters and execute

Because this dataset did not already exist, we had to dynamically define it (Figure 8-49).

```

----- Specify EXTRACT Parameters and Execute -----
Command ==>
                                                    SUBSYS: DSNC

+-----Allocate Dataset-----+
|
| Data Set Name:  DBA071.ALLDATA.EXTRACT
| Management Class ==> _          (Blank for Default Management Class)
| Storage Class   ==> _          (Blank for Default Storage Class)
| Use Stored Values ==> N        (Y-Yes, N-No, V-View Stored Values)
| Volume Serial   ==>           (Blank for Authorized Default Volume)
| Device Type     ==>           (Generic Unit or Device Address)
| Data Class      ==>           (Blank for Default Data Class)
| Space Units     : BLKS        (In 8KB BLKS)
| Primary Quantity ==> 112      (In Above Units)
| Secondary Quantity ==> 99     (In Above Units)
| Directory Blocks : 0
| Record Format    : FS
| Record Length   :             (Value Will Be Set to Block Size)
| Block Size      ==> 0         (Minimum 7944;0=Determine at Runtime)
|
| Calculated Values Shown, Change to Profiled Values ==> N (Y-Yes, N-No)
|
+-----+
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE
  
```

Figure 8-49 Specify EXTRACT parameters and execute

Pressing Enter processed the dataset definition and brought us back to the Specify Extract Parameters screen (Figure 8-50).

```

----- Specify EXTRACT Parameters and Execute -----
Command ==> _
                                                    SUBSYS: DSNC
Current AD Name      : EPSDEMO.DBA071.FULLCELLPHON
Extr
Extr +-----EXTRACT Errors & Warnings-----+
      |
      | EXTRACT Process Can Proceed Despite the Following Warnings:
      |   9 Relationship(s) in NEW Status
      |   5 Table(s) Have No RUNSTATS
      |                                     t)
      |                                     load)
      | Press ENTER Key to Proceed Despite Warnings
      | Enter END Command to Return to EXTRACT Menu to Correct Problems
      |-----+
Run
  If Batch, Review or Save JCL ==> R      (N-No, R-Review, S-Save)
Process Report Type      ==> D      (D-Detailed, S-Summary)

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT       11=RIGHT     12=RETRIEVE
  
```

Figure 8-50 Specify EXTRACT parameters and execute

Ignoring the warning messages, we pressed Enter to allow the extract process to complete. Pressing Enter when the system end of job message was received presented the report detail.

8.7 View extract report

Figure 8-52 shows the EXTRACT Process Report detail.

```

----- EXTRACT Process Report -----
Command ==> _                               SCROLL ==> PAGE
                                           ROW 0   OF 77
***** Top of Data *****

                EXTRACT Process Report

Extract File      : DBA071.ALLDATA.EXTRACT
Access Definition : EPSDEMO.DBA071.FULLCELLPHON
Created by       : Job DBA071, using SQLID DBA071 on DB2 Subsystem DSNC
Time Started    : 2008-05-21 13.12.59
Time Finished   : 2008-05-21 13.13.04

Process Options:
  Process Mode   : Online
  Retrieve Data using : DB2 / Legacy
  Limit Extract Rows : 99999

Total Number of Extract Tables      : 10
Total Number of Extracted Rows     : 345
Total Number of First Pass Start Table Rows : 105

PF 1=HELP    2=SPLIT    3=END    4=RETURN    5=RFIND    6=RCHANGE
PF 7=UP      8=DOWN     9=SWAP   10=LEFT    11=RIGHT   12=TERM
  
```

Figure 8-51 EXTRACT process report

PF8 and PF7 were used to scroll through the report (Figure 8-52).

```

----- EXTRACT Process Report -----
Command ==> _                               SCROLL ==> PAGE
                                           ROW 19  OF 77

  Extracted Object Types      Number
-----
  Table-List Tables          5
  Related Primary Keys       5
  Relationships               7
  Related Indexes            5

                Extracted
                Rows

  Extract Tables      Associated Legacy Informati
-----
  SYS030.OPT_CUSTOMER          105
  SYS030.OPT_CUST_ACCT         105
  SYS030.OPT_CUST_XACTION       8
  SYS030.OPT_ACCOUNT           99
  SYS030.OPT_ACCT_SRC_TYPE      14
  DBA071.ICUSTSSN                3  IMS: Segment-CUSTSSN, PSB-CSTPRVG,
  DBA071.ICUSTCC                 2  IMS: Segment-CUSTCC, PSB-CSTPRVG,
  DBA071.ICUSTCS                 2  IMS: Segment-CUSTCSCO, PSB-CSTPRVG
  DBA071.SRC@PROVIDER1XACTN      2  SYS030.PROVIDR1.XACTION

PF 1=HELP    2=SPLIT    3=END    4=RETURN    5=RFIND    6=RCHANGE
PF 7=UP      8=DOWN     9=SWAP   10=LEFT    11=RIGHT   12=TERM
  
```

Figure 8-52 EXTRACT process report (continued)

When we had satisfied ourselves with the content of the report, we used PF3 to return (Figure 8-53).

```

Command ==> _
SCROLL ==> PAGE
ROW 38 OF 77
10 DBA071.SRC@PROVIDER2XACTN      5 SYS030.PROVIDR2.XACTION

Selection Criteria in Use:

Table      Opr      Column      Criteria
-----
SYS030.OPT_CUSTOMER      SQL      (CUSTOMER_STATE = 'AR'
OR CUSTOMER_STATE = 'N'

Relationship Usage Report

Parent Table      Child Table      Relation Name      Access Type
-----
SYS030.OPT_CUSTOMER      DBA071.ICUSTSSN      CUST2IMS      **      KEY
CUSTSSN
SYS030.OPT_CUSTOMER      SYS030.OPT_CUST_ACCT      CUSTOMER      **      SCAN
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE

```

Figure 8-53 Extract process report (continued)

We pressed Enter to confirm the changes made to AD during the EXTRACT process (Figure 8-54).

```

----- EXTRACT Process -----
OPTION ==>
SCROLL ==> CSR
SUBSYS: DSNC
1 TABLES      - Specify Set of Tables and Selection Criteria
2 PATHS        - Specify Traversal Paths via Relationship List
3 OBJECTS      - Specify Object Definitions to Extract
4 PERFORM      - Specify EXTRACT Parameters and Perform EXTRACT

Type of Access Defini +-----Confirm AD Save-----+
If Permanent, Specify | AD was Modified During EXTRACT Process
Group ==> EPSDEMO     |
User ==> DBA071      | Press ENTER Key to Save AD Changes
Name ==> FULLCELL    | Enter END Command to Bypass Saving AD
Use '_' for DB2 LIK  | Enter CANCEL Command to Return to Menu
AD Name ==> EPSDEMO.DB071.FULLCELLPHON

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN      9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-54 EXTRACT process

PF3 returned us to the Optim menu screen.

8.8 Use table lookup to convert extracted data

Our next step was the conversion of the extracted data, using a table lookup. Selecting option 7 brought us to the Data Migration screen (Figure 8-55).

```
----- Data Migration -----
OPTION ==> 6_
                                SQLID ==> "DBA071"
                                SUBSYS ==> DSNC

1  EXTRACT  - Extract Data from Source Tables
2  INSERT   - Insert Data into Destination Tables
3  LOAD     - Create Load Files and Perform Load
4  DELETE   - Delete Data from Tables
5  CREATE   - Create Tables and Related Object Definitions
6  CONVERT  - Convert Extract File using Table and Column Maps

R  RETRY/RESTART  - Retry/Restart an Insert or Delete Process
B  BROWSE         - Browse Content of Extract or Control File
```

Figure 8-55 Data migration

Here we selected option 6 to start the conversion process (Figure 8-56).

```
----- Specify CONVERT Parameters and Execute -----
OPTION ==> 1_
                                SCROLL ==> CSR
                                More: - SUBSYS: DSNC

Specify Data Set Names for Source Extract File and Control File:
Source Extract File DSN ==> 'DBA071.ALLODATA.EXTRACT'
Control File DSN        ==> 'DBA071.ALLODATA.EXTRACT2.CONTROL'

Specify Data Set Name for Destination Extract File.  If Omitted,
Destination Extract DSN will be same as the Source Extract DSN:
Destination Extract DSN ==> 'DBA071.ALLODATA.EXTRACT2'

Age Date Values           ==> N      (Y-Yes, N-No)
Output to External Format  ==> N      (Y-Yes, N-No)
Limit Number of Discarded Rows ==>      (1-999999, Blank-No Limit)
If Destination Tables have a Cluster Index:
Sort Extract File Rows   ==> N      (Y-Yes, N-No)

Run Process in Batch or Online ==> O      (B-Batch, O-Online)
If Batch, Review or Save JCL ==> R      (N-No, R-Review, S-Save)

Process Report Type       ==> D      (D-Detailed, S-Summary)
PF 1=HELP      2=SPLIT    3=END      4=RETURN    5=RFIND    6=RCHANGE
PF 7=HELP      8=DOWN     9=SWAP    10=LEFT     11=RIGHT   12=RETRIEVE
```

Figure 8-56 Specify CONVERT parameters and execute

The first step was to specify datasets used by the convert process. This was achieved by specifying the source (extract) file, the control file, and the dataset to receive the converted extracted data, then Entering option 1 giving us the CONVERT Process Table Map (Figure 8-57).


```

----- CONVERT Process Table Map -----
Command ==>                               Scroll ==> CSR
SUBSYS: DSNC
Available Commands: APPLY,SAVE,LIST,MAP,POPULATE,CLEAR,END when Done

Src CID: SYS030                               Column
Dest CID: SYS030                               >> Map ID ==>

Extract Tables      Destination Table Name      Type      Column Map or "LOCAL
----->> ----->> ----->> ----->>
***** TOP *****
OPT_CUSTOMER      OPT_CUSTOMER_S15      UNKNOWN  LOCAL_
OPT_CUST_ACCT     OPT_CUST_ACCT         UNKNOWN
OPT_CUST_XACTION  OPT_CUST_XACTION     UNKNOWN
OPT_ACCOUNT       OPT_ACCOUNT           UNKNOWN
OPT_ACCT_SRC_TYPE OPT_ACCT_SRC_TYPE    UNKNOWN
DBA071.ICUSTSSN  DBA071.ICUSTSSN     LEGACY
DBA071.ICUSTCC   DBA071.ICUSTCC      LEGACY
DBA071.ICUSTCS   DBA071.ICUSTCS      LEGACY
DBA071.SRC@PROVIDER1X DBA071.SRC@PROVIDER1XACTN LEGACY
DBA071.SRC@PROVIDER2X DBA071.SRC@PROVIDER2XACTN LEGACY
***** BOTTOM *****
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-57 CONVERT process table map

In the Convert Process Table Map we entered the destination Creator ID (Dest CID) and changed the destination table name for the OPT_CUSTOMER extract table. Specifying LOCAL and pressing PF3 caused the column map editor to be invoked. A default column mapping was displayed. We had to use a lookup table to expand the CUSTOMER_STATE values (Figure 8-58).

```

-- Define Internal Column Map-----
Command ==>                               Scroll ==> CSR
SUBSYS: DSNC

Corresponding Columns MUST Have Compatible Data Types
Use LIST UNUSED Command for List of Unused Source Columns
Use LIST ALL Command for List of All Source Columns
VAL ON MOVE
1 OF 6

-----SYS030.OPT_CUSTOMER-----
Cmd      Source Column      Data Type      Num Destination Column Data Type      Status
----->> ----->> ----->> ----->>
***** TOP *****
__ CUSTOMER_ID      CH (10)        1 CUSTOMER_ID      CH (10)      EQUAL
__ CUSTOMER_NAME    CH (40)        2 CUSTOMER_NAME    CH (40)      EQUAL
__ CUSTOMER_ADDRESS CH (30)        3 CUSTOMER_ADDRESS CH (30)      EQUAL
__ CUSTOMER_CITY    CH (20)        4 CUSTOMER_CITY    CH (20)      EQUAL
EXP CUSTOMER_STATE  CH (2)         5 CUSTOMER_STATE    CH (15)      MAPPED
__ CUSTOMER_ZIP     CH (5)         6 CUSTOMER_ZIP     CH (5)       EQUAL
***** BOTTOM *****

```

Figure 8-58 Define internal column map

Using the EXP line command, we came to the following screen where the required expression was defined (Figure 8-59).

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
-----
EDIT      COLUMN-MAP-EXPRESSION-TEXT      Columns 00001 00072
Command ==> _____ Scroll ==> PAGE
***** ***** Top of Data *****
000001 LOOKUP(CUSTOMER_STATE, SYS030.OPT_LOOKUPSTATE(STATE_CODE, FULL_NAME))
***** ***** Bottom of Data *****

```

Figure 8-59 Extract expression definition

The expression is using STATE_CODE to get the FULL_NAME for CUSTOMER_STATE.

Using PF3 twice returned us to the CONVERT screen, with a message indicating that the LOCAL map had been defined successfully (Figure 8-60).

```

----- Specify CONVERT Parameters and Ex LOCAL MAP DEFINED
OPTION ==> 2_ SCROLL ==> CSR
More: + SUBSYS: DSNC
1 TABLE MAP - Specify Table Map and Column Maps
2 PERFORM - Execute Extract File Conversion Process

Specify Data Set Names for Source Extract File and Control File:
Source Extract File DSN ==> 'DBA071.ALLODATA.EXTRACT'
Control File DSN ==> 'DBA071.ALLODATA.EXTRACT2.CONTROL'

Specify Data Set Name for Destination Extract File. If Omitted,
Destination Extract DSN will be same as the Source Extract DSN:
Destination Extract DSN ==> 'DBA071.ALLODATA.EXTRACT2'

Age Date Values ==> N (Y-Yes, N-No)
Output to External Format ==> N (Y-Yes, N-No)
Limit Number of Discarded Rows ==> (1-999999, Blank-No Limit)
If Destination Tables have a Cluster Index:
Sort Extract File Rows ==> N (Y-Yes, N-No)

Run Process in Batch or Online ==> 0 (B-Batch, 0-Online)
If Batch, Review or Save JCL ==> R (N-No, R-Review, S-Save)
PF 1=HELP 2=SPLIT 3=END 4=RETURN 5=RFIND 6=RCHANGE
PF 7=UP 8=DOWN 9=SWAP 10=LEFT 11=RIGHT 12=RETRIEVE

```

Figure 8-60 Specify CONVERT parameters and execute

The next step was to convert the existing extract file. Selecting option 2 PERFORM and pressing Enter returned us a go / no go option selection (Figure 8-61).

```

----- Specify CONVERT Parameters and Execute -----
OPTION ==> _                               SCROLL ==> CSR
                                           SUBSYS: DSN
1  TABLE MAP          - Specify Table Map and Column Maps
2  PERFORM
                                           ccess
+-----Confirm Loss of Object Definitions-----+
Specify Data S :
Source Extra   : CONVERT Process will Result in Loss of
Control File   : Extracted Object Definitions
Specify Data S : Press ENTER Key to Start CONVERT Process
Destination Ex: Enter END Command to Cancel CONVERT Process
Destination    :
Age Date Values      ==> N      (Y-Yes, N-No)
Output to External Format ==> N      (Y-Yes, N-No)
Limit Number of Discarded Rows ==>      (1-999999, Blank-No Limit)
If Destination Tables have a Cluster Index:
Sort Extract File Rows ==> N      (Y-Yes, N-No)
Run Process in Batch or Online ==> 0      (B-Batch, 0-Online)
If Batch, Review or Save JCL ==> R      (N-No, R-Review, S-Save)
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE

```

Figure 8-61 Specify CONVERT parameters and execute

We chose to continue by pressing Enter. This brought us to the Conversion Report (Figure 8-62).

```

----- CONVERT Process Report -----
Command ==> _                               SCROLL ==> PAGE
                                           ROW 0   OF 51
***** Top of Data *****
                                CONVERT Process Report
Extract File      : DBA071.ALLODATA.EXTRACT
Control File     : DBA071.ALLODATA.EXTRACT2.CONTROL
Destination Extract File: DBA071.ALLODATA.EXTRACT2
User ID          : DBA071
Time Started     : 2008-05-21 13.57.15
Time Finished    : 2008-05-21 13.57.16
Totals:
Number of Convert Tables      : 10
Number of Original Extract Rows : 345
Number of Converted Rows     : 345
Number of Conversion Errors   : 0
Number of Exit Routine Discards : 0
PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=TERM

```

Figure 8-62 CONVERT process report

Using PF7 and PF8 we scrolled through the report. When finished we used PF3 twice to return us to the Data Migration Menu screen.

Here we Entered a **B** to start a browse function (Figure 8-63).

```

----- Data Migration -----
OPTION ==> B_
                                SQLID ==> "DBA071"
                                SUBSYS ==> DSNC

1 EXTRACT - Extract Data from Source Tables
2 INSERT  - Insert Data into Destination Tables
3 LOAD    - Create Load Files and Perform Load
4 DELETE  - Delete Data from Tables
5 CREATE  - Create Tables and Related Object Definitions
6 CONVERT - Convert Extract File using Table and Column Maps

R RETRY/RESTART - Retry/Restart an Insert or Delete Process
B BROWSE        - Browse Content of Extract or Control File
  
```

Figure 8-63 Data migration

We first browsed the input extract dataset, requesting the data to be returned in a report format (Figure 8-64).

```

----- Extract, Archive or Control File Browse Parameters -----
Command ==> _
                                More: +
                                SUBSYS: DSNC
                                SCROLL ==> CSR
Provide Extract, Archive or Control File Data Set Name:
DSN ==> 'DBA071.ALLDATA.EXTRACT'

Browse Mode ==> R (T-Table, R-Report
                  S-Summary, A-Access Def)

If Table Mode, specify
Table Name ==> >> (Blank for Start Table)
Begin with ==> D (D-Data, S-Sel Crit, Q-SQL)
If begin with S or Q
Case Sensitive ==> Y (Y-Yes, N-No If NO, any
                    dense indexes are skipped)

If Other than Table Mode, specify
Table Name ==> >> (Blank for all tables)
If output to Disk, specify
Output DSN ==> (Blank for temp dataset)

For Control File Only:
Show Row Status ==> Y (Y-Yes, N-No, X-Explain)
Filter Data ==> A (E-Error Rows Only, A-All)
PF 1=HELP 2=SPLIT 3=END 4=RETURN 5=RFIND 6=RCHANGE
PF 7=UP 8=DOWN 9=SWAP 10=LEFT 11=RIGHT 12=RETRIEVE
  
```

Figure 8-64 Extract, archive or control file browse parameters

With the report displayed, we entered a find command to locate CUSTOMER_STATE data (Figure 8-65).

```

Menu  Utilities  Compilers  Help
-----
BROWSE      SYS08142.T140359.RA000.DBA071.R0178048      Line 00000000 Col 001 080
Command ==> F CUSTOMER STATE                      Scroll ==> PAGE
***** Top of Data *****

Optim - Extract File Print Report

Extract File      : DBA071.ALLEXTDATA.EXTRACT
File Created By   : Job DBA071 using SQLID DBA071
File Created On   : May 21, 2008 at 01:13 PM from DB2 Subsystem DSN
Report Printed On: May 21, 2008 at 02:03 PM from DB2 SubSystem DSN

Number of Tables in the Extract File: 10
Number of Tables Processed in Report: 10

Table: SYS030.OPT_CUSTOMER

CUSTOMER_ID  CUSTOMER_NAME  CUSTOMER_ADDRESS
-----
222222222  BARBARA BESSINGER  6789 OCEAN PARKWAY
F1=Help  F2=Split  F3=Exit  F5=Rfind  F7=Up  F8=Down  F9=Swap
F10=Left F11=Right F12=Cancel

```

Figure 8-65 Browse extract file print report

Using PF8, we scrolled forward to show that the original extract data contained 2-character short names (Figure 8-66).

```

Menu  Utilities  Compilers  Help
-----
BROWSE      SYS08143.T143949.RA000.DBA071.R0186290      Line 00000018 Col 086 165
Command ==>                               Scroll ==> PAGE
EW YORK      NY      10001
ALICO ROCK   AR      72943
ENTONVILLE AR      72764
OTTER        AR      72632
AYETTEVILLE AR     72231
UREKA SPRINGS AR     72342
MOUNTAIN VIEW AR     72675
ILOAM SPRINGS AR     72896
ELSOR        AR     72436
USSELLVILLE AR     72985
PRINGDALE   AR     72439
ENTONVILLE AR     72568
ENTERVILLE AR     72528
ROCKETTS BLUFF AR     72847
ENTER RIDGE  AR     72421
ENTONVILLE AR     72895
ECATUR       AR     72747
OHNSON       AR     72532
F1=Help  F2=Split  F3=Exit  F5=Rfind  F7=Up  F8=Down  F9=Swap
F10=Left F11=Right F12=Cancel

```

Figure 8-66 Browse results

PF3 returned us to the screen where we could enter detail for another browse. We selected the option to obtain a report showing the contents of the converted extract dataset (Figure 8-67).

```

----- Extract, Archive or Control File Browse Parameters -----
Command ==>                                     SCROLL ==> CSR
More: + SUBSYS: DSNC
Provide Extract, Archive or Control File Data Set Name:
DSN ==> 'DBA071.ALLDATA.EXTRACT2' _

Browse Mode ==> R (T-Table, R-Report
                  S-Summary, A-Access Def)

If Table Mode, specify
Table Name ==> >> (Blank for Start Table)
Begin with ==> D (D-Data, S-Sel Crit, Q-SQL)
If begin with S or Q
Case Sensitive ==> Y (Y-Yes, N-No If NO, any
                    dense indexes are skipped)

If Other than Table Mode, specify
Table Name ==> >> (Blank for all tables)
If output to Disk, specify
Output DSN ==> (Blank for temp dataset)

For Control File Only:
Show Row Status ==> Y (Y-Yes, N-No, X-Explain)
Filter Data ==> A (E-Error Rows Only, A-All)
PF 1=HELP 2=SPLIT 3=END 4=RETURN 5=RFIND 6=RCHANGE
PF 7=UP 8=DOWN 9=SWAP 10=LEFT 11=RIGHT 12=RETRIEVE

```

Figure 8-67 Browse selection

Again we located the CUSTOMER_STATE data using a find command (Figure 8-68).

```

Menu Utilities Compilers Help
BROWSE SYS08143.T144321.RA000.DBA071.R0186299 Line 00000000 Col 001 080
Command ==> F CUSTOMER_STATE Scroll ==> PAGE
***** Top of Data *****
Optim - Extract File Print Report

Extract File : DBA071.ALLDATA.EXTRACT2
File Created By : Job DBA071 using SQLID DBA071
File Created On : May 22, 2008 at 02:37 PM from DB2 Subsystem DSNC
Report Printed On: May 22, 2008 at 02:43 PM from DB2 SubSystem DSNC

Number of Tables in the Extract File: 10
Number of Tables Processed in Report: 10

Table: SYS030.OPT_CUSTOMER_S15

CUSTOMER_ID CUSTOMER_NAME CUSTOMER_ADDRESS
-----
222222222 BARBARA BESSINGER 6789 OCEAN PARKWAY
F1=Help F2=Split F3=Exit F5=Rfind F7=Up F8=Down F9=Swap
F10=Left F11=Right F12=Cancel

```

Figure 8-68 Converted Extract file report

This showed that the CUSTOMER_STATE field had been converted from a short to a long name (Figure 8-69).

```
Menu Utilities Compilers Help
BROWSE SYS08153.T024959.RA000.DBA071.R0108557 Line 00000018 Col 099 178
Command ==> Scroll ==> PAGE
10001 NEW YORK
72943 ARKANSAS
72764 ARKANSAS
72632 ARKANSAS
72231 ARKANSAS
72342 ARKANSAS
72675 ARKANSAS
72896 ARKANSAS
72436 ARKANSAS
72985 ARKANSAS
72439 ARKANSAS
72568 ARKANSAS
72528 ARKANSAS
F 72847 ARKANSAS
72421 ARKANSAS
72895 ARKANSAS
72747 ARKANSAS
72532 ARKANSAS
F1=Help F2=Split F3=Exit F5=Rfind F7=Up F8=Down F9=Swap
F10=Left F11=Right F12=Cancel
```

Figure 8-69 Converted Extract file data

This was the end of our test data extract exercise.

8.9 Delete previously defined objects

Optim stores any object definitions it creates in a single repository. This section describes the process used to delete objects from the repository. Though normally this would not be required, it is useful if you have to work through a scenario multiple times as you develop your data extract. In real life we would expect that most customers would define their required objects and not delete the definitions until they were no longer required.

The objects that we had to delete were:

- ▶ Access Definitions
- ▶ Relationships
- ▶ Legacy Tables
- ▶ IMS Environment Definition

To begin the deletion process, we selected option 6 from the Optim menu and pressed Enter (Figure 8-70).

```
----- IBM's Optim -----
0  OPTIONS          - Site and User Options          SQLID  ==> DNET999
*  BROWSE TABLE   - Browse a DB2 Table             SUBSYS ==> DSNC
*  EDIT TABLE     - Edit a DB2 Table
*  BROWSE USING AD - Browse DB2 Tables Using Access Definition
*  EDIT USING AD   - Edit DB2 Tables Using Access Definition
5  ADS             - Create or Modify Access Definitions
6  DEFINITIONS     - Maintain Optim Definitions (Keys, Maps, ...)
7  MIGRATION       - Data Migration - Extract, Insert, Update, ...
*  COMPARE         - Compare Two Sets of Data
*  ARCHIVE         - Archive and Restore Data

T  TUTORIAL        - Information About IBM's Optim
C  CHANGES        - Changes from Prior Release(s)
X  EXIT            - Terminate Product Use

5855-U32 (C) Copyright IBM Corporation 1989, 2007.
All rights reserved. Licensed materials - property of IBM.
US Government Users Restricted Rights - Use, duplication or disclosure
restricted by GSA ADP schedule contract with IBM Corp.
Optim 5.5.2 Build: 276

OPTION ==>
```

Figure 8-70 Optim menu

To delete the existing Access Definition, we selected option 5 from the Choose a Definition Option screen (Figure 8-71).

```
----- Choose a Definition Option -----
OPTION ==> 5_
                                SQLID  ==> "DBA071"
                                SUBSYS ==> DSNC
1  PRIMARY KEYS   - Maintain Primary Keys
2  RELATIONSHIPS - Maintain Relationships
3  COLUMN MAPS   - Maintain Column Maps
4  TABLE MAPS   - Maintain Table Maps
5  ADS           - Maintain Access Definitions
6  LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7  IMS ENVIRONMENT - Maintain IMS Environment Definitions
8  IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E  EXPORT        - Export Optim Object Definitions
I  IMPORT        - Import Optim Object Definitions
```

Figure 8-71 Selecting a definition option

The next screen allows us to delete existing Access Definitions (Figure 8-72).

```

----- Choose an Access Definition -----
Command ==>
                                                    SUBSYS: DSN0

Access Definition:
  Group ==> EPSDEMO
  User  ==> DNET328
  Name  ==> FULLCELL%

Use '_' for DB2 LIKE character ==> YES   (Y-Yes, N-No)

To limit selection list to Access Definitions with certain start tables, enter
the start table name below.  A wild card is allowed at the end of each part.

  Start Table Creator ID ==>
  Start Table Name      ==>

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT      11=RIGHT     12=RETRIEVE
  
```

Figure 8-72 Choose an access definition

By entering the partial Access Definition name, we chose to select all those definitions with a name starting with FULLCELL. In our case, there was only one (Figure 8-73).

```

----- Select Access Definitions -----
Command ==>
                                                    Scroll ==> CSR
                                                    SUBSYS: DSN0
  Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 1

  ----- Access Definition ----- Last Modified -----
  Cmd  Group   User      Name          By           Date
  -----
  D_   EPSDEMO  DBA071   FULLCELLPHON DBA071       2008-05-21-13.26.47
  ***** TOP *****
  ***** BOTTOM *****
  
```

Figure 8-73 Select access definitions

By coding a **D** next to the Access Definition and pressing Enter, the existing object was deleted (Figure 8-74).

```

----- Select Access Definitions -----
Command ==> _                               Scroll ==> CSR
                                           SUBSYS: DSNC
Line Cnds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 1

----- Access Definition ----- Last Modified -----
Cmd  Group   User      Name      By      Date
-----
***** TOP *****
__ EPDDEMO DBA071 FULLCELLPHON DBA071 *DELETED
***** BOTTOM *****

```

Figure 8-74 Select Access Definitions

Using PF3, we returned to the Choose a Definition Option screen (Figure 8-75).

```

----- Choose a Definition Option -----
OPTION ==> 2_
                                           SQLID ==> "DBA071"
                                           SUBSYS ==> DSNC
1 PRIMARY KEYS - Maintain Primary Keys
2 RELATIONSHIPS - Maintain Relationships
3 COLUMN MAPS - Maintain Column Maps
4 TABLE MAPS - Maintain Table Maps
5 ADS - Maintain Access Definitions
6 LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7 IMS ENVIRONMENT - Maintain IMS Environment Definitions
8 IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E EXPORT - Export Optim Object Definitions
I IMPORT - Import Optim Object Definitions

```

Figure 8-75 Choose a Definition Option

Relationship objects were next to be deleted. Selecting option 2 took us to the relationship selection screen (Figure 8-76).

```

----- Choose a Relationship -----
OPTION ===> 3_                               SCROLL ===> CSR
                                           SUBSYS: DSNC
1 CREATE - Create a Relationship for Specified Parent or Child Table
2 MODIFY - Modify a Relationship for Specified Child Table
3 LIST   - List All Relationships for Specified Table

Specify Table Name (Child for OPTION 2, Parent or Child for OPTIONS 1 and 3)
Creator ID ===> SYS030                       >>
Table Name ===> OPT_CUSTOMER                 >>

Specify Relationship Name (OPTIONS 1 and 2)
Relationship Name ===> %                     >>

Specify Relationship Type (OPTIONS 2 and 3)
Relationship Type ===> ALL                   (P|O-OPT, D-DB2, I-IMS, A-All)

Use '_' for DB2 LIKE character ===> YES     (Y-Yes, N-No)

PF 1=HELP      2=SPLIT      3=END      4=RETURN      5=RFIND      6=RCHANGE
PF 7=UP        8=DOWN       9=SWAP     10=LEFT       11=RIGHT     12=RETRIEVE

```

Figure 8-76 Choose a relationship

To list all the relationships associated with our table OPT_CUSTOMER, we used option 3 with a wild card entry for the relationship name (Figure 8-77).

```

----- Select Relationships -----
Command ===>                               Scroll ===> CSR
                                           SUBSYS: DSNC
Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr, I-Info 1 OF 5
LR-List Rels for Other Table

----- Parent -----   ----- Child -----
Cmd Creator      Table      Creator      Table      Relation  Type
----->> ----->> ----->> ----->> ----->> ----->>
***** TOP *****
_ SYS030  OPT_CUSTOMER  SYS030  OPT_CUST_ACCT  CUSTOMER  DB2
_ SYS030  OPT_CUSTOMER  SYS030  OPT_CUST_XACTION  CUSTOMER  DB2
_ SYS030  OPT_CUSTOMER  DBA071  ICUSTSSN      CUST2IMS  *DELETED
_ SYS030  OPT_CUSTOMER  DBA071  SRC@PROVIDER1XACTN  CUST2PRO  *DELETED
_ SYS030  OPT_CUSTOMER  DBA071  SRC@PROVIDER2XACTN  CUST2PRO  *DELETED
***** BOTTOM *****

```

Figure 8-77 Select relationships

We deleted all except 2 objects from the list returned by entering a **D** in the left side column and pressing Enter.

Using PF3, we returned to select the next definition types for deletion (Figure 8-78).

```

----- Choose a Definition Option -----
OPTION ==> 6
SQLID ==> "DBA071"
SUBSYS ==> DSNC

1 PRIMARY KEYS - Maintain Primary Keys
2 RELATIONSHIPS - Maintain Relationships
3 COLUMN MAPS - Maintain Column Maps
4 TABLE MAPS - Maintain Table Maps
5 ADS - Maintain Access Definitions
6 LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7 IMS ENVIRONMENT - Maintain IMS Environment Definitions
8 IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E EXPORT - Export Optim Object Definitions
I IMPORT - Import Optim Object Definitions

```

Figure 8-78 Choose a definition option

Option 6 took us to the following screen (Figure 8-79).

```

----- Choose a Legacy Table -----
Command ==>
SUBSYS: DSNC

Legacy Table:
Creator ID ==> DBA071
Table Name ==> %

Use '_' for DB2 LIKE character ==> YES Y-Yes, N-No

```

Figure 8-79 Choose a legacy table

Here we used a wild card entry to provide a list of the existing Legacy Table definitions associated with userid DBA071 (Figure 8-80).

```

----- Select Legacy Table -----
Command ==> _
Scroll ==> CSR
SUBSYS: DSNC

Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attr 1 OF 5

----- Legacy Table ----- Last Modified -----
Cmd Creator Table Name By Date
-----
***** TOP *****
DBA071 ICUSTCC DBA071 *DELETED
DBA071 ICUSTCS DBA071 *DELETED
DBA071 ICUSTSSN DBA071 *DELETED
DBA071 SRC@PROVIDER1XACTN DBA071 *DELETED
DBA071 SRC@PROVIDER2XACTN DBA071 *DELETED
***** BOTTOM *****

```

Figure 8-80 Select a legacy table

All objects were deleted, again by keying a **D** in the left side column and pressing Enter.

PF3 returned us to the Choose a Definition screen. The next and final objects to be deleted were those associated with the IMS environment (Figure 8-81).

```

----- Choose a Definition Option -----
OPTION ==> 7_
                                SQLID ==> "DBA071"
                                SUBSYS ==> DSNC
1 PRIMARY KEYS   - Maintain Primary Keys
2 RELATIONSHIPS - Maintain Relationships
3 COLUMN MAPS   - Maintain Column Maps
4 TABLE MAPS   - Maintain Table Maps
5 ADS           - Maintain Access Definitions
6 LEGACY TABLES - Maintain Legacy Tables for Non-DB2 Data
7 IMS ENVIRONMENT - Maintain IMS Environment Definitions
8 IMS RETRIEVAL - Maintain IMS Retrieval Definitions

E EXPORT        - Export Optim Object Definitions
I IMPORT        - Import Optim Object Definitions
  
```

Figure 8-81 Choose a definition option - IMS

Selecting option 7 took us to this screen (Figure 8-82).

```

----- Choose an IMS Environment -----
Command ==>
                                SUBSYS: DSNC

IMS Environment:
  Environment Name ==> DBA071%_

Use '_' for DB2 LIKE character ==> YES  Y-Yes, N-No
  
```

Figure 8-82 Choose an IMS environment

Entering the userid with the wild card character ensured that all definitions created by userid DBA071 were found (Figure 8-83).

```

----- Choose An Environment Definition -----
Command ==>
                                Scroll ==> CSR
                                SUBSYS: DSNC

  Line Cmds: S-Select, D-Delete, C-Copy, R-Rename, AT-Attribute  1 OF 1

----- Last Modified -----
Cmd Env Name   By           Date
-----
***** TOP *****
d_ DBA071   DBA071   2008-05-21-12.11.45
***** BOTTOM *****
  
```

Figure 8-83 Choose an environment definition

Keying a **d** in the left side column and pressing Enter resulted in the lone IMS environment definition being deleted.

We have now deleted all Optim objects previously created for userid DBA071.

Archived

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 210. Note that some of the documents referenced here might be available in softcopy only.

- ▶ *IBM Application Development and Problem Determination Tools V7 for System z: Application Performance Analyzer, Debug Tool Utilities and Advanced Functions, Fault Analyzer, File Export, File Manager, and Workload Simulator*, SG24-7372

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Application Performance Analyzer for z/OS User's Guide*, SC23-8512-03
- ▶ *Debug Tool for z/OS Debug Tool Utilities and Advanced Functions for z/OS User's Guide*, SC19-1196-02
- ▶ *IBM Fault Analyzer for z/OS User's Guide and Reference*, SC19-1253-03
- ▶ *IBM File Manager for z/OS User's Guide and Reference*, SC19-1239-00
- ▶ *File Manager for z/OS User's Guide and Reference for CICS*, SC19-1242-00
- ▶ *Workload Simulator for z/OS and OS/390 User's Guide*, SC31-8948-00
- ▶ *Enterprise COBOL for z/OS Language Reference*, SC23-8528-00
- ▶ *Enterprise COBOL for z/OS Programming Guide*, SC23-8529-00

Online resources

These Web sites are also relevant as further information sources:

- ▶ For general information on the COBOL compilers:
<http://www.ibm.com/software/awdtools/cobol/>
- ▶ Release history of the various COBOL compilers:
<http://www.ibm.com/servers/eserver/zseries/zos/1e/history/history.html>
- ▶ For general information on the PLI compilers:
<http://www.ibm.com/software/awdtools/pli/>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

0C7 97

with Fault Analyzer 99

64-bit DB2 Universal Database 6

A

abend code

ABM0 74

ADATA 70

Add Watch Breakpoint 23

Address Breakpoints 20

Advanced Program-to-Program Communication (APPC)
52

Animated Step Into 21

ANSI 85 standard 5

APA

Batch

A01 panel - load source data 156

C02 report 154

C02 report - module expanded 155

C09 report 158

C09 report - modules expanded 159

Existing observation sessions 148

M01 report - real time monitoring 152

P01 report 157

R03 - job detail entered 150

R03 - select active jobs 151

R03 - specify job detail 149

S01 report 153

Selecting a previous window 161

Selecting C02 - Usage by Module report 154

Windows navigation 160

Building the observation session 119

CICS DB2

A01 - Confirmation that source code is loaded
137

A01 - Source Program Mapping 136

A02 - report JCL creation 146

A02 - report request screen 145

C01 report - SQL activity expanded 142

C01 report, one item expanded 131

D04 report 141

D05 report 140

D06 report 139

E03 report 133

E03 report, detail expanded 134

E03 report, showing request for source code 135

Existing observation sessions 118

F01 report 143

F04 report 144

Observation List 123

Observation session metrics 124

P01 report 138

Real time observation 122

Ro01 report 125

S01 report 127, 130

Specify CICS options 120

Specify job details 119

Specify subsystem options 121

W01 report 128

W01 report, expanded 129

Integration with OMEGAMON 46

report categories 125

start 118

to analyze batch performance 147

APAR

PK63876 53

APAR PK63876 53

Application Performance Analyzer for z/OS 1

Assembler 70

B

Batch reanalysis 8

BMS map 83

BMS maps 52

Breakpoints view 19

Browse minidump 28

Browse report 28

C

C and C++ (5694-A01) 71

CADP screen 76

Call Stack 19

CEEBINIT 72

CEEPIPI 72

CICS application performance 52

CICS Business Transaction Services (BTS) 52

CICS PA

Application Grouping 53

CICS Reports 52

Historical Database 53

CICS PA HDB 53

CICS System Resource Usage 52

CICS Web Support 52

CICS Web support 52

CMF data 52

COBOL 3, 62, 68

copybooks 38

COBOL for MVS and VM (5688-197) 64

Column Maps 13

Compiler listing 60, 67

Convert Process 14

Copybook

to describe the record format 35

Copybooks 176

CPU 155

CPU usage 4, 47

Create Object Definitions 13

Create Process 14
Creator ID (Dest CID) 195
Cross-system performance, 52
CSECT 47

D

DATASOURCE (DS) 177
DB2 7
DB2 Catalog 11
DB2 UNLOAD utility 9
DB2, 3
Debug Console 19
Debug Tool
 Integration with Rational Developer for System z 18
Debug Tool for z/OS and Debug Tool Utilities and Advanced Functions 1
Debug Tool Utilities 5
Debug Tool Utilities and advanced functions (DTUAF) 18
Debug view 18
Debugger editor 18
Delete cached data 28
Delete from view 28
DFSVSAMP 168
DSNTIAUL 9
DTUAF 18
DWARF 62

E

Eclipse platform 15
Enterprise COBOL 6
Enterprise COBOL V3 (5655-G53) 63
Enterprise COBOL V4.1 (5655-S71) 62
Enterprise PLI V3.4 and earlier (5655-H31) 69
Enterprise PLI V3.5 and V3.6 (5655-H31) 68
Enterprise PLI V3.7 and later (5655-H31) 67
Entry Breakpoints 20
EQALANGX 62
ESDS 34
Event Notification 45
Exception Breakpoints 20
EXEC CICS SEND MAP 81

F

F 'APA CICS DB2 123
FA
 Artefact 99
 Synopsis of abend 29
FA Artefacts View 25
Fault Analyzer 74
 Lookup view 33
 Source code display 31
Artifacts view 24
Dump browser 30
Integration with Rational Developer for System z 24
Report browser 28
Fault Analyzer for z/OS 1
Fault History details 100

Field Metadata 40
File and TSQueue resource usage 52
File Manager
 Integration for Rational Developer for System z 34
File Manager for z/OS 1
FTP 55

G

governance 166

H

Hex Editing 42
HFS datasets 9
High Level Assembler (HLASM) 18

I

IBM Fault Analyzer for z/OS 6
 Analysis options 7
 Support for IBM subsystems 7
IBM File Manager for z/OS 8
 components 9
 Highlights 8
IBM Parallel Sysplex Coupling Facility 3
IBM Tivoli Enterprise Portal. 3
IBM Tivoli OMEGAMON 3
IBM® Tivoli® OMEGAMON® 48
ICUSTSSN 172
Identify extract data sources 180
IMS 7, 168
 buffer parameters 171
 DBD 173
 environment 171
 Legacy Tables 171
 program libraries 171
 segment 173
IMS, 3
Insert Process 13
Insert Processing 13
Interactive reanalysis 8
Interactive System Productivity Facility (ISPF) 5
ISPF menu 148

J

Java 2 Platform, Standard Edition (J2SE) 4
Java Virtual Machine (JVM) support 52
JCL 97
JES 97

K

KSDS 34

L

LANGX 61, 67
LE COBOL 60
Load Breakpoints 20
Load Process 14

LPEX Editor 36

M

Macros 68
Memory view 19
Modules view 19
Monitor Expression 23
Monitor Memory 23
Monitors view 19
MP (Multiply Decimal) 159
Multiple Virtual Storage (MVS) 31
Multi-Region Operation (MRO) 52
MVS Workload Manager (WLM) 52

N

NOOPTIMIZE 66

O

OC7
 With Debug Tool 102
OMEGAMON with APA 48
Optim MOVE 1, 11
 Associate IMS segments with IMS database 189
 Browse results 199
 Choose a legacy table 174
 Choose a relationship 183
 Choose definition option - IMS 170
 Choose definition option - Legacy Tables 171
 CONVERT process report 197
 CONVERT process table map 195
 Converted Extract file data 201
 Converted Extract file report 200
 Data migration 180, 198
 Define extract selection criteria 187
 Define IMS environment 171
 Define legacy table 172–173
 Delete previously defined objects 201
 Environment definition selection 170
 EXTRACT data sources 187
 Extract expression definition 196
 EXTRACT process 180
 EXTRACT process report 192
 Extract the data 189
 Extract, archive or control file browse parameters 198
 Main menu screen 169
 Modify Legacy Table 175
 Processing Flow 12
 Select Tables/Views 181
 Specify CONVERT parameters and execute 196
 Specify copybooks for legacy table 172, 174
 Test Data Management 11
 Use table lookup to convert extracted data 194
 View extract report 192
Optim MOVE for DB2 165
Optim MOVE z/OS 11
Optimized code support for COBOL 2
OS/VS COBOL 5, 61
OS/VS COBOL (5740-CB1) 66

P

PD Tools
 with RDz 95
PDS 9, 60
PDSE 60
PL/I 73
 includes 38
PLI 67
PLI for MVS and VM (5688-235), 69
port number 103
Properties Window 97
PSB libraries 173

Q

QSAM 9, 177
Query Management Facility (QMF) 53

R

Rational Developer for System z 1, 6, 15
RDz LPEX editor 34
Real-time analysis 8
Redbooks Web site 210
 Contact us xi
Registers view 19

S

S0C7 abend 101
SAM data set 178
SAM2 104
Secure Sockets Layer (SSL) 52
SEQAMOD 72
Service oriented architecture (SOA) 2
Set encoding 28
SNA 14
SOAP for CICS 52
SQL 118
SRP (Shift and Round Decimal) 159
Statement breakpoints 20
Step Into 21
Step Over 21
Step Procedures 21
Step Return 21
SYSADATA 62, 71
SYSDEBUG 60, 67

T

Table Maps 13
TCP/IP 14
Temporary storage 52
temporary storage queues 10
TEMS (Tivoli Enterprise Management System) 48
TEP (Tivoli Enterprise Portal) 48
Tool Integration 48
Transaction Resource Usage 52
Transient data 52
transient data queues 10
TRT (Translate and Test) 159

TSO region 47

U

UNIX System Services 3

UNIX® System Services. 18

V

v3.5 Enterprise PLI compiler 60

Variable Tools 22

Variables view 19

VS COBOL II (5668-958) 65

VSAM (including IAM - Innovation Access Method™ files)
9

VTAM 102

W

WAIT state 122

Watch Breakpoints 20

WebSphere Application Server 7

WebSphere MQ 3, 46

Workload Simulator 1, 14

X

XML data from files 10

XML tags 10

Z

z/OS

file mapping for the sample data set 36

z/OS files 166

z/OS protected-storage 60

z/OS subsystems 6



IBM Application Development and Problem Determination



Integration with Rational Developer for System z

IBM Optim MOVE usage scenario

FA, FM, and APA usage scenarios

The Problem Determination toolset, or PD Tools for short, consists of a core group of IBM products designed to work in conjunction with compilers and run times to provide a start-to-finish development solution for the IT professional. This IBM Redbooks publication can provide you with an introduction to the tools, guidance for program preparation for use with them, an overview of their integration, and several scenarios for their use.

When developing a system, an iterative process is usually employed. You can decide to create a starting set of test data using Optim MOVE to strip or modify existing databases, files, and data relationships. File Manager can also scramble, alter, or otherwise process this data. Once created, this data can be used to test the newly developed or maintained software.

Should an abend during testing occur, Fault Analyzer enables the programmer to quickly and easily pinpoint the abending location and optionally, the failing line of code. Many times, this is all the programmer requires to correct the problem. But occasionally, it might be necessary to delve a little deeper into the code to figure out the problem. Debug Tool allows the programmer to step through the code at whatever level is required to determine where the error was introduced or encountered.

After the code or data is corrected, the same process is followed again until no errors are encountered. But, this is not the whole story. Volume testing or testing with multiple terminals is sometimes required to ensure real-world reliability. Workload Simulator can be utilized to perform this type of testing.

After all tests have been completed, running the application using Application Performance Analyzer will ensure that no performance bottlenecks are encountered. It also provides a baseline to ensure that future enhancements do not introduce new performance degradation into the application.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7661-00

ISBN 073843146X