IBM

Threadsafe Considerations for CICS

New threadsafe options in CICS Transaction Server for z/OS V4.2

The value of threadsafe applications in a CICS environment

CICS Tools to help migrate applications to be threadsafe

Chris Rayns
George Bogner
Philip Hale
Em James
Peter Klein
John Tilling

Redbooks

ibm.com/redbooks

International Technical Support Organization

Threadsafe Considerations for CICS

April 2012

SG24-6351-04

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

Fifth edition (April 2012)

This edition applies to CICS Transaction Server for z/OS V4.2.

© Copyright International Business Machines Corporation 2004, 2012. 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

© Copyright IBM Corp. 2004, 2012. All rights reserved.

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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

© Copyright IBM Corp. 2004, 2012. All rights reserved.

Trademarks

IBM, the IBM logo, and 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 Explorer®
CICSPlex®
CICS®
DB2®
eServer™
IBM®
IMS™
Language Environment®
MVS™
NetView®
OMEGAMON®
OS/390®
RACF®
Redbooks®
Redbooks (logo) ®
Resource Measurement Facility™
RMF™
Tivoli®
VTAM®
WebSphere®
z/OS®
zSeries®

The following terms are trademarks of other companies:

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

Originally, IBM® CICS® employed a single task control block (TCB) to process, for example, application code, task dispatching, terminal control, and file control that run on the *application* or quasi-reentrant (QR) TCB. Over time, IBM added specialized TCBs to CICS to help offload management tasks from the overcrowded QR TCB. Virtual Storage Access Method (VSAM) subtasking, the IBM VTAM® High Performance Option, and asynchronous journaling were implemented on separate TCBs. The IBM DB2® and MQ Series (now called *WebSphere® MQ*) attachment facilities also employed TCBs apart from the application TCB. Distributing processing among multiple TCBs in a single CICS address space is not new. However, customers and ISVs had little control over which TCB CICS was selected to dispatch a function.

Beginning with CICS Version 2, applications can run on TCBs apart from the QR TCB, which has positive implications for improving system throughput and for implementing new technologies inside of CICS. Examples of implementing new technologies include using the IBM MVS™ Java virtual machine (JVM) inside CICS and enabling listener tasks written for other platforms to be imported to run under CICS.

The newest release, CICS Transaction Server for z/OS® (CICS TS) V4.2, includes scalability enhancements so that you can perform more work more quickly in a single CICS system. The advantage of this enhancement is that you can increase vertical scaling and decrease the need to scale horizontally, reducing the number of regions that are required to run the production business applications. The scalability enhancements in CICS TS V4.2 fall into two broad areas, which are increased usage of open transaction environment (OTE) and of 64-bit storage.

This IBM Redbooks® publication is a comprehensive guide to threadsafe concepts and implementation for IBM CICS. This book explains how systems programmers, applications developers, and architects can implement threadsafe applications in an environment. It describes the real-world experiences of users, and our own experiences, of migrating applications to be threadsafe. This book also highlights the two most critical aspects of threadsafe applications: system performance and integrity.

© Copyright IBM Corp. 2004, 2012. All rights reserved.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Chris Rayns is an IT Specialist and Project Leader at the ITSO in Poughkeepsie, New York. He writes extensively on all areas of CICS, including CICS Tools, CICS Transaction Gateway, and CICS Transaction Server. Before joining the ITSO, he worked in IBM Global Services in the UK as a CICS IT Specialist.

George Bogner is a Software IT Specialist working in IBM Sales and Distribution and supporting the CICS Transaction Server product suite. George worked at IBM for 25 years, specializing in the DB/DC area, working with CICS, IBM IMS™, and DB2, supporting client accounts in IBM Global Services. He currently works out of Raleigh, North Carolina, supporting North American clients by providing CICS seminars, proofs of technology (POT), proofs of concept (POC), and consulting services for CICS-related topics.

Philip Hale is a software developer in the Australian Development Lab, in Perth, Australia. Phil has over 30 years of experience in application development and systems and production support roles. He has also worked over 20 years in commercial application mainframe environments, such as banking, insurance, health, oil, and utilities, using CICS DB2, DLI, VSAM Assembler, COBOL, PLI, PLX, and REXX. For the past 10 years, he has been performing mostly L3 development roles and L3 Change Team roles within the ADL. He has worked on IBM z/OS products including IBM Tivoli® Asset Discovery for z/OS, Tivoli Composite Application Manager for CICS, Tivoli NetView®, and Systems Automation for OS/390®.

Em James is a Technical Specialist for CICS Tools working at IBM Hursley Labs in the UK. He has 22 years of experience working with CICS as both an application programmer and a systems programmer. He most recently worked as the lead developer on the CICS Interdependency Analyzer product. He has a degree in computer science from Loughborough University.

Peter Klein is a CICS Team Leader at the IBM Germany Customer Support Center. He has 19 years of experience working as a Technical Support Specialist with IBM software products. His expertise includes WebSphere MQ, IBM CICSPlex® System Manager, and distributed transaction systems. Peter has contributed to several other Redbooks publications and ITSO projects sponsored by IBM Learning Services.

John Tilling is a Senior Software Engineer working in the CICS Strategy and Planning group at the IBM Hursley Laboratory in the United Kingdom. He joined IBM in 1985 having graduated from York University with a degree in Computer

Science and Mathematics. He has 21 years of experience developing CICS, working in data access components including file control, local DLI, CICS-DBCTL, and was responsible for restructuring the CICS-DB2 Attachment Facility to exploit OTE.

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

Richard M. Conway
ITSO, Poughkeepsie Center

George Burgess
John Tilling
IBM Hursley

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

- Fourth edition (November 2010): Edward Addison, Diana Blair, George Bogner, David Carey, Tony Fitzgerald, Scott McClure, Christen Plum, John Tilling, and Andy Wright
- Third edition (November 2007): Edward Addison, George Bogner, David Carey, Tony Fitzgerald, Steve Foley, Jim Grauel, Fabrice Jarassat, Scott McClure, Keith Patterson, Christen Plum, John Tilling, and Andy Wright
- Second edition (July 2006): George Bogner, Tony Fitzgerald, Steve Foley, Jim Grauel, Fabrice Jarassat, Scott McClure, Keith Patterson, John Tilling, and Andy Wright
- First edition (August 2004): George Bogner, Tony Fitzgerald, Steve Foley, Jim Grauel, Fabrice Jarassat, Scott McClure, Keith Patterson, and John Tilling

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email 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

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition might also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6351-04
for *Threadsafe Considerations for CICS*
as created or updated on April 5, 2012.

April 2012, Fifth Edition

This revision reflects the addition, deletion, or modification of new and changed information described in the following section.

Changed information

We updated Chapters 1 - 8 and Appendix A to reflect the following changes:

- A new concurrency option on the program definition that allows for greater usage of the open transaction environment (OTE) for threadsafe applications
- Usage of the OTE for function shipping, by allowing the mirror program, when started in a remote CICS region through use of an IPIC connection, to run on an open task control block (TCB)
- Making more of the application programming interfaces (API) and system programming interfaces (SPI) threadsafe, including access to IMS databases by using the CICS IMS Database Control (DBCTL) interface

November 2010, Fourth Edition

This revision reflects the addition, deletion, or modification of new and changed information described in the following section.

- In Chapter 5, we updated the chapter to include IBM CICS Explorer®.

© Copyright IBM Corp. 2004, 2012. All rights reserved.

November 2007, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information described in the following sections.

New information

- CICS File Control interface under CICS Transaction Server (CICS TS) V3.2
- Threadsafe CICS journaling commands under CICS TS V3.2
- Threadsafe definition for system autoinstalled global user exits (GLUEs)
- WebSphere MQ interface under CICS TS V3.2
- Performance case studies

Changed information

- In Chapter 1, we updated the OPENAPI information.
- In Chapter 2, we added the new functions for CICS TS V3.2.
- In Chapter 3, we made updates for CICS TS V3.2.
- In Chapter 5, we made updates for CICS Interdependency Analyzer and CICS VSAM Transparency.
- In Chapter 6, we added a file control application example.
- In Chapter 7, we updated the chapter with the functions for CICS TS V3.2.
- In Chapter 8, we updated the chapter with the functions for CICS TS V3.2.
- In Chapter 14, we updated the chapter with new questions about CICS TS V3.2.
- In Appendix A, we made updates to include CICS TS V3.2 APAR information.

July 2006, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described in the following sections.

New information

- Full OTE OPENAPI
- CICS Transaction Server V3.1
- XPLINK

Changed information

- In Chapter 5, we added the CICS Performance Analyzer and CICS Interdependency Analyzer sections.
- In Chapter 8, we added a section about OPENAPI.

Part 1

Threadsafe concepts and definitions

This part introduces threadsafe concepts and definitions, provides an overview of threadsafe considerations in CICS, and explains techniques for ensuring that applications will operate as expected in a multiprocessing environment.

This part includes the following chapters:

© Copyright IBM Corp. 2004, 2012. All rights reserved.

1

Introduction to CICS open transaction environment

This chapter provides introductory information about the CICS open transaction environment (OTE). It includes the following sections:

- The concept of the CICS open transaction environment
- CICS data integrity for shared resources
- Benefits of threadsafe-enabled applications

© Copyright IBM Corp. 2004, 2012. All rights reserved.

1.1 The concept of the CICS open transaction environment

CICS OTE is an architecture that was introduced for the following purposes:

- To allow CICS to make better use of the mainframe

  With OTE, CICS can run more processes in parallel, increasing the throughput of work through the system and resulting in more work being done in the same amount of time.

- To improve the performance of existing applications, particularly those applications that access external resources managers, such as IBM DB2, WebSphere MQ (WMQ), and IMS, by consuming less mainframe resources in getting the job done
- To augment the already rich set of capabilities provided by the CICS application programming interface (API), by providing application interfaces supplied by other software components, and allowing CICS applications to use these interfaces.

To benefit from the power of OTE, you must ensure that your applications are threadsafe. Having threadsafe applications ensures that, if the mainframe has many processors, and many processes are running in parallel, the threadsafe application runs correctly, and the right result is achieved. CICS ensures that its code runs correctly, but customers must ensure that their COBOL code, for example, implementing payroll, accounts, and ledger, runs correctly. If an application is threadsafe, it can be defined to CICS with a CONCURRENCY keyword so that it uses OTE. If an application is not threadsafe, CICS runs it without using OTE.

Applications that cannot use OTE must run on the main CICS task control block (TCB), the quasi-reentrant (QR) TCB. Applications that use OTE can run on a CICS open TCB. A CICS system has only one QR TCB, and the CICS dispatcher shares use of the QR TCB between all the tasks. However, a single CICS system can have many open TCBs. Using OTE effectively keeps an application running on an open TCB for as long as possible and minimizes the number of times it must switch back to the QR TCB. The result is processor savings and improved throughput because the open TCBs can run in parallel and take advantage of the multiprocessor mainframe.

OTE enhancements in CICS Transaction Server for z/OS (CICS TS) V4.2 fall into the following areas:

- The introduction of the REQUIRED concurrency attribute on the program definition, which allows for greater exploitation of OTE for threadsafe applications
- Usage of OTE for function shipping, by allowing the mirror program, when it is started in a remote CICS region by using an IP interconnectivity (IPIC) connection, to run on an open TCB
- Making more of the API and system programming interface (SPI) threadsafe, including access to IMS databases by using the CICS IMS Database Control (DBCTL) interface

> **OTE support:** OTE is supported from IMS 12, with program temporary fixes (PTFs) for APARs PM31420, PM47327, and PM45414 applied.

1.1.1 Improved throughput

OTE introduces a new class of TCB, called an *open TCB*, that can be used by applications. An open TCB is characterized by the fact that it is assigned to a CICS task for its sole use, and multiple OTE TCBs can run concurrently in CICS. Several modes of open TCBs are used to support various functions, such as Java in CICS, open API programs, and C and C++ programs, which are compiled with the XPLink option.

Other CICS tasks cannot be subdispatched on an open TCB.

The OTE introduces several new engines (TCBs) to CICS program execution. Each new TCB can run on one processor in parallel (concurrently), which provides the potential of increased throughput for a single CICS system if the necessary processor power is present.

1.1.2 Improved performance

Each new TCB represents a thread where a CICS program can run in parallel. When the CICS program continues to run on the open TCB, it is called a *threadsafe execution* of the program. The result is a reduced number of TCB switches between the open TCB and the QR TCB. In turn, the result is in reduced CPU consumption that corresponds to the number of saved TCB switches. The more CICS commands that are made threadsafe, the higher the probability that your programs will remain running on the open TCB.

1.2 CICS data integrity for shared resources

This section highlights the concept of quasi-reentrant execution and threadsafe execution in relation to access to shared resources.

1.2.1 Quasi-reentrant and threadsafe programs

Programs are said to be *quasi-reentrant programs* because they take advantage of the behavior of the CICS dispatcher and the QR TCB. In particular, only one CICS task is ever active under the QR TCB. That is, although the same program can be run by multiple CICS tasks, only one of those CICS tasks is active at any time. Compare this situation with one in which multiple instances of the same program are each running under a separate TCB. In this scenario, multiple tasks are active in the same program at the same time, and the program must be fully MVS reentrant at the least. For a program to be threadsafe, it must go beyond being fully reentrant and use appropriate serialization techniques when accessing shared resources.

QR programs always run under the QR TCB. They can access shared resources safe in the knowledge that they are the only CICS user tasks running at that time. Such shared resources include the common work area (CWA) or shared storage obtained by using EXEC CICS GETMAIN SHARED. Running under the QR TCB guarantees serialized access to those shared resources. An example is a program that updates a counter in the CWA. The program is alone to update this counter. When it stops or is suspended by the CICS dispatcher, it detects that the counter still has the value that was assigned.

1.2.2 Shared application resources

Multiple tasks can access shared resources simultaneously. Therefore, when running under an open TCB, applications that access shared resources must ensure the integrity of those resources by implementing an appropriate serialization technique.

1.2.3 Shared CICS resources

CICS ensures the integrity of all the resources it manages. The CICS code is amended to run on multiple TCBs safely (for example, the CICS code that handles temporary storage requests), or it ensures that the code runs on the QR TCB.

The use of nonthreadsafe CICS commands that must run on the QR TCB can, depending on the application, have a performance penalty. This penalty results because of the need to switch TCBs when a nonthreadsafe CICS command is encountered. If many nonthreadsafe CICS commands are in a program that is otherwise threadsafe, the extra switching back to the QR TCB has a detrimental effect on performance, but with no risk to data integrity.

In an example of a program that uses a CWA counter, by implementing an appropriate serialization technique, this former QR program runs in an OTE environment. Therefore, multiple instances of this program can run at the same time. The counter value in the CWA can be changed by multiple executors at the same time, and one instance is always sure about the counter value when it stops or is suspended.

1.2.4 Threadsafe applications

For the purposes of this book, the term *threadsafe application* is defined as a collection of application programs that employ an agreed-upon form of serialized access to shared application resources. A program written to *threadsafe standards* is a program that implements the agreed-upon serialization techniques. A single program operating without the agreed-upon serialization technique can negatively affect the predictability and, therefore, the integrity of an entire system of otherwise threadsafe programs. Therefore, an application system cannot be considered *threadsafe* until all programs that share a common resource implement the threadsafe standards of that application.

Threadsafe versus nonthreadsafe application: An application that does not use any of the shared resources can be considered threadsafe even if it uses nonthreadsafe CICS commands. Such application is not considered threadsafe if it is self-modifying and is, therefore, not re-entrant.

1.3 Benefits of threadsafe-enabled applications

The following potential business drivers, as explained in this section, lead CICS customers to enable their applications to a threadsafe environment:

- Improving performance
- Reducing the cost of computing
- Using OTE

However, risk associated with defining an application as threadsafe. You must understand this risk and eliminate it before you attempt threadsafe enablement. For more information, see 1.3.4, “The importance of understanding your applications” on page 10.

1.3.1 Improving performance

Customers who benefit most from enabling a threadsafe environment are those customers who experience poor response times for any of the following reasons:

- CICS QR TCB is CPU constrained.
- Application tasks are waiting excessively for the QR TCB.
- CICS region in general is CPU constrained.

The following sections examine these situations in detail.

CICS QR TCB is CPU constrained

In this scenario, the CICS QR TCB is consistently reaching system central processor (CP) SHARE (QR TCB is running at 100% CPU) and must wait to be dispatched by the operating system. Every task running under the QR TCB is being delayed.

Defining transactions as threadsafe, processing as many tasks as possible on an open TCB removes this constraint on the QR TCB. It also reduces the response times of both threadsafe and nonthreadsafe transactions.

CP SHARE calculation: CP SHARE is the amount of CP that a logical partition (LPAR) is guaranteed before it is eligible to have the CP removed. For CICS to perform well, the CP SHARE for the LPAR, where it is running, must be fairly high (>90% is great; 80% is good; 70% is workable). Use the following equation to calculate CP SHARE:

```
CP SHARE = ((# available physical CP x 100)/(# logical CP in LPAR)) x FAIR
SHARE
```

Application tasks are waiting excessively for the QR TCB

In this scenario, the QR TCB is not CPU constrained, but application tasks are contending for their share of QR. Again, defining transactions as threadsafe and moving as many tasks as possible to an open TCB reduces contention for the QR TCB. It also reduces the response times of threadsafe and nonthreadsafe transactions.

CICS region in general is CPU constrained

In this scenario, the system as a whole is at or approaching 100% busy, and CICS is being constrained with everything else. Depending on how an application is designed, defining it as threadsafe can significantly reduce the path length of application tasks.

The transactions that achieve the greatest CPU reduction are likely to be many different applications that have the following characteristics:

- A significant number of EXEC SQL, IMS, or WMQ calls are started per task.
- All programs started between the first and last EXEC SQL, IMS, or WMQ call in each task are defined as threadsafe.
- All exits started as part of an EXEC SQL call are defined as threadsafe and only contain threadsafe EXEC CICS commands.
- All exits started between the first and last EXEC SQL, IMS, or WMQ call in each task are defined as threadsafe.
- All EXEC CICS statements started between the first and last EXEC SQL, IMS, or WMQ call in each task are threadsafe.

Defining transactions with the preceding characteristics as threadsafe all but eliminates TCB switches for the associated CICS tasks.

1.3.2 Reducing the cost of computing

Reducing the CPU consumption of an application does not always necessarily result in improved response times. An application can be a heavy user of a CPU. However, if the CPU has spare capacity and the application is not CPU constrained, a reduction in path length can have a negligible impact on response times.

However, for many customers, the financial cost incurred running their applications is related to the amount of CPU that is used. Under these circumstances, the CPU savings gained by enabling the appropriate applications to a threadsafe environment can equate to financial savings. As show in Chapter 8, “Migration scenario” on page 203, the CPU savings for some applications can be substantial.

1.3.3 Using OTE

OTE in CICS was implemented in four stages, over several releases of the CICS TS:

- Stage 1: OTE function introduced (delivered in CICS TS V1.3)
- Stage 2: Task-related user exits (TRUEs) can use OTE (in CICS TS V2.2)
- Stage 3: Full application use of open TCBs (delivered in CICS TS V3.1)
- Stage 4: The following OTE enhancements (delivered in CICS TS V4.2)
  - The introduction of a new concurrency option, REQUIRED, on the program definition, which allows for greater usage of OTE for threadsafe applications
  - Usage of OTE for function shipping, by allowing the mirror program, when it is started in a remote CICS region through an IPIC connection, to run on an open TCB

– Making more of the APIs and SPIs threadsafe, including access to IMS databases by using the CICS DBCTL interface

Applications that can be defined as threadsafe in CICS TS V2 can use the enhancements provided in CICS TS V3.1 and later with minimum enablement effort. Moreover, all new application programs are written to threadsafe standards at the level of CICS that they are developed.

1.3.4 The importance of understanding your applications

A *threadsafe application* refers to a program that has one of the following conditions:

- Uses appropriate serialization techniques, such as Compare and Swap or enqueue, when accessing any shared application resources. It must be able to run concurrently on multiple TCBs and must not rely on QR to serialize access to shared resources and storage.
- Uses no shared application resources whatsoever.

For an application to meet these conditions and, therefore, be considered threadsafe, the application must have both of the following characteristics:

- Incorporate threadsafe application logic, meaning that the existing language code in between the EXEC CICS commands must be threadsafe.
- Be defined to CICS as threadsafe.

Rule for defining programs as threadsafe: Before you define any programs as threadsafe, you must understand whether an application is threadsafe and serialize all access to all shared resources. Otherwise unpredictable results can occur, placing the integrity of the application data at risk.

The following IBM CICS Tools can assist in understanding your applications:

- CICS Interdependency Analyzer
- CICS Performance Analyzer
- CICS Configuration Manager

For more information about how these tools can help you, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245. For more information about the CICS Tools, see Appendix A, “Overview of CICS Tools” on page 337.

2

Overview of an open transaction environment and threadsafe applications

This chapter addresses the following program types (definitions) in CICS:

- Quasi-reentrant (QR)
- Threadsafe:
  - CICSAPI
  - OPENAPI
- Required:
  - CICSAPI
  - OPENAPI

This chapter explains what determines each type, how to define the associated program definition, and the requirements CICS expects of each type. It examines the history of the open transaction environment (OTE) in CICS and the relationship of OTE to open task control blocks (TCBs), task-related user exits (TRUEs), and TCB limits. This chapter includes the following sections:

- Overview of quasi-reentrant and threadsafe programs
- Open transaction environment: A brief history
- Techniques to ensure threadsafe processing

© Copyright IBM Corp. 2004, 2012. All rights reserved.

- Program definition
- Task-related user exit APIs
- TCB limits
- Open TCB performance
- TCB considerations with UNIX System Services
- Static and dynamic calls
- Threadsafe API commands
- Threadsafe SPI commands
- Threadsafe XPI commands
- Nonthreadsafe SPI commands
- Function shipping considerations

2.1 Overview of quasi-reentrant and threadsafe programs

Definitions of important terms that are relevant to the open transaction environment are provided in this section.

2.1.1 Quasi-reentrant programs

CICS runs user programs under a TCB managed by CICS. If a program is defined as quasi-reentrant, using the CONCURENCY attribute of the program resource definition, CICS always starts the program under the CICS QR TCB. The requirements for a quasi-reentrant program, in a multithreading context, are less stringent than if the program were to run concurrently on multiple TCBs.

CICS requires an application program to be reentrant to guarantee a consistent state. A program is considered reentrant if it is read only and does not modify storage within itself. In practice, an application program might not be truly reentrant. CICS expects *quasi-reentrancy*, which means that the application program must be in a consistent state when control is passed to it, both on entry to the program and before and after each EXEC CICS command. Quasi-reentrancy guarantees that each invocation of an application program is unaffected by previous runs or by concurrent multithreading through the program by multiple CICS tasks.

CICS quasi-reentrant user programs (application programs, user-replaceable modules, global user exits (GLUEs), and TRUEs) are given control by the CICS dispatcher under the QR TCB. When running under this TCB, a program can be sure that no other quasi-reentrant program can run until it relinquishes control during a CICS request. The user task is suspended, leaving the program still *in use*. The same program can then be reinvoked by another task, meaning that the application program can be in use concurrently by more than one task although only one task is running at a time.

To ensure that programs cannot interfere with the working storage of each other, CICS obtains a separate copy of working storage for each application program start. Therefore, if a user application program is in use by 11 user tasks, 11 copies of working storage are in the appropriate dynamic storage area (DSA).

Quasi-reentrancy allows programs to access globally shared resources, such as the CICS common work area (CWA), without needing to protect those resources from concurrent access by other programs. Such resources are effectively locked by the running program until it relinquishes control. Therefore, an application can update a field in the CWA without using compare and swap (CS) instructions or locking (enqueuing on) the resource.

> **Important:** The CICS QR TCB provides protection through exclusive control of global resources *only* if all user tasks accessing those resources run under the QR TCB. It does not provide automatic protection from other tasks that run concurrently under another (*open*) TCB.

Specifying a value of `Quasirent` for the *COncurrency* attribute of the program definition is supported for all executable programs.

2.1.2 Threadsafe programs

In the CICS open transaction environment, threadsafe application programs, OPENAPI TRUEs, GLUE programs, and user-replaceable modules cannot rely on quasi-reentrancy because they can run concurrently on multiple open TCBs. Furthermore, even quasi-reentrant programs are at risk if they access resources that can also be accessed by a user task running concurrently under an open TCB. That is, the techniques used by user programs to access shared resources must take into account the possibility of simultaneous access by other programs. Programs that use appropriate serialization techniques when accessing shared resources are described as *threadsafe programs*.

> **Fully reentrant:** The term *fully reentrant* is sometimes used but can be misunderstood. Therefore, *threadsafe* is the preferred term.

CICS resources

For CICS resources, such as temporary storage queues, transient data queues and Virtual Storage Access Method (VSAM) files, CICS processing automatically ensures access in a threadsafe manner. CICS ensures that its resources are accessed in a threadsafe way for either of the following reasons:

- The CICS application programming interface (API) code is made threadsafe.
- CICS ensures that the command is run on the QR TCB, which effectively serializes access to the resource.

Application resources

For application-maintained shared resources, the application program is responsible for ensuring that the resource is accessed in a threadsafe manner. Typical examples of shared storage are the CICS CWA, GLUE global work areas, and storage acquired explicitly by the application program with the shared option.

You can check whether your application programs use these types of shared storage by looking for occurrences of the following EXEC CICS commands:

- `ADDRESS CWA`
- `EXTRACT EXIT`
- `GETMAIN SHARED`

Application programs that use these commands *might* not be threadsafe because they allow access to global storage areas that can be updated concurrently by several tasks running on different open TCBs. To ensure that an application is threadsafe, it must include the necessary synchronization logic to guard against concurrent updates. To help you find occurrences of these commands, CICS provides the DFHEIDTH table, which is a sample command table that you can use with the load module scanner utility, `DFHEISUP`.

The CICS Interdependency Analyzer tool provides real-time information about the EXEC CICS commands used by a program. For more information, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245.

> **Important:** The DFHEIDTH table does not test the scanned programs for nonthreadsafe CICS commands. It merely identifies whether the application is using CICS commands that give rise to the *possibility* that the application logic is nonthreadsafe.

During your investigation of identifying programs that use shared resources, you must include any program that modifies itself. Such program is effectively sharing storage and must be considered at risk.

2.1.3 CICSAPI programs

A CICSAPI program is restricted to use only the CICS API, which by definition includes the following interfaces:

- The command-level API
- The system programming interface (SPI)
- The resource manager interface (RMI)
- The exit programming interface (XPI), for GLUEs
- The system application architecture common programming interfaces, CPI-C and CPI-RR
- IBM Language Environment® callable services

A CICSAPI program starts running on the QR TCB. Calls to an OPENAPI-enabled TRUE cause a switch to an open TCB to run the TRUE. Whether the program is defined as threadsafe or quasi-reentrant dictates whether control returns to the application from the TRUE on the open TCB or the QR TCB.

A new **`CONCURRENCY(REQUIRED)`** parameter was introduced with CICS Transaction Server for z/OS (CICS TS) V4.2. When **`CONCURRENCY(REQUIRED)`** is specified, the program receives control on an open TCB and continues running on the open TCB until a switch to the QR TCB is forced (for example, an EXEC CICS call to a nonthreadsafe command). After the work is completed on the QR TCB, the program then continues to run on the open TCB.

2.1.4 OPENAPI programs

With CICS TS V3 and later, programs can run on an open TCB from the start of the program, called an OPENAPI program. An *OPENAPI program* is a program that is written to threadsafe standards and does not rely on a call to a TRUE to move the program to an open TCB. An OPENAPI program *must* be run on an open TCB.

An OPENAPI program is also not restricted to the CICS API. An OPENAPI program can use APIs regardless of whether they are for CICS. However, the documentation for CICS TS V3.1 and later states that using an API that is not for CICS is at the risk of the user. IBM has not tested the usage of APIs that are not for CICS.

Threadsafe program: An OPENAPI program must always be threadsafe.

2.2 Open transaction environment: A brief history

This section charts the history of the open transaction environment and outlines the enhancements that have been introduced in each release of CICS TS for z/OS.

Figure 2-1 shows the key OTE enhancements introduced in recent releases of CICS, which are explained in more detail in the following sections.

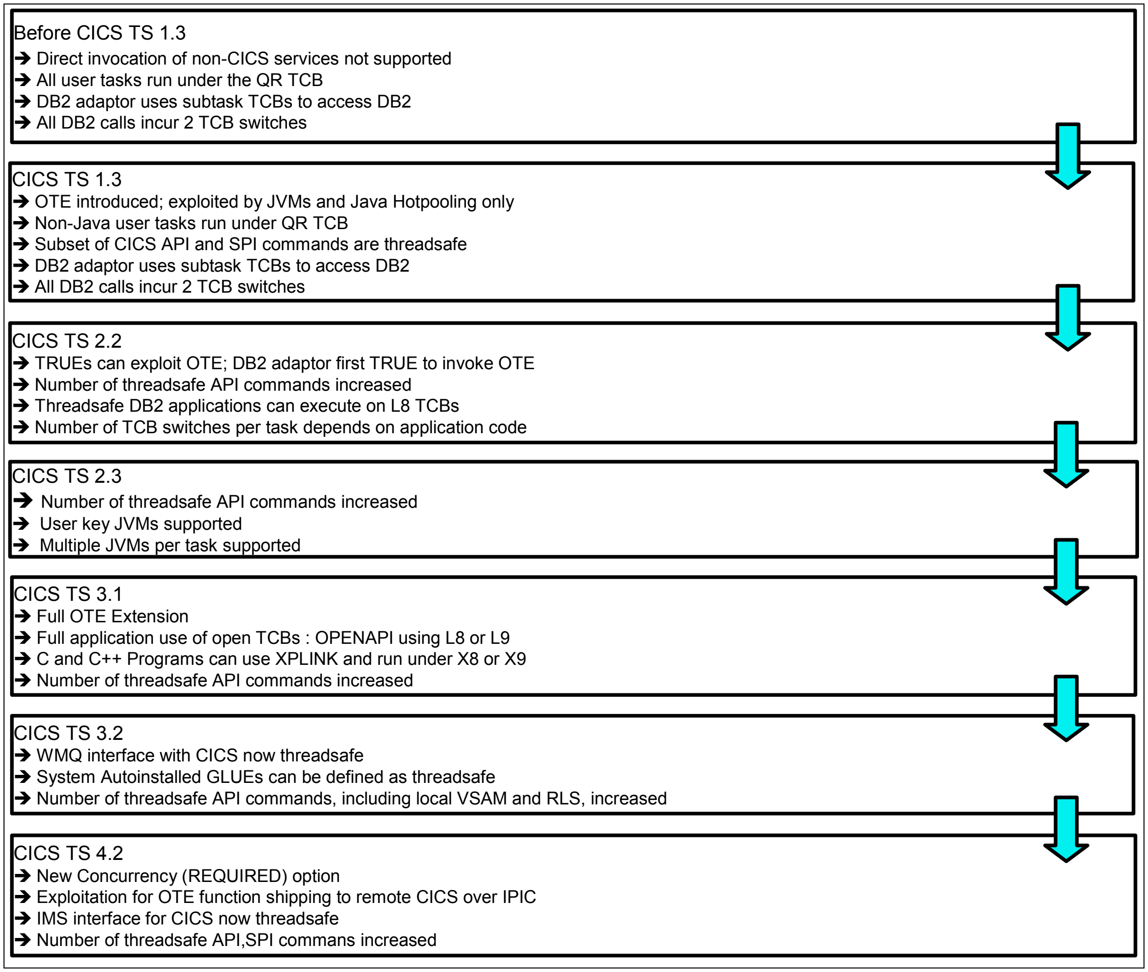


Figure 2-1 OTE enhancements in recent releases of CICS TS

2.2.1 Before CICS Transaction Server V1.3

Before CICS TS for OS/390 V1.3, user applications and user exits operated in a restricted, or closed, environment. Although the applications can use the functionally rich CICS API, direct invocation of other services was not supported, because CICS ran all user transactions under a single z/OS TCB, known as the *CICS quasi-reentrant TCB*. Direct invocation of other services that are outside the scope of the interfaces that are permitted by CICS can interfere with the use by CICS of the QR TCB. In particular, requests resulting in the suspension (*blocking*) of the QR TCB, which happens when an MVS wait is issued, causes all CICS tasks to wait.

CICS DB2 interface before CICS Transaction Server V1.3

The CICS DB2 attachment facility created and managed its own subtask thread TCBs with which to access DB2 resources, ensuring that waits for DB2 resources do not block the QR TCB. CICS used the QR TCB for the CICS DB2 TRUE and for the code of the application program. The subtask thread TCBs were used for requests to DB2. Switching between the subtask TCB and the QR TCB took place for every DB2 request, which continues in CICS TS V1.3, as explained in 2.2.2, “CICS Transaction Server V1.3”.

2.2.2 CICS Transaction Server V1.3

The OTE function was introduced in CICS TS for OS/390 V1.3 for use initially by Java virtual machines (JVMs) and Java hot-pooling applications.

OTE is an environment where CICS application code can use services that are not for CICS (facilities outside the scope of the CICS API) within the CICS address space, without interfering with other transactions. Applications that use OTE run on their own open TCB, rather than on the QR TCB. Unlike the QR TCB, CICS does not perform subdispatching on an open TCB. If the application running on an open TCB starts a service that is not for CICS that blocks the TCB, the TCB blocking does not affect other CICS tasks. For example, some services provided by DB2, MVS, UNIX System Services, or TCP/IP might result in TCB blocking.

CICS DB2 interface under CICS TS V1.3

Although OTE became available in CICS TS V1.3, it was not yet enabled for TRUEs and, therefore, not yet used by the CICS DB2 attachment facility. Similar to previous CICS releases, subtask thread TCBs were used to access DB2 resources to ensure that waits for DB2 resources do not block the QR TCB.

CICS continued to use the QR TCB for the CICS DB2 TRUE and for application program code. Subtask thread TCBs are used for requests to DB2, and switching between the subtask TCB and the QR TCB took place for every DB2 request.

Figure 2-2 shows the TCB switches involved in typical DB2 transactions running under CICS TS V1.3.

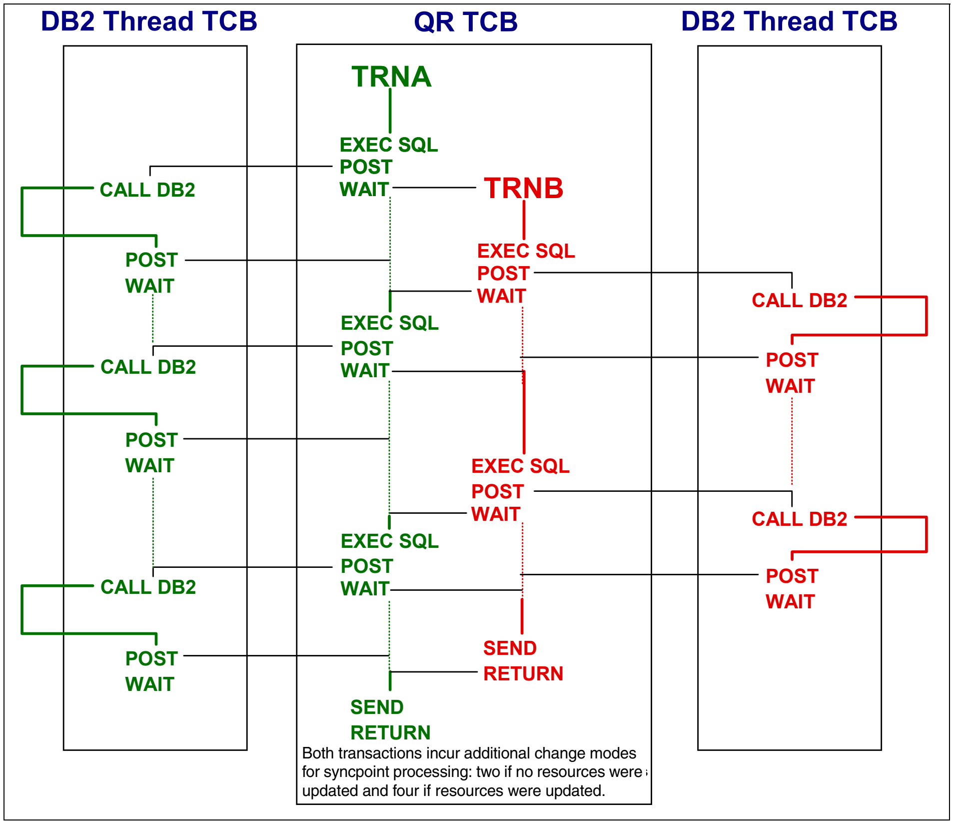


Figure 2-2 DB2 transactions in CICS TS V1.3

2.2.3 CICS Transaction Server V2.2

Enhancements introduced in CICS TS for z/OS V2.2 make it possible for TRUEs to use the OTE. The CICS DB2 adapter supplied with this release was the first TRUE to use OTE.

CICS DB2 TRUE: The CICS DB2 TRUE was converted to use this feature and operate as an open API TRUE when CICS is connected to DB2 6 or later. It uses L8 TCBs for DB2 request processing.

Applications with a TRUE enabled by using the OPENAPI option on the **ENABLE PROGRAM** command can use OTE to provide performance benefits. Such TRUEs are known as *OPENAPI TRUEs*. An OPENAPI TRUE is given control under an open TCB in L8 mode (known as an *L8 TCB*) and can use APIs that are not for CICS without creating, managing, and switching between subtask TCBs.

CICS DB2 interface under CICS TS V2.2

Starting with CICS TS V2.2, the CICS DB2 attachment facility no longer creates subtask thread TCBs to access DB2 resources, unless they are connected to DB2 V5 or earlier. Instead, by using OTE, L8 TCBs are used to process EXEC SQL statements. If an application is *not* defined as threadsafe (the default), each task returns to the QR TCB on completion of the EXEC SQL statement.

Existing or new CICS DB2 applications written in any language that accesses DB2 6 or later can now gain performance benefits that are provided by OTE. These performance benefits can be gained because open TCBs, unlike the QR TCB or subtask thread TCBs, might be used for API requests that are not for CICS (including requests to DB2) and for application code. Because application code can be run on the open TCB, the number of TCB switches is significantly reduced.

With OTE, the same L8 TCB can be used by the CICS DB2 TRUE.

Figure 2-3 on page 20 shows the TCB switches that are involved in typical DB2 transactions running under CICS TS V2.2. Threadsafe and nonthreadsafe tasks are both shown.

2.2.4 CICS Transaction Server V2.3

CICS TS for z/OS V2.3 does not introduce any fundamental changes to the OTE. However, this release makes it easier to maximize the performance improvements that can be achieved by defining appropriate applications as threadsafe.

Issuing nonthreadsafe **EXEC CICS** commands causes a threadsafe program running on an L8 TCB to switch back to the QR TCB. CICS TS V2.3 helps to prevent this switching by increasing the number of threadsafe **EXEC CICS** commands to include ASKTIME and FORMATTIME, among others.

CICS DB2 interface under CICS TS V2.3

The CICS DB2 attachment facility in CICS TS V2.3 operates the same as it does in V2.2. For details, see “CICS DB2 interface under CICS TS V2.2” on page 19. The **EXEC CICS** commands made threadsafe in CICS TS V2.3 make it easier for some applications to reap the full performance benefits that are associated with being defined as threadsafe.

Figure 2-3 shows the TCB switches that are involved in typical DB2 transactions running under CICS TS V2.3. Threadsafe and nonthreadsafe tasks are both shown.

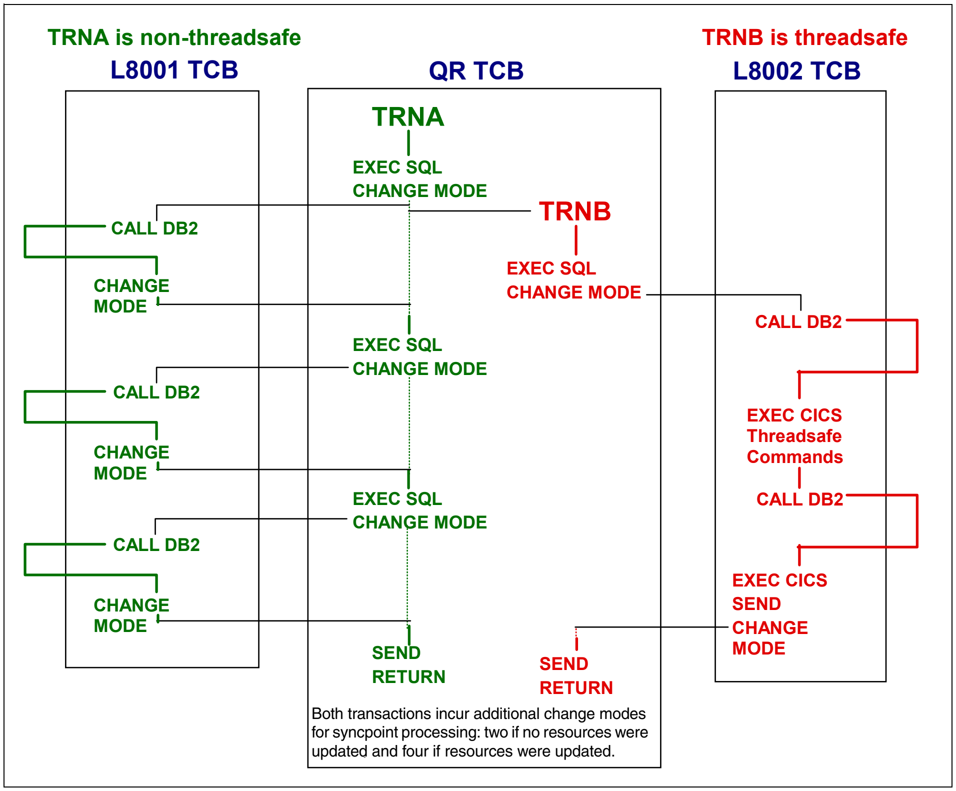


Figure 2-3 DB2 transactions in CICS TS V2.2 and V2.3

2.2.5 CICS Transaction Server V3.1

Starting with CICS TS V3.1, programs can be defined with API(OPENAPI) and run almost independently of the QR TCB. Any program defined this way runs on an L8 or L9 open TCB depending on its EXECKEY value. Any program that can be defined as COncurrency (Threadsafe) can also be defined as API(OPENAPI) and take advantage of the benefits of running on an open TCB regardless of whether it accesses DB2. For this reason, *all* programs must be written to threadsafe standards.

Before CICS TS V3.1, the OPENAPI option was only available to TRUEs.

Figure 2-4 shows the effect of using the new OPENAPI definition.

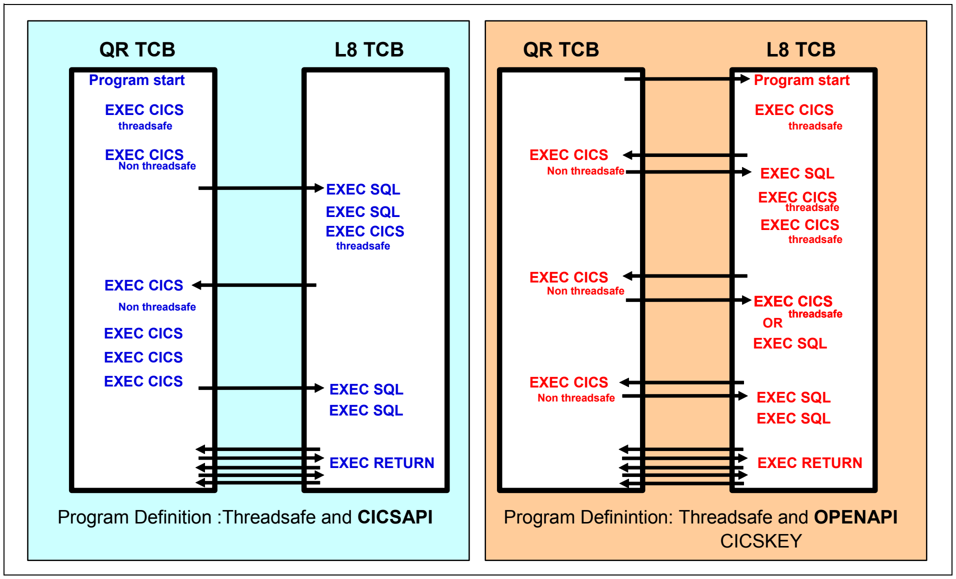


Figure 2-4 OPENAPI programs in CICS TS V3.1

The left side of Figure 2-4 shows a program defined as API(CICSAPI). In this example, the behavior is the same as in CICS TS V2. CICS switches to an L8 TCB when a DB2 command is encountered and remains there until a nonthreadsafe CICS command causes a switch back to the QR TCB. The switch to the L8 TCB in this case is made because the CICS DB2 TRUE is enabled in OPENAPI mode.

The right side of Figure 2-4 shows the behavior when an application program is defined using the API(OPENAPI). Using OPENAPI for this program indicates to CICS that this program *must* run on an open TCB. CICS immediately moves the task to an L8 or L9 TCB at the start of the program. Only if a nonthreadsafe CICS command is encountered does CICS move the task to the QR TCB and then *only* for the duration of the CICS command.

API(OPENAPI): A program defined as API(OPENAPI) is not required to have *any* DB2 or WebSphere MQ commands.

If the CICS program is now defined as API(OPENAPI) and with EXECKey (UserKey), CICS switches to an L9 TCB for execution rather than an L8 TCB. However, CICS switches the task to an L8 TCB for every DB2 command because OPENAPI TRUEs *must* run in a CICS key on an L8 TCB, as shown in Figure 2-5.

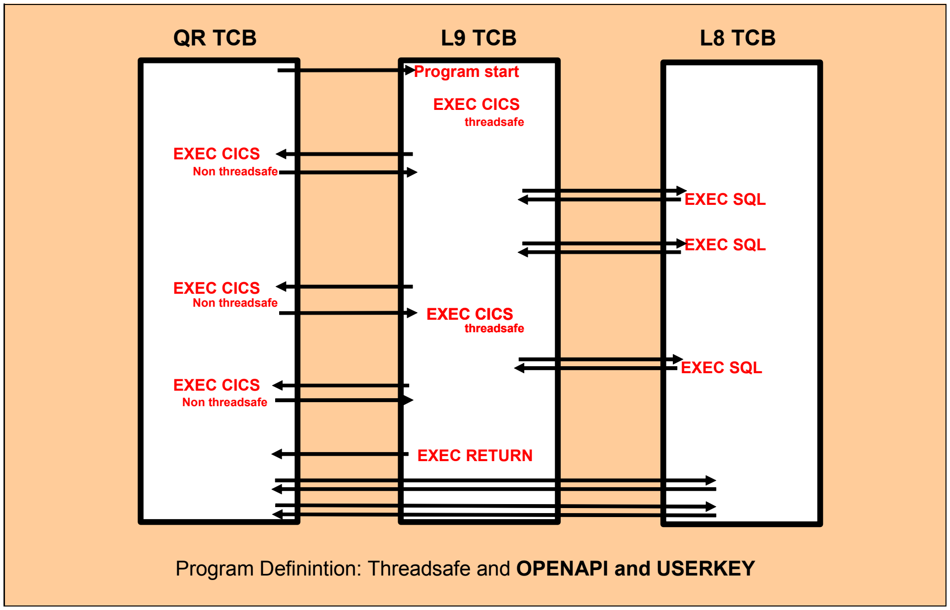


Figure 2-5 Program defined as OPENAPI and EXECKEY(USER)

OPENAPI and CICSAPI candidates

The combination of OPENAPI and EXECKEY attributes in CICS TS V3.1 can lead to extra TCB switching, which is undesirable. Applications must be analyzed correctly before using the OPENAPI attribute, because some rules define what is a good candidate for the OPENAPI attribute (and, by implication, what is a bad candidate). For a summary of good candidates for OPENAPI and CICSAPI, see “OPENAPI good and bad candidates” on page 63.

XPLINK

Another enhancement to OTE in CICS TS V3.1 is that C and C++ programs compiled with the Extra Performance Linkage (XPLINK) option can run under an X8 or X9 open TCB depending on its EXECKey definition. XPLink was introduced in OS/390 V2.10 to provide a high performance subroutine call and return mechanism for C and C++ programs. XPLink is enabled by using the XPLINK compiler option when compiling C and C++ programs.

A C or C++ program compiled with the XPLINK option runs on an X8 or an X9 TCB depending on the EXECKEY attribute of the program definition.

TCB switching still occurs in the following situations:

- When using nonthreadsafe CICS commands
- When making SQL calls
- When linking (LINK) to a different program
- When using the CICS C++ foundation classes (Currently only versions that are not XPLINK are available.)

XPLINK programs can be considered a special case of OPENAPI programs. They run on X8 and X9 because XPLINK uses batch Language Environment rather than CICS Language Environment. Each X8 and X9 TCB has a separate Language Environment enclave. However, the storage is still allocated from CICS storage. You can use DFHAPXPO to change the batch Language Environment runtime options.

> **Considerations:** The same considerations for OPENAPI programs apply to XPLINK programs.

2.2.6 CICS Transaction Server V3.2

CICS TS V3R2 includes the following enhancements:

- Local and record-level sharing (RLS) file control threadsafe commands
- Threadsafe CICS journaling commands
- Threadsafe definition for system autoinstalled GLUEs
- Threadsafe WebSphere MQ commands

CICS file control interface under CICS TS V3.2

Starting in CICS TS 3.2, the commands for accessing local and RLS VSAM files are threadsafe. These changes result in improved performance for threadsafe applications that contain a mixture of DB2 and file control. Also, pure VSAM applications running on an open TCB have a higher throughput due to usage of concurrent CPUs. The number of TCB switches is also reduced.

The following commands are threadsafe:

- `READ`
- `READ UPDATE`
- `REWRITE`
- `DELETE`
- `UNLOCK`
- `STARTBR`
- `RESETBR`
- `READNEXT`

- **READPREV**
- **ENDBR**

In addition, the SPI **INQUIRE FILE** command is threadsafe.

> **Nonthreadsafe commands:** The commands for accessing files using other methods (remote files, shared data tables, coupling facility data tables, and basic direct-access method (BDAM) files) remain nonthreadsafe.

Threadsafe CICS journaling commands under CICS TS Version 3.2

The following journaling commands are threadsafe:

- **WRITE JOURNALNAME**
- **WRITE JOURNALNUM**
- **WAIT JOURNALNAME**
- **WAIT JOURNALNUM**

In addition, the XPI **WRITE_JOURNAL_DATA** command is threadsafe.

Threadsafe definition for system autoinstalled GLUEs

With CICS TS V3.2, system autoinstalled GLUE programs can be defined as threadsafe. GLUE programs that are required early during CICS initialization are required to be configured to CICS by using the **ENABLE** command. The **ENABLE** command can be specified with an override of THREADSAFE.

WebSphere MQ interface under CICS TS V3.2

The following components to connect CICS TS V3.2 and WebSphere MQ are integrated into CICS so that the components can become threadsafe:

- CICS MQ adapter
- CICS MQ trigger monitor
- CICS MQ bridge

2.2.7 CICS Transaction Server V4.2

CICS TS V4.2 offers further enhancements to OTE:

- The introduction of a new concurrency option on the program definition that allows for greater exploitation of OTE for threadsafe applications
- Usage of OTE for function shipping, by allowing the mirror program, when it is started in a remote CICS region with IP interconnectivity (IPIC), to run on an open TCB

- Making more of the API and SPI threadsafe, including access to IMS databases by using the CICS IMS Database Control (DBCTL) interface.

CONCURRENCY(REQUIRED) option in CICS TS V4.2

Before CICS TS V4.2, an application program was defined as **CONCURRENCY(QUASIRENT)** or **CONCURRENCY(THREADSAFE)**.

- A **CONCURRENCY(QUASIRENT)** program always runs on the QR TCB.
- A **CONCURRENCY(THREADSAFE)** program is a program that is coded to threadsafe standards and contains threadsafe logic. It can run on the QR TCB or an open TCB. It starts running on the QR TCB. If processing a DB2 request, for example, causes a switch to an open TCB, then on return to the program, the program continues on the open TCB.

CICS TS V4.2 provides a new **CONCURRENCY(REQUIRED)** setting. As with **CONCURRENCY(THREADSAFE)**, the new setting specifies that the program is coded to threadsafe standards and contains threadsafe logic, but the program must also run on an open TCB. Therefore, the program runs on an open TCB from the start. If CICS must switch to the QR TCB to process a nonthreadsafe CICS command, CICS returns to the open TCB when it returns control to the application program.

With the **CONCURRENCY(REQUIRED)** option, the user defines the program to start on an open TCB independently of defining what APIs it uses:

- If the program uses only APIs that are for CICS (including access to external resource managers such as DB2, IMS, and WebSphere MQ), it must be defined with program attribute API(CICSAPI). In this case, CICS always uses an L8 open TCB, regardless of the execution key of the program, because CICS commands do not rely on the key of the TCB.
- If the program uses other APIs that are not for CICS, it must be defined with program attribute API(OPENAPI). In this case, CICS uses an L9 TCB or an L8 TCB depending on the execution key of the program so that the APIs that are not for CICS can operate correctly. This OPENAPI behavior is the same as in previous releases.

However, not all applications are suitable. For example, a threadsafe application that issues many EXEC SQL requests and then issues many **EXEC CICS** commands that are not threadsafe is best left as **CONCURRENCY(THREADSAFE)**. Defining the application as **CONCURRENCY(REQUIRED)** means two TCB switches for each nonthreadsafe CICS command, because control always returns to the application on the open TCB as shown in Figure 2-6 on page 26.

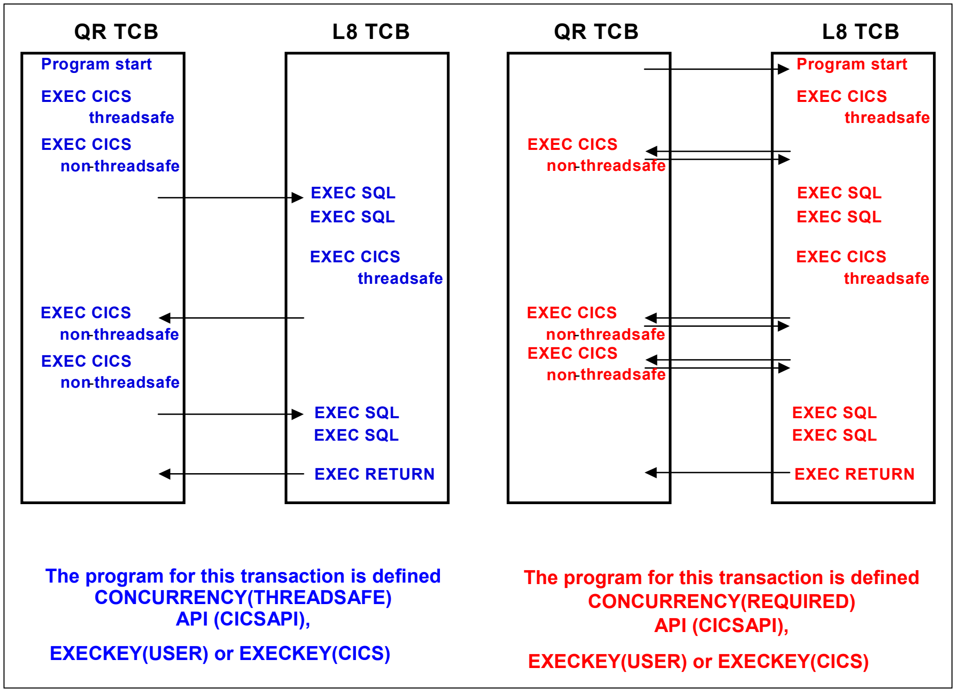


Figure 2-6 Comparison of CONCURRENCY(THREADSAFE) and CONCURRENCY(REQUIRED)

This situation demonstrates the importance of knowing what the application does. For example, such tools as CICS Interdependency Analyzer for z/OS (CICS IA) help you to discover application execution paths. In particular, its command flow feature shows the order in which CICS, IMS, WebSphere MQ, and DB2 commands run, and what TCB each command runs on. Other tools, such as CICS Performance Analyzer for z/OS (CICS PA), analyze CICS performance System Management Facilities (SMF) data. They show, for example, how much of the CPU is consumed on which TCBs and how many TCB switches occurred. Such tools as CICS IA and CICS PA are invaluable aids to have in your toolbox when embarking upon a threadsafe project.

Function shipping enhancements in CICS TS V4.2

The mirror program DFHMIRS, which is used by all mirror transactions and is supplied by CICS, is now defined as threadsafe. In addition, the IPIC transformers are now threadsafe. For IPIC connections only, CICS runs the mirror program on an L8 open TCB whenever possible. For threadsafe applications that function ship commands to other CICS regions using IPIC, the resulting reduction in TCB switching improves the performance of the application compared to other intercommunication methods. To gain the performance improvement for remote files, you must specify the system initialization parameter **FCQRONLY=NO** in the file-owning region (FOR).

For remote file control or temporary storage requests shipped over IPIC connections, CICS no longer forces a switch to the QR TCB in the application-owning region (AOR) if it is running currently on an open TCB. The requests are shipped running on the open TCB.

In the FOR or queue-owning region (QOR), the mirror decides when to switch to an open TCB. It does this task for the first File Control or temporary storage request received over an IPIC connection. The idea is for long-running mirrors to keep the mirror running on an open TCB.

A new option MIRRORLIFE is added to the IPCONN attributes for function-shipped file control and temporary storage requests using an IPIC connection. MIRRORLIFE improves efficiency and provides performance benefits by specifying the lifetime of mirror tasks and the amount of time a session is held.

API and SPI enhancements in CICS TS V4.2

CICS commands that are threadsafe in CICS TS V4.2 include the following named counter server commands:

- **QUERY SECURITY**
- **SIGNON**
- **SIGNOFF**
- **VERIFY PASSWORD**
- **CHANGE PASSWORD**
- **EXTRACT TCPIP**
- **EXTRACT CERTIFICATE**

In addition, several new SPI commands are available, of which the most significant command is the **SYNCPOINT** command.

The CICS Recovery Manager domain now processes a **SYNCPOINT** command on an open TCB wherever possible to minimize TCB switching. Syncpoint processing can take place on an open TCB for all resource types declared as threadsafe that were accessed in the unit of work (UOW). If resource types not declared as threadsafe are accessed in the UOW, the Recovery Manager switches to the QR TCB for those resource types. Before CICS TS V4.2, CICS switched to the QR TCB before the end of task sync point. In CICS TS V4.2, the application remains on an open TCB, if it is running on one, until the end of task sync point is called. Afterward, CICS switches to QR for the task detach logic.

Before CICS TS V4.2, a threadsafe application running on an open TCB that had, for example, updated DB2 and WebSphere MQ and then issued a sync point required nine TCB switches:

- A switch to QR was made at the start of the sync point.
- Switches to L8 and back to QR occurred when calling DB2 for PREPARE.
- Switches to L8 and back to QR occurred when calling WebSphere MQ for PREPARE.
- Switches to L8 and back to QR occurred when calling DB2 for COMMIT.
- Switches to L8 and back to QR occurred when calling WebSphere MQ for COMMIT.

In CICS TS V4.2, for a terminal-driven transaction, one TCB switch to QR occurs. For a non-terminal-driven transaction (and assuming no other nonthreadsafe resources were touched), no TCB switches occur.

For more information, see the following tables:

- Table 2-4 on page 48 shows threadsafe API commands for CICS TS V4.
- Table 2-5 on page 49 shows the existing threadsafe API commands in CICS TS V4.2.
- Table 2-7 on page 50 shows the threadsafe SPI commands in CICS TS V4.1.
- Table 2-8 on page 50 shows the threadsafe SPI commands in CICS TS V4.2.
- Table 2-9 on page 51 shows the nonthreadsafe SPI commands in CICS TS V4.1.

THREADSAFE CICS DBCTL interface in CICS TS V4.2

CICS provides the CICS DBCTL interface to support `CALL DLI` and `EXEC DLI` command requests that are issued by applications running in a CICS region. In CICS TS V4.2, the CICS DBCTL interface is enhanced to use OTE, and CICS can run the CICS DBCTL TRUE on an L8 open TCB.

OTE is supported from IMS 12, with PTFs for APAR PM31420, PM47327, and PM45414 applied. IMS indicates to CICS during the connection process that the OTE is supported, and therefore, CICS defines the CICS DBCTL TRUE as an OPENAPI TRUE. For IMS 11 and earlier, OTE is not supported. CICS runs the CICS DBCTL TRUE on the QR TCB, and the IMS code switches to an IMS thread TCB.

Running an application on an open TCB improves throughput and performance by reducing the use of the QR TCB. Threadsafe CICS applications that run on an L8 open TCB and issue `CALL DLI` or `EXEC DLI` commands can avoid two TCB switches for each call to IMS:

- For a nonthreadsafe application, the amount of switching is not reduced. Instead of switching from the QR TCB to an IMS thread TCB and back again for each IMS request, the application switches from QR to L8 and back again.
- For a threadsafe application, if it is running on the QR TCB, it switches to L8 and then stays on L8 when control is returned to the application.
- For a threadsafe application that is already running on an L8 TCB, or for a `CONCURRENCY(REQUIRED)` application that is running on an L8 TCB, no TCB switching occurs for the IMS request.

2.2.8 Open TCB modes in CICS Transaction Server V2

The following open TCB modes are available starting in CICS TS V2:

J8 CICS key JVM requirements
J9 USER key JVM requirements (only in CICS TS V2.3)
L8 OPENAPI TRUEs (TRUEs must run in the CICS key.)
H8 High performance Java programs

2.2.9 Open TCB modes in CICS Transaction Server V3 and later

CICS TS V3.1 extends the number of TCB modes that are available to CICS. The following open TCB modes are now available for application use:

J8 CICS key JVM requirements
J9 User key JVM requirements
L8 OPENAPI TRUEs (TRUEs must run in the CICS key.)
CICS key OPENAPI applications
`CONCURRENCY(THREADSAFE)`
`CONCURRENCY(REQUIRED)` for CICS TS V4.2
L9 User key OPENAPI applications
X8 CICS key C and C++ applications compiled with XPLINK
X9 User key C and C++ applications compiled with XPLINK

In addition, the S8 TCB mode is used internally by CICS for Secure Sockets Layer (SSL).

2.3 Techniques to ensure threadsafe processing

You can use many techniques to ensure threadsafe processing when accessing a shared resource. The following techniques are a subset of the possible options:

- Enqueue on the resource to obtain exclusive control and ensure that no other program can access the resource:
  - An `EXEC CICS ENQ` command within an application program
  - An XPI ENQUEUE function call within a GLUE program
- Perform access to shared resources only in a program defined as *quasirent*.

  A linked-to program defined as quasi-reentrant runs under the QR TCB and can take advantage of the serialization provided by CICS quasi-reentrancy. Even in quasi-reentrant mode, serialization is provided only if the program retains control and does not wait. Do not use this technique.
- Place all transactions that access the shared resource into a restricted transaction class (TRANCLASS) defined with the number of active tasks specified as MAXACTIVE(1).

  This approach effectively provides a coarse locking mechanism, but might have an impact on performance.

Attention: Although the term *threadsafe* is defined in the context of individual programs, a user application as a whole can be considered threadsafe *only* if all the application programs that access shared resources obey the rules. A program written to threadsafe standards cannot safely update shared resources if another program accessing the same resources does not obey the threadsafe rules.

For more information, see Chapter 3, “Threadsafe techniques” on page 53.

2.4 Program definition

The program definition has two attributes:

- CONCURRENCY attribute
- API attribute

CONCURRENCY attribute

The CONCURRENCY attribute of the program definition defines a program as QUASIRENT, THREADSAFE, or REQUIRED (for CICS TS V4.2). QUASIRENT is the default value.

The CONCURRENCY attribute applies to the following programs:

- User-application programs
- Program list table (PLT) programs
- User-replaceable programs
- GLUE programs
- TRUE programs

API attribute

The API attribute, which applies to CICS TS V3.1 and later, specifies whether the program is to be defined as CICSAPI or OPENAPI.

The API attribute applies to the following programs:

- User-application programs
- PLT programs
- User-replaceable programs
- GLUE programs (CICS always forces CICSAPI.)
- TRUE programs

A program defined as API(CICSAPI) begins on the QR TCB. The resulting behavior is the same as in versions before CICS TS V3.1. A new **`CONCURRENCY(REQUIRED)`** parameter was introduced in CICS TS V4.2. When the **`CONCURRENCY(REQUIRED)`** parameter is specified, the program receives control on an open TCB. It continues running on the open TCB until a switch to the QR TCB is forced (for example, an EXEC CICS call to a nonthreadsafe command). After the work is completed on the QR TCB, the program then continues to run on the open TCB.

A program that is defined as API(OPENAPI) begins its execution on an L8 or an L9 TCB depending on the value of its EXECKEY attribute. It switches to the QR TCB for nonthreadsafe CICS commands and to the L8 TCB (if it started on L9) to run SQL commands. Defining a program as API(OPENAPI) automatically implies that the program is also threadsafe.

The benefit of using the OPENAPI attribute at CICS TS V3.1 is that you can move more applications off the QR TCB. Applications that are not DB2 supported and highly CPU intensive applications can benefit from running on an open TCB.

When a program is defined with the **`CONCURRENCY(REQUIRED)`** parameter in CICS TS V4.2, it receives control on the L8 TCB. Now you use OPENAPI only if you are going to use APIs that are not from CICS and not as a back door to have the program start on an open TCB.

Figure 2-7 shows an example program definition as viewed by the CEDA transaction.

```
OBJECT CHARACTERISTICS                                        CICS RELEASE = 0640
 CEDA  View PROGram( DB2PROG5 )
  PROGram        : DB2PROG5
  Group          : THDSAFE
  DEscription    :
  Language       :                   CObol | Assembler | Le370 | C | Pli
  RELoad         : No                No | Yes
  RESident       : No                No | Yes
  USAge          : Normal            Normal | Transient
  USElpacopy     : No                No | Yes
  Status         : Enabled           Enabled | Disabled
  RSl            : 00                0-24 | Public
  CEdf           : Yes               Yes | No
  DAtalocation   : Any               Below | Any
  EXECKey        : User              User | Cics
  COncurrency    : Quasirent          Quasirent | Threadsafe | Required
  Api            : Cicsapi           Cicsapi | Openapi
 REMOTE ATTRIBUTES
  DYnamic        : No                No | Yes

                                                          SYSID=PJA6
APPLID=SCSCPJA6
```

Figure 2-7 Program definition

The CONCURRENCY and API attribute can both be specified by using a program autoinstall exit. The sample program autoinstall exit that is supplied by IBM defaults to QUASIRENT and CICSAPI.

> **Important:** The program definition keywords **`CONCURRENCY(THREADSAFE)`** and **`CONCURRENCY(REQUIRED)`** for CICS TS V4.2 indicate to CICS that the application logic is threadsafe. The keywords do not indicate whether CICS commands are threadsafe. CICS ensures threadsafety of its own logic because CICS logic can either run on an open TCB or it cannot. Therefore, it is switched to the QR TCB before it runs. In either case, the resource is accessed in a threadsafe way.

A threadsafe application can use nonthreadsafe CICS commands. It suffers the overhead of TCB switching, but resource integrity is maintained.

If an application that contains nonthreadsafe logic is incorrectly defined to CICS as **`CONCURRENCY(THREADSAFE)`** or **`CONCURRENCY(REQUIRED)`** for CICS TS V4.2, the results are unpredictable.

2.5 Task-related user exit APIs

TRUEs can be enabled with or without the OPENAPI option. Without the OPENAPI option, the TRUE is enabled as CICSAPI.

For CICSAPI, the TRUE is enabled as QUASIRENT, THREADSAFE, or REQUIRED (for CICS TS V4.2) without the OPENAPI option. The TRUE is restricted to the programming interfaces that are permitted by CICS.

For OPENAPI, the TRUE is also enabled as THREADSAFE or REQUIRED (for CICS TS V4.2) when the OPENAPI option is specified. The program is assumed to be written to threadsafe standards (serially reusable) and is permitted to use APIs that are not for CICS.

- When **`CONCURRENCY(THREADSAFE)`** CICS give control to the TRUE under an L8 or L9 mode open TCB depending on the EXECKEY attribute
  - For a CICS access key, the program is given control on the L8 mode open TCB.
  - For a User access key, the program is given control to the L9 mode open TCB.
- When **`CONCURRENCY(REQUIRED)`** CICS gives control to the TRUE under an L8 mode open TCB

For more information about the OPENAPI option, see the CICS Transaction Server for z/OS V4.2 Information Center at the following address, and search for *CICS Customization Guide*:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

2.5.1 CICS DB2 task-related user exit

The CICS DB2 adapter supplied by CICS is the first one to include a TRUE that can be enabled with the OPENAPI attribute. This adapter was first supplied in CICS TS V2.2 and enabled DB2 calls to be run on an open TCB. As mentioned previously, with this adapter, you can create applications that can remain on the open TCB following a DB2 call depending on the CONCURRENCY attribute of the program definition.

2.5.2 CICS IMS task-related user exit

For CICS TS V4.2, the definitions for the CICS DBCTL adapter TRUE DFHDBAT and the EXEC DLI TRUE DFHEDP changed to specify **`CONCURRENCY(THREADSAFE)`**. However DFHDBAT uses only the OTE when the release level of IMS supports the OTE. Therefore, the definition is overridden at run time by the ENABLE options.

If IMS supports the OTE, when CICS connects to DBCTL, DFHDBAT is enabled as OPENAPI and uses an open TCB. If IMS does not support the OTE, DFHDBAT is enabled as QUASIRENT. That is, DFHDBAT is started on the QR TCB, and the IMS database resource adapter (DRA) switches to use an IMS thread TCB when processing an IMS request.

2.5.3 CICS WebSphere MQ task-related user exit

The components that are threadsafe in CICS TS V3.2 are the CICS MQ adapter, the CICS MQ trigger monitor, and the CICS MQ bridge. Using the OTE benefits threadsafe applications that use WebSphere MQ. You can avoid TCB switching, resulting in CPU savings and an increase in throughput because WebSphere MQ applications can now run multiple open TCBs.

2.5.4 IP sockets task-related user exit

With the IP CICS Sockets component of the Communications Server at z/OS V1.7, the calls to the IP CICS Sockets TRUEs can now run by using the CICS OTE. In the same way that a DB2 call runs on an L8 TCB, an IP socket call can now run on an L8 TCB.

However, for IP CICS Sockets API calls to use OTE, the IP CICS Socket configuration file must be updated to turn on this facility. Unlike DB2, the TRUE for IP CICS Sockets can be enabled as OPENAPI or CICSAPI. The default action is for IP sockets to continue managing its own subtask TCBs. That is, they must be enabled as CICSAPI.

For information about the installation and configuration of IP CICS Sockets, see *z/OS Communications Server IP CICS Sockets Guide Version 1 Release 7* SC31-8807. The following sections summarize where you must define the OTE-related parameters.

Building the configuration file using the EZACICD macro

The IP CICS Sockets configuration file is initially built from a macro called EZACICD. After it is created, you can incorporate the file into CICS by using RDO and modify it by using the supplied configuration transactions.

The macro creates configuration records for each CICS region that uses IP sockets and a configuration record for every listener within each CICS region.

The definition of the CICS region is where OTE for IP sockets is enabled, as shown in Example 2-1.

Example 2-1 CICS region definition in an EZACICD macro

```
EZACICD TYPE=CICS,       CICS record definition              X
      APPLID=CICSPRDB,  APPLID of CICS region using OTE      X
      TCPADDR=TCPIP,    Job/Step name for TCP/IP             X
      CACHMIN=15,       Minimum refresh time for cache       X
      CACHMAX=30,       Maximum refresh time for cache       X
      CACHRES=10,       Maximum number of resident resolvers X
      ERRORTD=CSMT,     Transient data queue for error msgs  X
      TCBLIM=12,        Open API TCB Limit                   X
      OTE=YES,          Use Open Transaction Environment     X
      TRACE=NO,         Trace CICS Sockets                   X
      SMSGSUP=NO        STARTED Messages Suppressed?
```

The two parameters that are related to OTE are **OTE=YES** and **TCBLIM=12**.

OTE

When **OTE=YES** is specified, the IP CICS Sockets interface enables TRUE for OPENAPI. Therefore CICS switches all EZASOKET calls and all IP CICS C socket functions from the QR TCB to an L8 TCB.

TCBLIM

The TCBLIM parameter defines that maximum number of OTE TCBs that the IP CICS Socket TRUE can use. It is a subset of the number of TCBs that are allocated to the pool of TCBs defined by the **MAXOPENTCBS** system initialization table (SIT) parameter in CICS. It is the same pool of TCBs used by the DB2 TRUE if DB2 is also in use.

After the socket call is complete, CICS leaves the task on the L8 TCB or returns to the QR TCB depending on the CONCURRENCY attribute of the application program definition. If the program is defined as **CONCURRENCY(THREADSAFE)**, the program remains on the L8 until task end or a nonthreadsafe CICS API command is encountered. If the program is defined as **CONCURRENCY(QUASIRENT)**, the task is moved back to the QR TCB on completion of the IP socket call, which is the same behavior as for the DB2 TRUEs. Additionally, starting in CICS TS V3.1, the application program might be defined as **API(OPENAPI)** if appropriate, which enables the program to begin execution on an open TCB. For more information, see “OPENAPI good and bad candidates” on page 63.

OTE=YES: If you intend to use **OTE=YES** for IP sockets programs *and* to define the IP sockets application program as threadsafe, you *must* ensure that the programs *are* threadsafe before defining them as such.

Customizing the configuration file

After the configuration file is created and defined to the CICS region, you can modify it by using the supplied configuration transaction EZAC. For example, by using EZAC, you can turn on or off OTE and modify the TCBLIM attribute.

For more information about this transaction, see *z/OS Communications ServerIP CICS Sockets Guide Version 1 Release 7,* SC31-8807.

2.6 TCB limits

As explained previously, CICS manages several TCB pools. For example, CICS TS V3.1 has pools for JVM TCBs (J8 and J9), OPENAPI and TRUE TCBs (L8 and L9), SSL TCBs (S8), and XPLINK TCBs (X8 and X9).

CICS imposes a limit for each of these TCB pools by using a SIT parameter for each one:

MAXOPENTCBS Limits the number of TCBs in the pool of L8 and L9 mode open TCBs.

MAXSSLTCBS Limits the number of TCBs in the pool of S8 mode open TCBs.

MAXXPTCBS Limits the number of TCBs in the pool of X8 and X9 mode open TCBs.

MAXJVMTCBS Limits the number of TCBs in the pool of J8 and J9 mode open TCBs.

2.6.1 MAXOPENTCBS

The pool of L8 and L9 mode TCBs is managed by the CICS dispatcher. The maximum number of TCBs that are allocated to the pool is defined by the `MAXOPENTCBS` SIT parameter. Any combination of L8 and L9 TCBs can be in use (allocated to running tasks) and free.

`MAXOPENTCBS` has a default value of 12 in CICS TS V3.1. You must understand which functions now use the pool of TCBs defined by `MAXOPENTCBS` so that you can assign a sensible value to `MAXOPENTCBS`. In addition to application programs defined with the OPENAPI attribute or programs calling TRUEs enabled with OPENAPI, CICS itself performs tasks on an open TCB taken from the pool of `MAXOPENTCBS`.

Using L8 and L9 TCBs can be summarized as follows:

- L9 mode TCBs are used for user key OPENAPI application programs.
- L8 mode TCBs are used in the following ways:
  - For CICS key OPENAPI application programs
  - For OPENAPI TRUEs (TRUEs always run in CICS key):
    - The CICS DB2 attachment facility
    - The IP CICS Sockets interface
    - The CICS MQ adapter
  - For CICSAPI and **CONCURRENCY(REQUIRED)** for CICS TS V4.2:
    - The CICS DB2 attachment facility
    - The IP CICS Sockets interface
    - The CICS MQ adapter
    - CICS IMS through the CICS DBCTL interface
  - By CICS itself, because CICS uses OPENAPI CICS key programs that run on L8 TCBs:
    - When accessing document templates and HTTP static responses that are stored on the hierarchical file system (HFS)
    - When processing web service requests and parsing XML

Choosing a value for **MAXOPENTCBS**, therefore, must take into account all of these factors depending on which are being used.

Task-related user exit imposed limits

In CICS TS V3.2, the following TRUEs can be enabled in CICS by using the OPENAPI attribute:

- TRUEs supplied by the CICS DB2 attachment facility
- TRUEs supplied by the CICS WebSphere MQ attachment facility
- TRUEs supplied by the IP CICS Sockets interface.

In CICS TS V4.2, an additional TRUE can be enabled in CICS using the OPENAPI attribute, TRUEs supplied by the CICS DBCTL for CICS IMS.

Some of these TRUEs have their own parameter that can be set to limit the number of TCBs that can be used by that TRUE. The TCB limit for each of these TRUEs is part of the TCBs allocated to the pool defined by **MAXOPENTCBS**.

DB2

The DB2 parameter is TCBLIMIT, which is specified in the DB2CONN definition. TCBLIMIT defines the maximum number of TCBs that can be associated with the CICS DB2 attachment.

IMS

The CICS DBCTL interface for CICS IMS was introduced in CICS TS V4.2. The `MAXTHRD` parameter specified in the DRA startup table DFSPZPxx defines the maximum number of IMS threads from a CICS.

WMQ

No parameter limits the number of open TCBs used by WebSphere MQ. Therefore, the limit for WebSphere MQ is the same as the `MAXOPENTCBS` parameter.

IP CICS Sockets

The TCBLIM parameter limits the number of open TCBs that can be associated with the IP CICS Sockets TRUE. This parameter is used when the IP CICS Sockets interface is configured with `OTE=YES`.

Transaction isolation

When transaction isolation (TRANISO) is used, `MAXOPENTCBS` must be set equal to or more than the max task value. When a task defined as using TRANISO is initiated and has accessed DB2 in previous executions of the transaction, CICS assigns an L8 TCB with the correct subspace. This setting eliminates TCB stealing on the first DB2 access.

Nontransaction isolation

If you are not using transaction isolation, you can calculate `MAXOPENTCBS` by using the following steps:

1. Find the value specified for TCBLIMIT in your DB2CONN definition. This value represents the number of L8 TCBs required for your DB2 workload.
2. Add a value for the expected peak number of concurrent CICS tasks accessing WebSphere MQ.
3. Add a value for the expected peak number of tasks using web services, XML, or DOCTEMPLATEs on z/OS UNIX.
4. Add a value for the expected peak number of tasks running as OPENAPI applications that are not for DB2.

Considerations for allocating L8 and L9 mode TCBs

Keep in mind the following considerations for allocating an L8 or L9 mode TCB:

- If the transaction already has an L8 or L9 mode TCB allocated, it is used. At most, only one L8 and L9 TCB is allocated to a task.
- If a free L8 or L9 mode TCB exists for the correct subspace, it is allocated and used.

  > **TRANISO:** If TRANISO is not in use, all tasks use the same space.

- If the number of open TCBs is below the `MAXOPENTCBS` limit, a new L8 or L9 mode TCB is created and associated with the subspace of the task.
- If the number of open TCBs is at the `MAXOPENTCBS` limit and there are free L8 or L9 mode TCBs with the wrong subspace, the dispatcher deletes the free TCB and creates a TCB for the required subspace. This approach avoids suspending the task until the number of TCBs is reduced below the pool limit. This action is reflected in the count of *TCB steals* in the CICS dispatcher TCB mode statistics.
- If the number of open TCBs is at the `MAXOPENTCBS` limit and no TCBs are available to steal, the task is suspended with an OPENPOOL wait, until one becomes free or the `MAXOPENTCBS` limit is increased.

> **Important:** CICS TS V2.2 APAR PQ75405 changes the allocation algorithm and must be installed. This code is included in the base level of subsequent CICS releases.

2.7 Open TCB performance

Currently, the following IBM software uses an OTE within CICS:

- The CICS DB2 attachment facility
- CICS IMS through the CICS DBCTL interface (for CICS TS V4.2)
- The CICS MQ adapter
- The IP CICS Sockets interface

2.7.1 DB2

The CICS DB2 attachment facility includes a CICS DB2 TRUE, DFHD2EX1, which is written to threadsafe standards and enabled as an open API TRUE program. The TRUE is automatically enabled with the OPENAPI option on the `ENABLE PROGRAM` command during startup of the CICS DB2 attachment facility. With this method, the TRUE can receive control on an open L8 mode TCB. DB2 calls are made on this same L8 TCB. Therefore, it also acts as the thread TCB, resulting in better performance, because you do not need to switch to a subtask TCB.

2.7.2 IMS

The CICS DBCTL interface for CICS IMS that was introduced in CICS TS V4.2 provides an interface for IMS to satisfy DL/I requests that are issued by applications running in a CICS region. The CICS DBCTL interface is defined as threadsafe, and CICS can run the CICS DBCTL TRUE on an L8 open TCB.

The OTE is supported from IMS 12, with PTFs for APAR PM31420, PM47327, and PM45414 applied. During the connection process, IMS indicates to CICS that the OTE is supported. Therefore, CICS defines the CICS DBCTL TRUE as an open API TRUE.

An open API TRUE is run on an L8 open TCB, which is dedicated for use by the calling CICS task. Running an application on an open TCB improves throughput and performance by reducing the use of the QR TCB. Threadsafe CICS applications that run on an L8 open TCB and use threadsafe CICS DBCTL commands now avoid up to four TCB switches for each call to IMS.

2.7.3 WebSphere MQ

Starting in CICS TS V3.2 and WebSphere MQ for z/OS, the CICS WebSphere MQ attachment facility includes a TRUE, DFHMQTRU, which is written to threadsafe standards and is enabled as an open API TRUE program. The TRUE is automatically enabled with the OPENAPI option on the ENABLE PROGRAM during startup of the CICS MQ adapter. This way, the TRUE can receive control on an open L8 mode TCB.

2.7.4 IP CICS Sockets interface

The IP CICS Sockets interface includes a TRUE, EZACIC01, which is written to threadsafe standards and can be enabled as an open API TRUE program. It is enabled as OPENAPI only if the OTE parameter in the IP CICS Sockets configuration file for that CICS region is set to YES.

2.7.5 Performance considerations

To gain the best possible performance within an OTE environment, keep in mind the following considerations:

- Ensure that all applications and exits within the TRUE path are written to threadsafe standards and are defined to CICS as **CONCURRENCY(THREADSAFE)** or **CONCURRENCY(REQUIRED)** for CICS TS V4.2. Consider the common exits: XPCFTCH, XEIIN, XEIOUT, XRMIIN, XRMIOUT, and Dynamic Plan.

  For DB2, the default sample Dynamic Plan exit, DSNCUEXT, is not defined to CICS as threadsafe:

  – CICS TS V2.3 and V3.1 ship an alternative sample Dynamic Plan exit, DFHD2PXT, which is defined to CICS as threadsafe.

  – For CICS TS V2.2, APAR PQ67351 supplies the alternative sample Dynamic Plan exit, DFHD2PXT.

- Minimize or eliminate the use of nonthreadsafe CICS commands. For more information, see the following sections:

  – "Threadsafe API commands" on page 45
  – "Threadsafe SPI commands" on page 49
  – "Threadsafe XPI commands" on page 50

  If you are unable to eliminate all nonthreadsafe commands, if possible, consider rearranging the commands within your application so that they are not interspersed with SQL calls, IMS calls (for CICS Transaction Server V4.2) or IP CICS Sockets calls.

- When using transaction isolation (TRANISO), set the **MAXOPENTCBS** parameter equal to or greater than max task (MXT) coded within the CICS SIT.

Mode switching, in regard to OTE, is the act of switching from the QR TCB to an open TCB or vice versa:

- For nonthreadsafe exits, a switch occurs from the open TCB to the QR TCB and returns to the open TCB when the exit program completes.
- For nonthreadsafe commands issued from a program defined as **CONCURRENCY(THREADSAFE)**, a switch occurs from the L8 TCB to the QR TCB. It remains there until the next SQL, IMS (for CICS TS V4.2), or WMQ call, which causes a switchback from the QR TCB to the L8 TCB.
- For nonthreadsafe commands issued from a program defined as **CONCURRENCY(REQUIRED)** in CICS TS V4.2, a switch occurs from the L8 TCB to the QR TCB. It reverts to the L8 TCB after the nonthreadsafe command is completed.
- For nonthreadsafe commands issued from an OPENAPI program, a switch occurs from the open TCB to the QR TCB for the duration of the **EXEC CICS**

command. Upon return to the application, a switch occurs from the QR TCB back to the open TCB.

2.8 TCB considerations with UNIX System Services

When defining the numbers of TCBs that are allowed in a CICS region, you must also consider the settings in UNIX System Services that control the number of processes that can run within a CICS region. In UNIX System Services, the **MAXPROCUSER** parameter specifies the maximum number of processes one UNIX user identifier (UID) can have concurrently active, regardless of how the processes were created. The value can be in the range 3 - 32767. The default value is 25. The **MAXPROCUSER** parameter is specified in SYS1.PARMLIB member BPXPRMxx. For guidance about the **MAXPROCUSER** setting, see *z/OS MVS Initialization and Tuning Reference,* SA22-7592.

The **MAXPROCUSER** parameter is independent of any particular user ID. However, an equivalent IBM Resource Access Control Facility (RACF®) setting, called **PROCUSERMAX**, limits the number of processes by user ID for a particular user. **PROCUSERMAX** sets the maximum number of processes per user ID field of the RACF OMVS SEGMENT of a user ID profile.

The following TCBs contribute to the potential number of processes associated with a particular CICS region:

MAXOPENTCBS The maximum number of L8 and L9 TCBs that can exist.

MAXJVMTCBS The maximum number of J8 and J9 TCBs that can exist.

MAXSSLTCBS The maximum number of S8 TCBs that can exist.

MAXXPTCBS The maximum number of X8 and X9 TCBs that exist.

SO TCB Used to issue the necessary UNIX System Services and CEEPIPI calls for the socket domain.

SL TCB Provides a listening environment for sockets domain requests.

SP TCB Owns the S8 TCBs and the SSL cache.

In addition, TCBs used by the separate *TCP/IP Socket Interface for CICS* component of the z/OS Communications Server (if applicable) contribute to the number of processes associated with a particular CICS region.

By adding the number of TCBs from the previous list, you can obtain the total number of processes that might be associated with a CICS region. This total represents a possible upper limit for the region.

If the CICS systems share the user ID, add the totals to get the maximum number of processes associated with that user ID, because the **MAXPROCUSER** value indicates the number of processes for a UID, not for each job.

After you determine the total possible number of processes associated with each user ID for your CICS regions, use the largest number, and add 10% to it when calculating the value of **MAXPROCUSER**.

If you have a particular user ID with a high result for the total number of processes required, due to several CICS systems sharing the user ID, setting **MAXPROCUSER** to such a figure might not be appropriate. In this situation, use the **PROCUSERMAX** parameter on the OMVS segment of the RACF profile for the user ID to set a suitably high value to accommodate the requirements of the user ID.

The setting of the **MAXPROCUSER** and **PROCUSERMAX** parameters does not use extra resources. These values are limiting. CICS does not generate the open TCBs until they are needed, meaning that processes and system resources are not associated with TCBs until required. TCBs specified in the SSLTCBS system initialization parameter are created at CICS system initialization. The setting of the TCPIP system initialization parameter does not affect the use of open TCBs by OTE. Also, if you specify **TCPIP=NO** and no OTE-managed services are used by CICS, then two of the **MAXPROCUSER** entries are used in the initialization of the sockets domain.

2.8.1 Implications of setting MAXPROCUSER too low

If you do not set a large enough value for **MAXPROCUSER** for the CICS environment, you might see message BPXI040I or message DFHKE0500.

Message BPXI040I

Message BPXI040I is a UNIX System Services message that alerts the operator that system resources are being consumed. The message notifies the operator when a threshold of 85% of the **MAXPROCUSER** value for a UNIX process identifier (PID) is reached.

The percentage can exceed 100%, because two special UIDs are allowed to create more processes than **MAXPROCUSER** normally allows. The superuser ID (UID=0) can exceed many of the limits set in BPXPRMxx. Also, the default UID can exceed the **MAXPROCUSER** setting because many users can use the default UID, and they each have independent processes. If each user is given an individual UID, each user is subject to **MAXPROCUSER** independently. The default UID refers to an RACF user ID without an OMVS segment defined for it. As such, it uses the default OMVS segment. Do not confuse the default UID with the CICS default user.

Message DFHKE0500

Message DFHKE0500 is issued by the CICS TS V3.1 kernel when the **MAXPROCUSER** value is exceeded for the user ID of the CICS system. This result can occur because several CICS systems share the UID on UNIX System Services. They have a requirement to use several TCBs that is greater than the value defined in the **MAXPROCUSER** parameter.

2.9 Static and dynamic calls

If you defined a program with **CONCURRENCY(THREADSAFE)** or **CONCURRENCY(REQUIRED)** for CICS TS V4.2, you must also code to threadsafe standards all routines that are statically or dynamically called from this program (such as COBOL routines).

When you use an **EXEC CICS LINK** command to link from one program to another, the program-link stack level is incremented. However, a routine that is statically or dynamically called does not involve passing through the CICS command-level interface (CLI). Therefore, it does not cause the program-link stack level to be incremented.

With COBOL routines, a static call causes a simple branch and link to an address resolved at link-edit time. For a dynamic call, a program definition is required so that Language Environment can load the program. After the load, a simple branch and link is still used. When a routine is called using either method, CICS does not receive control and is, therefore, unaware of the program execution change. The program that called the routine is still considered to be running, and its program definition is still considered to be the current program definition.

If the program definition for the calling program states **CONCURRENCY(THREADSAFE)** or **CONCURRENCY(REQUIRED)** for CICS TS V4.2, the called routine must also comply with threadsafe standards. Programs with these states for CICS TS V.2 remain on an open TCB when they return from a DB2 call, IMS call (for CICS TS V4.2), or any threadsafe **EXEC CICS** command, which is inappropriate for a nonthreadsafe program.

For example, consider a situation in which the initial program of a transaction, program A, issues a dynamic call to program B, which is a COBOL routine. Because the CICS CLI was not involved, CICS is unaware of the call to program B and considers the current program to be program A. Program B issues a DB2 call.

Upon return from the DB2 call, CICS must determine whether the program can remain on the open TCB or must switch back to the QR TCB to ensure threadsafe processing. To make this determination, CICS examines the **CONCURRENCY** attribute of what it considers as the current program (program A in this example). If program A is defined as **CONCURRENCY(THREADSAFE)** or

CONCURRENCY(REQUIRED) for CICS TS V4.2, CICS allows processing to continue on the open TCB. Program B is currently running. Therefore, if processing must continue safely, program B must be coded to threadsafe standards. For more information, see 7.5, "COBOL calls" on page 167.

2.10 Threadsafe and nonthreadsafe commands in CICS Transaction Server

CICS Transaction Server has several threadsafe commands and commands that have become threadsafe from release to release. It also has nonthreadsafe SPI commands.

2.10.1 Threadsafe API commands

If you write and define a CICS program as threadsafe, it can receive control on an OTE TCB. To obtain the maximum performance benefit from OTE, write your CICS programs in a threadsafe manner to prevent CICS from switching TCBs. However, keep in mind that not all **EXEC CICS** commands are threadsafe. Issuing any of the nonthreadsafe commands causes CICS to switch your task back to the QR TCB to ensure serialization. The CICS API commands that are threadsafe are noted in the command syntax diagrams in the appropriate *CICS Application Programming Reference*, with the statement `This command is threadsafe`.

For more information, see the CICS Transaction Server V4.2 Information Center at the following address, and search for *application programming reference*:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

Table 2-1 lists the threadsafe API commands for CICS TS V1 and V2.

Table 2-1 Threadsafe API commands in CICS TS V1R3, V2R2, and V2R3

CICS TS V1.3	CICS TS V2.2	CICS TS V2.3
ABEND ADDRESS ASSIGN DELETEQ TS ENTER TRACENUM FREEMAIN GETMAIN HANDLE ABEND HANDLE AID HANDLE CONDITION IGNORE CONDITION LINK LOAD MONITOR POP HANDLE PUSH HANDLE READQ TS RELEASE RETURN WRITEQ TS XCTL	DEQ ENQ SUSPEND WAIT EXTERNAL	ASKTIME CHANGE TASK DOCUMENT CREATE DOCUMENT INSERT DOCUMENT RETRIEVE DOCUMENT SET FORMATTIME

Table 2-2 shows the threadsafe API commands in CICS TS V3.1.

Table 2-2 Threadsafe commands in CICS TS V3.1

New commands that are threadsafe	Existing commands that are now threadsafe
CONVERTTIME DELETE CONTAINER (CHANNEL) GET CONTAINER (CHANNEL) INVOKE WEBSERVICE MOVE CONTAINER (CHANNEL) PUT CONTAINER (CHANNEL) SOAPFAULT ADD SOAPFAULT CREATE SOAPFAULT DELETE WEB CONVERSE WEB CLOSE WEB OPEN WEB PARSE URL WEB RECEIVE (Client) WEB SEND (Client)	WEB ENDBROWSE FORMFIELD WEB ENDBROWSE HTTPHEADER WEB EXTRACT WEB READ FORMFIELD WEB READ HTTPHEADER WEB READNEXT FORMFIELD WEB READNEXT HTTPHEADER WEB RECEIVE (Server) WEB RETRIEVE WEB SEND (Server) WEB STARTBROWSE FORMFIELD WEB STARTBROWSE HTTPHEADER WEB WRITE HTTPHEADER

Table 2-3 shows the threadsafe API commands in CICS TS V3.2.

> **File Control API commands:** The File Control API commands in Table 2-3 are threadsafe if the file to which they refer is defined as local VSAM or RLS. If the file is defined as remote, is a shared data table, a coupling facility data table, or is a BDAM file, the commands are not threadsafe.

Table 2-3 Threadsafe API commands in CICS TS V3.2

New commands that are threadsafe	Existing commands that are now threadsafe

`DOCUMENT DELETE`	`WAIT JOURNALNAME`
	`WAIT JOURNALNUM`
	`WRITE JOURNALNAME`
	`WRITE JOURNALNUM`
	`DELETE`
	`ENDBR`
	`READ`
	`READNEXT`
	`READPREV`
	`RESETBR`
	`REWRITE`
	`STARTBR`
	`UNLOCK`
	`WRITE`

Table 2-4 lists the new threadsafe API commands in CICS TS V4.1 and V4.2.

Table 2-4 New threadsafe API commands in CICS TS V4

CICS TS V4.1	CICS TS V4.2

`INVOKE SERVICE`
`SIGNAL EVENT`
`TRANSFORM DATATOXML`
`TRANSFORM XMLTODATA`
`WEB ENDBROWSE QUERYPARM`
`WEB READ QUERYPARM`
`WEB READNEXT QUERYPARM`
`WEB STARTBROWSE`
`QUERYPARM`
`WSACONTEXT BUILD`
`WSAEPR CREATE`
`WSACONTEXT DELETE`
`WSACONTEXT GET`
	`CHANGE PHRASE`
	`VERIFY PHRASE`

Table 2-5 lists the existing threadsafe API commands in CICS TS V4.2.

Table 2-5 Existing threadsafe API commands in CICS TS V4.2

BIF DEEDIT	READNEXT
BIF DIGEST	READPREV
CHANGE PASSWORD	RESETBR
DEFINE COUNTER and DEFINE DCOUNTER	REWIND COUNTER and REWIND DCOUNTER
DELETE	REWRITE
DELETE COUNTER and DELETE DCOUNTER	SIGNOFF
ENDBR	SIGNON
EXEC DLI	STARTBR
EXTRACT CERTIFICATE	SYNCPOINT
EXTRACT TCPIP	SYNCPOINT ROLLBACK
GET COUNTER and GET DCOUNTER	UNLOCK: UPDATE COUNTER and UPDATE
LINK	DCOUNTER
QUERY COUNTER and QUERY DCOUNTER	VERIFY PASSWORD
QUERY SECURITY	WRITE
READ	

2.10.2 Threadsafe SPI commands

The CICS SPI commands that are threadsafe are noted in the command syntax diagrams in the appropriate *CICS System Programming Reference* manual, with the statement `This command is threadsafe`. You can find this manual in the CICS Transaction Server V4.2 Information Center at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

Table 2-6 shows the threadsafe SPI commands in CICS TS V1, V2, and V3.

Table 2-6 Threadsafe SPI commands in CICS TS V1, V2, and V3

CICS TS V1.3	CICS TS V2.2	CICS TS V2.3	CICS TS V3.2
INQUIRE EXITPROGRAM	DISCARD DB2CONN	INQUIRE WORKREQUEST	INQUIRE ASSOCIATION
INQUIRE TASK	DISCARD DB2ENTRY	SETWORKREQUEST	INQUIRE ASSOCIATION LIST
	DISCARD DB2TRAN	INQUIRE DOCTEMPLATE	INQUIRE IPCONN
	INQUIRE DB2CONN	DISCARD DOCTEMPLATE	INQUIRE LIBRARY
	INQUIRE DB2ENTRY		SET IPCONN
	INQUIRE DB2TRAN		PERFORM JVMPOOL
	SET DB2CONN		SET DOCTAMPLATE
	SET DB2ENTRY		INQUIRE FILE
	SET DB2TRAN		

Table 2-7 shows the threadsafe SPI commands in CICS TS V4.1.

Table 2-7 Threadsafe SPI commands in CICS TS V4.1

DISCARD ATOMSERVICE DISCARD BUNDLE DISCARD EVENTBINDING DISCARD JVMSERVER DISCARD MQCONN INQUIRE ATOMSERVICE INQUIRE EVENTBINDING INQUIRE BUNDLE INQUIRE BUNDLEPART INQUIRE CAPTURESPEC INQUIRE EVENTPROCESS	INQUIRE JVMSERVER INQUIRE MQCONN INQUIRE MQINI INQUIRE XMLTRANSFORM SET ATOMSERVICE SET BUNDLE SET EVENTBINDING SET EVENTPROCESS SET JVMSERVER SET MQCONN SET XMLTRANSFORM

Table 2-8 shows the threadsafe SPI commands in CICS TS V4.2.

Table 2-8 Threadsafe SPI commands in CICS TS V4.2

New SPI commands	Existing SPI commands
INQUIRE CAPDATAPRED INQUIRE CAPINFOSRCE INQUIRE CAPOPTPRED INQUIRE EPADAPTER INQUIRE OSGIBUNDLE INQUIRE OSGISERVICE INQUIRE TEMPSTORAGE SET EPADAPTER SET TEMPSTORAGE	INQUIRE CLASSCACHE INQUIRE JVM INQUIRE JVMPOOL INQUIRE JVMPROFILE PERFORM CLASSCACHE PERFORM JVM POOL RESYNC ENTRYNAME SET CLASSCACHE SET JVMPOOL

2.10.3 Threadsafe XPI commands

All the XPI commands are threadsafe, except for the **DFHDUDUX TRANSACTION_DUMP** command.

2.10.4 Nonthreadsafe SPI commands

Table 2-9 shows the nonthreadsafe SPI commands in CICS TS V4.1.

Table 2-9 Nonthreadsafe SPI commands in CICS TS V4.1

CREATE ATOMSERVICE CREATE BUNDLE CREATE JVMSERVER CREATE MQCONN CSD ADD CSD ALTER CSD APPEND CSD COPY CSD DEFINE CSD DELETE CSD DISCONNECT CSD ENDBRGROUP CSD ENDBRLIST CSD ENDBRRSRCE CSD GETNEXTGROUP	CSD GETNEXTLIST CSD GETNEXTRSRCE CSD INQUIREGROUP CSD INQUIRELIST CSD INQUIRERSRCE CSD INSTALL CSD LOCK CSD REMOVE CSD RENAME CSD STARTBRGROUP CSD STARTBRLIST CSD STARTBRRSRCE CSD UNLOCK CSD USERDEFINE

2.11 Function shipping considerations

Depending on your version of CICS Transaction Server, you must keep in mind the function shipping considerations as explained in the following sections.

2.11.1 Before CICS Transaction Server V4.2

Terminal control, including multiregion operation (MRO) and intersystem communication (ISC), is not threadsafe. Therefore, CICS must issue a mode switch to the QR TCB to function ship a request to a remote region. Issuing this mode switch means that any command that is listed as threadsafe is treated as such when it is run locally, but incurs the overhead of a TCB switch if function shipped. For more information, see 7.4, “Function shipped commands” on page 162.

2.11.2 CICS Transaction Server V4.2 and later

The mirror program DFHMIRS (supplied with CICS), which is used by all mirror transactions, is now defined as threadsafe. In addition, the IPIC transformers are now threadsafe. For IPIC connections only, CICS runs the mirror program on an L8 open TCB whenever possible. For threadsafe applications that function ship commands to other CICS regions using IPIC, the resulting reduction in TCB switching improves the performance of the application compared to other intercommunication methods. To gain the performance improvement for remote files, you must specify the system initialization parameter **FCQRONLY=NO** in the FOR.

For remote file control or temporary storage requests shipped over IPIC connections, CICS no longer forces a switch to the QR TCB in the AOR if it is running currently on an open TCB. The requests are shipped running on the open TCB.

In the FOR or QOR, the mirror determines when to switch to an open TCB. It switches for the first file control or temporary storage request received over an IPIC connection. The idea is for long-running mirrors to keep the mirror running on an open TCB.

A new option, MIRRORLIFE, is now added to the IPCONN attributes for function-shipped file control and temporary storage requests using an IPIC connection. MIRRORLIFE improves efficiency and provides performance benefits by specifying the lifetime of mirror tasks and the amount of time a session is held.

3

Threadsafe techniques

This chapter highlights the techniques that you can use when migrating to CICS threadsafe applications.

This chapter includes the following sections:

- Threadsafe standards
- Serialization techniques
- Application design considerations

© Copyright IBM Corp. 2004, 2012. All rights reserved.

3.1 Threadsafe standards

Application and system programmers must write all new CICS application programs to threadsafe standards as explained in this section. By applying the following guidance, you can ensure that existing and new applications can maximize the benefits to be gained from being defined as threadsafe:

- Ensure that all programs are written to current CICS standards, as documented in the *CICS Application Programming Guide* in the CICS Transaction Server V4.2 Information Center at:

  http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

  In particular, programs must meet the following standards:

  – Be compiled and link-edited as reentrant, and be in read-only storage (system initialization table (SIT) parameter **RENTPGM=PROTECT**).

    This standard is not an absolute requirement for threadsafe programming. However, if a program can overwrite itself, the program is effectively shared storage, and access to it must be serialized. For information about appropriate serialization techniques, see 3.2, "Serialization techniques" on page 56.

  – Use only published CICS interfaces to external resources.

    Again, this standard is not an absolute requirement for threadsafe programming. However, using existing MVS calls under CICS, before open transaction environment (OTE), is most likely to cause the quasi-reentrant (QR) task control block (TCB) to enter an MVS WAIT state, stopping CICS. For this reason, such MVS calls are not allowed.

    This restriction is removed in CICS Transaction Server for z/OS (CICS TS) V3.1 and later by using OPENAPI programs because they never run application code on the QR TCB. However, use of application programming interfaces (APIs) not supplied by CICS is at your own risk. Such usage is not formally supported in CICS TS V3.1.

    If existing programs are accessing shared application resources, then access must be serialized before defining the programs as threadsafe. For information about appropriate serialization techniques, see 3.2, "Serialization techniques" on page 56.

- Avoid use of the CICS common work area (CWA) if at all possible (that is, set SIT parameter **WRKAREA=0**). Instead use shared resources that are accessed through CICS APIs (for example, CICS temporary storage). If you cannot avoid use of the CWA, and the data in it is updated, ensure that all programs use an appropriate serialization technique to access it. For information about appropriate serialization techniques, see 3.2, "Serialization techniques" on page 56.

- Ensure that all programs (including PLT programs, user exits, and user-replaceable modules) do not create or access shared storage (that is, as created by the **EXEC CICS GETMAIN SHARED** command). Instead use shared resources that are accessed by using CICS APIs (for example, CICS temporary storage). If you cannot avoid using shared storage, and the data in it is updated, ensure that all programs use an appropriate serialization technique to access it. For information about appropriate serialization techniques, see 3.2, "Serialization techniques" on page 56.
- Avoid using global work areas (GWAs) in user exits, as created by the `GALENGTH` option of the **EXEC CICS ENABLE PROGRAM** command. These GWAs might also be referenced by using **UEPGAA** parameter in the exit or through the **EXTRACT EXIT** command from other application programs. Depending on the exit point, instead you might be able to use shared resources that are accessed through CICS APIs.

  If use of a GWA is necessary, and the data in it is updated, ensure that all user exits and application programs use an appropriate serialization technique to access it. For example, an application program can use the **EXEC CICS ENQ** or **EXEC CICS DEQ** command, and a user exit can use XPI ENQUEUE and DEQUEUE functions if they both use the same resource argument. For information about appropriate serialization techniques, see 3.2, "Serialization techniques" on page 56.
- Ensure that all programs, user exits, and user-replaceable modules (URMs) use only threadsafe **EXEC CICS** commands. Check the command syntax diagrams in the *CICS Application Programming Reference* and the *CICS System Programming Reference* guides for the statement: `This command is threadsafe`. You can search for these reference guides in the CICS Transaction Server V4.2 Information Center at:

  http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

  If you cannot avoid using nonthreadsafe commands, design the application to minimize the performance impact. For information about threadsafe application design, see 3.3, "Application design considerations" on page 61.
- Ensure that all programs that are written or identified as threadsafe are defined to CICS with the **CONCURRENCY(THREADSAFE)** or **CONCURRENCY(REQUIRED)** attribute. If program autoinstall is enabled, amend your autoinstall control program to ensure that the correct **CONCURRENCY** value is set for each program. Alternatively, use the CICS environment variable **CICSVAR** as explained in 5.2.2, "CICS environment variable CICSVAR" on page 98.
- Review the use of function shipping within the application. Usage of OTE for function shipping was introduced with CICS TS V4.2 For more information, see 7.4, "Function shipped commands" on page 162.

- Check with IBM for the latest threadsafe-related APARs, and apply any maintenance that is appropriate to your environment.
- Check with your independent software vendors (ISVs) to ensure that their programs and exits comply with threadsafe standards and are defined as threadsafe. If they are not threadsafe, or issue nonthreadsafe **EXEC CICS** commands, understand the implications for your application.

3.2 Serialization techniques

As explained in Chapter 5, "Application review" on page 83, all access to application shared resources (if they exist) that can be updated must be serialized before defining the associated programs as threadsafe. This section outlines several techniques that you can use to achieve this state.

Regardless of the technique that you select, establish a standard for your organization, so that all programs that access the same resource use the same serialization technique. No program is threadsafe until *all* programs that access the resource are changed to include serialization.

3.2.1 Recommended serialization techniques

We encourage use of the following serialization techniques:

- CICS API enqueue or dequeue
- CICS XPI enqueue or dequeue
- Compare and swap

CICS API enqueue or dequeue

The **EXEC CICS ENQUEUE** and **EXEC CICS DEQUEUE** commands are ideally suited for CICS application programs to serialize access to shared resources. Both commands are threadsafe. Therefore, they do not incur the performance overhead of switching a task back to the QR TCB.

For information about coding **EXEC CICS ENQUEUE** and **EXEC CICS DEQUEUE** commands, see the *CICS Application Programming Reference* in the CICS Transaction Server V4.2 Information Center at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

CICS XPI enqueue or dequeue

The DFHNQEDX macro function call is an enhancement to the exit programming interface (XPI) that was introduced with CICS TS V1.3. This macro function call provides the same ENQUEUE and DEQUEUE capability as the CICS API. The

XPI commands are threadsafe. Therefore, they do not incur the performance overhead of switching a task back to the QR TCB.

The XPI ENQUEUE or DEQUEUE function call is ideal for use within a user exit to serialize access to a GWA or any other shared resource. For information about coding XPI commands, see the CICS Transaction Server for z/OS V4.2 Information Center at the following address, and search for *CICS Customization Guide*:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

Tip: If you want the ENQUEUE or DEQUEUE function call around the same resource using both API and XPI, use the ENQUEUE function on different enqueue pools. In this case, you must use the new **`XPI ENQUEUE_TYPE`** option, which is needed when you are using exits and GWAs.

Compare and swap

Assembler applications and user exits can use one of the conditional swapping instructions to serialize access to shared resources:

- Compare and swap (CS)
- Compare double and swap (CDS)

For information about coding these instructions, see *z/Architecture Principles of Operation*, SA22-7832.

Comparison of recommended options

Table 3-1 compares the preferred options.

Table 3-1 Comparison of options

Option	Advantages	Disadvantages
Use compare and swap assembler instructions on a shared data element.	► It offers potentially the best performance. ► It is the easiest implementation of an API that is not a CICS API. ► The new locking mechanism is nondisruptive. It can be installed one program at a time.	► You cannot use it for fields greater than 4 bytes (8 bytes for CDS instruction). ► For fields less than 4 bytes, activity on adjacent bytes can cause additional failed lock attempts. ► Storage access is not threadsafe until all programs are converted. ► Requires an assembly language program or subroutine.

Option	Advantages	Disadvantages
Use test and set or compare and swap assembler instruction on separate *lock* byte.	► The new locking mechanism is nondisruptive. It can be installed one program at a time. ► *Lock* granularity is single byte or word. ► *Lock* might be defined for noncontiguous areas. ► If using CS, a *Locked* status can indicate which CICS task owns the resource (such as task number, and terminal identifier).	► An application failure while holding a lock causes other TCBs to spin until the lock is manually cleared. The effects can be mitigated somewhat by adding a retry counter to the lock loop. However, access to the resource is denied until the lock is cleared. ► Storage access is not threadsafe until all programs are converted. ► Requires an assembly language program or subroutine.
Use a compare and swap assembler instruction after moving the shared data element to a new fullword.	► Unrelated tasks do not interfere. ► It guarantees that all accesses to shared resource are identified. ► This option is viable if a limited number of programs is involved.	► No migration path is available. All affected programs must be installed at the same time. ► This option is not viable if a many programs are involved. ► Requires an assembly language program or subroutine.
CICS ENQ (API or XPI)	► *Lock* granularity is single byte. ► Application failure does not result in a held lock. ► No knowledge of an assembly language is required.	► It requires more CPU than API techniques require that are not CICS APIs. ► You must always perform ENQ or DEQ even when the resource is not relevant to any other tasks. ► You must consider implications of the MAXLIFETIME option.

3.2.2 Generalized compare and swap routine

In this section, the assumption is made that most accesses to shared resources are for maintaining flags, counters, or chain pointers. In general, where this assumption applies, it might be possible to implement a single subroutine (written in assembly language) with the following characteristics:

- That protects the integrity of the shared resources
- Is generally more efficient than ENQ/DEQ
- Insulates the application programmer from the details of implementing CS instructions for every shared data element.

Except for the actual operation to be performed (such as increment, decrement, OR, or AND), most CS implementations follow the same pattern. For example, to increment a 4-byte counter, the code always follows the pattern shown in Example 3-1.

Example 3-1 Compare and swap implementation

```
* Increment a 4-byte field
INCREMENT DS    0H
          L     ROLD,SHARED           get shared data
RETRY     LR    RNEW,ROLD             save Shared value
          LA    RNEW,1(,RNEW)         increment value
          CS    ROLD,RNEW,SHARED      store new value
          BNZ   RETRY                 serialization failed
          B     RETURN                successful completion
```

Trying the operation again without embedding a form of delay is disconcerting to some in that it looks as though there is a high potential for a CPU loop. This point is addressed in *z/Architecture Principles of Operation*, SA22-7832, and explained in the following note.

> **Loops:** A *CPU loop* differs from the typical *bitspin loop*. In a bitspin loop, the program continues to loop until the bit changes. In this example, the program continues to loop only if the value changes during each iteration. If the number of CPUs simultaneously attempt to modify a single location by using the sample instruction sequence, one CPU falls through on the first try, another one loops one time, and so on until all CPUs succeed.

Implementing a retry counter mitigates this problem. A retry counter also provides a convenient method for tracking potential resource contention at a granular level. You log the retry count somewhere, such as in a CICS trace or monitor entry for offline analysis. Adding a retry counter in the code yields the results shown in Example 3-2. The symbol RCOUNT is a register other than ROLD or RNEW.

Example 3-2 Retry count

```
* Increment a 4-byte field
INCREMENT DS    0H
          XR    RCOUNT,RCOUNT         clear retry counter
          L     ROLD,SHARED           get shared data
RETRY     LA    RCOUNT,1(,RCOUNT)     increment retry count
          CL    RCOUNT,MAXTRIES       too many attempts?
          BNL   ERROR                 yes, quit trying
          LR    RNEW,ROLD             save original value
          LA    RNEW,1(,RNEW)         increment value
          CS    ROLD,RNEW,SHARED      store new value
          BNZ   RETRY                 serialization failed-retry
          B     RETURN                return to caller
```

```
ERROR     DS    0H
     < Too many retry attempts >
```

You can implement the retry counter in many ways. However, the important point is that the logic required to set up the CS instruction is always the same.

Likewise, the *increment value* instruction [`LA     RNEW,1(,RNEW)`] is the only instruction in either pattern that must change to implement a different operation (such as decrement, AND, or OR). Placing this code in a subroutine in which SHARED is passed by reference allows the creation of a generalized routine for manipulating shared memory elements. Such a subroutine must handle the most common updates of shared memory.

3.2.3 Nonrecommended techniques

You can also use the following techniques to serialize access to resources. However, we discourage use of these techniques because of the disadvantages associated with each one.

LINK to a QUASIRENT program

A linked-to program, defined as QUASIRENT, runs under the QR TCB and can, therefore, take advantage of the serialization provided by CICS quasi-reentrancy. Even in QR mode, serialization is provided only if the program retains control and does not wait.

Therefore, a valid serialization technique is to move all shared resource access to a single program and define it as quasi-reentrant. All other application programs can then be defined as threadsafe if they always link to the QR program to access the shared resource.

Although this technique is valid, in that it protects the integrity of the shared resource, it does not result in the same performance gain as one of the recommended techniques, such as enqueue or dequeue. Where the recommended techniques allow the program to remain on an open TCB (assuming it is there already), this technique incurs the performance overhead of a TCB switch to QR.

CICS transaction class

With user-defined CICS transaction classes (TRANCLASS), the systems programmer can limit the number of concurrent tasks for transactions that belong to each class. Creating a transaction class with a MAXACTIVE value of 1 is a crude method of serializing resource access. All transactions that belong to the class are single threaded.

This technique has one advantage in that it can be achieved without changing any application code. However, even in a moderately busy system, it is likely to have a severe impact on transaction response times. It also runs contrary to the objective of implementing threadsafe applications in the first place, which is improved performance.

MVS enqueue or dequeue

Before the release of CICS TS V3, issuing API calls (that are not CICS API calls) from a CICS program is not supported because CICS is unable to guarantee that the QR TCB can issue such calls. If the application and system programmers design the system so that such a call is issued from an open TCB, a future program change might cause the call to be issued from QR and block all CICS tasks. For example, such a program change might be the insertion of a nonthreadsafe EXEC CICS command.

The same concept applies in CICS TS V3 unless the program is defined as THREADSAFE and OPENAPI, which ensures that the program runs on an open TCB. Even in this situation, use CICS services, because CICS provides better facilities to release enqueues in error situations.

3.3 Application design considerations

An ideal candidate application to define as THREADSAFE (or REQUIRED, introduced in CICS TS V4.2) and CICSAPI, and therefore to use OTE, has the following characteristics:

- Contains threadsafe application code
- Contains only threadsafe EXEC CICS commands
- Uses only threadsafe user exit programs

An application that is defined as THREADSAFE moves to an L8 TCB when it makes its first call to an OPENAPI task-related user exit (TRUE). Examples might be an SQL request, an IMS call, an IP CICS sockets request, or a WebSphere MQ (WMQ) request. Then it continues to run on the L8 TCB through any number of such requests and application code, requiring no TCB switching.

An application that is defined as REQUIRED is given control on an L8 TCB and then continues to run on the L8 TCB through any number of such requests and application code. It does no require any TCB switching.

If many application programs are not threadsafe, or programs contain nonthreadsafe EXEC CICS commands, you can still design application transactions to minimize the number of TCB switches and obtain the performance benefits associated with running threadsafe.

As shown in Figure 2-3 on page 20, the execution path between the first and the last SQL call is key to the performance of a CICS DB2 task running under OTE. Then, by placing nonthreadsafe code and commands before the first SQL call or after the final SQL call, the application avoids incurring the CPU overhead that placing the same code between SQL calls incurs.

We return to the example of the application with both DB2 and Virtual Storage Access Method (VSAM) data. In releases before CICS TS V3.2, by designing the transactions so that the VSAM and DB2 calls are not interspersed, such an application can at least partially use OTE.

3.3.1 Application design considerations for CICS TS V3.2

With CICS TS V3.2, OTE has additional enhancements. For example, you can define an application program to begin execution on an open TCB. You do not have to wait for a call to an OPENAPI TRUE (DB2, MQ or IP CICS Sockets) to move the task to an open TCB.

However, you must use care with applications that call OPENAPI-enabled TRUEs. You might be tempted to define an application OPENAPI that is currently defined as threadsafe, so that it begins running on the open TCB rather than waiting for the call to an OPENAPI TRUE. The issue is that, if an application program is defined as OPENAPI and as EXECKEY(USER), the task begins on an L9 TCB. Then when a call to an OPENAPI TRUE is encountered, a switch to an L8 TCB occurs because OPENAPI-enabled TRUEs always run in the CICS key. This situation can lead to TCB switching across three TCBs (QR, L8, and L9). If nonthreadsafe CICS API commands are also in the program, the performance impact can be undesirable. Figure 2-5 on page 22 illustrates this situation, which depends on storage protection being active within the CICS region.

File control

File control for local VSAM and VSAM record-level sharing (RLS) access is now available by using the following threadsafe API and system programming interface (SPI) commands:

- **READ**
- **REWRITE**
- **WRITE**
- **DELETE**
- **UNLOCK**
- **STARTBR, READNEXT, READPREV, RESETBR, ENDBR**
- **SPI - INQUIRE FILE**

Basic direct-access methods (BDAMs), system dump tables (SDTs), coupling facility data tables (CFDTs), and remote files have no threadsafe API.

The following file control functions that are not threadsafe still run on the QR TCB:

- Open/close
- Enable/disable
- Quiesce functions
- INQ DSNAME
- SET SPI functions

You must ensure that file control exits are made threadsafe. Otherwise, when the exit is called, a switch is made to the QR TCB, and then a switch back is made when the exit processing completes. You must change products that previously located the file control table (FCT) by using control block interrogation. Otherwise, your application can become corrupted.

Use the INQUIRE FILE SPI, the official interface, for access to information in the FCT. No interface is available to return the addresses of FCT entries, data set name blocks (DSNBs), or any other file control block.

OPENAPI good and bad candidates

The example in the previous section shows that not all threadsafe application programs are necessarily good candidates to be defined as OPENAPI. If the program being defined is written to threadsafe standards, you must decide whether to define the program as CICSAPI or OPENAPI.

Consider the following guidelines for OPENAPI and CICSAPI:

- Candidates for OPENAPI with THREADSAFE
  - Programs with threadsafe APIs only
  - SQL or WebSphere MQ programs with a CICS key
  - CPU-intensive programs
- Candidates for CICSAPI with THREADSAFE
  - SQL or WebSphere MQ programs with some or many nonthreadsafe APIs
  - SQL or WebSphere MQ programs with a user key

Bad candidates for OPENAPI are user key DB2, IP CICS Sockets, and WebSphere MQ programs because the application starts on an L9 TCB and must switch to an L8 TCB and back again for each call to an OPENAPI TRUE. Likewise for nonthreadsafe CICS commands, you switch to QR and back again.

The best candidates for OPENAPI are DB2, IP CICS Sockets, or WebSphere MQ programs that have only threadsafe CICS API commands and are defined as EXECKEY(CICS). Also, CPU-intensive programs (that is, programs that do a lot of processing without giving up control to CICS) are good candidates for OPENAPI. They can perform the intensive processing without affecting other tasks that might be waiting to run on the QR TCB.

User key for in active storage protection: If storage protection is not active (`STGPROT=NO`), the user key is the same as the CICS key. Both types of programs run on L8 TCBs, if defined as OPENAPI.

3.3.2 Application design considerations for CICS TS V4.2

CICS TS V4.2 now offers the following enhancements to OTE:

- The introduction of a new concurrency option on the program definition that allows for greater exploitation of OTE for threadsafe applications
- Using an OTE for function shipping, by allowing the mirror program, when it is started in a remote CICS region through an IP interconnectivity (IPIC) connection, to run on an open TCB
- Making more of the API and SPI threadsafe, including access to IMS databases by using the CICS IMS Database Control (DBCTL) interface.

For an overview of the threadsafe commands in CICS TS, see 2.10, "Threadsafe and nonthreadsafe commands in CICS Transaction Server" on page 45.

Good and bad OPENAPI candidates

CICS TS V4.2 provides a new `CONCURRENCY(REQUIRED)` setting. As with `CONCURRENCY(THREADSAFE)`, the new setting specifies that the program is coded to threadsafe standards and contains threadsafe logic. However, in addition, the program must run on an open TCB. Therefore, when given control, the program runs on an open TCB from the start. If CICS must switch to the QR TCB to process a nonthreadsafe CICS command, CICS returns to the open TCB when it returns control to the application program. Use API(OPENAPI) only if you are going to use APIs that are not from CICS and not as a back door to starting the program on an open TCB.

With the `CONCURRENCY(REQUIRED)` option, the user defines that the program must start on an open TCB independently of defining the APIs that it uses:

- If the program uses only CICS supported APIs (including access to external resource managers, such as DB2, IMS, and WebSphere MQ), it must be defined with program attribute API(CICSAPI). In this case, CICS always uses an L8 open TCB, regardless of the execution key of the program, because CICS commands do not rely on the key of the TCB.
- If the program uses other APIs that are not CICS APIs, it must be defined with program attribute API(OPENAPI). In this case, CICS uses an L9 TCB or an L8 TCB depending on the execution key of the program so that the APIs that are not from CICS can operate correctly. This OPENAPI behavior is the same as in previous releases.

Existing threadsafe applications take advantage of the performance gains of being able to run on the same TCB as the call to an external resource manager, such as DB2, IMS or WebSphere MQ, by being defined as **CONCURRENCY(THREADSAFE) API(CICSAPI)**. These applications might be able to gain further throughput advantages by being defined as **CONCURRENCY(REQUIRED) API(CICSAPI)**. Throughput gains are achieved when an application can run for longer periods of time on an open TCB.

However, not all applications are suitable. For example, a threadsafe application that issues a large number of EXEC SQL requests and then issues many EXEC CICS commands that are not threadsafe is best left as **CONCURRENCY(THREADSAFE)**. Defining the application as **CONCURRENCY(REQUIRED)** indicates two TCB switches for each nonthreadsafe CICS command, because control always returns to the application on the open TCB. This example demonstrates the importance of knowing what the application does.

CONCURRENCY(THREADSAFE) API(OPENAPI): To maintain compatibility with previous releases, **CONCURRENCY(THREADSAFE) API(OPENAPI)** is still a supported combination, but it is deprecated. It has the same meaning as **CONCURRENCY(REQUIRED) API(OPENAPI)**. That is, the program runs on open TCBs from the start and uses APIs that are not supplied by CICS.

When a program is defined as **CONCURRENCY(REQUIRED)** or **API(OPENAPI)**, the program is always given control on the open TCB. All task-attach processing still runs on the QR TCB. Therefore, if the QR TCB is blocked, no work comes into CICS regardless of the program definitions.

File control

The CICS supplied mirror program, DFHMIRS, which is used by all mirror transactions, is now defined as threadsafe. In addition, the IPIC transformers are now threadsafe. For IPIC connections only, CICS runs the mirror program on an L8 open TCB whenever possible. For threadsafe applications that function ship commands to other CICS regions using IPIC, the resulting reduction in TCB switching improves the performance of the application compared to other intercommunication methods. To gain the performance improvement for remote files, you must specify the system initialization parameter **FCQRONLY=NO** in the file-owning region (FOR).

For remote file control or temporary storage requests shipped over IPIC connections, CICS no longer forces a switch to the QR TCB in the application-owning region (AOR) if it is running currently on an open TCB. The requests are shipped running on the open TCB.

In the FOR or queue-owning region (QOR), the mirror determines when to switch to an open TCB for the first file control or temporary storage request received over an IPIC connection. The idea is for long-running mirrors to keep the mirror running on an open TCB.

A new option, MIRRORLIFE, is now added to the IPCONN attributes for function-shipped file control and temporary storage requests using an IPIC connection. MIRRORLIFE improves efficiency and provides performance benefits by specifying the lifetime of mirror tasks and the amount of time a session is held.

Threadsafe CICS DBCTL interface

CICS provides a CICS DBCTL interface to support **`CALL DLI`** and **`EXEC DLI`** command requests that are issued by applications running in a CICS region. In CICS TS V4.2, the CICS DBCTL interface can now use OTE, and CICS can run the CICS DBCTL TRUE on an L8 open TCB.

OTE is supported since the release of IMS 12 with PTFs for APAR PM31420 applied. During the connection process, IMS indicates to CICS that the OTE is supported, and therefore, CICS defines the CICS DBCTL TRUE as an OPENAPI TRUE. For IMS 11 and earlier, OTE is not supported. CICS runs the CICS DBCTL TRUE on the QR TCB, and the IMS code switches to an IMS thread TCB.

Running an application on an open TCB improves throughput and performance by reducing the use of the QR TCB. Threadsafe CICS applications that run on an L8 open TCB and issue **`CALL DLI`** or **`EXEC DLI`** commands can avoid two TCB switches for each call to IMS:

- For a nonthreadsafe application, the amount of switching is not reduced. Instead of switching from the QR TCB to an IMS thread TCB and back again for each IMS request, the application switches from QR to L8 and back again.
- For a threadsafe application, if it is running on the QR TCB, it switches to L8 and then stays on L8 when control is returned to the application.

For a threadsafe application that is already running on an L8 TCB, or for a **`CONCURRENCY(REQUIRED)`** application that is running on an L8 TCB, no TCB switching occurs for the IMS request.

When to use OPENAPI versus CICSAPI

To help you determine when to use OPENAPI versus CICSAPI, use this guide:

- Candidates for OPENAPI with THREADSAFE or REQUIRED have an OPENAPI behavior that is the same as in previous releases.
- Candidates for CICSAPI with REQUIRED are existing threadsafe applications that might be able to gain further throughput advantages by being defined as **`CONCURRENCY(REQUIRED)`**.
- Candidates for CICSAPI with THREADSAFE are threadsafe applications that issue many EXEC SQL requests followed by many EXEC CICS commands that are *not* threadsafe.

Part 2

Threadsafe implementation

This part highlights the implementation tasks and system programmer tasks. It provides a review of the application code and includes a migration scenario.

© Copyright IBM Corp. 2004, 2012. All rights reserved.

4

Threadsafe tasks

This chapter identifies the tasks to make a CICS DB2 application threadsafe, so that it can continue to run on an L8 task control block (TCB), after a DB2 command is issued. This chapter also identifies the tasks to make a DB2 application threadsafe. The same principles apply for an application calling one of the other OPENAPI task-related user exits (TRUEs), namely IMS, WebSphere MQ, and IP sockets for CICS.

In addition, this chapter explains how to use utilities supplied with CICS that can assist in identifying programs that contain commands that cause an application to switch to the QR TCB or wrongly use shared resources. In particular, this chapter highlights the CICS load module scanner (**DFHEISUP**) utility.

This chapter includes the following sections:

- Threadsafe enablement planning
- Load module scanner: The DFHEISUP utility

To learn how the IBM CICS Tools products can further assist in making applications threadsafe, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245.

© Copyright IBM Corp. 2004, 2012. All rights reserved.

4.1 Threadsafe enablement planning

Making your application threadsafe is more complex than just defining your application programs as threadsafe and then reaping the performance benefits. Without careful planning and a staged implementation, you can cause a performance degradation to your system or more seriously jeopardize the data integrity of your application.

This section highlights a high-level plan to safely convert from an existing nonthreadsafe environment to a fully functional threadsafe environment.

4.1.1 CICS Transaction Server upgrade and enablement path

To achieve your threadsafe goals, you must run CICS Transaction Server for z/OS (CICS TS) V2 or later and DB2 8 or later. Because of the open transaction environment (OTE) enhancements to the CICS DB2 attachment facility, you must run the correct release of DB2 to realize the benefits of threadsafe technology.

If you upgrade to CICS TS V2 or later, and change program definitions to **`CONCURRENCY(THREADSAFE)`** or to **`CONCURRENCY(REQUIRED)`** for CICS TS V4.2 without performing a review of your exits, you do more harm than good.

The order in which you must you upgrade your CICS and DB2 products depends on the method you use. You can approach your threadsafe implementation in a couple of ways, as shown in Table 4-1 and Table 4-2 on page 73. The method in Table 4-2 on page 73 is the preferred approach.

Table 4-1 Converting to CICS TS V2 or later first

Task	Description
1	Upgrade to CICS TS V2 or later and DB2 V8 or later.
2	Perform a threadsafe analysis of *all* exits that are defined to CICS.
3	Make adjustments or conversions to your exits.
4	Use CEDA to define your exit programs as threadsafe.
5	Analyze and convert your applications to be threadsafe.

Table 4-2 Reviewing your exits first

Task	Description
1	Perform a threadsafe analysis of all exits defined to CICS.
2	Make adjustments or conversions to your exits.
3	Use CEDA to define your exit programs as threadsafe.
4	Upgrade to CICS TS V2 or later and DB2 V8 or later.
5	Retest your exits.
6	Analyze and convert your applications to be threadsafe.

Review your exits first (Table 4-2), because of the way that CICS TS V2 and later handles exits in the threadsafe environment. After you run it on the new L8 TCBs, each call to a nonthreadsafe defined exit in the DB2 path forces a return to the QR TCB to run the exit. Then it returns to the L8 TCB, incurring extra TCB switches. For more information, see Figure 7-2 on page 153.

4.1.2 High-level threadsafe enablement path

By reviewing your exits first, the system exits can increase the number of TCB switches that you incur when they are defined as nonthreadsafe. You can still run a CICS TS V2 system or later without converting your exits because not all exits are in the DB2 path.

The CICS system exits are a critical point of analysis for ensuring that you receive the benefits of threadsafe applications. Therefore, you must convert *all exits* and define them as threadsafe as part of your upgrade to CICS TS V2 or later.

Apart from the exits that you wrote, you must contact the vendors of any OEM products that you installed. These vendors can advise whether their exits are already threadsafe or if you need to apply any maintenance to make them threadsafe. Additionally, you can find information about problems known to IBM by clicking the links on the following CICS web page and searching the related support information:

http://www.ibm.com/software/htp/cics/tserver/

The **DFHOSTAT** utility can produce a list of all your exits and indicates whether they are already defined as threadsafe. Some exits might be threadsafe if you installed a vendor package that installed the exits as threadsafe. For an example of the **DFHOSTAT** utility, see 6.4.3, “Running the DFH0STAT utility” on page 124.

Table 4-3 outlines a safe upgrade path that you can follow regardless of the CICS or DB2 release that you are currently running.

Table 4-3 High-level threadsafe enablement plan

Task	Description
1	Upgrade to DB2 V8 or later.
2	Install the prerequisite CICS PTFs.
3	Install the prerequisite DB2 PTFs.
4	Review the **FORCEQR SIT** parameter.
5	Address your exits: 1. Identify all your exits. 2. Contact vendors if necessary about their exits. 3. Review each exit for nonthreadsafe commands. 4. Review each exit for use of shared resources. 5. Make any coding adjustments and test. 6. Define them as threadsafe.
6	Review the following system parameters and make adjustments: ► **MAXOPENTCBS** ► **TCBLIMIT** ► **THREADLIMIT** ► **MXT**
7	Upgrade to CICS TS V2 or later.
8	Retest the exits in a threadsafe environment.
9	Create a threadsafe application review plan.
10	Review and identify your candidate applications.
11	Make necessary program changes to conform to threadsafe standards.
12	Define applications that passed your review or that are converted to threadsafe practices as THREADSAFE to CICS.

The next two chapters break down the steps in Table 4-3 into further detail, but first explain how to use the tools to analyze your applications.

In this book, for our analysis, we treat system exits and application code as simple applications. You review all your code for two nonthreadsafe practices:

- EXEC CICS commands that generate TCB switches to the QR TCB.
- EXEC CICS commands that reference shared resources:
  - **ADDRESS CWA**
  - **EXTRACT EXIT**
  - **GETMAIN SHARED**

You can use the load module scanner (**DFHEISUP** utility), supplied by CICS, for the following purposes:

- To scan code for occurrences of nonthreadsafe commands that generate a switchback to the QR TCB.
- To help you find occurrences of the three previously listed CICS commands.

In addition, the CICS Interdependency Analyzer tool provides a more comprehensive function. To use CICS IA and other CICS Tools, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245.

4.2 Load module scanner: The DFHEISUP utility

The **DFHEISUP** utility is provided by CICS so that you can search load modules for specific CICS API and SPI commands. It locates all the EXEC CICS commands in your load modules and then applies the filter to report on the commands that you specified.

The **DFHEISUP** utility returns one of the following types of report:

- A *summary report* with a list of modules that contain the commands specified by your filter and the number of these commands in each module. You can use this information as input to the detailed report to obtain more information about those modules.
- A *detailed report* that shows, for each module, the specific commands that it contains and the offset of the command. It also includes EDF information, if it is available.

CICS provides an example job DFHEILMS, in SDFHINST, that you can edit and use to run the load module scanner. For information about using this job for CICS TS V2 and later, see the *CICS Operations and Utilities Guide* in the CICS Transaction Server Information Center for your version.

Important: Users of CICS TS V2.2 must apply APAR PQ78531 before running the **DFHEISUP** utility. This APAR fixes storage problems that occur when running the **DFHEISUP** utility on large load libraries or large load library concatenations. The APAR fix is present at the base code level in later releases of CICS.

4.2.1 DFHEISUP filter tables

Two sample filter tables are provided for use in determining whether an application is threadsafe:

- DFHEIDTH
- DFHEIDNT

You can find these filter tables supplied by CICS in the SDFHSAMP library on your system.

The DFHEIDTH table

The DFHEIDTH table contains the following three commands that are a *threadsafe inhibitor*. That is, these commands *might* cause the program not to be threadsafe because they allow access to shared storage.

- **EXTRACT EXIT GASET**
- **GETMAIN SHARED**
- **ADDRESS CWA**

All of these commands return addresses of data areas that can be shared between programs. Multiple updates of the data areas pointed at by these addresses can occur by concurrently running tasks.

If your installation has an application standard that allows assembler data tables as a form of shared storage, consider amending DFHEIDTH to add the **LOAD *** command to find which applications load, and use this form of shared storage. By default, the **LOAD** command is not included as part of DFHEIDTH because it finds *too* many legitimate uses of **EXEC CICS LOAD** (for example, loading a read-only program into a read-only dynamic storage area (DSA)).

If any of these commands are identified as being used in any one application program, you *must* perform a more detailed analysis of the whole application. This analysis identifies how and when the addresses returned by these commands are used to access the underlying data. The address returned by one of these commands can be passed to another program that does none of the commands itself, but still modifies the data at the address passed to it. After you identify how the address is used, only then can you decide how to serialize access to the data.

The DFHEIDNT table

The DFHEIDNT filter table contains a list of all commands that cause a TCB switchback to the QR TCB. This table is provided by APAR PQ82603 for both CICS TS V2.2 and V2.3. The tables are provided at the base code level in CICS TS V3 or later.

Use of these commands does not prevent you from defining the program as threadsafe. However, they can prevent your application from achieving the performance benefits of allowing programs to stay on an open TCB following a DB2 call.

4.2.2 DFHEISUP summary mode

Running the load module scanner in summary mode produces two groups of information. Both groups are written to SYSPRINT DD:

- A summary of the whole load library (Figure 4-1). This summary shows the number of modules that were scanned, that are in the library, and that were not scanned, and the number of requested commands were found in the library.

```
LOAD LIBRARY STATISTICS
 ======================
 Total modules in library                               =     41
 Total modules Scanned                                  =     41
 Total CICS modules/tables not scanned                  =      0
 Total modules possibly containing requested commands =     19
```

Figure 4-1 Load library statistics

- A list of members in the library that contain any of the commands that were specified in the filter table (Figure 4-2). The list specifies the number of commands that are in the load module and the language in which the program was originally written.

```
SUMMARY LISTING OF CICSRS3.U.LOAD
=====================================
Module Name               Commands Found  Language
'CICSRS3.U.LOAD(DB2MANY)'              2  Assembler
'CICSRS3.U.LOAD(DB2ONCE)'              3  Assembler
'CICSRS3.U.LOAD(DB2PGMA)'              1  Assembler
'CICSRS3.U.LOAD(DB2PGMB)'              1  Assembler
'CICSRS3.U.LOAD(DB2PGMO)'              1  Assembler
'CICSRS3.U.LOAD(DB2PROGA)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG1)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG2)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG3)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG4)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG5)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG6)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG7)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG8)'             2  Assembler
'CICSRS3.U.LOAD(DB2PROG9)'             2  Assembler
'CICSRS3.U.LOAD(DB2SAMPL)'             1  Assembler
'CICSRS3.U.LOAD(FUNCSHIP)'             1  Assembler
'CICSRS3.U.LOAD(INITXIT)'              1  Assembler
'CICSRS3.U.LOAD(INITXIT2)'             1  Assembler
```

Figure 4-2 Module listing from summary report

The list of modules can also be optionally written to a file that is allocated to the DFHDTL DD statement by specifying the DETAILMODS parameter with the SUMMARY parm on the EXEC statement of the job step (Example 4-1).

Example 4-1 DFHEILMS summary run

```
//DFHSCNR  JOB (accounting information),CLASS=A,MSGCLASS=A
//DFHSCAN  EXEC PGM=DFHEISUP,PARM=('SUMMARY, DETAILMODS'),REGION=512M
//STEPLIB  DD DSN=&HLQ.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//* Filter table
//DFHFLTR  DD DSN=&HLQ.FILTER,DISP=SHR
//* Module list for input to detail run
//DFHDTL   DD DSN=&HLQ.MODLIST,DISP=(NEW,CATLG,DELETE),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),SPACE=(CYL,(1,1))
//DFHIN    DD DSN=&HLQ.SDFHLOAD,DISP=SHR
```

This file can then be fed into the detail run through the DFHLIST DD statement. Again the report is written to the SYSPRINT DD statement.

4.2.3 DFHEISUP detail mode

The load module scanner, when run in detail mode, writes a report to the SYSPRINT DD statement showing which commands are in each load module that is scanned. Example 4-2 shows the JCL to run the detail report.

Example 4-2 DFHEILMS detail run

```
//DFHSCNR  JOB (accounting information),CLASS=A,MSGCLASS=A
//DFHSCAN  EXEC PGM=DFHEISUP,PARM=('DETAIL'),REGION=512M
//STEPLIB  DD DSN=&HLQ.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//* Filter table
//DFHFLTR  DD DSN=&HLQ.FILTER,DISP=SHR
//* Module list for input to detail run
//DFHIN    DD DSN=&HLQ.SDFHLOAD,DISP=SHR
//* Module list from the summary run - DO NOT SPECIFY ALL with this
//DFHLIST  DD DSN=&HLQ.MODLIST,DISP=SHR
```

The detail run scans only those modules listed in the input file pointed to by DD DFHLIST unless you add ALL to the parm statement.

Figure 4-3 shows an example of the output from a detail run. Most of the entries were edited from the example to save space.

```
 CICS LOAD MODULE SCANNER UTILITY
  SCAN PERFORMED ON Thu May  6 16:18:04 2004 USING TABLE RSTABLE2.3

  DETAILED LISTING OF DD:DFHLIST
  =====================================

  Module Name        'CICSRS3.U.LOAD(DB2MANY)'
  Module Language    Assembler
  Offset/EDF         Command
  -----------------  ------------------------------------------
  00001962/no-edf    START TRANSID FROM LENGTH INTERVAL
  00001971/no-edf    SEND TEXT FROM LENGTH FREEKB TERMINAL

  Module Name        'CICSRS3.U.LOAD(DB2ONCE)'
  Module Language    Assembler
  Offset/EDF         Command
  -----------------  ------------------------------------------
  00001961/no-edf    INQUIRE CLASSCACHE PROFILE
  00001974/no-edf    INQUIRE JVM PROFILE
  00002000/no-edf    ASKTIME ABSTIME

  Module Name        'CICSRS3.U.LOAD(DB2PGMA)'
  Module Language    Assembler
  Offset/EDF         Command
  -----------------  ------------------------------------------
  00000840/no-edf    START TRANSID INTERVAL

Module Name        'CICSRS3.U.LOAD(INITXIT2)'
  Module Language    Assembler
  Offset/EDF         Command
  -----------------  ------------------------------------------
  00000716/no-edf    EXTRACT EXIT PROGRAM GASET GALENGTH

  Total possible commands located = 32

  LOAD LIBRARY STATISTICS
  =======================
  Total modules in library                            =     19
  Total modules Scanned                               =     19
  Total CICS modules/tables not scanned               =      0
  Total modules possibly containing requested commands =     19
```

Figure 4-3 Detail report from the DFHEISUP utility

4.2.4 DFHEISUP summary

For CICS TS V2.2, ensure that you apply APAR PQ78531 if you intend to scan large libraries of load modules in a single run (the APAR fix is present at the base code level in higher releases). This APAR prevents possible storage problems when running against load libraries with 80 or more load modules. The **DFHEISUP** utility is still a CPU-intensive program and takes longer to run against larger load libraries or load library concatenations.

The summary run is specified by **PARM=SUMMARY** on the PARM statement. Specifying **PARM='SUMMARY,DETAILMODS'** directs a copy of the load module list to the file pointed to by DFHDTL and writes this information to SYSPRINT. You can then use this file as input to the detail run.

The detail run is specified by **PARM=DETAIL** on the PARM statement. If you supply, as input to the detail run, the module list generated by the summary run, do not specify **PARM='DETAIL,ALL'**. This parameter overrides the list of modules in the file and scans the whole library again. If ALL is omitted, only those modules listed in the DFHLIST DD are scanned.

5

Application review

This chapter explains how to make a CICS DB2 application threadsafe, so that it can continue running on an L8 task control block (TCB) after a DB2 command is run. The same principles apply for an application that calls one of the other OPENAPI task-related user exits (TRUEs), such as WebSphere MQ and IP sockets for CICS.

This chapter addresses three different areas to investigate before defining an application as threadsafe:

- Use of nonthreadsafe local code
- Use of shared resources
- Use of nonthreadsafe CICS commands

The chapter concludes with an example of a COBOL program that uses file control commands to show how they run on an open TCB at CICS Transaction Server for z/OS (CICS TS) V3.2.

This chapter includes the following sections:

- Reviewing the application code
- Changing the program definitions

© Copyright IBM Corp. 2004, 2012. All rights reserved.

5.1 Reviewing the application code

Before you enable any application as threadsafe, you *must* review the application code for the two reasons.

First, application data integrity must be maintained. Before CICS TS V2.2, all user applications and exits ran on the quasi-reentrant (QR) TCB, which is a restricted or closed environment. CICS provided the serialization needed to ensure that application data integrity was never compromised. In this environment, programs can be sure that no more than one QR program can run at the same time.

Now, for applications that make calls to TRUEs that are enabled as OPENAPI or, for application programs that are defined as OPENAPI, two or more programs can run concurrently on different open TCBs and the QR TCB. Therefore, shared resources used by an application must be serialized to prevent any application integrity problems due to more than one program accessing the same resource at the same time.

The second reason for reviewing your application code is to ensure that, after CICS moves an application to an open TCB, it must remain there for as long as possible. CICS switches the application program back to the QR TCB to run CICS application programming interface (API) or system programming interface (SPI) commands that are nonthreadsafe. CICS must do this switch to maintain the integrity of, for example, the common system area (CSA) and other control blocks used by these commands.

5.1.1 Ensuring that the program logic is threadsafe

To ensure that the program logic is threadsafe, you must review the local code and the shared resources.

Checking the local code

The local language logic is the application code in between any CICS commands. This code must also be threadsafe. If you define a program to be threadsafe, but the application logic is not threadsafe, unpredictable results can occur that can compromise your data integrity.

For a program to be threadsafe, the program must be reentrant. Language Environment programs can be guaranteed reentrant by compiling with the RENT option, meaning that the compiler for the language generates fully reentrant (and therefore) threadsafe code. Language compilers before the introduction of Language Environment cannot be guaranteed to be reentrant. Therefore, programs that are compiled by using such compilers cannot be made threadsafe.

Assembler programs are probably the most commonplace where nonthreadsafe code can be generated, which can be achieved, for example, by storing variable data in a define constant (DC) in a CSECT. In this case, the program alters itself to store variable data and, therefore, creates a shared resource that can be updated by more than one transaction running the same program at the same time.

Testing for nonreentrant local code

The simplest way to check that the local code between EXEC CICS commands is reentrant is to link-edit the program with the RENT option. CICS then places any program linked with the RENT option into a read-only dynamic storage area (DSA). (This DSA is the read-only dynamic storage area (RDSA) for RMODE(24) programs and the extended read-only dynamic storage area (ERDSA) for RMODE(ANY) programs).

By default, the storage for these DSAs is allocated from read-only, key-0, protected storage. This storage protects any modules that are loaded into a read-only DSA from being modified by all programs except those running in key-0 or in a supervisor state. Therefore, if CICS is *not* initialized with **RENTPGM=NOPROTECT**, any attempt by a program to modify itself results in an ASRA abend. Test for nonreentrant native code in a preproduction environment where you can thoroughly test the application to identify any possible programs that are not reentrant.

Checking for shared resources

When identifying issues that can make an application nonthreadsafe, you must also analyze the use of shared resources by your applications. Shared resources are those storage areas that result from use of the following resources:

- Common work area (CWA)
- Shared getmains
- Global work areas (GWAs) for global user exits (GLUEs)
- Loaded assembler data tables

Using these resources does not imply that a program is not threadsafe. You must analyze the application to determine how these areas are then used by the application as a whole. In particular, if the shared area is updated at any point, then *all* access to the shared area must be serialized.

The DFHEISUP utility

CICS provides the **DFHEISUP** utility that you can use to scan load modules to identify the CICS commands associated with these shared areas. Use the load module scanner against the application load modules with the supplied DFHEIDTH filter table to identify all of the programs that contain any of the commands mentioned in this chapter. To learn more about this utility, see 4.2, “Load module scanner: The DFHEISUP utility” on page 75.

In addition to the `DFHEISUP` utility, by using CICS Interdependency Analyzer (CICS IA), you can scan for these CICS commands statically and at run time.

Important: If any of the CICS commands are identified as being used in any application program, you must perform a more detailed analysis of the whole application. In this analysis, you can identify how and when the addresses returned by these commands are used to access the underlying data.

The address returned by one of the CICS commands can be passed to another program that does none of the commands itself but still modifies the data at the address passed. Therefore, a module is not necessarily threadsafe just because the scanner utility does not report any of these commands as being present in a particular load module.

If you determine that the shared resource is *never* updated by any of your application programs, no further action is necessary for that shared resource. For example, a program list table (PLT) startup program might have initialized the resource and then read only the rest of the application.

Serializing access to the data

After analysis of your application determines that the shared data area is updated, decide how to serialize access to the data by using techniques such as the following examples:

- Compare and swap
- Enqueue/dequeue
- Accessing the shared storage only from QR programs

For details about these techniques, see 3.2, “Serialization techniques” on page 56.

5.1.2 Example showing the use of shared resources

The following example application demonstrates how use of shared resources can compromise data integrity if resources are not serialized.

Starter program CWAPROG

The example application consists of one starter program that initializes shared storage, which is CWA in this case, and then passes on the address of the CWA to five transactions. Each instance of these five transactions uses this address to access and update the data in the shared area. The five transactions are started 25 times each, and each transaction start the same TXNPROG program. Example 5-1 on page 87 shows the CWAPROG starter program.

Example 5-1 The CWAPROG starter program

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CWAPROG.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01  ws-queue                    pic x(08)
      value 'OUTPUTQ'.
 01  ws-ptr                      pointer.

 LINKAGE SECTION.
 01  common-work-area.
     03  cwa-counter             pic s9(8) comp.

 PROCEDURE DIVISION.
*    Delete the output TSQ - don't worry if it is not there
     EXEC CICS DELETEQ TS QUEUE(WS-QUEUE) NOHANDLE END-EXEC.

*    Access our shared storage area - this time the CWA
     EXEC CICS ADDRESS CWA(ADDRESS OF COMMON-WORK-AREA) END-EXEC.

*    Save address of our shared area so we can pass it on
     set ws-ptr to address of common-work-area.

*    Initialize the counter in our shared area
     move zero to cwa-counter.

*
*    Start our 5 transactions 25 times passing the address of the
*      CWA (which contains our counter) so that each transaction
*      can access it
*
     Perform 25 times
         EXEC CICS START TRANSID('TXN1') FROM(WS-PTR) END-EXEC

         EXEC CICS START TRANSID('TXN2') FROM(WS-PTR) END-EXEC

         EXEC CICS START TRANSID('TXN3') FROM(WS-PTR) END-EXEC

         EXEC CICS START TRANSID('TXN4') FROM(WS-PTR) END-EXEC

         EXEC CICS START TRANSID('TXN5') FROM(WS-PTR) END-EXEC
     End-Perform.

     EXEC CICS RETURN
END-EXEC.
```

The TXNPROG program

The TXNPROG program runs each of the following transactions:

- Retrieves the address of the shared storage passed to it
- Makes an EXEC SQL call that causes a switch to an L8 TCB
- Increments a copy of the counter value in the shared storage by one
- Performs processing
- Writes the new counter value back to the shared storage
- Writes the result to a temporary storage queue

Example 5-2 shows the program.

Example 5-2 TXNPROG

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TXNPROG.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 ws-enq-queue                     pic x(08) value ‘ENQUEUE’.
01 ws-enqueue-yes-no                pic x(03) value ‘NO’.
   88 enqueue-yes                             value ‘YES’.

01 ws-ptr                           pointer.
01 ws-counter2                      pic s9(8) comp.
01 ws-count                         pic s9(8) comp.
01 ws-queue                         pic x(08)
       VALUE ‘OUTPUTQ’.
01 WS-MSG.
   03 WS-TXN                        pic x(05).
   03 filler                        pic x(17)
       value “Counter value :- “.
   03 ws-counter                    pic 9(8).

01 ws-cwa-ptr                       usage is pointer.

    EXEC SQL
          DECLARE DSN8710.EMP TABLE (
          EMPNO                      CHAR(6),
          FIRSTNME                   CHAR(12),
          MIDINIT                    CHAR(1),
          LASTNAME                   CHAR(15),
          WORKDEPT                   CHAR(3),
          PHONENO                    CHAR(4),
          HIREDATE                   DATE,
          JOB                        CHAR(8),
          EDLEVEL                    SMALLINT,
          SEX                        CHAR(1),
```

```
          BIRTHDATE                   DATE,
          SALARY                      DECIMAL,
          BONUS                       DECIMAL,
          COMM                        DECIMAL )
    END-EXEC.

    EXEC SQL INCLUDE SQLCA END-EXEC.

LINKAGE SECTION.
01  SHARED-AREA.
    03  SHARED-COUNTER                  PIC S9(8) COMP.

PROCEDURE DIVISION.
    EXEC CICS READQ TS
        QUEUE(WS-ENQ-QUEUE)
        ITEM(1)
        INTO(WS-ENQUEUE-YES-NO)
        NOHANDLE
     END-EXEC.
     MOVE EIBTRNID TO WS-TXN.

*    get the address of the shared area which has been passed
     EXEC CICS RETRIEVE INTO(WS-PTR) END-EXEC.

*    map our linkage section to the address of the shared area
     Set address of shared-area to ws-ptr.
*
*    Make DB2 Call which will transfer to the L8
*
     EXEC SQL
        SELECT count(*)
        INTO :ws-count FROM DSN8710.EMP
               WHERE EMPNO = "000990"
     END-EXEC.

     if enqueue-yes
*        enqueue before we change the shared storage
         EXEC CICS
              ENQ RESOURCE(shared-area)
         END-EXEC
     end-if.

*    read the value in shared storage.
     move shared-counter to ws-counter.

*    ... and change its value
     Add 1 to ws-counter.

*    ** Do some important processing **
```

```
     move ws-counter to ws-counter2.
     Perform 100000 Times
       add 2 to ws-counter2
       subtract 1 from ws-counter2
     End-Perform.
*    **********************************

*    update the shared storage with our new value
     Move ws-counter to shared-counter.

     if enqueue-yes
*        remove the enqueue now we have finished updating
*          the shared storage
         EXEC CICS
             DEQ RESOURCE(shared-area)
         END-EXEC
     end-if.

*    output the results ......
     EXEC CICS
       WRITEQ TS MAIN QUEUE(WS-QUEUE) FROM(WS-MSG)
     END-EXEC.

     EXEC CICS RETURN END-EXEC.
```

Results when run as quasi-reentrant

When the TXNPROG program is defined as quasi-reentrant, each occurrence of the transaction is serialized by CICS. Only one occurrence of the program can be running at any one time and always on the QR TCB. Therefore, the results are as expected, which is that each program processes a unique counter, as shown by the output in Figure 5-1.

```
 CEBR  TSQ OUTPUTQ          SYSID PJA6 REC     1 OF    25    COL     1 OF 30
 ENTER COMMAND ===>
      **************************  TOP OF QUEUE
00001 TXN1 Counter value :- 00000001
00002 TXN2 Counter value :- 00000002
00003 TXN1 Counter value :- 00000003
00004 TXN5 Counter value :- 00000004
00005 TXN4 Counter value :- 00000005
00006 TXN3 Counter value :- 00000006
00007 TXN2 Counter value :- 00000007
00008 TXN3 Counter value :- 00000008
00009 TXN4 Counter value :- 00000009
00010 TXN5 Counter value :- 00000010
00011 TXN1 Counter value :- 00000011
00012 TXN2 Counter value :- 00000012
00013 TXN3 Counter value :- 00000013
00014 TXN4 Counter value :- 00000014
00015 TXN5 Counter value :- 00000015
00016 TXN1 Counter value :- 00000016

PF1 : HELP               PF2 : SWITCH HEX/CHAR     PF3 : TERMINATE BROWSE
PF4 : VIEW TOP           PF5 : VIEW BOTTOM         PF6 : REPEAT LAST FIND
PF7 : SCROLL BACK HALF    PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL    PF11: SCROLL FORWARD FULL PF12: UNDEFINED
```

Figure 5-1 Results when run as quasi-reentrant

Results when defined as threadsafe without enqueue

If you change the definition of the TXNPROG program to be threadsafe *without* ensuring that the update of the CWA is serialized, the output looks different. Each instance of the TXNPROG program remains on an L8 TCB after completing the EXEC SQL call. The result is that several instances of the TXNPROG program are running concurrently on multiple TCBs. Each instance of the program cannot rely on the value of the counter in the CWA because access to it is not serialized. Therefore, the result is a scenario with the following pattern:

- TXN1 reads counter (0).
- TXN1 increments counter (1).
- TXN2 reads counter (0).
- TXN1 writes incremented value (1).

- TXN2 writes incremented value (1).
- TXN3 reads counter (1).

Figure 5-2 shows the output written to the temporary storage queue.

```
CEBR  TSQ OUTPUTQ          SYSID PJA6 REC     1 OF    25    COL     1 OF    30
 ENTER COMMAND ===>
      **************************  TOP OF QUEUE  ***************************
00001 TXN4 Counter value :- 00000001
00002 TXN1 Counter value :- 00000001
00003 TXN3 Counter value :- 00000001
00004 TXN2 Counter value :- 00000002
00005 TXN5 Counter value :- 00000002
00006 TXN1 Counter value :- 00000002
00007 TXN2 Counter value :- 00000003
00008 TXN3 Counter value :- 00000003
00009 TXN5 Counter value :- 00000004
00010 TXN1 Counter value :- 00000004
00011 TXN4 Counter value :- 00000004
00012 TXN2 Counter value :- 00000005
00013 TXN3 Counter value :- 00000005
00014 TXN4 Counter value :- 00000005
00015 TXN1 Counter value :- 00000006
00016 TXN5 Counter value :- 00000006

PF1 : HELP               PF2 : SWITCH HEX/CHAR     PF3 : TERMINATE BROWSE
PF4 : VIEW TOP           PF5 : VIEW BOTTOM         PF6 : REPEAT LAST FIND
PF7 : SCROLL BACK HALF    PF8 : SCROLL FORWARD HALF PF9 : UNDEFINED
PF10: SCROLL BACK FULL    PF11: SCROLL FORWARD FULL PF12: UNDEFINED
```

Figure 5-2   Results when running the program as threadsafe without enqueues

Figure 5-2 shows that the data is compromised because each transaction is attempting to concurrently update the counter value.

The solution to this problem is to add an **ENQ** and **DEQ** command around the code that reads and then updates the counter value. In this example, we enqueued upon the address of the shared area, which is currently pointed to by the linkage section item *shared-area*. Adding the enqueue and dequeue commands causes the results to return to the results when the program was defined as quasi-reentrant, as shown in Figure 5-1 on page 91.

The example program shown in Figure 5-3 uses an IF statement to enclose the **ENQUEUE** and **DEQUEUE** commands so that they can be switched on and off easily without recompiling the program.

```
 if enqueue-yes
* enqueue before we change the shared storage
    EXEC CICS
       ENQ RESOURCE(shared-area)
    END-EXEC
 end-if.
```

Figure 5-3 Enqueue statement

To switch on the enqueue dynamically, write the word `YES` to a temporary storage queue called ENQUEUE.

Example summary

The example of using shared resources shows how a program uses the CWA to store a counter value. In a QR scenario, access to this counter value is serialized by CICS, and the counter value returned is always unique and the next in the series. However, in a threadsafe scenario, the application must do the serialization. Otherwise, with concurrent tasks running on separate TCBs, the counter returned can no longer be relied upon to be unique.

The **ENQUEUE** and **DEQUEUE** commands enclose only the minimum number of program statements that are necessary to ensure that the resource is not updated before this program is ready. In this example, we can make the enqueue-to-dequeue path shorter by updating the shared resource and dequeuing before we do the *important processing* section of code.

In this example, the CWA is used as the shared resource. To change the example to use any of the other shared resources listed in "Checking for shared resources" on page 85, replace the **ADDRESS CWA** command in Example 5-1 on page 87 with one of the other commands.

Important: If several programs (as in the example) access the shared resource, they must *all* use the *same* serialization technique to serialize access to the shared resource. In this example, if another program was accessing the CWA, it must also use ENQUEUE and DEQUEUE on the same resource (in this case, the address of the CWA).

5.1.3 Assembler data tables

A technique that was often used in the past is to load a data-only assembler program that contains only DC entries in a CSECT. If the load is done with the HOLD option, the empty assembler program remains in storage and, therefore, becomes a shared resource that can be updated concurrently from several programs. For example, you can assemble and link-edit the program in Example 5-3 into a library on the DFHRPL concatenation.

Example 5-3 Assembler data table

```
TABLE    CSECT
FILLER1  DC    CL16'COUNTER VALUE >>'
COUNTER  DC    F'99'
FILLER2  DC    CL16'<< COUNTER VALUE'
         END
```

Then you can load the program into storage by any program, with an **EXEC CICS LOAD** command. This command can map the data in the table onto a linkage section structure, such as in Example 5-4.

Example 5-4 Linkage section

```
LINKAGE SECTION.
01  TABLE-AREA.
    03  filler                pic x(16).
    03  LS-COUNTER            PIC S9(8) COMP.
    03  filler                pic x(16).
```

The address of this area can be passed on and used in the same that way the address of the CWA is used in the previous example. This technique does not work if the table is linked with the RENT option and CICS is started with **RENTPGM=PROTECT**.

5.1.4 Ensuring usage of only threadsafe CICS commands

After a program is switched over to an open TCB, minimize the number of times that CICS switches back to the QR TCB. By using this method, applications can reap the benefits of running multiple tasks concurrently across multiple TCBs.

The CICS API and SPI commands that are not threadsafe are the main inhibitor to staying on an open TCB. When a CICS API or SPI command is run, CICS runs the code that can update any number of CICS control blocks, such as the CSA.

The integrity of CICS can be compromised and unpredictable results can occur in the following situations:

- CICS was not changed to serialize access to these control blocks.
- Multiple tasks are allowed to run that cause these control blocks to be updated concurrently from several tasks.

To ensure that CICS is not compromised, CICS automatically switches back to the QR TCB when it is about to run any API or SPI command that it knows to be nonthreadsafe. For a current list of commands that *are* threadsafe, see the *CICS Application Programming Reference* and the *CICS System Programming Reference* in the CICS Transaction Server V4.2 Information Center at:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp

Nonthreadsafe commands: If a command is *not* listed in the appendixes of either guide, it is *not* threadsafe and it *will* cause a switch to the QR TCB.

These commands can be identified, again, by using the load module scanner, the **DFHEISUP** utility with filter table DFHEIDNT, or CICS IA.

Penalty for using nonthreadsafe commands: Using a CICS command that is nonthreadsafe does not prevent the program from being defined as threadsafe and does not compromise data integrity. However, including these commands causes CICS to switch back to the QR TCB each time a nonthreadsafe command is encountered during an open TCB. Therefore, using nonthreadsafe commands has a *performance* penalty, not an *integrity* penalty.

In terms of performance, the worst case scenario for a DB2 application program is where many EXEC SQL calls are interspersed with nonthreadsafe EXEC CICS commands as shown in Example 5-5 on page 96.

Example 5-5 Nonthreadsafe commands causing TCB switches

```
EXEC CICS
EXEC SQL
EXEC CICS <nonthreadsafe>
EXEC SQL
EXEC CICS <threadsafe>
EXEC CICS <nonthreadsafe>
EXEC SQL
EXEC CICS <nonthreadsafe>
EXEC SQL
EXEC CICS <nonthreadsafe
EXEC SQL

RETURN
```

Example 5-5 shows a TCB switch for each DB2 request and then a switch back to the QR TCB when a nonthreadsafe EXEC CICS command follows an EXEC SQL call. This method does not provide optimal threadsafe performance that applications can deliver due to the number of nonthreadsafe CICS commands and their distribution throughout the program.

Example 5-5 can be restructured in such a way that most, or all, of the nonthreadsafe commands can be removed or moved to the start of the program before any DB2 request is made. This restructure removes the excessive number of mode switches and delivers the performance benefits that we want, as shown in Example 5-6.

Example 5-6 Nonthreadsafe commands that are moved or deleted

```
EXEC CICS
EXEC CICS <nonthreadsafe>
EXEC CICS <nonthreadsafe>
EXEC CICS <nonthreadsafe>

EXEC SQL
EXEC SQL
EXEC CICS <threadsafe>
EXEC SQL
EXEC SQL

EXEC SQL

RETURN
```

This simple reorganization is not possible for every program. After the commands are identified as being present in the program, only then can you assess what changes, if any, can be made.

5.2 Changing the program definitions

After the applications are changed or verified to be threadsafe, to keep the application on the open TCB, you change the definition of all of the programs concerned to be threadsafe. You can change the program definitions in the following ways:

- Change the resource definition online (RDO) definition of the program.
- Modify the autoinstall exit to install the program as threadsafe.
- Use the Language Environment variable CICSVAR, which is explained in 5.2.2, “CICS environment variable CICSVAR” on page 98.

5.2.1 Changing the RDO definition

Figure 5-4 shows the RDO definition.

```
OBJECT CHARACTERISTICS                                      CICS RELEASE = 0670
  CEDA  View PROGram( DB2MANY  )
   PROGram        : DB2MANY
   Group          : THDSAFE
   DEscription    :
   Language       :                  COBol | Assembler | Le370 | C | Pli
   RELoad         : No                No | Yes
   RESident       : No                No | Yes
   USAge          : Normal            Normal | Transient
   USElpacopy     : No                No | Yes
   Status         : Enabled           Enabled | Disabled
   RS1            : 00                0-24 | Public
   CEdf           : Yes               Yes | No
   DAtalocation   : Any               Below | Any
   EXECKey        : User              User | Cics
   COncurrency    : Threadsafe        Quasirent | Threadsafe | Required
   Api            : Cicsapi           Cicsapi | Openapi
  REMOTE ATTRIBUTES
   DYnamic        : No                No | Yes
 + REMOTESystem   :

                                                      SYSID=PJA7 APPLID=SCSCPJA7

 PF 1 HELP 2 COM 3 END        6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 5-4 RDO program definition using CEDA

5.2.2 CICS environment variable CICSVAR

Before CICS TS V3.1, changing the definitions of autoinstalled programs that are now threadsafe required the introduction of logic into the autoinstall program. This logic helped CICS TS to detect which programs to autoinstall as threadsafe and which programs were not threadsafe.

CICS TS V3.1 introduced the CICSVAR environment variable so that the CONCURRENCY and API program attributes can be associated with the application program by using the ENVAR runtime option. You might use the CICSVAR environment variable in a CEEDOPT CSECT to set an installation default. Therefore, the variable is most useful when set in the following ways:

- In a CEEUOPT CSECT link-edited with an individual program
- Through a #pragma statement in the source of a C or C++ program
- Through a PLIXOPT statement in a PL/I program

For example, when a program is coded to threadsafe standards, you can define it without changing a PROGRAM resource definition. Alternatively, the program can adhere to an installation-defined naming standard so that a program autoinstall exit can install it with the correct attributes.

You can use CICSVAR for the following programs that are compiled by using a Language Environment conforming compiler:

- Language Environment conforming assembler
- PLI
- COBOL
- C and C++ programs (compiled with or without the XPLINK option)

You cannot use CICSVAR for assembler programs that are not Language Environment conforming or for Java programs.

> **PROGRAM resource definition:** Use of CICSVAR overrides the settings on a PROGRAM resource definition installed by using standard RDO interfaces or program autoinstall.

Until a program is run the first time, an `INQUIRE PROGRAM` command shows the keyword settings from the program definition. After the application runs one time, an INQUIRE PROGRAM command shows the settings with any CICSVAR overrides applied.

5.2.3 CICSVAR values

The following values for CICSVAR result in the values shown for CONCURRENCY and API:

CICSVAR=QUASIRENT Results in a program with the QUASIRENT and CICSAPI attributes.

CICSVAR=THREADSAFE Results in a program with the THREADSAFE and CICSAPI attributes.

CICSVAR=OPENAPI Results in a program with the THREADSAFE and OPENAPI attributes.

5.2.4 Coding ENVAR

You can code the ENVAR runtime option for different programming environments. This section shows the following examples:

- ENVAR CSECT example
- ENVAR pragma runopts example
- ENVAR PLIXOPT example

ENVAR CSECT example

The following example shows the ENVAR runtime option coded in a CEEUOPT CSECT:

```
CEEUOPT CSECT
CEEUOPT AMODE ANY
CEEUOPT RMODE ANY
        CEEXOPT ENVAR=('CICSVAR=THREADSAFE')
        END
```

This example can be assembled and link-edited into a load module. Then the CEEUOPT load module can be link-edited with any language program that is supported by Language Environment, as mentioned previously.

ENVAR pragma runopts example

For C and C++ programs, you add the following statement at the start of the program source before any other C statements:

```
#pragma runopts(ENVAR(CICSVAR=THREADSAFE))
```

ENVAR PLIXOPT example

For PL/I programs, you add the following statement after the PL/I MAIN procedure statement:

```
DCL PLIXOPT CHAR(25) VAR STATIC EXTERNAL INIT('ENVAR(CICSVAR=THREADSAFE)');
```

5.2.5 Example of file control application

CICS TS V3.2, released in June 2007, expanded the number of API commands that were made threadsafe to include the file control commands. This way, applications that currently are not good candidates to be made threadsafe can be converted and enabled as threadsafe.

The following example is of a program that browses through a file. It demonstrates that a file control API call will remain on an OPEN TCB, where in previous releases of CICS, it switched back the QR TCB to run the command.

The program in Figure 5-5 on page 101 browses through the entire FILEA file that is supplied by CICS.

```
Identification Division.
Program-ID. KSDSPR01.
Environment Division.
Data Division.
Working-storage Section.

01  ws-file                  pic x(8) value 'FILEA'.

01  ws-record                pic x(80) value low-values.
01  ws-rid                   pic x(06) value low-values.
01  ws-browse-rid            pic x(06) value low-values.

01  ws-resp                  pic s9(8) comp value zero.
01  ws-resp2                 pic s9(8) comp value zero.


Procedure Division.

    exec cics
      startbr file(ws-file)
      ridfld(ws-browse-rid)
      resp(ws-resp)
      resp2(ws-resp2)
    end-exec.

    perform until ws-resp not = dfhresp(normal)
       initialize ws-record
       exec cics
         readnext file(ws-file)
             into(ws-record)
             ridfld(ws-rid)
             resp(ws-resp)
             resp2(ws-resp2)
       end-exec
    end-perform.

    exec cics
       endbr file(ws-file)
    end-exec.

    exec cics return end-exec.
```

Figure 5-5 Example file control program

This program was defined as OPENAPI. Therefore, as soon as it begins, CICS switches the task to an open TCB, where it remains until the end of the task or until a nonthreadsafe CICS command is encountered. In this case, an L8 TCB (L8005) is used, which is in the trace snippet in Figure 5-6 that shows trace entries for *one* of the **READNEXT** commands.

```
00061 L8005 AP 00E1 EIP   ENTRY READNEXT                           0004,266A9900 ..r.,0800060E ....          =000898=
00061 L8005 AP E110 EISR  ENTRY TRACE_ENTRY     266A99F0                                                     =000899=
00061 L8005 AP E160 EXEC  ENTRY READNEXT     'FILEA   ' AT X'266AB040',AT X'266AB048',80 AT X'266AAD18',AT X'266AB
                                                                                                             =000900=
00061 L8005 AP E111 EISR  EXIT  TRACE_ENTRY/OK                                                               =000901=
00061 L8005 AP 04F0 EIFC  ENTRY PROCESS_EXEC_ARGUMENTS 266A99F0,0005C308                                     =000902=
00061 L8005 AP 04E0 FCFR  ENTRY READ_NEXT_INTO  FILEA,266AB048,50,00000000,266AB098,0,FCT_VALUE,KEY,NO,NO     =000903=
00061 L8005 AP 04B0 FCVS  ENTRY READ_NEXT_INTO  FILEA,266AB048,50,00000000,26F393B0,266AB098,26DA2100,0,FCT_VALUE,KEY
                                                                                                             =000904=
00061 L8005 AP 0492 FCVR  EVENT ISSUE_VSAM_RPL_REQUEST GET,SEQ ASY,000000000000                              =000905=
00061 L8005 AP 0493 FCVR  EVENT RETURN_FROM_VSAM   0000,F0F0F0F1F0F0                                         =000906=
00061 L8005 AP 04B1 FCVS  EXIT  READ_NEXT_INTO/OK  50,50,6,00000000,,LENGTH_OK,NO,NO                         =000907=
00061 L8005 AP 04E1 FCFR  EXIT  READ_NEXT_INTO/OK  50,50,6,00000000,,LENGTH_OK,NO,NO                         =000908=
00061 L8005 AP 04E2 FCFR  EXIT  FRAB_FLAB_AND_FRTE                                                           =000909=
00061 L8005 AP 04F1 EIFC  EXIT  PROCESS_EXEC_ARGUMENTS/OK                                                    =000910=
00061 L8005 AP E110 EISR  ENTRY TRACE_EXIT         266A99F0                                                  =000911=
00061 L8005 AP E161 EXEC  EXIT  READNEXT     'FILEA ' AT X'266AB040',' 000100S. D. BORMAN    SURREY, ENGLAND  =000912=
00061 L8005 AP E111 EISR  EXIT  TRACE_EXIT/OK                                                                =000913=
00061 L8005 AP 00E1 EIP   EXIT  READNEXT    OK                         00F4,00000000 ....,0000060E ....      =000914=
```

Figure 5-6   File control trace snippet

6

System programmer tasks

The CICS System Programmer normally performs the tasks to implement threadsafe applications as described in this chapter. Among these tasks, CICS Transaction Server for z/OS (CICS TS) V4.2 provides significant enhancements for the open transaction environment (OTE) that are also addressed in this chapter.

This chapter includes the following sections:

- Role of the system programmer
- Understanding threadsafe operation
- Analyzing the CICS regions
- Providing a threadsafe CICS operating environment
- Making your exits threadsafe
- Nonthreadsafe data integrity example
- Coordinating and driving individual application conversions
- Post-conversion monitoring
- Summary

© Copyright IBM Corp. 2004, 2012. All rights reserved.

6.1 Role of the system programmer

The system programmer does not make an application threadsafe, but rather prepares and makes the environment threadsafe so that it can efficiently run the customer applications. Additionally, the system programmer can coordinate and guide the conversion of the applications.

The system programmer might perform the following actions:

- Analyze the CICS regions:
  - Ensure that software prerequisites are in place.
  - Ensure that CICS regions have sufficient processor capacity to run tasks concurrently on multiple task control blocks (TCBs).
- Provide a threadsafe CICS operating environment.
- Coordinate and drive individual application conversions.
- Monitor and tune the CICS regions to ensure that they are efficiently using the open TCBs.

6.2 Understanding threadsafe operation

Before you start analyzing and preparing your CICS regions, you must understand a few of concepts about why some of the conversions are necessary.

6.2.1 Threadsafe performance issues

To gain the performance benefit of threadsafe applications, you must achieve the following objectives:

- Eliminate TCB switches between the quasi-reentrant (QR) and open TCBs.
- Improve the throughput of your CICS workload.

Threadsafe applications in previous versions of CICS Transaction Server

In CICS TS V2, a program begins running on the QR TCB, and when a DB2 call is encountered, the program is swapped over to run on an open TCB. If the program is defined to CICS as `CONCURRENCY(THREADSAFE)`, it can continue to run all additional instructions on the open TCB until program termination or until a nonthreadsafe command or exit is encountered.

By default, this behavior is also true in CICS TS V3. The API attribute of a program definition that was introduced in CICS TS V3.1 defaults to a value of CICSAPI. The CICSAPI value means that the program uses the traditional CICS programming interfaces. CICSAPI mirrors the behavior of a threadsafe application in CICS TS V2, meaning that CICS TS V3 CICSAPI applications use open TCBs in the same way as in CICS TS V2.

> **API(OPENAPI):** In CICS TS V3, a threadsafe program can be defined as API(OPENAPI), in which case it is switched to run under an L8 or L9 TCB *during its initialization*. The open TCB mode that it runs under depends on the EXECKEY parameter of the program. An OPENAPI program continues to run its instructions under its L8 or L9 TCB *until program termination*. Any calls to exits or nonthreadsafe commands requiring TCB switches are handled by CICS. Upon completion, the OPENAPI program receives control back under its open TCB again. Therefore, OPENAPI programs have more extensive threadsafe zones than CICSAPI programs.

Concurrency enhancements in CICS Transaction Server V4.2

CICS TS V4.2 has an additional option that can be specified on the **`CONCURRENCY`** parameter. You can define an application program as **`CONCURRENCY(REQUIRED)`**, which means that from the start of the program, it always runs on an open TCB, which is true for CICSAPI and OPENAPI programs.

For CICS application programs that are logically threadsafe, use **`CONCURRENCY(THREADSAFE`** or **`REQUIRED)`**.

> **Open TCB type:** The type of open TCB used depends upon the API setting. By defining your program with **`CONCURRENCY(REQUIRED)`**, CICSAPI programs can start on an L8 open TCB without matching the TCB with the execution key of the program. This setting is suitable for programs that access resource managers, such as DB2, IMS, and WebSphere MQ, which also require an L8 TCB.
>
> For OPENAPI programs, CICS uses an L9 TCB if **`EXECKEY(USER)`** is set and uses an L8 TCB if **`EXECKEY(CICS)`** is set.

From a performance perspective, run eligible programs by using the **`CONCURRENCY(REQUIRED)`** option. Enabling programs to start on an open TCB reduces contention for resources on the CICS QR TCB and reduces TCB switching. The new REQUIRED setting offers the following advantages:

- If CICS switches to the QR TCB to run a CICS command, it switches back to the open TCB before handing control back to the application program.
- You can reduce QR TCB delays. If your CICS regions show QR TCB delays (> 100 ms during peak time), you can define your programs using options `REQUIRED` or `THREADSAFE` on the **`CONCURRENCY`** parameter.

Originally, only the CICS DB2 task-related user exist (TRUE) made productive use of OTE. When reviewing the use of L8 TCBs and TCB switching, it was reasonable to discuss TRUE in terms of CICS DB2 applications. Since then, CICS TS V3.2 provides an OTE-enabled TRUE for WebSphere MQ (WMQ). Also, the IP CICS Sockets for z/OS Communications Server were written to use OTE if enabled for this purpose. In addition to these enhancements, CICS TS V3 can define applications as OPENAPI programs to run under their own L8 or L9 TCBs.

A major enhancement to threadsafe support in CICS TS V3.2 is making the CICS file control API threadsafe for applications. That is, you must also review the path for EXEC CICS file control commands to ensure that this path does not result in unwanted TCB switching activity.

With CICS TS V4.2, the CICS system programmer must take additional considerations into account. The CICS system programmer can enable CICS and IMS applications to use the OTE. CICS must be connected to IMS 12 or later to use the OTE.

Another major enhancement to threadsafe support in CICS TS V4.2 is the new design of DFHMIRS. DFHMIRS is now defined as a threadsafe program. For IP interconnectivity (IPIC) connections only, CICS runs DFHMIRS on an L8 open TCB whenever possible. For other connection types, CICS does not run DFHMIRS on an open TCB. Running remote function shipping requests on an open TCB offers a performance benefit. Therefore, systems programmer must prove whether existing non-IPIC connection can be replaced by IPIC connections to run DFHMIRS on an open TCB.

Function-shipped requests on open TCB: Only the following requests that are function-shipped over IPIC run on an open TCB:

- File control
- Temporary storage
- Distributed program link (if the target program is defined as threadsafe and the mirror already on an open TCB)

The use of open TCBs within CICS has grown. In regard to preparing CICS for OTE, the objectives of the system programmer role are still described in terms of calling DB2. The reason is that the same basic principles of serialization and data integrity apply, regardless of the reason why open TCBs are being used.

An important objective in reviewing your application or exit programs is to ensure that, if a program is running on an L8 TCB, it stays there until all DB2, WMQ, IMS, or IP CICS sockets work is completed. Another objective is to ensure that programs defined as OPENAPI in CICS TS V3 or later avoid TCB switching if possible. Minimizing TCB switches is a key performance goal for threadsafe implementation.

Consider the general case for threadsafe code logic, where a threadsafe CICS application program issues calls to an OTE-enabled TRUE such as DB2, WMQ, IMS, or IP sockets. For simplicity, assume that the application program is defined as CICSAPI in the CICS TS V4.2 environment, as illustrated in Figure 6-1. After the program starts to run on the L8 TCB, it is in the threadsafe zone. You must ensure that it is not moved off the L8 TCB by running a nonthreadsafe command or exit.

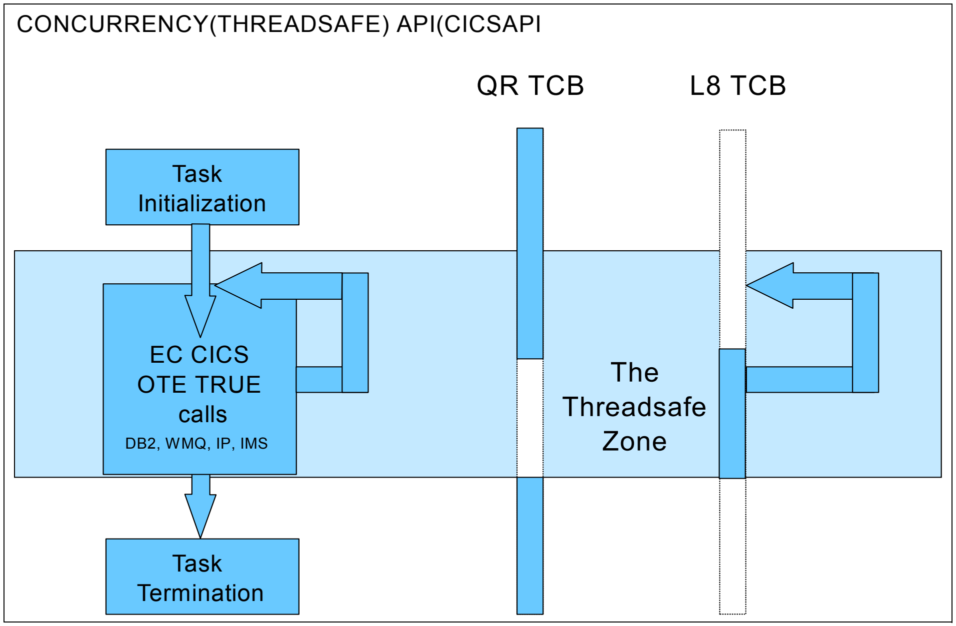


Figure 6-1 CICSAPI CONCURRENCY(THREADSAFE) program on an L8 TCB in the threadsafe zone

Regardless of how your CICSAPI program is coded, when you define the program as **`CONCURRENCY(THREADSAFE)`**, it always starts on the QR TCB and finishes on the QR TCB. Therefore, ideally you can place all your nonthreadsafe EXEC CICS commands at the beginning and end of your application program.

Figure 6-1 illustrates that a threadsafe program starts on the QR TCB and stays there until the first call to an OTE TRUE is issued. In CICS TS V4.2, support for threadsafe programs is further enhanced as shown in Figure 6-2 on page 108. For programs defined with **`CONCURRENCY(REQUIRED)`**, CICS starts the application on an open L8 TCB. CICSAPI programs can run with both **`EXECKEY(USER)`** and **`EXECKEY(CICS)`** on an L8 TCB. For OPENAPI programs, CICS uses an L9 TCB if **`EXECKEY(USER)`** is set and uses an L8 TCB if **`EXECKEY(CICS)`** is set.

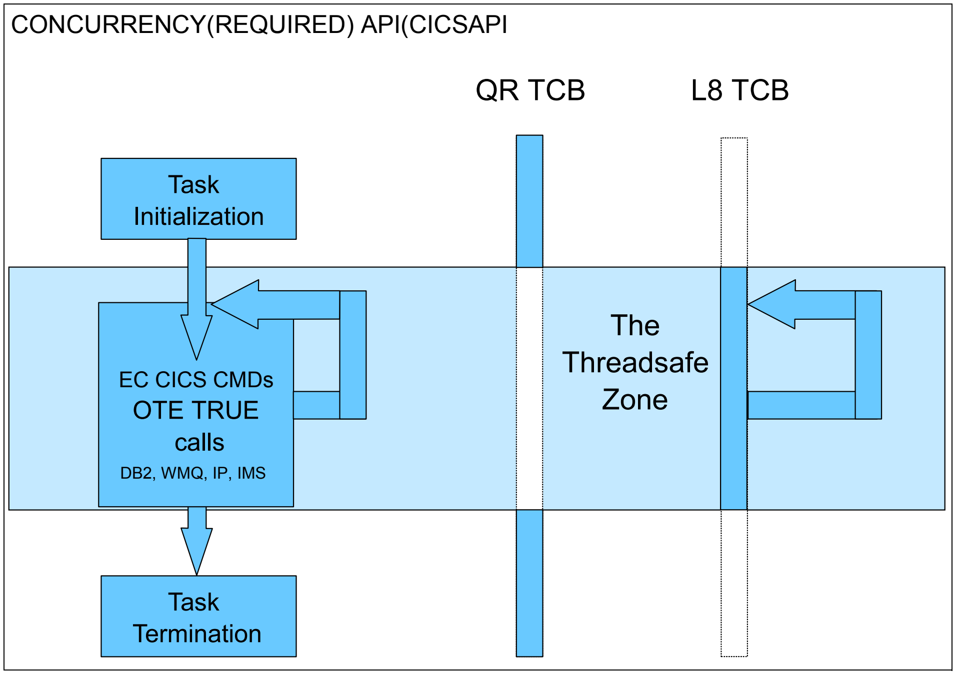


Figure 6-2 CICSAPI CONCURRENCY(REQUIRED) program on an L8 TCB in the threadsafe zone

The system programmer has a goal to keep the application programs running in the threadsafe zone. TCB switches must not be generated to the QR TCB in system exits, such as global user exits (GLUEs) or user-replaceable module (URMs), DB2 dynamic plan exits, or the CICS WMQ API crossing exit CSQCAPX.

You can now compare this situation with the case in which an OPENAPI CICS TS V4.2 application program is defined with **EXECKEY(USER)**. For OPENAPI programs, the key of the TCB must match the EXECKEY setting. The program issues various threadsafe and nonthreadsafe EXEC CICS commands with a call to WebSphere MQ (for example, an MQGET) and an EXEC SQL call to DB2. Do *not* use this combination, because it results in additional TCB switching between L9 and L8 TCBs. For this reason, Figure 6-3 on page 109 is provided to help you visualize this type of scenario.

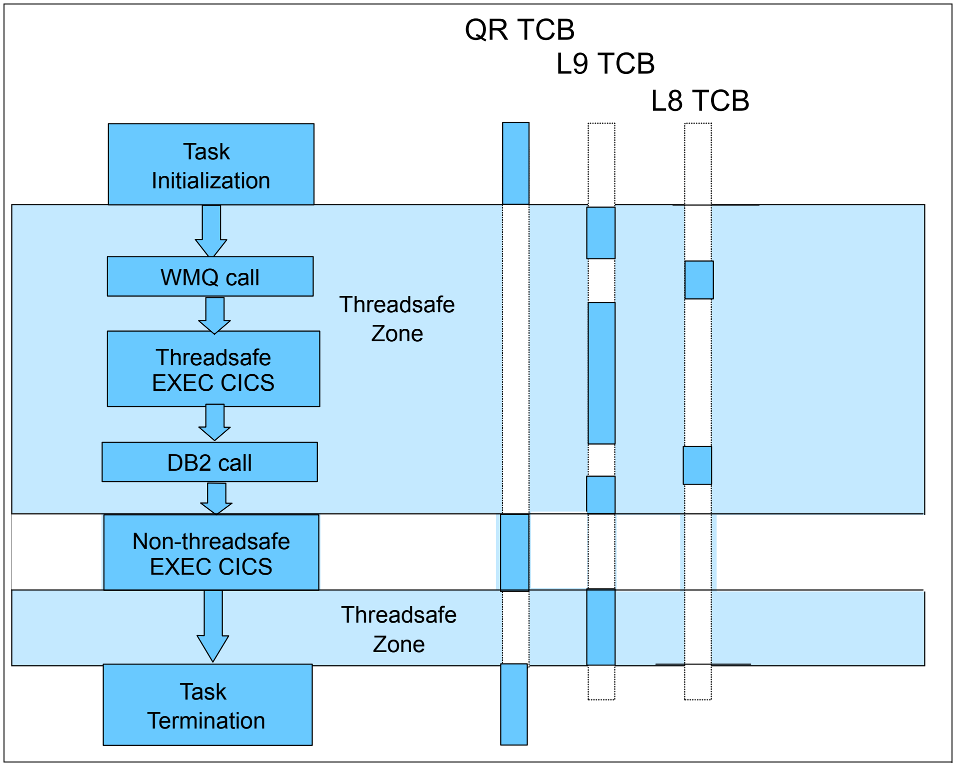


Figure 6-3 OPENAPI user key program running on QR, L9 and L8 TCBs

Figure 6-3 shows an OPENAPI program that is entering the threadsafe zone during its program initialization. In fact, an OPENAPI application receives its initial control under an open TCB. Therefore, it must be threadsafe by definition. OPENAPI programs always receive control under their open TCB, both when they start to run and when they receive control back after an EXEC CICS command or a call to a TRUE. The application logic must run under the open TCB. The key of this open TCB depends upon the EXECKEY attribute of the program. Because a user key program is shown, the open TCB selected by CICS is from the pool of L9 TCBs.

In this example, the application runs under this TCB until it issues a call to WebSphere MQ. Because WebSphere MQ calls run under an L8 TCB in CICS TS V3.2 and later, CICS switches TCBs for the duration of the request. Upon completion, the application receives control back on its L9 TCB. Here we demonstrate what can happen if the application then issues threadsafe EXEC CICS commands. Threadsafe EXEC CICS commands have no TCB affinity and, therefore, can be processed under the L9 TCB of the program. The application then issues an EXEC SQL call to DB2. As before, the flow of control moves from the L9 TCB to the L8 TCB for the duration of the call to the other OTE-enabled TRUE. The TCBs are then switched back from L8 to L9 at the end of the call. The application then issues a nonthreadsafe EXEC CICS command, which must be processed under the QR TCB so that CICS switches from the L9 to the QR TCB for the duration of the command. It switches back to the original L9 TCB when the command completes. The application then terminates. Eventually the L9 TCB returns control to the QR TCB during program termination.

This example does not show any additional TCB switches t hat are required during any syncpoint processing, for example, at end of task processing if no further programs are in the task.

6.2.2 Threadsafe data integrity issues

Another type of threadsafe issue that you can encounter is data integrity exposures. After your programs are enabled to run concurrently on multiple open TCBs, you expose your *shared resources* to update conflicts due to the multiple concurrent program instances running on parallel open TCBs.

Shared resources: The term *shared resources* in this context refers to application shared resources (for example, `EXEC CICS GETMAIN SHARED` storage), not resources managed by CICS.

In general, removing nonthreadsafe commands and changing the CICS definition to threadsafe is only half of the conversion process. You must then ensure that any shared resource is serialized to prevent data corruption.

Important: You must not remove nonthreadsafe CICS API commands to run a program in threadsafe enabled mode. CICS switches back to the QR TCB to process nonthreadsafe CICS API commands.

To correct data integrity issues, application programmers can modify the coding in their programs. Figure 6-4 emphasizes that, after leaving the serialized zone and entering the threadsafe zone, a program is in an execution zone where CICS does not provide single-threaded program execution. All shared resources now rely on the programmer to code the logic into the program to ensure that multiple instances of the program running concurrently do not corrupt the shared data. Again, for this example, a `CICSAPI CONCURRENCY(THREADSAFE)` program environment with an L8 TCB is shown.

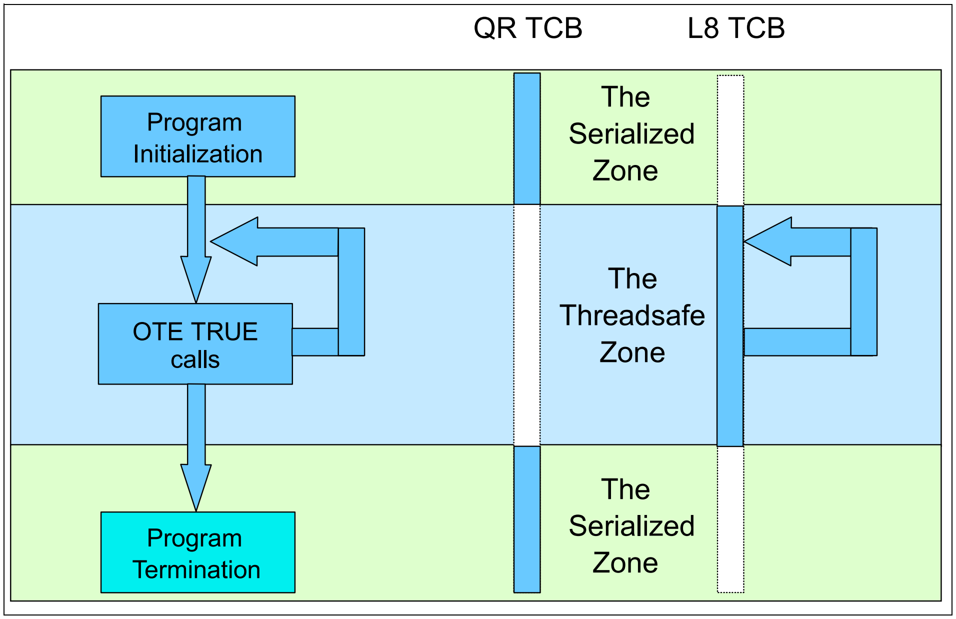


Figure 6-4 Shared data in threadsafe zone serialized by in-house code logic

Figure 6-5 shows how a nonthreadsafe program works. Because all programs run on the QR TCB, a shared resource is always updated in a serialized fashion forced by the single QR TCB. All secondary instances of the program are waiting for their chance to run on the QR TCB.

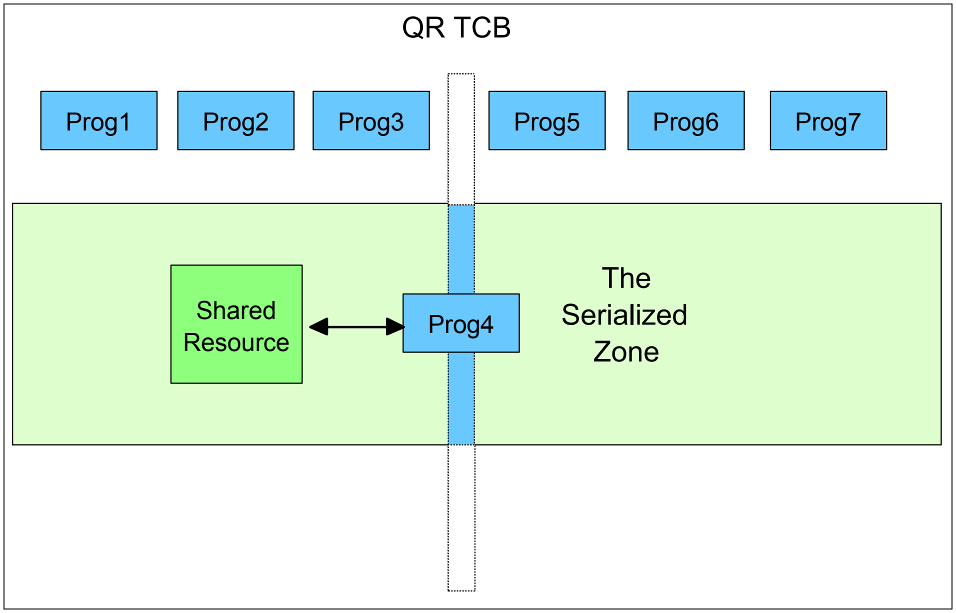


Figure 6-5 Single-threaded serialized resource on the QR TCB

Figure 6-6 shows a threadsafe environment where concurrent instances of the CICSAPI program are all running at the same time on their own L8 TCB, all sharing a common resource.

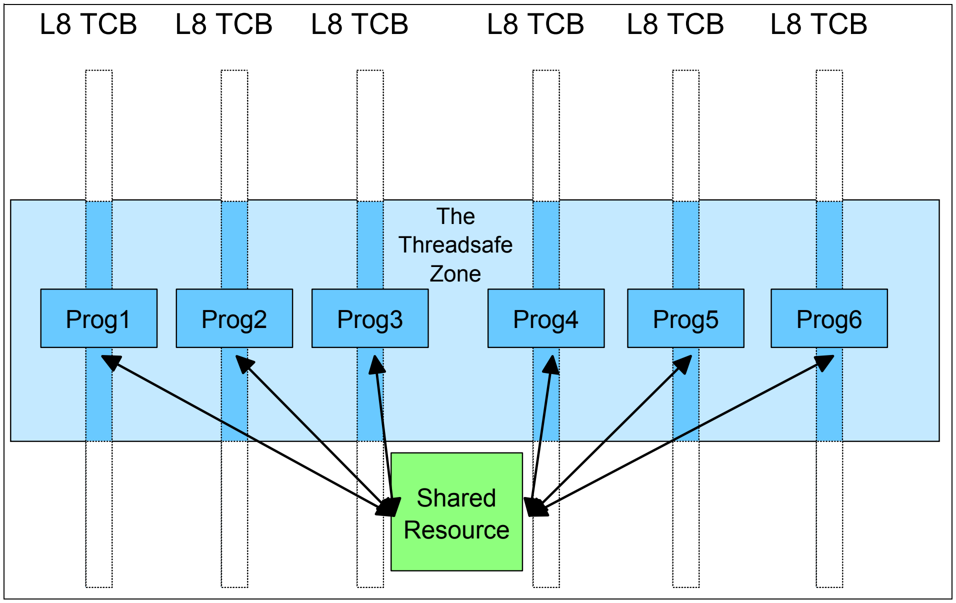


Figure 6-6 Multithreaded shared resource on multiple concurrent L8 TCBs

Figure 6-4 on page 111 through Figure 6-6 help to illustrate how a shared resource can become corrupted due to the nature of threadsafe programs running on open TCBs. Therefore, just checking CICS auxiliary trace reports and defining a program as threadsafe does not guarantee that a program is threadsafe. An application programmer or systems programmer must review the source code carefully and verify that every shared resource is properly serialized. This type of detailed and thorough analysis is vital because corruption of shared resources might not become apparent until considerable time after it occurs, *if it is noticed at all*. For example, not all such corruption results in an abend.

6.3 Analyzing the CICS regions

Before converting and running your applications and system exits in threadsafe mode, review the status of each CICS region to ensure that you have the prerequisites in place. The next few sections highlight the software and system parameters that you must review.

6.3.1 DB2 version

All supported CICS TS versions provide a DB2 attachment facility that uses the OTE. By using the OTE, CICS DB2 TRUE can run on an open TCB. The CICS DB2 attachment facility is shipped with CICS TS and works with all DB2 releases that are supported for the relevant CICS TS version. When CICS is connected to DB2 7 or later, it uses the OTE. In the OTE, the CICS DB2 attachment facility uses open TCBs (L8 mode) as the thread TCBs, rather than using specially created subtask TCBs.

Supported CICS TS versions: The following CICS TS versions were supported at the time this book was published:

- CICS TS V3.1
- CICS TS V3.2
- CICS TS V4.1
- CICS TS V4.2

6.3.2 WebSphere MQ version

Starting at CICS TS V3.2, the CICS WMQ attach code was redesigned to take advantage of the OTE. The CICS WMQ attach code provided with this release of CICS works with all of the currently supported releases of WebSphere MQ (that is, versions 6 and 7).

Also starting in CICS TS V3.2, the CICS WMQ attach code uses L8 open TCBs. Similar to the CICS DB2 attach mechanism, these TCBs give the potential for performance improvements over the previous CICS WMQ attachment mechanism with its proprietary subtask TCBs. From this CICS release onwards, your applications automatically uses L8 TCBs for their calls to WebSphere MQ.

WebSphere MQ Version 6 APAR PK42616 is shipped to provide support for the CICS MQ adapter in CICS TS V3.2. You must not install a PTF if you connect to WebSphere MQ Version 7. When WebSphere MQ is connected with a CICS TS V3.2 system, use the CICS shipped versions of the CICS WMQ adapter, the CICS WMQ trigger monitor, and the CICS WMQ bridge. The WMQ APAR ensures that the WebSphere MQ shipped versions of the components are immediately terminated if they are run when WebSphere MQ is connected with a CICS TS V3.2 system. In this situation, message CSQC330E is written to the CICS system log and to the CSMT transient data destination.

CICS WMQ attachment mechanism: WebSphere MQ continues to ship its original version of the CICS WMQ attachment mechanism for use with CICS TS V3.1 and earlier.

6.3.3 IMS version

With CICS TS V4.2, the CICS IMS interface code was redesigned to take advantage of the OTE. CICS provides a CICS IMS interface to satisfy DL/I requests that are issued by applications running in a CICS region. In CICS TS V4.2, the CICS IMS interface is defined as threadsafe, and CICS can run the CICS IMS task-related user exit on an L8 open TCB.

The OTE is supported starting in IMS 12 with PTFs for APAR PM31420 applied. During the connection process, IMS indicates to CICS that the OTE is supported, and then CICS defines the CICS IMS TRUE as an open API TRUE.

If your IMS version does not support the OTE, CICS runs the CICS IMS TRUE on the QR TCB.

6.3.4 DB2 system parameters

The **`CTHREAD`** parameter, also listed as **`MAX USERS`**, is a DB2 subsystem tuning parameter that defines the maximum number of threads that can be concurrently allocated to a DB2 subsystem from any source except for DDF. Because CICS is just one possible front end to DB2, you must ensure that the value you set for **`TCBLIMIT`** is well below the **`CTHREAD`** threshold. This parameter is relevant to all currently supported releases of CICS TS. However, if you did not check this parameter before, you might want to check with your DB2 support team. These parameters are set in the DB2 ZPARM.

6.3.5 WebSphere MQ system parameters

The **`CTHREAD`** parameter is a subsystem tuning parameter in WebSphere MQ that specifies the total number of threads that can connect to a queue manager, including batch, TSO, IMS, and CICS.

Before CICS TS V3.2, each CICS region took up nine of the threads specified here, plus one thread for each task initiator (CKTI), because the original CICS WMQ attachment mechanism used a pool of eight subtask TCBs. In CICS TS v3.2, now hard-coded number of TCBs is used for the CICS WMQ attachment. TCBs are allocated from the OTE pool of L8 TCBs and are subject to availability and the limitation set by the **`MAXOPENTCBS`** parameter. Therefore, to account for the extra threads of work resulting from CICS TS V3.2, you might need to increase the **`CTHREAD`** parameter. These parameters are set by the WebSphere MQ **`SET SYSTEM`** command.

> **CTHREAD parameter:** Starting in WebSphere MQ Version 7, you cannot adjust the **`CTHREAD`** parameter in WebSphere MQ.

6.3.6 IMS DBCTL system parameters

If you plan to enable your CICS IMS applications to run threadsafe, you can improve the throughput depending on the number of CICS IMS Database Control (DBCTL) commands that the applications issue. Therefore, you might consider reviewing the number of threads specified for the CICS IMS interface. From the performance perspective, it is essential to specify the optimum values for the number of threads that the CICS IMS interface can use.

Unlike with DB2, you specify the number of threads using two parameters:

- **`MINTHRD`**. These threads are available for the duration of the DBCTL connection. All required control blocks and resources are preallocated. Therefore, no path-length overhead exists for collapsing and reallocating thread-related storage, and throughput might be faster.
- **`MAXTHRD`**. After the **`MINTHRD`** limit is exceeded, threads continue build up to the **`MAXTHRD`** limit. However, because the control blocks of each thread are allocated during PSB scheduling, the path length is greater for the tasks running after the **`MINTHRD`** limit is reached.

After your CICS DBCTL applications are enabled to run threadsafe, you can use the DBCTL session termination statistics (Example 6-1) to investigate if thread usage changed. You can adjust the **`MINTHRD`** and **`MAXTHRD`** parameters if threads are unavailable and tasks are queued waiting for threads and PSBs.

Example 6-1 CICS DBCTL session statistics

```
Unsolicited Statistics Report      Collection Date-Time 09/16/93-15:15:18  Last Reset 15:06:46
------------------------------------------------------------------------------------------------
DBCTL SESSION TERMINATION STATISTICS
------------------------------------
  CICS DBCTL Session Number          :             2
    DBCTL identifier                 :          SYS2
    DBCTL RSE name                   :      DBCTLSY2
    Time CICS connected to DBCTL     : 15:14:02.8506
    Time CICS disconnected from DBCTL : 15:16:18.3689
    Minimum number of threads        :             1
    Maximum number of threads        :             3
    Times minimum threads hit        :             1
    Times maximum threads hit        :             1
    Elapsed time at maximum threads  : 00:00:09.4371
    Peak number of thread TCBs       :             3
    Successful PSB schedules         :             9
```

6.3.7 CICS system parameters

To effectively enable threadsafe applications, you must set or tune several CICS system parameters. The parameters described in this section are in different areas within CICS, and you can dynamically alter some of the parameters by using the CEMT commands.

SIT Parm: MXT

SIT Parm: MXT is not directly a threadsafe-related parameter, but it comes into play when setting your **MAXOPENTCBS** and **TCBLIMIT** parameters. If you are running with transaction isolation turned on, you must set the **MAXOPENTCBS** parameter greater than or equal to MXT to prevent possible TCB stealing.

You set this parameter in the system initialization table (SIT) parameter or SYSIN control statement. You can override the parameter by using the **CEMT SET SYSTEM** command.

SIT Parm: MAXOPENTCBS

The **MAXOPENTCBS** parameter sets the maximum number of L8 and L9 TCBs allowed for the CICS region. For information about setting this parameter, see 2.6.1, “MAXOPENTCBS” on page 36.

You can set this parameter in the SIT parameter or SYSIN control statement. You can override this parameter by using the **CEMT SET DISPATCHER** command.

> **Important:** You must always set the **MAXOPENTCBS** parameter greater than or equal to the **TCBLIMIT** parameter value. Additionally, when running with transaction isolation turned on, you must set the **MAXOPENTCBS** parameter equal to or higher than MXT.

DB2CONN/DB2ENTRY Parm: PRIORITY

The **PRIORITY** parameter controls the priority of the CICS open L8 thread TCBs relative to the CICS QR TCB. This parameter has three options:

- **PRIORITY=HIGH**
- **PRIORITY=LOW**
- **PRIORITY=EQUAL**

When **PRIORITY=HIGH** is specified, L8 TCBs run at a higher priority than the CICS QR TCB. Many nonthreadsafe CICS environments suffer QR TCB delays when high volume tasks with SQL requests returns to the QR TCB. As a good practice, reduce the priority of L8 TCBs to EQUAL or even LOW to avoid QR TCB constraints. If applications become threadsafe enabled in such environments, reconsider the priority parameter setting to improve workload throughput.

DB2CONN Parm: TCBLIMIT

The **TCBLIMIT** parameter specifies the maximum number of TCBs that can be used to run DB2 threads. This parameter is a subset of the **MAXOPENTCBS** parameter (see “SIT Parm: MAXOPENTCBS” on page 117).

You can set this parameter in the DB2CONN RDO definition. You can override this parameter by using the **CEMT SET DB2CONN** command.

> **Important:** You must set your TCBLIMIT greater than or equal to the total of all your **THREADLIMIT** parameters.

DB2CONN Parm: THREADLIMIT

The **DB2CONN THREADLIMIT** parameter specifies the maximum number of active DB2 threads for the pool.

You can set this parameter in the DB2CONN RDO definition. You can override this parameter by using the **CEMT SET DB2CONN** command.

DB2ENTRY Parm: THREADLIMIT

The **DB2ENTRY THREADLIMIT** parameter specifies the maximum number of active DB2 threads for a specific transaction or group of transactions.

You can set this parameter in the DB2ENTRY RDO definition. You can override this parameter by using the **CEMT SET DB2ENTRY** command.

In general, you start at the top of the preceding list, making sure that DB2 and WebSphere MQ can handle your thread volume. Then you move up into CICS, setting your MXT to the total number of active tasks that can run in your CICS region. Then you set your limit for open TCBs using the **MAXOPENTCBS** parameter.

Furthermore, for DB2, you can then use the **TCBLIMIT** parameter to throttle the number of L8 TCBs used from the **MAXOPENTCBS** pool. Also ensure that your DB2 entry and pool THREADLIMITS total a value less than or equal to your TCBLIMIT.

The **MAXSSLTCBS**, **MAXJVMTCBS**, and **MAXXPTCBS** parameters also relate to the other types of open TCBs used by OTE. These parameters are not directly relevant when implementing a threadsafe environment on CICS. However, they are important from the overall CICS system programming perspective.

Summary CICS DB2 attachment parameters

The CICS system programmer must also frequently check that entry threads have sufficient values for thread reuse. The CICS system programmer must ensure that consistent thread limits are defined by using the following rules of thumb:

- The number of thread TCBs equals the number of pool threads, plus the sum of all entry threads, plus the number command threads. The number of thread TCBs is specified in the **TCBLIMIT** parameter in DB2CONN resource parameter.
- The number of thread TCBs must equal the limit of open L8 TCBs that CICS uses. You can use the **MAXOPENTCB** system initialization parameter to specify the limit of open L8 TCBs. You might require additional open TCBs for WebSphere MQ, TCP/IP socket application programs, or CICS DBCTL application programs.

The following CICS system parameters are related to threadsafe applications.

SIT Parm: FORCEQR

The **FORCEQR** parameter can be confusing because people often think that it turns off TCB switching, which is not true, nor possible. The **FORCEQR** parameter is used only as an emergency stopgap. It shifts programs back onto the QR TCB to provide resource serialization if you realize that your supposedly threadsafe programs are not threadsafe with respect to data integrity.

The **FORCEQR** parameter overrides all **API(CICSAPI) CONCURRENCY(THREADSAFE)** program definitions in the CICS region so that they all run as though defined as **API(CICSAPI),CONCURRENCY(QUASIRENT)**.

The **FORCEQR** parameter does not affect the fact that the CICS DB2 and CICS WMQ attachment facilities now use L8 TCBs. All DB2 and WebSphere MQ calls run on an L8 TCB. The **FORCEQR** parameter ensures that you swap back to the QR TCB when returning to the application.

You can set this parameter in the SIT parameter or SYSIN control statement. You can override this parameter by using the **CEMT SET SYSTEM** command.

Affect of changing the FORCEQR parameter on other programs: A change to the **FORCEQR** parameter does not affect programs that are already running. New tasks that start use the new **FORCEQR** setting, but long-running tasks can expect a delay to pick up the change.

The **FORCEQR** parameter does not affect the following programs:

- Java programs that are run in a JVM
- C/C++ programs using XPLINK
- OPENAPI programs
- Programs defined with **CONCURRENCY(REQUIRED)**

None of these programs can run on the QR TCB.

SIT Parm: FCQRONLY

The **FCQRONLY** parameter forces CICS TS V3.2 to run file control requests under the QR TCB, in the same manner as in previous CICS releases. By default, these commands are now threadsafe and, therefore, run under an open TCB if an application was running under an L8 or L9 TCB at the time a file control command was issued. The **FCQRONLY** parameter also bypasses some of the shared storage locking and concurrency implementations that are required for threadsafe file control support. The **FCQRONLY** parameter defaults to `NO`. This parameter is provided as a means of deactivating threadsafe file control support for environments that might choose to do so, such as during application testing or validation.

Function shipped file control requests: If you function ship file control requests from application-owning regions to file-owning regions, choose one of the following settings for the **FCQRONLY** parameter:

- For FORs at CICS TS V4.2 or later that use IPIC connections over TCP/IP, specify **FCQRONLY=NO** to optimize performance for those connections.
- For FORs that use multiregion operation (MRO) links or intersystem communication (ISC) over Systems Network Architecture (SNA) connections, specify **FCQRONLY=YES** to optimize performance for those connections. Also use **FCQRONLY=YES** for all FORs before CICS TS V4.2.

You can set this parameter in the SIT parameter or SYSIN control statement. You can override this parameter by using the **CEMT SET SYSTEM** command.

6.4 Providing a threadsafe CICS operating environment

After checking each region to verify that the correct software is running and after reviewing the system parameters, the CICS system programmer must take several steps to ensure a threadsafe CICS operating environment.

6.4.1 CICS exits

GLUEs are a primary area of concern for system programmers because a poorly tuned CICS subsystem can experience a performance degradation due to excessive TCB switching caused by nonthreadsafe exits. Therefore, system programmers must make all GLUEs threadsafe before moving threadsafe-enabled application programs into production.

With the usage of the OTE in CICS TS V2.2 and later, the switchover to an L8 TCB happens earlier in processing a DB2 request than the switchover to the subtask TCB in CICS TS released before V2.2. Therefore, all exits now run, or try to run, on L8 TCBs. If you did not convert your GLUEs and define them as `CONCURRENCY(THREADSAFE)`, invocation of the exit programs causes a switchback to the QR TCB for processing and then immediately returns to the L8 TCB to continue processing the DB2 call. The process can generate a TCB thrashing effect that results in poor performance.

The same effect is true for exit programs driven during calls to WebSphere MQ in CICS TS V3.2 and later versions.

The design of CICS forces CICSAPI application programs and exit programs to react differently. When an exit program is swapped back to the QR TCB for processing, it always swaps back to the L8 TCB on return. Then if the application program encounters a nonthreadsafe CICS command, it again swaps back to the QR TCB. However, unlike exits, threadsafe application programs stay on the QR TCB until another call to an OTE-enabled TRUE, such as DB2 or WebSphere MQ, is encountered. In CICS TS V3 and later versions, OPENAPI programs are treated in a similar manner to exits in this respect. That is, they always receive control back under their open TCB, if they start nonthreadsafe commands that require switching to the QR TCB for processing.

Figure 6-7 shows the flow of a DB2 call from a threadsafe CICSAPI program, showing how the GLUEs cause processing to bounce between the QR TCB and an L8 TCB.

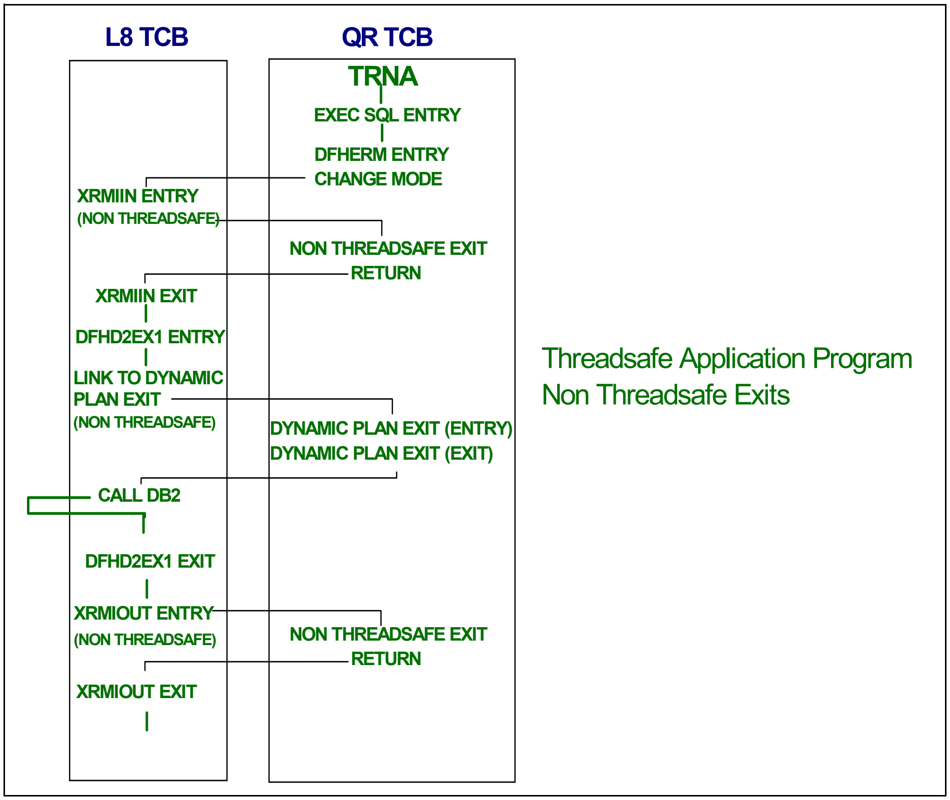


Figure 6-7   Exit flow between the QR and L8 TCBs

The flow in Figure 6-7 shows the application program starting on the QR TCB when a DB2 call is encountered and execution is swapped over to an L8 TCB to process the DB2 call. On the L8 TCB, the XRMIIN exit is encountered. Because it is nonthreadsafe, its processing is swapped back over to the QR TCB. When the XRMIIN exit is complete, the process flow is returned to the L8 TCB. CICS always returns to the TCB where the exit was started.

Processing continues on the L8 TCB until the dynamic plan exit is started, at which point processing is again swapped back to the QR TCB. Upon completion of the dynamic plan exit, processing is swapped back onto the L8 TCB to make the actual DB2 call.

After the DB2 call is complete, the XRMIOUT exit is started and processing swaps over to the QR TCB to process the exit and then back to the L8 TCB after the exit is complete. All processing continues on the L8 TCB until the program terminates, a nonthreadsafe command is encountered, or a nonthreadsafe exit is encountered.

Figure 6-7 on page 122 shows an example of the switching that might be encountered for exits on the path of a DB2 call. If calls are made to WebSphere MQ in CICS TS V3.2, the same principles apply now that the CICS MQ adapter is an OTE-enabled TRUE. For WebSphere MQ, a dynamic plan exit does not exist, but instead there is the API crossing exit CSQCAPX. This exit is defined as threadsafe in the program definition as supplied in CICS TS V3.2, meaning, by default, the supplied version of CSQCAPX is threadsafe. Therefore, if it is active, it does not result in a switch back from the L8 to the QR TCB for the link to the CICS WMQ API crossing exit. If this exit is changed, ensure that any alterations to its logic are implemented in a threadsafe manner.

Now that you understand that nonthreadsafe exits cause extra switching in the threadsafe zone, you can start the analysis and conversion process. The preceding sections highlighted that you must review the key exits: the XRMIIN, XRMIOUT, DB2 dynamic plan exit, and CICS WMQ API crossing exit. Any EXEC CICS command can potentially drive exit programs defined to run at the XEIIN and XEIOUT exit points. Finally, starting with CICS TS V3.2, the EXEC CICS file control commands are provided as threadsafe. Therefore, you must review exits started from within file control for potential TCB switching activity and analyze the XFCREQ and XFCREQC exit points.

Additionally, any EXEC CICS calls made in one of the key exit programs can pull in other exits, such as XEIIN, XEIOUT, XPCFTCH, or XTSQRIN. Therefore, all exits that are started during the execution path of a DB2, DBCTL, WMQ call, or an EXEC CICS file control request must be converted to be threadsafe to eliminate a TCB switching.

6.4.2 Analyzing your exits

Before you begin analyzing your exits, you must first identify which exits (TRUEs, GLUEs, and dynamic plan exits) are in your system. You must also determine whether they need any modifications to the code or their definition to make them threadsafe. In an exception case, if you have no exits, you can skip the rest of this section and move on to converting the applications themselves.

Tools used to identify your exits

The easiest way to understand which exits (TRUEs, GLUEs, and dynamic plan exits) are running in your CICS region, you can use the `DFHOSTAT` utility that ships with CICS. Running the STAT transaction generates a report that lists all of your TRUE and GLUE exits with a listing of your DB2 Pool and Entry resources.

You can also use the IBM CICS Interdependency Analyzer (CICS IA) to identify your exits. For information, see 9.1, “Four-step CICS Tools process” on page 246.

You can use the report to identify which exits you have and whether they are defined as THREADSAFE or QUASIRENT. The report also helps you identify whether your system exits are using a GWA, which can be a shared resource.

For information about CSQCAPX (the CICS WMQ API crossing exit), use CEMT or an equivalent to review its program definition CONCURRENCY attribute. As mentioned previously, as provided, it is defined as threadsafe and written to threadsafe standards. The name of the crossing exit is fixed and cannot be changed. The CKQC panel shows whether it is active.

Shared GWA as a nonserialized shared resource: You can use a shared GWA as a nonserialized shared resource and, therefore, classify your exit program as nonthreadsafe, in which case you must add serialization techniques to your code.

6.4.3 Running the DFH0STAT utility

To run the **DFH0STAT** utility:

1. Ensure that the CICS system definition data set (CSD) group DFH$STAT is installed.
2. Run the STAT transaction to access the main menu for the **DFH0STAT** utility (Figure 6-8).

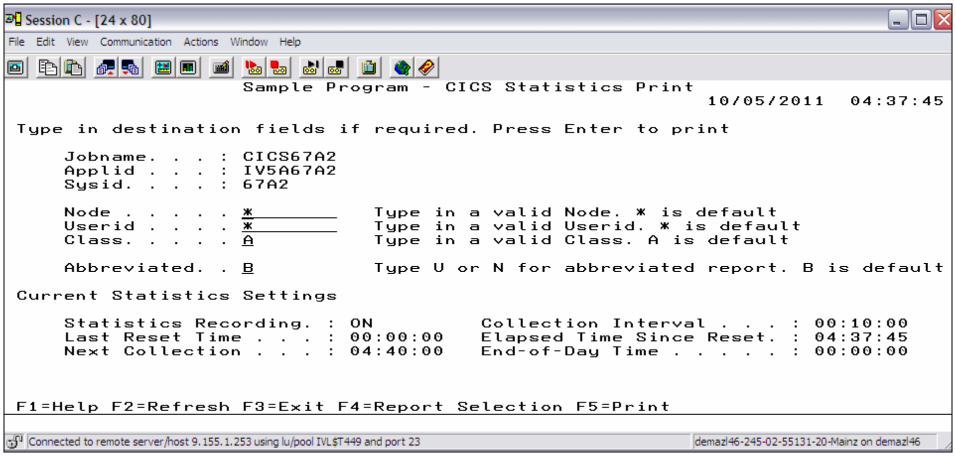


Figure 6-8 CICS STAT/DFH0STAT panel

3. Press the PF4 key to access the report selection menu (Figure 6-9).
4. For **DB2 Connection and Entries** and **User Exit Pgms/Global User Exits**, type Y, and then press Enter.

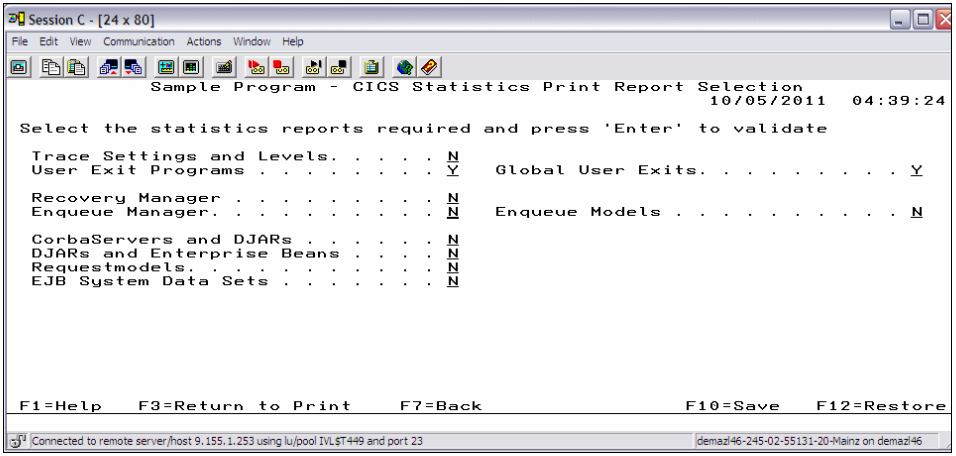


Figure 6-9 DFH0STAT report selection menu

5. Press PF3 to return to the main menu.
6. Press Enter to print your report.

Example 6-2 shows the User Exit Programs and Global User Exits sections of the DFH0STAT report.

Example 6-2 Output from the DFH0STAT utility showing the exits reports

```
User Exit Programs
__________________
                    <---- Global Area ---->   No.
 Program    Entry    Entry            Use    of    Program  Exit   Program    LIBRARY
  Name      Name     Name    Length  Count  Exits  Status   Use    Count      Name     LIBRARY Dataset Name
 ___________________________________________________________________________________________________________
 PFEI       PFEI     PFEI      200       1      1  Started          16,411    DFHRPL    CICS67.CAPALOAD
 PFEIO      PFEIO                0       0      1  Started          12,600    DFHRPL    CICS67.CAPALOAD
 PFRMI      PFRMI                0       0      1  Started               0    DFHRPL    CICS67.CAPALOAD
 PFRMIO     PFRMIO               0       0      1  Started               0    DFHRPL    CICS67.CAPALOAD

Global User Exits
_________________
   Exit    Program    Entry    <------- Global Area ------->   Number   Program    Program     Concurrency
   Name     Name      Name     Entry Name  Length  Use Count  of Exits  Status   Concurrency     Status
 ______________________________________________________________________________________________________________
 XDUREQ   EYU9NLDR  EYU9NLDR   EYU9NXSD       0          0         1    Started  Threadsafe    Threadsafe
 XMEOUT   EYU9NLME  EYU9NLME   EYU9NXSD       0          0         1    Started  Threadsafe    Threadsafe
 XMNOUT   EYU9NMTE  EYU9NMTE   EYU9NXSD       0          0         1    Started  Quasirent     Quasirent
```

```
XPCTA     EYU9XLGR  EYU9XLGR   EYU9XLGR      40          2         1   Started  Quasirent    Quasirent
XSTOUT    EYU9NMST  EYU9NMST   EYU9NXSD       0          0         1   Started  Quasirent    Quasirent
XEIIN     PFEI      PFEI       PFEI         200          1         1   Started  Threadsafe   Threadsafe
XEIOUT    PFEIO     PFEIO                     0          0         1   Started  Threadsafe   Threadsafe
XSRAB     EYU9XLSR  EYU9XLSR   EYU9XLGR       0          0         1   Started  Threadsafe   Threadsafe
XRSINDI   EYU9NLID  EYU9NLID   EYU9NXSD       0          0         1   Started  Threadsafe   Threadsafe
XSNOFF    EYU9NLSO  EYU9NLSO   EYU9NXSD       0          0         1   Started  Threadsafe   Threadsafe
XRMIIN    PFRMI     PFRMI                     0          0         1   Started  Threadsafe   Threadsafe
XRMIOUT   PFRMIO    PFRMIO                    0          0         1   Started  Threadsafe   Threadsafe
XDUREQC   EYU9NLDC  EYU9NLDC   EYU9NXSD       0          0         1   Started  Threadsafe   Threadsafe
```

The first section of the report, User Exit Programs, lists the exit programs in the system. The Concurrency Status column shows the concurrency setting for each program.

The other item of interest is the Global Area section of the exit programs or GLUEs reports. The Use Count column identifies whether an exit is using a GWA.

The report in Example 6-2 has exits XEIIN, XEIOUT, XRMIIN, and XRMIOUT in use. These exits have a GWA in use, which can be a shared resource, as identified by the Length and Use Count fields.

The User Exit Programs and Global User Exits reports identify the exits. However, the dynamic plan exit is not defined as a CICS exit. Therefore, you must search the DB2 connection and DB2 entries reports to identify and list all DB2 dynamic plan exits.

For the CICS WMQ API crossing exit CSQCAPX, the same is true as for dynamic plan exits. Nor is it defined as a CICS exit. Therefore, you must investigate the exit by using, for example, CEMT and CKQC.

You can use the sample DFH0STAT report to see whether a pool dynamic plan exit is defined. This report also shows if any entry definitions have any plan exits in use. Unfortunately, the DFH0STAT report does not indicate whether PLANEXIT is defined as threadsafe or Quasirent. You must issue a **`CEMT I PROG`** command to determine its status. If you do not want to use the sample DFH0STAT report, you can also use the **`CEMT INQ DB2CONN`** and **`CEMT INQ DB2ENTRY`** commands to determine if DB2 plan exits are used.

Figure 6-10 shows the output of the **CEMT INQ DB2CONN** command. The PLANEXIT pool plan exit is in use.

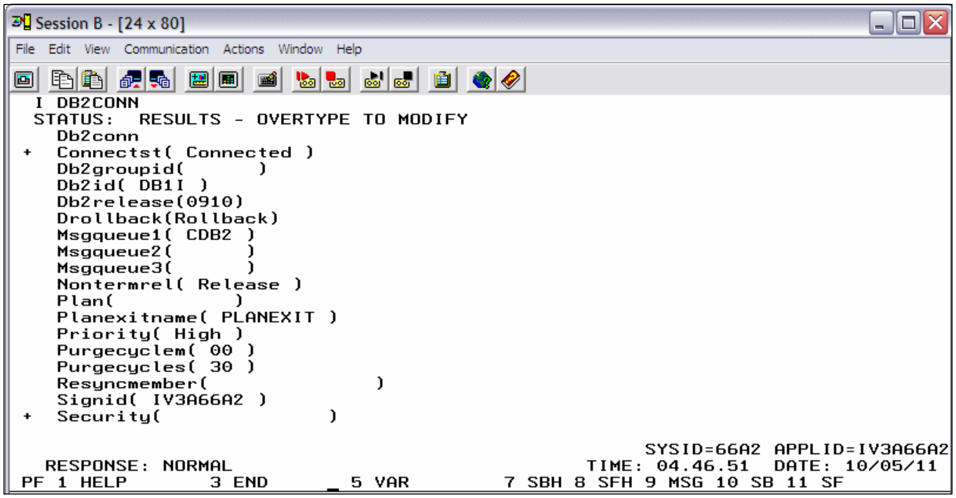


Figure 6-10 Results of the CEMT INQUIRE DB2CONN command

Figure 6-11 shows the result of the **CEMT INQ DB2ENTRY** command that is issued for this DB2 entry. Again the PLANEXIT is in use.

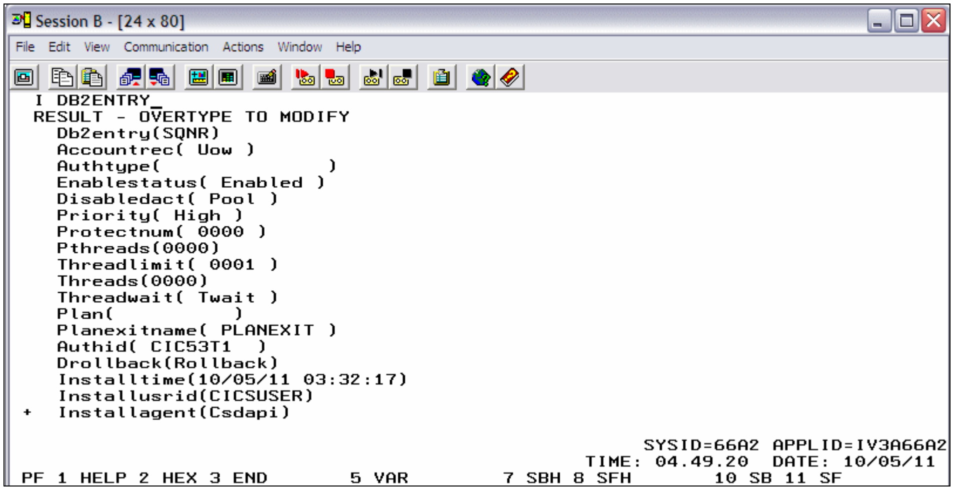


Figure 6-11 Results of the CEMT INQ DB2ENTRY command

6.4.4 Exits that must be reviewed

As previously mentioned, all system exits must be threadsafe before you run threadsafe-enabled applications in production. You must review all your exits. However, you must convert exits in the DB2, IMS, and WMQ call path to be threadsafe, in addition to exits driven during the processing of threadsafe EXEC CICS commands. This conversion particularly applies to those commands for heavily used API options such as EXEC CICS file control requests.

The DFH0STAT report helps you to identify which exits are in your system. However, you might have many and might be unsure about which ones to convert. To determine which exits are in the DB2, WMQ, or EXEC CICS file control command path, turn on a CICS auxiliary trace for a specific DB2, WMQ, or file control application program. Then review the trace for that transaction, noting all the TCB switching and identifying which exits were involved.

The CICS auxiliary trace in Example 6-3 shows a typical TCB switch (change mode) from the QR to the L8 TCB to process a DB2 call.

Example 6-3 CICS auxiliary trace output for a single task

```
04488 QR    AP 00E1 EIP   EXIT  ASKTIME               OK
04488 QR    AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )
04488 QR    US 0401 USXM  ENTRY INQUIRE_TRANSACTION_USER
04488 QR    US 0402 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK 00000000
04488 QR    RM 0301 RMLN  ENTRY ADD_LINK              RMI,2302E1E4 , 01010101 ,
04488 QR    RM 0302 RMLN  EXIT  ADD_LINK/OK           0111005A,2302E1E4 , 010101
04488 QR    DS 0002 DSAT  ENTRY CHANGE_MODE        L8      Example 6-4 on page 129
04488 QR    DS 0018 DSDS4 ENTRY ALLOC_OPEN            1,22E4A060
04488 QR    DS 0019 DSDS4 EXIT  ALLOC_OPEN/OK
DSTCB QR    DS 0016 DSDS3 ENTRY PARTITION_EXIT        21D03030
DSTCB QR    DS 0032 DSDS3 EVENT DSDS3_SCAN_HAND_POSTABLES
DSTCB QR    DS 0022 DSDS3 EVENT MVS_WAIT_ENTRY
04488 L8014 DS 0003 DSAT  EXIT  CHANGE_MODE/OK
04488 L8014 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM TSTXEII AT EXIT POINT XRMIIN
04488 L8014 DS 0002 DSAT  ENTRY CHANGE_MODE           QR
DSTCB L8014 DS 0016 DSDS3 ENTRY PARTITION_EXIT        23133148
DSTCB L8014 DS 0022 DSDS3 EVENT MVS_WAIT_ENTRY
DSTCB QR    DS 0023 DSDS3 EVENT MVS_WAIT_EXIT
DSTCB QR    DS 0017 DSDS3 EXIT  PARTITION_EXIT/OK
DSTCB QR    DS 0042 DSTCB EVENT TRACE_DOUBLE_CHAIN_GET
04488 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK
04488 QR    AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM TSTXEII WITH RETURN CODE 0
04488 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           L8
DSTCB QR    DS 0032 DSDS3 EVENT DSDS3_SCAN_HAND_POSTABLES
DSTCB QR    DS 0022 DSDS3 EVENT MVS_WAIT_ENTRY
DSTCB QR    DS 0016 DSDS3 ENTRY PARTITION_EXIT        21D03030
DSTCB L8014 DS 0023 DSDS3 EVENT MVS_WAIT_EXIT
```

```
DSTCB L8014 DS 0017 DSDS3 EXIT  PARTITION_EXIT/OK
DSTCB L8014 DS 0042 DSTCB EVENT TRACE_DOUBLE_CHAIN_GET
04488 L8014 DS 0003 DSAT  EXIT  CHANGE_MODE/OK
04488 L8014 AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
```

In Example 6-3, the program runs on the QR TCB when it performs a DB2 call and then it jumps to an L8 TCB. The program then encounters a nonthreadsafe exit and jumps back to the QR TCB to run it. Upon completion of the exit, the program returns to the L8 TCB to process the SQL request of the application.

6.4.5 Identifying exits in the DB2, WMQ, and file control call paths

The find exits that are started during the DB2, IMS, WMQ, or file control call path:

1. Turn on a CICS auxiliary trace.
2. Search the output report looking for CHANGE_MODE entry records to the QR TCB followed by the RETURN-FROM-USER-EXIT-PROGRAM events.
3. Note the exit involved.

As an alternative approach, use the CICS IA Command Flow feature as explained in “Command Flow feature” on page 251.

You might have to make several iterations of your report. For example, you might need to turn on your CICS auxiliary trace for a short time, run a general report, and then look for the CHANGE_MODE records. Then you must choose one task and rerun the report for that single task only to eliminate extraneous report records and then use the following technique.

The CICS auxiliary trace snippet in Example 6-4 is an excerpt from the auxiliary trace in Example 6-3 on page 128. It shows a task already running on the L8 TCB linking to exit TSTXEII and then issuing a CHANGE_MODE to the QR TCB. Shortly after the change mode to the QR TCB, you see a few trace records on the QR TCB. One of them is the RETURN-FROM-USER-EXIT-PROGRAM event record showing a return from TSTXEII.

Example 6-4 CICS auxiliary trace entries showing the return from an exit

```
04488 L8014 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM TSTXEII AT EXIT POINT XRMIIN
04488 L8014 DS 0002 DSAT  ENTRY CHANGE_MODE          QR
DSTCB L8014 DS 0016 DSDS3 ENTRY PARTITION_EXIT       23133148
DSTCB L8014 DS 0022 DSDS3 EVENT MVS_WAIT_ENTRY
DSTCB QR    DS 0023 DSDS3 EVENT MVS_WAIT_EXIT
DSTCB QR    DS 0017 DSDS3 EXIT  PARTITION_EXIT/OK
DSTCB QR    DS 0042 DSTCB EVENT TRACE_DOUBLE_CHAIN_GET
```

```
04488 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK
04488 QR    AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM TSTXEII WITH
04488 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           L8
```

The four trace records in this sequence mean that you just started a nonthreadsafe exit, TSTXEII. You can now add TSTXEII to the list of exits that you will review.

Repeat the preceding process to identify all your exits in the DB2 or WMQ call paths and in the EXEC CICS file control operations.

6.4.6 Identifying dynamic plan exits in the DB2 call path

To identify the dynamic plan exits in the DB2 call path, again use the CICS auxiliary trace output. The auxiliary trace sample in Example 6-5 shows the invocation of a dynamic plan exit called *DB2PLAN*. The output is already on an L8 TCB, as indicated by the second column containing L8000, which is the name of the TCB. Therefore, this trace is in the middle of the DB2 call path.

Example 6-5 Auxiliary trace showing invocation of a dynamic plan exit DB2PLAN

```
00258 L8000 AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL SELECT
00258 L8000 PG 0A01 PGLU  ENTRY LINK_URM              DB2PLAN,22BE7678 , 0000001C,NO,NO
00258 L8000 DD 0301 DDLO  ENTRY LOCATE                21C27B70,22BE7574,PPT,DB2PLAN
00258 L8000 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 22DA5B88
00258 L8000 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       22C87258
00258 L8000 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK
A43002D8,243002B0,D8,0,REUSABLE,ESDSA,OLD_COPY
00258 L8000 AP 1940 APLI  ENTRY START_PROGRAM
DB2PLAN,CEDF,FULLAPI,URM,NO,22F89C10,22BE7678
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE           QR
00258 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK
00258 QR    SM 0C01 SMMG  ENTRY GETMAIN               190,YES,00,TASK
00258 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788
00258 QR    AP 00E1 EIP   ENTRY RETURN 0004,226E1798 .>.q,08000E08 ....
00258 QR    AP E160 EXEC  ENTRY RETURN                ASM
00258 QR    SM 0301 SMGF  ENTRY FREEMAIN              226E1788,TASK
00258 QR    SM 0302 SMGF  EXIT  FREEMAIN/OK
00258 QR    AP 1941 APLI  EXIT  START_PROGRAM/OK      ....,NO,DB2PLAN
00258 QR    LD 0001 LDLD  ENTRY RELEASE_PROGRAM       22C87258,A43002D8
00258 QR    LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK    243002B0,D8,ESDSA
00258 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           L8
00258 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK        QR
00258 L8000 PG 0A02 PGLU  EXIT  LINK_URM/OK
00258 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL          230D7030
00258 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK
```

```
00258 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT
00258 L8000 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL
```

As part of starting the dynamic plan exit, DB2PLAN, CICS detects that the exit is not threadsafe and immediately switches to the QR TCB. Upon return from the dynamic plan exit, it switches back to the L8 TCB.

The CICS auxiliary trace snippet in Example 6-6 is taken from the auxiliary trace in Example 6-5 on page 130. This snippet shows a task already running on an L8 TCB that starts a URM, which in this case is the dynamic plan exit DB2PLAN. Shortly after the LINK_URM record, CICS starts program DB2PLAN and then detects that it is not threadsafe, causing a switch to the QR TCB.

Example 6-6 Auxiliary trace entries showing a task running on an L8 TCM that starts a URM

```
00258 L8000 PG 0A01 PGLU  ENTRY LINK_URM              DB2PLAN,22BE7678 , 0000001C,NO,NO
00258 L8000 DD 0301 DDLO  ENTRY LOCATE                21C27B70,22BE7574,PPT,DB2PLAN
00258 L8000 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 22DA5B88
00258 L8000 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       22C87258
00258 L8000 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK
A43002D8,243002B0,D8,0,REUSABLE,ESDSA,OLD_COPY
00258 L8000 AP 1940 APLI  ENTRY START_PROGRAM         DB2PLAN,CEDF,FULLAPI,URM,NO,
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE           QR
```

The sequence of trace records in Example 6-6 identifies all of the dynamic plan exits in the DB2 call path. You can now add DB2PLAN to the list of exits to review.

Repeat this process to identify all dynamic plan exits in the DB2 call path.

6.4.7 Contacting the owner of vendor product exits

If various monitors and debugging products are installed, you might see their product exits in the DB2, IMS, and WMQ call paths. If you do, contact the product vendor to have them determine the threadsafe status of the exit and make adjustments according to the guidance provided.

6.5 Making your exits threadsafe

After you identify which exits are in your system and review their source code for potential code changes, make the code adjustments to ensure that they are threadsafe.

Make programs threadsafe requires the following tasks:

1. Serializing shared resources.
2. Changing the CONCURRENCY definition of exit programs to THREADSAFE

Additionally, remove nonthreadsafe EXEC CICS commands if possible. Although having such commands in the exit does not make the exit nonthreadsafe, it causes additional TCB switching, which you must avoid if possible, especially for exits that are started on mainline DB2 and WMQ call paths.

6.5.1 Removing nonthreadsafe commands

For exit programs that run at exit points where CICS allows use of the CICS command-level API, review your exit code for use of EXEC CICS commands that can cause a switch to the QR TCB. Consider ways to eliminate use of these commands, such as whether you can use an XPI command instead. For a complete list of threadsafe commands, see the following publications:

- Appendix L, “Threadsafe Command List,” in the *CICS Application Programming Reference*, SC34-6232, for CICS TS V2, and *CICS Application Programming Reference,* SC34-6434, for CICS TS V3
- Appendix D, “Threadsafe SPI commands,” in the *CICS System Programming Reference*, SC34-6233, for CICS TS V2, and *CICS System Programming Reference*, SC34-6435, for CICS TS V3
- Appendix G, “Threadsafe XPI commands,” in the *CICS Customization Guide*, SC34-6227, for CICS TS V2, and *CICS Customization Guide*, SC34-6429, for CICS TS V3

If your assembler exit program links other high-level language programs, you must review them and change them as required to make them threadsafe.

You can use the **`DFHEISUP`** utility to search for nonthreadsafe commands.

6.5.2 Serializing shared resources

Serializing shared resources is a bit more difficult. It involves modifying the program code to add serialization techniques around code that is accessing application-shared resources.

The following EXEC CICS commands are the key commands to search for in your code and that you can use to access a shared resource:

- `ADDRESS CWA`
- `EXTRACT EXIT`
- `GETMAIN SHARED`

You can also use these techniques to serialize access to your shared storage:

- Assembly language COMPARE AND SWAP instructions
- `EXEC CICS ENQ` and `EXEC CICS DEQ` commands
- `XPI ENQUEUE` and `XPI DEQUEUE` commands

For a walk through the process to convert an exit and to see usage of the COMPARE AND SWAP instruction to serialize an application shared resource, see “Serializing access to GWAs” on page 224.

Compare and swap techniques

You can use compare and swap as a possible serialization technique. For information about compare and swap, see *z/Architecture Principles of Operations*, SA22-7832, which lists the following techniques:

- COMPARE AND SWAP
- COMPARE AND SWAP AND PURGE
- COMPARE DOUBLE AND SWAP
- TEST AND SET
- PERFORM LOCKED OPERATION

Additionally, when using one of these techniques, you must also pay attention to which instructions you used to load your data initially. In the example code, we use a COMPARE DOUBLE AND SWAP technique, which acts on two words of data at a time. Therefore, when we loaded the initial two words of data, each individual word being loaded had a small window of opportunity to change before the next word was loaded. *z/Architecture Principles of Operations*, SA22-7832, indicates to use the LOAD MULTIPLE instruction to load your data. It acts similar to the COMPARE AND SWAP instruction where the storage is locked during the time the instruction runs.

6.5.3 Changing the CONCURRENCY definition of the exit program to THREADSAFE

The sample exit is run in both Quasirent and threadsafe modes. The code is modified to remove a data integrity issue created by running it in threadsafe mode. You view the program in question by using CEMT, and if necessary, you change its definition by using CEDA to be threadsafe.

In Figure 6-12, program RMIXIT is shown as `Thr`, which means it is defined to CICS as **`CONCURRENCY(THREADSAFE)`**. Program RMIXIT2 is shown as `Qua`, which means it is defined to CICS as **`CONCURRENCY(QUAISRENT)`**. Both programs are defined to CICS as **`API(CICSAPI)`**.

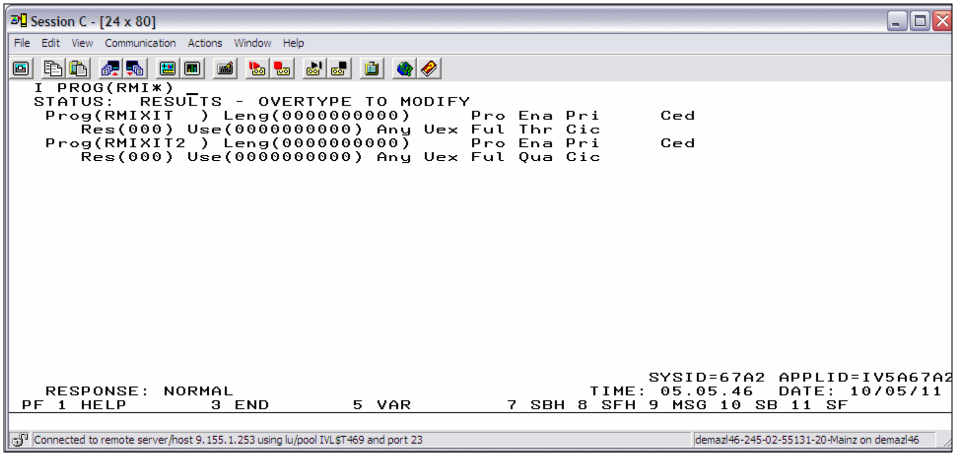


Figure 6-12 CEMT display showing Quasirent and threadsafe programs

Figure 6-13 shows the CEDA definitions for the Quasirent and threadsafe programs in Figure 6-12.

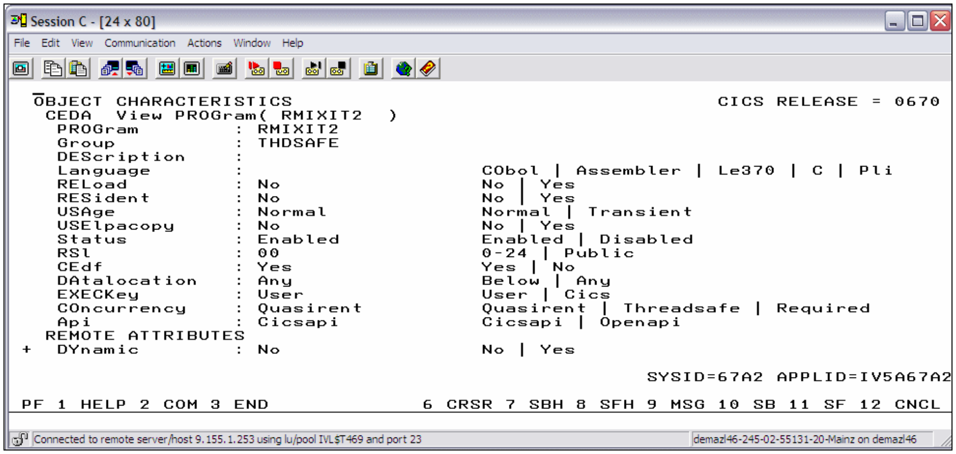
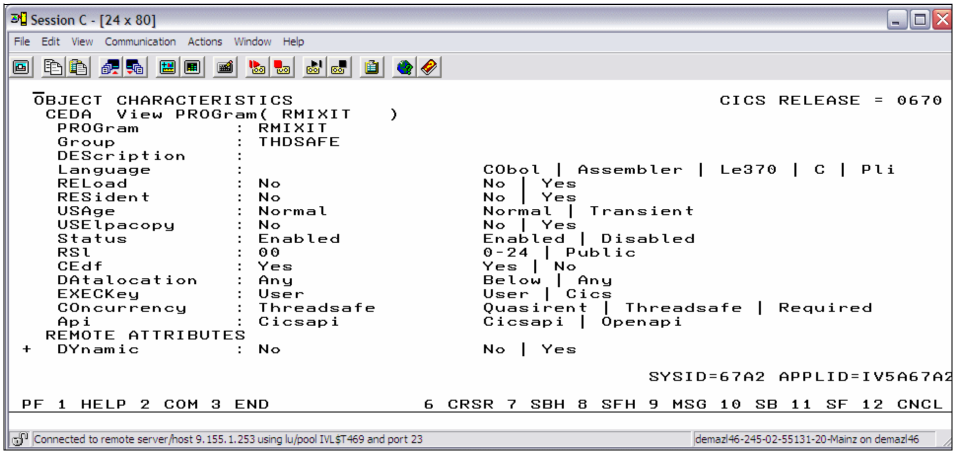


Figure 6-13 Quasirent program definition

Figure 6-14 shows the threadsafe program RMIXIT.



```
Session C - [24 x 80]
File  Edit  View  Communication  Actions  Window  Help

 OBJECT CHARACTERISTICS                                       CICS RELEASE = 0670
  CEDA  View PROGram( RMIXIT   )
   PROGram        : RMIXIT
   Group          : THDSAFE
   DEScription    :
   Language       :                    COBol | Assembler | Le370 | C | Pli
   RELoad         : No                 No | Yes
   RESident       : No                 No | Yes
   USAge          : Normal             Normal | Transient
   USElpacopy     : No                 No | Yes
   Status         : Enabled            Enabled | Disabled
   RSl            : 00                 0-24 | Public
   CEdf           : Yes                Yes | No
   DAtalocation   : Any                Below | Any
   EXECKey        : User               User | Cics
   COncurrency    : Threadsafe         Quasirent | Threadsafe | Required
   Api            : Cicsapi            Cicsapi | Openapi
  REMOTE ATTRIBUTES
 + DYnamic        : No                 No | Yes

                                                  SYSID=67A2 APPLID=IV5A67A2
 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

Connected to remote server/host 9.155.1.253 using lu/pool IVL$T469 and port 23     demazl46-245-02-55131-20-Mainz on demazl46
```

Figure 6-14 Threadsafe program definition RMIXIT

6.6 Nonthreadsafe data integrity example

Because data integrity threadsafe exposures are difficult to diagnose and find, this section provides an example of an exit that does not serialize a shared GWA and shows the disastrous effects. A short, simple assembler GLUE exit program is used that shares a GWA that simulates a storage chain. The data structure used by the sample exit program is a two-word header that contains the next available address of storage to update and a counter of how many updates are made.

The program reads in the two-word header, bumps the address value in the first word to the next address, and increments the counter in the second word by one. Then it saves the header back into the shared storage area and processes the header chain by using the address value to store a word of information. This example shows exclusive ORing of ones into memory to demonstrate the changes.

If the storage is serialized, each program picks up the next sequential chain address stored in the header, builds a sequential list of ones, and increments the counter by one. When the program fails to serialize the shared storage, the address value gets reused. Then instead of ORing ones into the shared storage, it can reverse the effect and turn a word of ones into zeros. Therefore, you can see pockets of zeros interspersed throughout the shared storage area. Additionally, the counter is not incremented sequentially and has an invalid total.

6.6.1 Nonthreadsafe code example

Example 6-7 shows a GLUE exit program, called RMIXIT, which is not threadsafe, because the shared storage is not serialized.

Example 6-7 GLUE with nonthreadsafe code

```
RMIXIT   DFHEIENT
RMIXIT   AMODE 31
RMIXIT   RMODE ANY
         LR    R2,R1                DFHUEPAR PLIST PROVIDED BY CALLER
         USING DFHUEPAR,R2          ADDRESS UEPAR PLIST
         L     R4,UEPGAA            GET GWA ADDRESS
         LA    R4,12(R4)            BUMP TO A DOUBLE WORD ADDR
         USING GWA,R4               ADDRESSABILITY
*
         L    R6,0(R4)              LOAD SAVED PGM ADDR
         L    R7,4(R4)              LOAD CTR
         LA   R8,4(R6)              BUMP SAVED PGM ADDR BY 4
         LA   R9,1(R7)              BUMP CTR BY 1
         L    R5,LOOPCTR            DELAY LOOP TO GET SOME OVERLAP
LOOP     EQU  *                      SO THAT WE CAN GENERATE SOME
         BCT  R5,LOOP                TCB CONTENTION
         ST   R8,0(R4)              STORE PGM ADDR AT HEADER ADDR
         ST   R9,4(R4)              STORE THE CTR AT WORD 2 IN HEADER
         L    R7,0(R8)              LOAD DATA AT PGM ADDR
         X    R7,ONES               FLIP THE BITS
         ST   R7,0(R8)              STORE THE DATA AT PGM ADDR
*
         LA    R15,UERCNORM         SET OK RESPONSE
         ST    R15,RETCODE            IN WORKING STORAGE
*
RETURN   EQU   *
         L     R15,RETCODE          FETCH RETURN CODE
         DFHEIRET RCREG=15          RETURN TO CICS
*********************************************************************
          DC    F'0'
ONES      DC    X'11111111'         ONES
LOOPCTR   DC    F'00777777'         TIME DELAY LOOP
*********************************************************************
         LTORG
         END   RMIXIT
```

DFH0STAT report showing RMIXIT defined to the system

By using the **DFH0STAT** utility, we generated a report (Example 6-8) to show how the program is defined to the system and to show the GWA that we are using.

Example 6-8 DFH0STAT report showing XRMIIN and XRMIOUT using a GWA

```
-----------------------------------------------------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY CICS67A2 JOB01695  DSID   114 LINE 1,711   COLUMNS 02- 133
COMMAND INPUT ===>                                            SCROLL ===> CSR
 EYU9XLAP  EYU9XLOP  Cics  Quasirent    Quasirent              0      No      No     No     No Wait   No      No
 EYU9NMTE  EYU9NMTE  Cics  Quasirent    Quasirent              0      No      No     No     No Wait   No      No
 RMIXIT    RMIXIT    Cics  Quasirent    Quasirent              0      No      No     No     No Wait   No      No
pplid IV5A67A2  Sysid 67A2  Jobname CICS67A2    Date 10/05/2011  Time 07:10:55                CICS 6.7.0           PAGE
_____________________________________________________________________________________________________________________________
Global User Exits
__________________
  Exit   Program   Entry   <------- Global Area -------> Number  Program   Program   Concurrency
  Name    Name     Name   Entry Name Length  Use Count  of Exits Status  Concurrency   Status
______________________________________________________________________________________________________
 XDUREQ   EYU9NLDR EYU9NLDR  EYU9NXSD     0        0        1  Started Threadsafe  Threadsafe
 XMEOUT   EYU9NLME EYU9NLME  EYU9NXSD     0        0        1  Started Threadsafe  Threadsafe
 XMNOUT   EYU9NMTE EYU9NMTE  EYU9NXSD     0        0        1  Started Quasirent   Quasirent
 XPCTA    EYU9XLGR EYU9XLGR  EYU9XLGR    40        2        1  Started Quasirent   Quasirent
 XSTOUT   EYU9NMST EYU9NMST  EYU9NXSD     0        0        1  Started Quasirent   Quasirent
 XSRAB    EYU9XLSR EYU9XLSR  EYU9XLGR     0        0        1  Started Threadsafe  Threadsafe
 XRSINDI  EYU9NLID EYU9NLID  EYU9NXSD     0        0        1  Started Threadsafe  Threadsafe
 XSNOFF   EYU9NLSO EYU9NLSO  EYU9NXSD     0        0        1  Started Threadsafe  Threadsafe
 XRMIIN   RMIXIT   RMIXIT    RMIXIT   2,008        1        2  Started Quasirent   Quasirent
 XRMIOUT   RMIXIT    RMIXIT     RMIXIT     2,008        1        2   Started  Quasirent    Quasirent
```

The report shows that program RMIXIT is in use at two exit points, XRMIIN and XRMIOUT, which means that it will start twice for each DB2 call. Both exits share the GWA, using the first two words as a header to communicate between programs.

Finding installed global user exits by using CECI INQ EXITPROGRAM

We also used CECI INQ EXITPROGRAM to show how the program is defined to the system and to show the GWA that we are using. Figure 6-15 shows that exit program RMIXIT is started at exit point XRMIIN. A GWA of 2008 bytes is used.

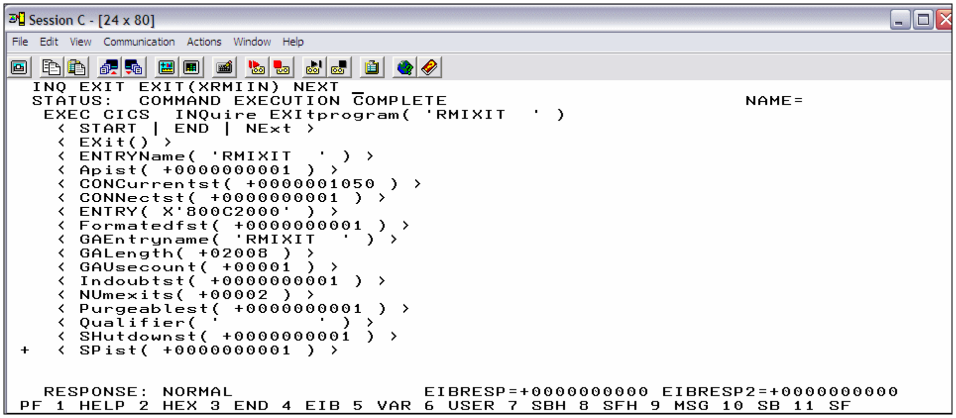


Figure 6-15 CECI INQ EXITPROGRAM EXIT(XRMIIN)

Figure 6-16 shows that the same exit program RMIXIT is also enabled at exit point XRMIOUT, using the same shared GWA.

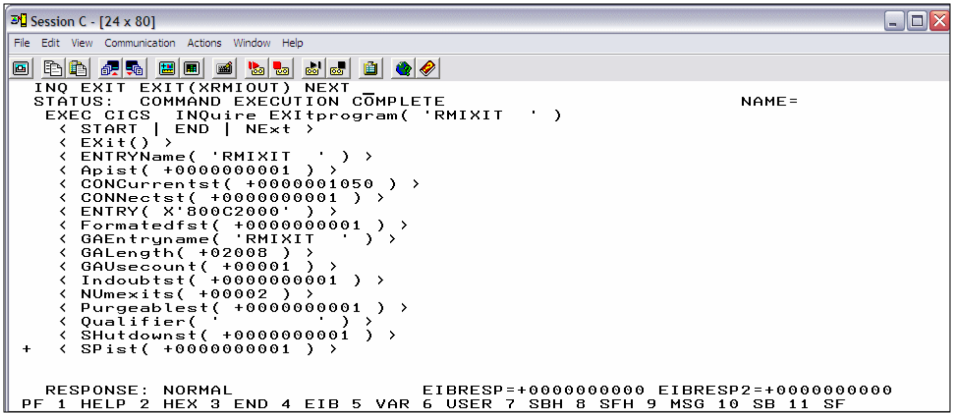


Figure 6-16 CECI INQ EXITPROGRAM EXIT(XRMIOUT)

Figure 6-15 and Figure 6-16 show that program RMIXIT is in use at two exit points, XRMIIN and XRMIOUT, which means that it is started twice for each DB2 call. Both exits share the GWA, using the first two words as a header to communicate between programs.

QUASIRENT results running on the QR TCB

We initially defined the program as QUASIRENT to show that the program runs successfully on the QR TCB.

Figure 6-17 shows the header format used by RMIXIT. The first word is the next available storage address in the GWA to be updated, and the second word is the number of words updated in the GWA.

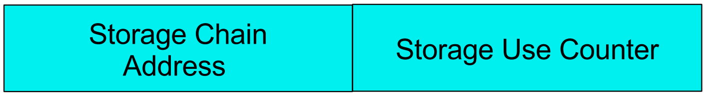


Figure 6-17 Sample header format

When running in QUASIRENT mode on the QR TCB, you can see that the last address updated was 00412B8 (Example 6-9). If you look down to the end of the storage area, you can see that the word at 00412BC is all zeros because it is not used yet.

The third word of the storage area is always skipped and left blank. Therefore, technically you can say that the header is three words. Starting at 004103C, program RMIXIT inserted sequential words of ones all the way up to and including address 00412B8.

The storage use counter has a hex value of A0, which translates to decimal 160. When RMIXIT starts, the header address is initialized to 0041038, so that `00412B8 - 0041038 = 000280`, which translates to decimal 640. Then dividing 640 by 4 gives you 160 words updated (4 bytes per word) (`640 / 4 = 160`).

Example 6-9 Results of running the nonthreadsafe GLUE defined as QUASIRENT

```
0041030   00000C   000412B8 000000A0 00000000 11111111
0041040   00001C   11111111 11111111 11111111 11111111
0041050   00002C   11111111 11111111 11111111 11111111
0041060   00003C   11111111 11111111 11111111 11111111
0041070   00004C   11111111 11111111 11111111 11111111
0041080   00005C   11111111 11111111 11111111 11111111
0041090   00006C   11111111 11111111 11111111 11111111
00410A0   00007C   11111111 11111111 11111111 11111111
00410B0   00008C   11111111 11111111 11111111 11111111
00410C0   00009C   11111111 11111111 11111111 11111111
00410D0   0000AC   11111111 11111111 11111111 11111111
00410E0   0000BC   11111111 11111111 11111111 11111111
00410F0   0000CC   11111111 11111111 11111111 11111111
0041100   0000DC   11111111 11111111 11111111 11111111
0041110   0000EC   11111111 11111111 11111111 11111111
0041120   00000C   11111111 11111111 11111111 11111111
0041130   00001C   11111111 11111111 11111111 11111111
```

```
0041140   00002C   11111111 11111111 11111111 11111111
0041150   00003C   11111111 11111111 11111111 11111111
0041160   00004C   11111111 11111111 11111111 11111111
0041170   00005C   11111111 11111111 11111111 11111111
0041180   00006C   11111111 11111111 11111111 11111111
0041190   00007C   11111111 11111111 11111111 11111111
00411A0   00008C   11111111 11111111 11111111 11111111
00411B0   00009C   11111111 11111111 11111111 11111111
00411C0   0000AC   11111111 11111111 11111111 11111111
00411D0   0000BC   11111111 11111111 11111111 11111111
00411E0   0000CC   11111111 11111111 11111111 11111111
00411F0   0000DC   11111111 11111111 11111111 11111111
0041200   0000EC   11111111 11111111 11111111 11111111
0041210   00000C   11111111 11111111 11111111 11111111
0041220   00001C   11111111 11111111 11111111 11111111
0041230   00002C   11111111 11111111 11111111 11111111
0041240   00003C   11111111 11111111 11111111 11111111
0041250   00004C   11111111 11111111 11111111 11111111
0041260   00005C   11111111 11111111 11111111 11111111
0041270   00006C   11111111 11111111 11111111 11111111
0041280   00007C   11111111 11111111 11111111 11111111
0041290   00008C   11111111 11111111 11111111 11111111
00412A0   00009C   11111111 11111111 11111111 11111111
00412B0   0000AC   11111111 11111111 11111111 00000000
```

THREADSAFE results of running on an L8 TCB

Running the same exit with no modifications but redefining it as THREADSAFE shows that we exposed an underlying data integrity problem.

The RMIXIT program does not contain any EXEC CICS commands that can move it off the L8 TCB. Therefore, after it is defined as THREADSAFE to CICS, it always runs on an L8 TCB. If a programmer performed a quick code review, someone might think the code is threadsafe and allow it to be defined as THREADSAFE. However, as you can see in Example 6-10, the data can become corrupted and not be realized for a while.

Example 6-10   Results of running the nonthreadsafe GLUE defined as THREADSAFE

```
0041030   00000C   00041104 00000033 00000000 11111111
0041040   00001C   11111111 11111111 11111111 11111111
0041050   00002C   11111111 00000000 00000000 00000000
0041060   00003C   11111111 00000000 11111111 00000000
0041070   00004C   11111111 11111111 11111111 11111111
0041080   00005C   00000000 00000000 00000000 00000000
0041090   00006C   11111111 11111111 11111111 11111111
00410A0   00007C   11111111 11111111 11111111 11111111
00410B0   00008C   11111111 11111111 11111111 11111111
```

```
00410C0   00009C   11111111 11111111 11111111 11111111
00410D0   0000AC   11111111 11111111 11111111 11111111
00410E0   0000BC   11111111 11111111 11111111 11111111
00410F0   0000CC   11111111 11111111 11111111 11111111
0041100   0000DC   11111111 11111111 00000000 00000000
0041110   0000EC   00000000 00000000 00000000 00000000
```

We added a loop in the middle of the program to slow it down to generate the contention. Without the loop, the program seems to run as expected, meaning it might run this way for years and then, all of a sudden, corrupt some data.

6.6.2 Threadsafe code example

To make the code threadsafe in regard to data integrity, we make a few code changes. We review the necessary changes, but for now, we provide the complete code snippet with the adjustments (Example 6-11). In the example, we use the COMPARE AND SWAP method to serialize the storage. Because the header is two words long, we used the COMPARE DOUBLE AND SWAP instruction.

By using the COMPARE DOUBLE AND SWAP method, you can read the data and update it. Then, a single instruction that is serialized across all processors in the logical partition (LPAR) does a final compare against the storage. It verifies that what you originally read in is still in storage. Then it stores the new changed results or fails and makes you try it again by using a code loop.

Example 6-11 GLUE with threadsafe code

```
RMIXIT   DFHEIENT
RMIXIT   AMODE 31
RMIXIT   RMODE ANY
         LR    R2,R1               DFHUEPAR PLIST PROVIDED BY CALLER
         USING DFHUEPAR,R2         ADDRESS UEPAR PLIST
         L     R4,UEPGAA           GET GWA ADDRESS
         LA    R4,12(R4)           BUMP TO A DOUBLE WORD ADDR
         USING GWA,R4              ADDRESSABILITY
*
         LM   R6,R7,0(R4)          LOAD PGM ADDR AND CTR
AGAIN    EQU  *
         LA   R8,4(R6)             BUMP SAVED PGM ADDR BY 4
         LA   R9,1(R7)             BUMP CTR BY 1
         L    R5,LOOPCTR           DELAY LOOP TO GET SOME OVERLAP
LOOP     EQU  *                     SO THAT WE CAN GENERATE SOME
         BCT  R5,LOOP               TCB CONTENTION
         CDS  R6,R8,0(R4)          SAVE DATA USING THD SAFE CMD
         BC   7,AGAIN              THD SAFE COMP LOOP
         L    R7,0(R8)             LOAD DATA AT PGM ADDR
```

```
         X    R7,ONES                FLIP THE BITS
         ST   R7,0(R8)               STORE THE DATA AT PGM ADDR
*
         LA    R15,UERCNORM          SET OK RESPONSE
         ST    R15,RETCODE             IN WORKING STORAGE
RETURN   EQU   *
         L     R15,RETCODE           FETCH RETURN CODE
         DFHEIRET RCREG=15           RETURN TO CICS
*********************************************************************
          DC    F'0'
ONES      DC    X'11111111'          ONES
LOOPCTR   DC    F'00777777'          TIME DELAY LOOP
*********************************************************************
         LTORG
         END   RMIXIT
```

QUASIRENT results running on the QR TCB

Running the fully threadsafe version of the exit in QUASIRENT mode worked as we expected. Therefore, we do not show the results, and move on to the real test.

THREADSAFE results of running on an L8 TCB

We redefined the RMIXIT program as THREADSAFE, disabled it, copied it, and re-enabled the GLUE at XRMIIN and XRMIOUT for another test.

By running in THREADSAFE mode on L8 TCBs, the program now accesses the single shared GWA from multiple programs running concurrently. The data integrity is maintained due to the COMPARE DOUBLE AND SWAP logic we added to the program. We can now run the new RMIXIT in any mode with the knowledge that we are not going to corrupt any data.

By comparing the first two words of data with the run in Example 6-9 on page 139, Example 6-12 shows that our count is again correct at 0A0 and that the next address is again 00412B8.

Example 6-12 Results of running the threadsafe GLUE defined as THREADSAFE

```
0041030   00000C   000412B8 000000A0 00000000 11111111
0041040   00001C   11111111 11111111 11111111 11111111
0041050   00002C   11111111 11111111 11111111 11111111
0041060   00003C   11111111 11111111 11111111 11111111
0041070   00004C   11111111 11111111 11111111 11111111
0041080   00005C   11111111 11111111 11111111 11111111
0041090   00006C   11111111 11111111 11111111 11111111
00410A0   00007C   11111111 11111111 11111111 11111111
00410B0   00008C   11111111 11111111 11111111 11111111
00410C0   00009C   11111111 11111111 11111111 11111111
00410D0   0000AC   11111111 11111111 11111111 11111111
```

```
00410E0   0000BC   11111111 11111111 11111111 11111111
00410F0   0000CC   11111111 11111111 11111111 11111111
0041100   0000DC   11111111 11111111 11111111 11111111
0041110   0000EC   11111111 11111111 11111111 11111111
0041120   00000C   11111111 11111111 11111111 11111111
0041130   00001C   11111111 11111111 11111111 11111111
0041140   00002C   11111111 11111111 11111111 11111111
0041150   00003C   11111111 11111111 11111111 11111111
0041160   00004C   11111111 11111111 11111111 11111111
0041170   00005C   11111111 11111111 11111111 11111111
0041180   00006C   11111111 11111111 11111111 11111111
0041190   00007C   11111111 11111111 11111111 11111111
00411A0   00008C   11111111 11111111 11111111 11111111
00411B0   00009C   11111111 11111111 11111111 11111111
00411C0   0000AC   11111111 11111111 11111111 11111111
00411D0   0000BC   11111111 11111111 11111111 11111111
00411E0   0000CC   11111111 11111111 11111111 11111111
00411F0   0000DC   11111111 11111111 11111111 11111111
0041200   0000EC   11111111 11111111 11111111 11111111
0041210   00000C   11111111 11111111 11111111 11111111
0041220   00001C   11111111 11111111 11111111 11111111
0041230   00002C   11111111 11111111 11111111 11111111
0041240   00003C   11111111 11111111 11111111 11111111
0041250   00004C   11111111 11111111 11111111 11111111
0041260   00005C   11111111 11111111 11111111 11111111
0041270   00006C   11111111 11111111 11111111 11111111
0041280   00007C   11111111 11111111 11111111 11111111
0041290   00008C   11111111 11111111 11111111 11111111
00412A0   00009C   11111111 11111111 11111111 11111111
00412B0   0000AC   11111111 11111111 11111111 00000000
```

6.6.3 Code changes to make RMIXIT threadsafe

First we break down the code by function so that the new changes make sense (Example 6-13). In the example, we used the COMPARE DOUBLE AND SWAP instruction to serialize the data. We can also use the ENQ/DEQ method, as described in 5.1.2, “Example showing the use of shared resources” on page 86.

Example 6-13 Code broken down into function

```
         (A) Load the two word header

        L    R6,0(R4)              LOAD SAVED PGM ADDR
        L    R7,4(R4)              LOAD CTR

         (B) Update the Header Address value and counter
```

```
         LA   R8,4(R6)              BUMP SAVED PGM ADDR BY 4
         LA   R9,1(R7)              BUMP CTR BY 1

          (C) Artificial loop used to simulate real program workload

         L    R5,LOOPCTR            DELAY LOOP TO GET SOME OVERLAP
LOOP     EQU  *                      SO THAT WE CAN GENERATE SOME
         BCT  R5,LOOP                TCB CONTENTION

          (D) Save the updated header back into the shared storage

         ST   R8,0(R4)              STORE PGM ADDR AT HEADER ADDR
         ST   R9,4(R4)              STORE THE CTR AT WORD 2 IN HEADER

          (E) Store Ones into the shared storage using the header address value

         L    R7,0(R8)              LOAD DATA AT PGM ADDR
         X    R7,ONES               FLIP THE BITS
         ST   R7,0(R8)              STORE THE DATA AT PGM ADDR
```

Sections (A) and (D) must be modified. Sections (B) and (E) stay the same. Section (C) is an artificial loop added to the code to create real-world processing time delays to generate concurrent TCB contention.

Section (A) loads the header from shared storage by using two load instructions. For our case only, use a load multiple to load both registers without introducing a window between the two loads where the data can be changed, as indicated in *z/Architecture Principles of Operation*, SA22-7832. See Example 6-14.

Example 6-14 Modifying section (A), loading the header

```
LM   R6,R7,0(R4)           LOAD PGM ADDR AND CTR
```

In section (D), we convert the two store instructions into a single compare double and swap instruction (Example 6-15).

Example 6-15 Modifying section (D), saving the updated header

```
AGAIN    EQU  *
         LA   R8,4(R6)              BUMP SAVED PGM ADDR BY 4
         LA   R9,1(R7)              BUMP CTR BY 1
         L    R5,LOOPCTR            DELAY LOOP TO GET SOME OVERLAP
LOOP     EQU  *                      SO THAT WE CAN GENERATE SOME
         BCT  R5,LOOP                TCB CONTENTION
         CDS  R6,R8,0(R4)           SAVE DATA USING THD SAFE CMD
         BC   7,AGAIN               THD SAFE COMP LOOP
```

The COMPARE AND SWAP instruction compares what was originally loaded with what is in storage. Based on the result, it does one of following actions:

- If the original data is unchanged, it stores the new updates, as in registers 8 and 9, in the storage location.
- If the original data changed, it reloads registers 6 and 7 with the new values from storage.

You then check the return codes from the COMPARE AND DOUBLE SWAP command and branch accordingly. In our case, for option 1, we drop through the code into unchanged section (E) to store our ones into memory. The address we have is already locked in and is ours, so that we can perform this function afterward.

For option 2, the COMPARE AND DOUBLE SWA instruction simulates section (A). Therefore, we must go backwards in the code, redo our updates, and then try to store our data again. Notice that this process is coded as an infinite loop, which can use all resources and hang the CPU. It might be cleaner to put a loop counter in there and abend the transaction if it cannot serialize the data. However, because it was difficult getting contention, we felt it was a low chance to go into an infinite loop.

6.7 Coordinating and driving individual application conversions

After you convert all appropriate GLUEs, TRUEs, URMs, and exits, next in the conversion process is to convert the application programs themselves. Depending on how your environment is set up, you might have a varying role in helping to coordinate the application conversions to threadsafe applications.

Your key role might be to identify which CICS region is ready for the conversion. Alternatively, in your region-by-region analysis, you might have collected statistics on how many TCB switches are occurring for individual application programs. Armed with performance data on TCB switches, you might be the key person to help identify the application conversion selection order.

Obviously applications that perform large amounts of DB2 calls, as opposed to single table lookups, benefit the most from the conversion. The same principle applies to applications with large volumes of WMQ calls.

By using such tools as CICS Performance Analyzer (CICS PA), the systems programmer can help identify which applications are the best candidates for conversion first.

For a description of the process to make applications threadsafe, see Chapter 5, “Application review” on page 83.

6.7.1 Changing your program definitions

After the applications are changed or verified to be threadsafe, the final step to make the application stay on the L8 TCB is to change the definition of all the programs concerned to define them as threadsafe. If the application program must start directly on an open L8 TCB, you can use the `REQUIRED` option on the **`CONCURRENCY`** parameter (Figure 6-18).

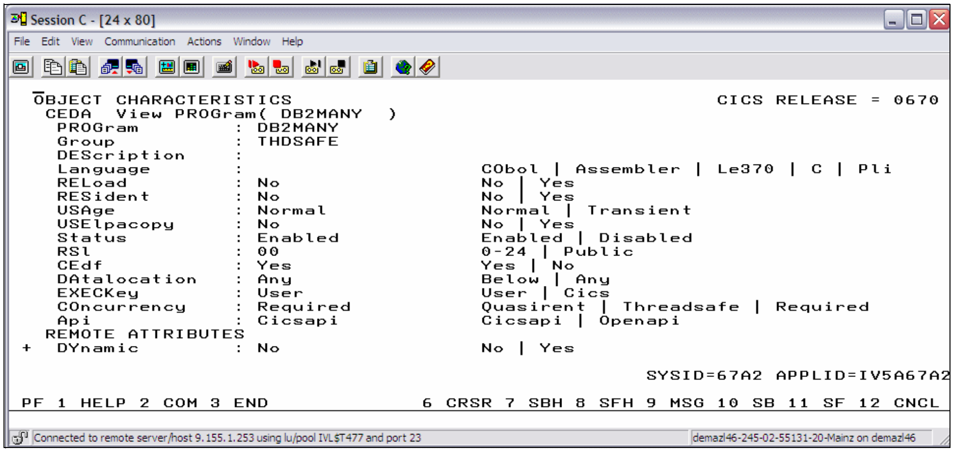


Figure 6-18 Changing program definitions

For more information about changing a the concurrency definition of a program, see 6.5.3, “Changing the CONCURRENCY definition of the exit program to THREADSAFE” on page 133.

6.8 Post-conversion monitoring

The concept of making a program threadsafe can seem simple. However, it can be complex. Identifying which EXEC CICS commands might cause a program to have excessive TCB switching is straightforward. You can look up a list of all the threadsafe commands, search your code, and then make the appropriate adjustments.

Making a program threadsafe is much harder because you must first identify any shared resources. This process can be disguised because the address of shared storage is obtained from a commarea passed into the program.

No tool or process is available for you to monitor for changes in application programs. However, you can periodically monitor your region for TCB switches using tools, such as CICS PA. A programmer can introduce a nonthreadsafe EXEC CICS command into a program that was already converted and, therefore, introduce extra TCB switches.

To help address nonthreadsafe EXEC CICS commands, you might want to alter any existing performance reports that you currently run to add change mode counts to your reports if you can identify any changes in TCB switching.

6.9 Summary

The system programmer is responsible for making the CICS environment threadsafe so that application programs can take advantage of the performance benefits of threadsafe DB2, IMS, and WebSphere MQ applications.

To make the CICS environment threadsafe, the system programmer must complete the following tasks:

- Review the DB2 version and system parameters.
- Review the environment and system parameters for WebSphere MQ.
- Review the CICS DBCTL environment and system parameters
- Review and adjust the CICS system parameters.
- Review and convert any GLUEs in the DB2, IMS, and WMQ call paths to threadsafe standards.
- Review and convert any GLUEs on the path of threadsafe EXEC CICS commands, particularly for heavily used API options such as file control.
- Assist application programmers with analyzing their programs by using utilities, such as the **DFHOSTAT** and **DFHEISUP** utilities, or tools, such as CICS IA and CICS PA.
- Potentially work with the application teams to help prioritize their application threadsafe migrations.
- Convert the actual program definition changes to **CONCURRENCY(THREADSAFE** or **REQUIRED)**.
- Modify the autoinstall program to change the **CONCURRENCY** value if used or to use the environment variable CICSVAR.

Additionally, system programmers might perform periodic reviews of their CICS regions using such tools as CICS PA to monitor the L8 to QR TCB statistics, checking to see whether applications are really running on the L8 TCBs.

7

Threadsafe conversion considerations

This chapter highlights several pitfalls that you might encounter during a threadsafe conversion project. It includes the following sections:

- Global user exit considerations
- Migrating WebSphere MQ regions
- Threadsafe program definition considerations
- Function shipped commands
- COBOL calls
- The CSACDTA field
- Ensuring CICS performance and capacity
- Diagnosing performance problems

© Copyright IBM Corp. 2004, 2012. All rights reserved.

7.1 Global user exit considerations

Before you start converting your DB2, WebSphere MQ (WMQ), or CICS IMS Database Control (DBCTL) application to run threadsafe enabled, investigate whether any RMI global user exits (GLUEs) are in use. User exits XRMIIN and XRMIOUT must be defined as threadsafe to avoid any adverse effects because of excessive TCB Mode switches. The RMI exit is driven for DB2-SQL, WebSphere MQ, and CICS DBCTL requests.

7.1.1 A potential pitfall

The CICS DB2 adapter includes the task-related user exit (TRUE) DFHD2EX1. This TRUE is THREADSAFE and is automatically enabled with the OPENAPI option on the `ENABLE PROGRAM` command during the connect process. If your program is defined as THREADSAFE (not OPENAPI), when your program makes a DB2 call, and it is running on the quasi-reentrant (QR) task control block (TCB), CICS switches the task to an OPEN L8 TCB. CICS performs a TCB switch from the QR TCB to the L8 TCB.

If your program is defined as QUASIRENT and you are running exits XRMIIN, XRMIOUT, or a dynamic plan exit enabled as QUASIRENT, you might experience additional TCB switches back to the QR TCB. You can easily avoid these switches if the exits are written to threadsafe standards and enabled as THREADSAFE.

The following scenario shows the program flow and TCB switches of a program making one DB2 call. The sample shows the application running in CICS Transaction Server for z/OS (CICS TS) V3 or V4, with the XRMIIN and XRMIOUT exits and a dynamic plan exit all enabled as QUASIRENT.

DB2 application in CICS Transaction Server Versions 3 and 4

Figure 7-1 on page 151 shows the same transaction, TRANA, running in a CICS TS V3 or V4 environment and making one DB2 call. The transaction was migrated to CICS TS V3 or V4 with *no* consideration of threadsafety. That is, the program associated with transaction TRANA was defined as QUASIRENT and that all exits were enabled as QUASIRENT.

This diagram shows a potential to experience additional TCB switches from the L8 TCB to the QR TCB and back. The nonthreadsafe exits must run on the QR TCB to ensure that serialization occurs. In this example, eight TCB switches occur. If the exits are written to threadsafe standards and then enabled as THREADSAFE, their associated programs are allowed to continue running on the L8 TCB. Also, the additional switches are not necessary as shown in the next section.

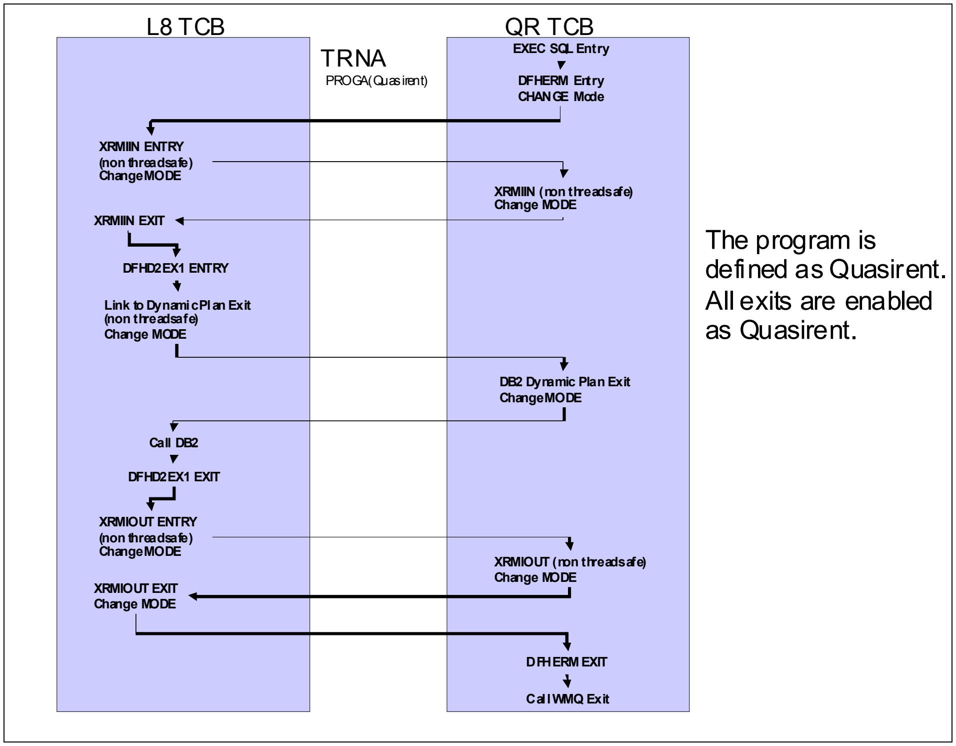


Figure 7-1 TCB switches before exits enabled as threadsafe on CICS TS V3 and V4

Example 7-1 is a CICS auxiliary trace showing the additional TCB switches shown in Figure 7-1.

Example 7-1 CICS trace of potential switches with nonthreadsafe exits

```
00258 QR    AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                   =003356=
00258 QR    US 0401 USXM  ENTRY INQUIRE_TRANSACTION_USER                                                       =003357=
00258 QR    US 0402 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK 00000000                                           =003358=
00258 QR    RM 0301 RMLN  ENTRY ADD_LINK              RMI,22F914A4 , 00000000 , 00000008,000949D0 , 00000000 , 00000008,22F =003359=
00258 QR    RM 0302 RMLN  EXIT  ADD_LINK/OK           01C80006,22F914A4 , 00000000 , 00000008,000949D0 , 00000000 , 0000000 =003360=
00258 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           0000000C                                                 =003361=
00258 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                 =003369=
00258 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIIN                          =003370=
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE           QR                                                       =003371=
00258 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                 =003377=
00258 QR    SM 0C01 SMMG  ENTRY GETMAIN               198,YES,00,TASK                                          =003378=
00258 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788                                                 =003379=
00258 QR    SM 0D01 SMMF  ENTRY FREEMAIN              226E1788                                                 =003380=
00258 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK           USER storage at 226E1788                                 =003381=
00258 QR    AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                        =003382=
00258 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           L8                                                       =003383=
00258 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                 =003384=
00258 L8000 AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL SELECT                                  =003385=
00258 L8000 PG 0A01 PGLU  ENTRY LINK_URM              DB2PLAN,22BE7678 , 0000001C,NO,NO                        =003386=
00258 L8000 DD 0301 DDLO  ENTRY LOCATE                21C27B70,22BE7574,PPT,DB2PLAN                            =003387=
00258 L8000 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 22DA5B88                                      =003388=
00258 L8000 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       22C87258                                                 =003389=
00258 L8000 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK    A43002D8,243002B0,D8,0,REUSABLE,ESDSA,OLD_COPY           =003390=
```

```
00258 L8000 AP 1940 APLI  ENTRY START_PROGRAM        DB2PLAN,CEDF,FULLAPI,URM,NO,22F89C10,22BE7678 , 0000001C,3           =003391=
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE          QR                                                                   =003392=
00258 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                            =003393=
00258 QR    SM 0C01 SMMG  ENTRY GETMAIN              190,YES,00,TASK                                                      =003394=
00258 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                             =003395=
00258 QR    AP 00E1 EIP   ENTRY RETURN                                            0004,226E1798 .>.q,08000E08 ....        =003396=
00258 QR    AP E160 EXEC  ENTRY RETURN               ASM                                                                  =003397=
00258 QR    SM 0301 SMGF  ENTRY FREEMAIN             226E1788,TASK                                                        =003398=
00258 QR    SM 0302 SMGF  EXIT  FREEMAIN/OK                                                                               =003399=
00258 QR    AP 1941 APLI  EXIT  START_PROGRAM/OK     ....,NO,DB2PLAN                                                      =003400=
00258 QR    LD 0001 LDLD  ENTRY RELEASE_PROGRAM      22C87258,A43002D8                                                    =003401=
00258 QR    LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK   243002B0,D8,ESDSA                                                    =003402=
00258 QR    DS 0002 DSAT  ENTRY CHANGE_MODE          L8                                                                   =003403=
00258 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK       QR                                                                   =003404=
00258 L8000 PG 0A02 PGLU  EXIT  LINK_URM/OK                                                                               =003405=
00258 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL         230D7030                                                             =003406=
00258 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                           =003407=
00258 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE 0 RETURNED ON EXEC SQL SELECT                                =003408=
00258 L8000 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME  DSNCSQL                                                              =003409=
00258 L8000 MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                    =003410=
00258 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIOUT                                    =003411=
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE          QR                                                                   =003412=
00258 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                            =003413=
00258 QR    SM 0C01 SMMG  ENTRY GETMAIN              198,YES,00,TASK                                                      =003414=
00258 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                             =003415=
00258 QR    SM 0D01 SMMF  ENTRY FREEMAIN             226E1788                                                             =003416=
00258 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK          USER storage at 226E1788                                             =003417=
00258 QR    AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                   =003418=
00258 QR    DS 0002 DSAT  ENTRY CHANGE_MODE          L8                                                                   =003419=
00258 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                            =003420=
00258 L8000 RM 0301 RMLN  ENTRY SET_LINK             01C80006,22F914AC , 0000000C , 00000008,YES,NECESSARY                =003421=
00258 L8000 RM 0302 RMLN  EXIT  SET_LINK/OK          22F914AC , 0000000C , 00000008,                                      =003422=
00258 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE          00000001                                                             =003423=
00258 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                            =003424=
00258 QR    AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                              =003425=
```

7.1.2 The solution

To demonstrate how to avoid the pitfall described in 7.1.1, "A potential pitfall" on page 150, consider the following scenarios:

- The first scenario shows the effect of having only the exits on the DB2 call path written to threadsafe standards and enabled as threadsafe (XRMIIN and XRMIOUT exits and the dynamic plan exit).
- The second scenario shows the benefit that threadsafe offers. *Both* the application program and programs are associated with all the exits on the DB2 call path to threadsafe standards, and they are defined as THREADSAFE.

Enabling exits on the DB2 call path to be THREADSAFE

Figure 7-2 on page 153 shows the TRNA transaction running in a CICS TS V3 or V4 environment and making one DB2 call. The transaction was migrated to CICS TS V3 or V4 *with* threadsafe considerations in mind.

The program associated with TRANA transaction is still defined as Quasirent. However, XRMIIN, XRMIOUT, and the dynamic plan exits are coded to threadsafe standards and then enabled as THREADSAFE. Figure 7-2 on page 153 shows that the number of TCB switches is reduced to just two switches. However, a TCB switchback to the QR TCB must still take place upon completion of the DB2 call because the TRNA program is not threadsafe. Therefore, each DB2 call has two TCB switches.

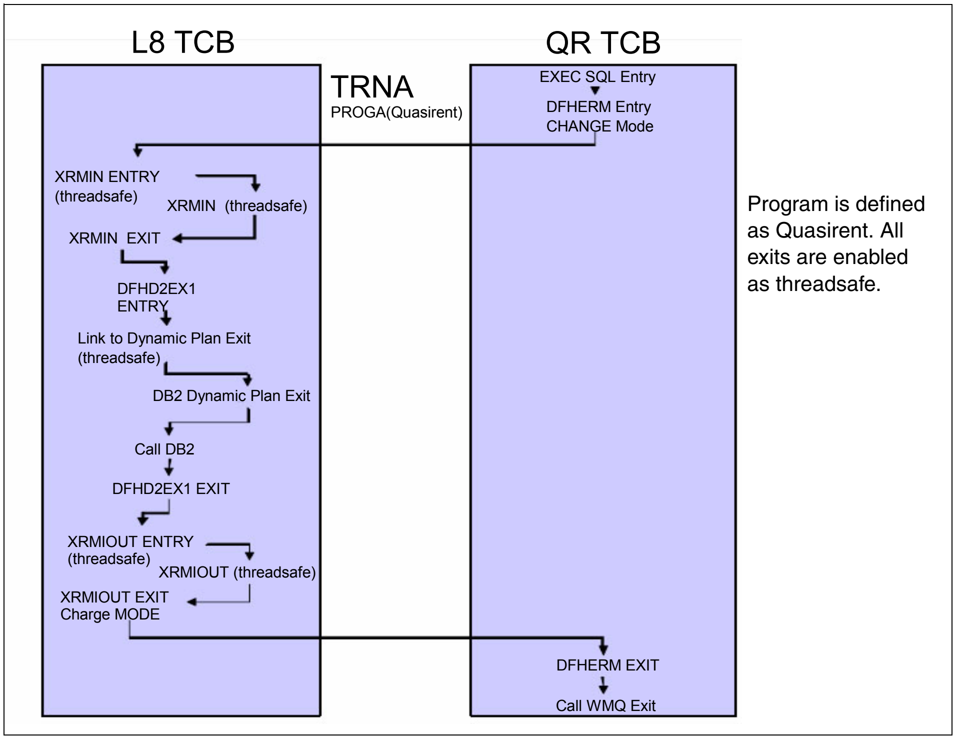


Figure 7-2 TCB switches after exits are made threadsafe on CICS TS V3 and V4

Example 7-2 shows a CICS auxiliary trace that demonstrates the TCB switches shown in Figure 7-2.

Example 7-2 CICS trace of TCB switches with threadsafe exits

```
00307 QR    AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                          =000266=
00307 QR    US 0401 USXM  ENTRY INQUIRE_TRANSACTION_USER                                                                 =000267=
00307 QR    US 0402 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK 00000000                                                     =000268=
00307 QR    RM 0301 RMLN  ENTRY ADD_LINK              RMI,22F914A4 , 00000000 , 00000008,000949D0 , 00000000 , 00000008,22F =000269=
00307 QR    RM 0302 RMLN  EXIT  ADD_LINK/OK           01C80011,22F914A4 , 00000000 , 00000008,000949D0 , 00000000 , 0000000 =000270=
00307 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           0000000C                                                           =000271=
00307 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                           =000279=
00307 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIIN                                    =000280=
00307 L8000 SM 0C01 SMMG  ENTRY GETMAIN               198,YES,00,TASK                                                    =000281=
00307 L8000 SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788                                                           =000282=
00307 L8000 SM 0D01 SMMF  ENTRY FREEMAIN              226E1788                                                           =000283=
00307 L8000 SM 0D02 SMMF  EXIT  FREEMAIN/OK           USER storage at 226E1788                                           =000284=
00307 L8000 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                  =000285=
00307 L8000 AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL SELECT                                            =000286=
00307 L8000 PG 0A01 PGLU  ENTRY LINK_URM              DB2PLAN,22BE7678 , 0000001C,NO,NO                                  =000287=
00307 L8000 DD 0301 DDLO  ENTRY LOCATE                21C27B70,22BE7574,PPT,DB2PLAN                                      =000288=
00307 L8000 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 22DA5B30                                                =000289=
```

```
00307 L8000 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       22C871A0                                                          =000290=
00307 L8000 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK    A43002D8,243002B0,D8,0,REUSABLE,ESDSA,OLD_COPY                    =000291=
00307 L8000 AP 1940 APLI  ENTRY START_PROGRAM         DB2PLAN,CEDF,FULLAPI,URM,NO,22F89A94,22BE7678 , 0000001C,3        =000292=
00307 L8000 SM 0C01 SMMG  ENTRY GETMAIN               190,YES,00,TASK                                                   =000293=
00307 L8000 SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788                                                          =000294=
00307 L8000 AP 00E1 EIP   ENTRY RETURN                                            0004,226E1798 .>.q,08000E08 ....      =000295=
00307 L8000 AP E160 EXEC  ENTRY RETURN                ASM                                                               =000296=
00307 L8000 SM 0301 SMGF  ENTRY FREEMAIN              226E1788,TASK                                                     =000297=
00307 L8000 SM 0302 SMGF  EXIT  FREEMAIN/OK                                                                             =000298=
00307 L8000 AP 1941 APLI  EXIT  START_PROGRAM/OK      ....,NO,DB2PLAN                                                   =000299=
00307 L8000 LD 0001 LDLD  ENTRY RELEASE_PROGRAM       22C871A0,A43002D8                                                 =000300=
00307 L8000 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK    243002B0,D8,ESDSA                                                 =000301=
00307 L8000 PG 0A02 PGLU  EXIT  LINK_URM/OK                                                                             =000302=
00307 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL          230D7030                                                          =000303=
00307 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                         =000309=
00307 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT                             =000310=
00307 L8000 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                           =000311=
00307 L8000 MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                  =000312=
00307 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIOUT                                  =000313=
00307 L8000 SM 0C01 SMMG  ENTRY GETMAIN               198,YES,00,TASK                                                   =000314=
00307 L8000 SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788                                                          =000315=
00307 L8000 SM 0D01 SMMF  ENTRY FREEMAIN              226E1788                                                          =000316=
00307 L8000 SM 0D02 SMMF  EXIT  FREEMAIN/OK           USER storage at 226E1788                                          =000317=
00307 L8000 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                 =000318=
00307 L8000 RM 0301 RMLN  ENTRY SET_LINK              01C80011,22F914AC , 0000000C , 00000008,YES,NECESSARY             =000319=
00307 L8000 RM 0302 RMLN  EXIT  SET_LINK/OK           22F914AC , 0000000C , 00000008,                                   =000320=
00307 L8000 DS 0002 DSAT  ENTRY CHANGE_MODE           00000001                                                          =000321=
00307 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                          =000322=
00307 QR    AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                            =000323=
00307 QR    AP 00E1 EIP   ENTRY RETURN                                            0004,226E1458 .>..,08000E08 ....      =000324=
```

Enabling both the application program and all exits on the DB2 call path to be THREADSAFE

Figure 7-3 on page 155 shows the TRANA transaction running in a CICS TS V3 or V4 environment and making one DB2 call. The transaction was migrated to CICS TS V3 or V4 *with* threadsafe considerations in mind. The program associated with the TRANA transaction *and* the programs associated with XRMIIN, XRMIOUT, and the dynamic plan exits are all written to threadsafe standards and defined as THREADSAFE.

Figure 7-3 on page 155 shows a TCB switch from the QR TCB to the L8 TCB for the first DB2 call. Upon completion of the DB2 call, the program remains on the L8 TCB. The number of DB2 calls that can be made without another TCB switch is limited only by the design of the application. Only a TCB switchback to the QR TCB is needed at task termination time, unless nonthreadsafe EXEC CICS commands were issued. Starting here, you begin to see what having threadsafe programs can offer regarding potential savings in both CPU and response time.

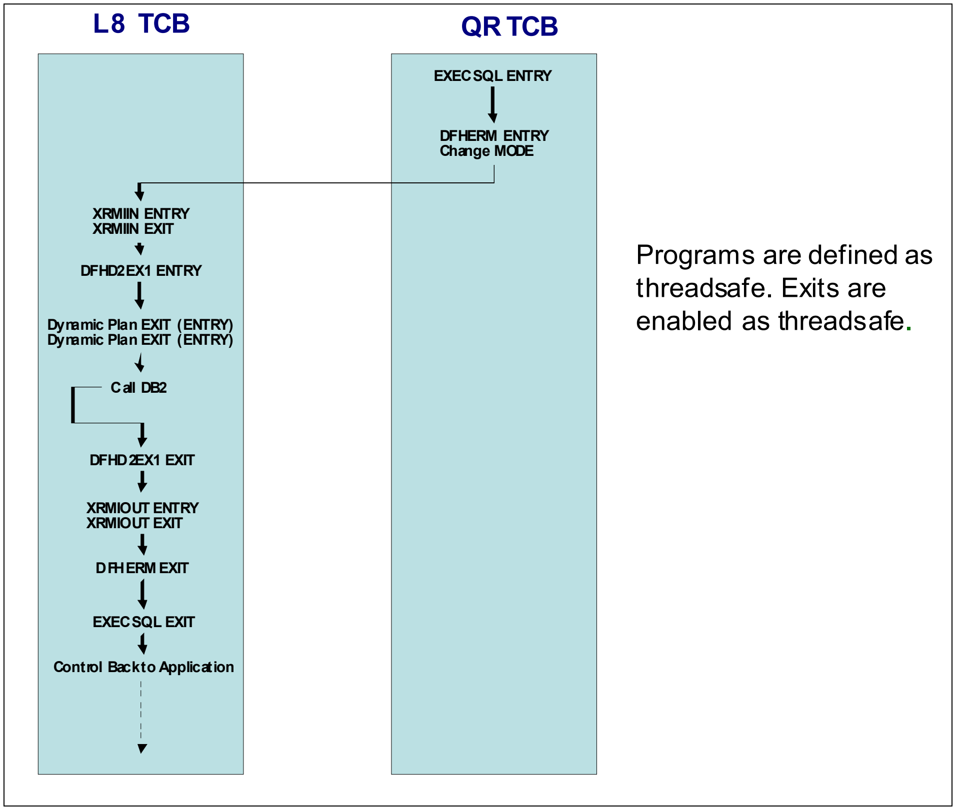


Figure 7-3 TCB switches with programs and exits as threadsafe on CICS TS V3 or V4

Example 7-3 is a CICS trace. It shows the TCB switches in Figure 7-3 after XRMIIN, XRMIOUT, the dynamic plan exit, *and* the application program associated with transaction TRNA are written to threadsafe standards and then defined or enabled as THREADSAFE.

To be consistent, Figure 7-3 shows only one DB2 call. However, the associated trace in Example 7-3 continues on, reflecting a second DB2 call. You can see that the second DB2 call runs on the L8 TCB and that no TCB switch was made.

Example 7-3 CICS trace of TCB switches with threadsafe program and exits

```
00772 QR    AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                 =000242=
00772 QR    US 0401 USXM  ENTRY INQUIRE_TRANSACTION_USER                                                     =000243=
00772 QR    US 0402 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK 00000000                                         =000244=
00772 QR    RM 0301 RMLN  ENTRY ADD_LINK              RMI,24C57CE4 , 00000000 , 00000008,000949D0 , 00000000 , 00000008,24C =000245=
00772 QR    RM 0302 RMLN  EXIT  ADD_LINK/OK           0154000C,24C57CE4 , 00000000 , 00000008,000949D0 , 00000000 , 0000000 =000246=
00772 QR    DS 0002 DSAT  ENTRY CHANGE_MODE           0000000C                                               =000247=
00772 L8001 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                               =000255=
00772 L8001 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIIN                        =000256=
00772 L8001 SM 0C01 SMMG  ENTRY GETMAIN               198,YES,00,TASK                                        =000257=
00772 L8001 SM 0C02 SMMG  EXIT  GETMAIN/OK            226E1788                                               =000258=
00772 L8001 SM 0D01 SMMF  ENTRY FREEMAIN              226E1788                                               =000259=
```

```
00772 L8001 SM 0D02 SMMF  EXIT  FREEMAIN/OK          USER storage at 226E1788                                                    =000260=
00772 L8001 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                        =000261=
00772 L8001 AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT                                                   =000262=
00772 L8001 PG 0A01 PGLU  ENTRY LINK_URM             DB2PLAN,22C1E678 , 0000001C,NO,NO                                         =000263=
00772 L8001 DD 0301 DDLO  ENTRY LOCATE               21C27B70,22C1E574,PPT,DB2PLAN                                             =000264=
00772 L8001 DD 0302 DDLO  EXIT  LOCATE/OK            D7D7E3C5 , 22DA5B30                                                       =000265=
00772 L8001 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM      22C871A0                                                                  =000266=
00772 L8001 LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK   A43002D8,243002B0,D8,0,REUSABLE,ESDSA,OLD_COPY                            =000267=
00772 L8001 AP 1940 APLI  ENTRY START_PROGRAM        DB2PLAN,CEDF,FULLAPI,URM,NO,22F89A94,22C1E678 , 0000001C,3                =000268=
00772 L8001 SM 0C01 SMMG  ENTRY GETMAIN              190,YES,00,TASK                                                           =000269=
00772 L8001 SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                                  =000270=
00772 L8001 AP 00E1 EIP   ENTRY RETURN                                            0004,226E1798 .>.q,08000E08 ....             =000271=
00772 L8001 AP E160 EXEC  ENTRY RETURN               ASM                                                                       =000272=
00772 L8001 SM 0301 SMGF  ENTRY FREEMAIN             226E1788,TASK                                                             =000273=
00772 L8001 SM 0302 SMGF  EXIT  FREEMAIN/OK                                                                                    =000274=
00772 L8001 AP 1941 APLI  EXIT  START_PROGRAM/OK     ....,NO,DB2PLAN                                                           =000275=
00772 L8001 LD 0001 LDLD  ENTRY RELEASE_PROGRAM      22C871A0,A43002D8                                                         =000276=
00772 L8001 LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK   243002B0,D8,ESDSA                                                         =000277=
00772 L8001 PG 0A02 PGLU  EXIT  LINK_URM/OK                                                                                    =000278=
00772 L8001 AP 3250 D2D2  ENTRY DB2_API_CALL         230D7030                                                                  =000279=
00772 L8001 AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                                =000285=
00772 L8001 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT                                    =000286=
00772 L8001 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                                  =000287=
00772 L8001 MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                         =000288=
00772 L8001 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIOUT                                         =000289=
00772 L8001 SM 0C01 SMMG  ENTRY GETMAIN              198,YES,00,TASK                                                           =000290=
00772 L8001 SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                                  =000291=
00772 L8001 SM 0D01 SMMF  ENTRY FREEMAIN             226E1788                                                                  =000292=
00772 L8001 SM 0D02 SMMF  EXIT  FREEMAIN/OK          USER storage at 226E1788                                                  =000293=
00772 L8001 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                        =000294=
00772 L8001 RM 0301 RMLN  ENTRY SET_LINK             0154000C,24C57CEC , 0000000C , 00000008,YES,NECESSARY                     =000295=
00772 L8001 RM 0302 RMLN  EXIT  SET_LINK/OK          24C57CEC , 0000000C , 00000008,                                           =000296=
00772 L8001 AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                                   =000297=
00772 L8001 AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                                   =000298=
00772 L8001 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIIN                                          =000299=
00772 L8001 SM 0C01 SMMG  ENTRY GETMAIN              198,YES,00,TASK                                                           =000300=
00772 L8001 SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                                  =000301=
00772 L8001 SM 0D01 SMMF  ENTRY FREEMAIN             226E1788                                                                  =000302=
00772 L8001 SM 0D02 SMMF  EXIT  FREEMAIN/OK          USER storage at 226E1788                                                  =000303=
00772 L8001 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                        =000304=
00772 L8001 AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT                                                   =000305=
00772 L8001 AP 3250 D2D2  ENTRY DB2_API_CALL         230D7030                                                                  =000306=
00772 L8001 AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                                =000307=
00772 L8001 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT                                    =000308=
00772 L8001 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                                  =000309=
00772 L8001 MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                         =000310=
00772 L8001 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIOUT                                         =000311=
00772 L8001 SM 0C01 SMMG  ENTRY GETMAIN              198,YES,00,TASK                                                           =000312=
00772 L8001 SM 0C02 SMMG  EXIT  GETMAIN/OK           226E1788                                                                  =000313=
00772 L8001 SM 0D01 SMMF  ENTRY FREEMAIN             226E1788                                                                  =000314=
00772 L8001 SM 0D02 SMMF  EXIT  FREEMAIN/OK          USER storage at 226E1788                                                  =000315=
00772 L8001 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                        =000316=
00772 L8001 AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                                   =000317=
00772 L8001 AP 00E1 EIP   ENTRY RETURN                                            0004,226E1458 .>..,08000E08 ....             =000318=
```

7.2 Migrating WebSphere MQ regions

The WMQ adapter supplied by CICS TS V3.2 is now enabled as OPENAPI by CICS. Therefore, the CICS WMQ TRUE now uses L8 TCBs, and not the eight private TCBs used by previous versions of the TRUE.

The potential for unnecessary TCB switches for WMQ applications is similar for DB2 applications. As for DB2 calls, WMQ calls invoke the RMI XRMIIN and

XRMIOUT exits. In addition, for WebSphere MQ, the API crossing exit is run before and after each WMQ call.

> **CSQCAPX definition:** The definition for the API crossing exit (CSQCAPX) is supplied by CICS in CICS system definition data set (CSD) group DFHMQ. By default, it is defined as THREADSAFE and *cannot* be changed.

Figure 7-4 shows the flow of a single WMQ call from CICS. The XRMIIN and XRMIOUT exits are defined as QUASIRENT, and the application program is defined as QUASIRENT.

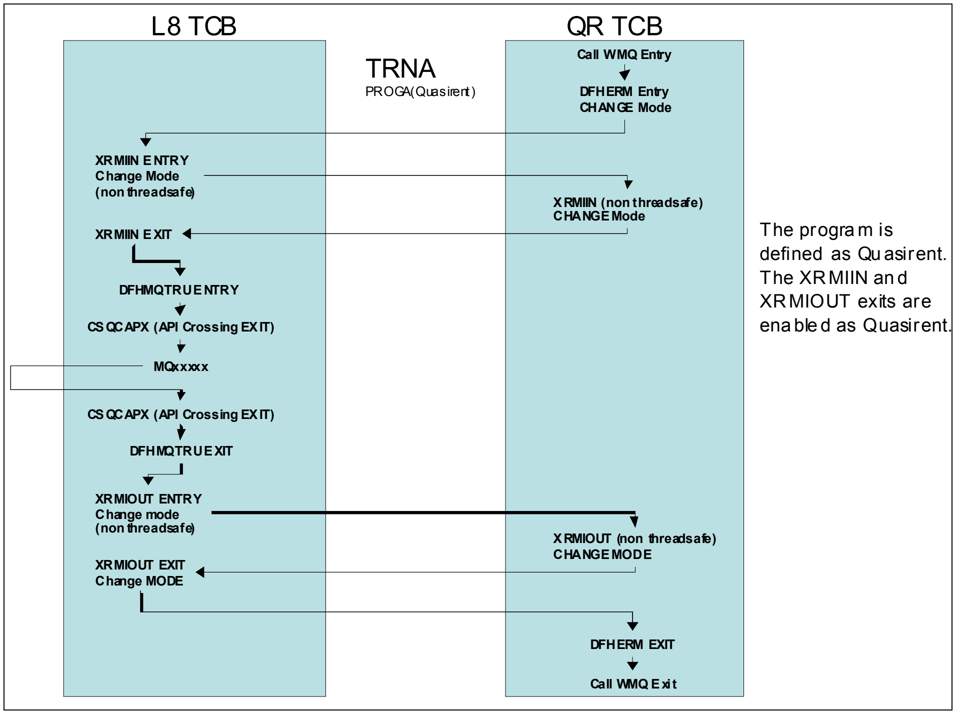


Figure 7-4 Call flow for a WMQ call with nonthreadsafe XRMIIN and XRMIOUT exits

The definition of the API crossing exit was not changed from its default of THREADSAFE. If it was changed, both calls to the crossing exit also run over on the QR TCB, adding four more TCB switches to the call.

If you enable the XRMI exits to be threadsafe, no switches occur back to the QR TCB when they are run as shown in Figure 7-5.

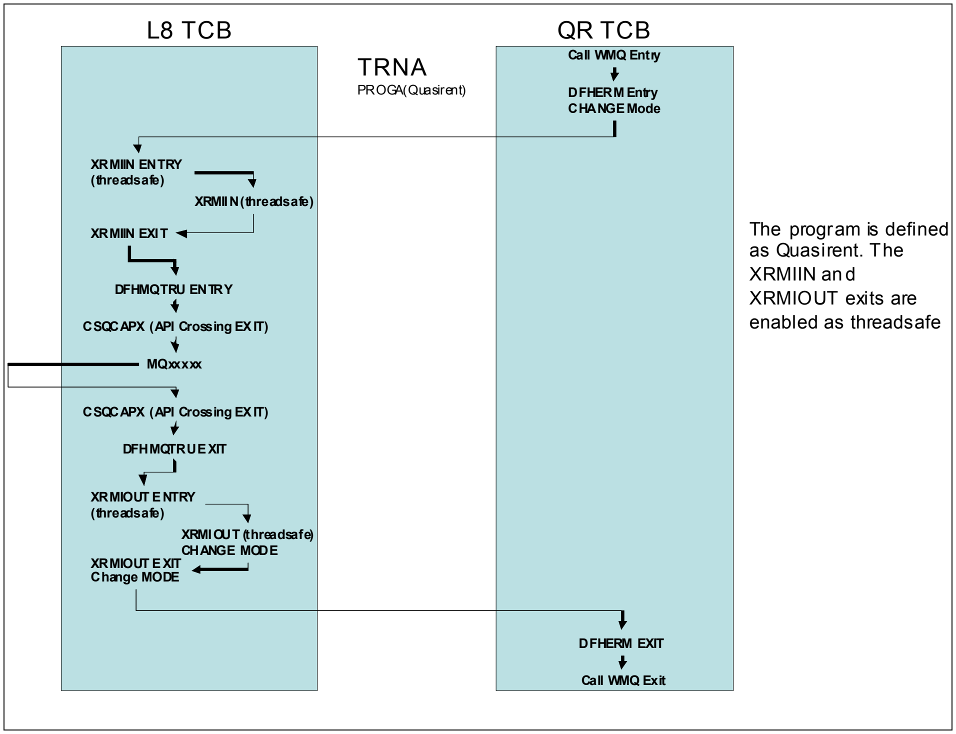


Figure 7-5 Call flow for a WMQ transaction with the XRMI exits enabled as threadsafe

Example 7-4 shows the trace of an MQPUT operation.

Example 7-4 Trace of an MQPUT operation

```
***
       ***  MQPUT
       ***
 00052 L8001 AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(MQM     )                                      =001236=
 00052 L8001 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM GENGEXIT AT EXIT POINT XRMIIN                       =001237=
 00052 L8001 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM GENGEXIT WITH RETURN CODE 790568                =001238=
 00052 L8001 AP 2522 ERM   EVENT PASSING-CONTROL-TO-OPENAPI-TRUE(MQM     )                                     =001239=
 00052 L8001 AP A090 MQTRU ENTRY APPLICATION-REQUEST   MQPUT                                                   =001240=
 00052 L8001 AP A099 MQTRU EVENT CSQCPMGH              & CSQCPMGD ABOUT TO ISSUE MQPUT                         =001241=


       ***
       ***  CSQCAPX
       ***
 00052 L8001 AP 00E1 EIP   ENTRY LINK                                            0004,266ADE6C ...%,08000E02 ....          =001242=
 00052 L8001 AP E110 EISR  ENTRY TRACE_ENTRY           266AD734                                                  =001243=
 00052 L8001 AP E160 EXEC  ENTRY LINK                  'CSQCAPX ' AT X'279F2A96','..Q........&...Y...............-y...' AT X =001244=
 00052 L8001 AP E111 EISR  EXIT  TRACE_ENTRY/OK                                                                  =001245=
 00052 L8001 PG 1101 PGLE  ENTRY LINK_EXEC             CSQCAPX,266AEB98 , 00000024,NO,NO                         =001246=
 00052 L8001 DD 0301 DDLO  ENTRY LOCATE                2592AE60,26CD46AC,PPT,CSQCAPX                             =001247=
 00052 L8001 DD 0302 DDLO  EXIT  LOCATE/OK             D7D7E3C5 , 26DF0108                                       =001248=
 00052 L8001 LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM       26DEF750                                                  =001249=
```

```
00052 L8001 LD 0002 LDLD   EXIT  ACQUIRE_PROGRAM/OK    A6D84828,26D84800,550,REUSABLE,ECDSA,OLD_COPY,                          =001250=
00052 L8001 AP 1940 APLI   ENTRY START_PROGRAM         CSQCAPX,NOCEDF,FULLAPI,EXEC,NO,2594C880,266AEB98 , 00000024,3,NO        =001251=
00052 L8001 SM 0C01 SMMG   ENTRY GETMAIN               20C,YES,00,TASK                                                         =001252=
00052 L8001 SM 0C02 SMMG   EXIT  GETMAIN/OK            266AEBD8                                                                =001253=
00052 L8001 AP 00E1 EIP    ENTRY RETURN                                            0004,266AEBE8 ...Y,08000E08 ....            =001254=
00052 L8001 AP E110 EISR   ENTRY TRACE_ENTRY           266AEC50                                                                =001255=
00052 L8001 AP E160 EXEC   ENTRY RETURN                ASM                                                                     =001256=
00052 L8001 AP E111 EISR   EXIT  TRACE_ENTRY/OK                                                                                =001257=
00052 L8001 SM 0301 SMGF   ENTRY FREEMAIN              266AEBD8,TASK                                                           =001258=
00052 L8001 SM 0302 SMGF   EXIT  FREEMAIN/OK                                                                                   =001259=
00052 L8001 AP 1941 APLI   EXIT  START_PROGRAM/OK      ,NO,CSQCAPX                                                             =001260=
00052 L8001 LD 0001 LDLD   ENTRY RELEASE_PROGRAM       26DEF750,A6D84828                                                       =001261=
00052 L8001 LD 0002 LDLD   EXIT  RELEASE_PROGRAM/OK    26D84800,550,ECDSA                                                      =001262=
00052 L8001 PG 1700 PGCH   ENTRY DELETE_OWNED_CHANNELS                                                                         =001263=
00052 L8001 PG 1701 PGCH   EXIT  DELETE_OWNED_CHANNELS/OK                                                                      =001264=
00052 L8001 PG 1102 PGLE   EXIT  LINK_EXEC/OK          ,,,                                                                     =001265=
00052 L8001 AP E110 EISR   ENTRY TRACE_EXIT            266AD734                                                                =001266=
00052 L8001 AP E161 EXEC   EXIT  LINK                  'CSQCAPX ' AT X'279F2A96','..Q........&...Y...............-y...' AT X =001267=
00052 L8001 AP E111 EISR   EXIT  TRACE_EXIT/OK                                                                                 =001268=
00052 L8001 AP 00E1 EIP    EXIT  LINK                  OK                          00F4,00000000 ....,00000E02 ....            =001269=


       ***
       ***  CSQCAPX
       ***
00052 L8001 AP 00E1 EIP    ENTRY LINK                                              0004,266ADE6C ...%,08000E02 ....            =001270=
00052 L8001 AP E110 EISR   ENTRY TRACE_ENTRY           266AD734                                                                =001271=
00052 L8001 AP E160 EXEC   ENTRY LINK                  'CSQCAPX ' AT X'279F2A96','..Q........&...Y...............-y...' AT X =001272=
00052 L8001 AP E111 EISR   EXIT  TRACE_ENTRY/OK                                                                                =001273=
00052 L8001 PG 1101 PGLE   ENTRY LINK_EXEC             CSQCAPX,266AEB98 , 00000024,NO,NO                                       =001274=
00052 L8001 DD 0301 DDLO   ENTRY LOCATE                2592AE60,26CD46AC,PPT,CSQCAPX                                           =001275=
00052 L8001 DD 0302 DDLO   EXIT  LOCATE/OK             D7D7E3C5 , 26DF0108                                                     =001276=
00052 L8001 LD 0001 LDLD   ENTRY ACQUIRE_PROGRAM       26DEF750                                                                =001277=
00052 L8001 LD 0002 LDLD   EXIT  ACQUIRE_PROGRAM/OK    A6D84828,26D84800,550,REUSABLE,ECDSA,OLD_COPY,                          =001278=
00052 L8001 AP 1940 APLI   ENTRY START_PROGRAM         CSQCAPX,NOCEDF,FULLAPI,EXEC,NO,2594C880,266AEB98 , 00000024,3,NO        =001279=
00052 L8001 SM 0C01 SMMG   ENTRY GETMAIN               20C,YES,00,TASK                                                         =001280=
00052 L8001 SM 0C02 SMMG   EXIT  GETMAIN/OK            266AEBD8                                                                =001281=
00052 L8001 AP 00E1 EIP    ENTRY RETURN                                            0004,266AEBE8 ...Y,08000E08 ....            =001282=
00052 L8001 AP E110 EISR   ENTRY TRACE_ENTRY           266AEC50                                                                =001283=
00052 L8001 AP E160 EXEC   ENTRY RETURN                ASM                                                                     =001284=
00052 L8001 AP E111 EISR   EXIT  TRACE_ENTRY/OK                                                                                =001285=
00052 L8001 SM 0301 SMGF   ENTRY FREEMAIN              266AEBD8,TASK                                                           =001286=
00052 L8001 SM 0302 SMGF   EXIT  FREEMAIN/OK                                                                                   =001287=
00052 L8001 AP 1941 APLI   EXIT  START_PROGRAM/OK      ,NO,CSQCAPX                                                             =001288=
00052 L8001 LD 0001 LDLD   ENTRY RELEASE_PROGRAM       26DEF750,A6D84828                                                       =001289=
00052 L8001 LD 0002 LDLD   EXIT  RELEASE_PROGRAM/OK    26D84800,550,ECDSA                                                      =001290=
00052 L8001 PG 1700 PGCH   ENTRY DELETE_OWNED_CHANNELS                                                                         =001291=
00052 L8001 PG 1701 PGCH   EXIT  DELETE_OWNED_CHANNELS/OK                                                                      =001292=
00052 L8001 PG 1102 PGLE   EXIT  LINK_EXEC/OK          ,,,                                                                     =001293=
00052 L8001 AP E110 EISR   ENTRY TRACE_EXIT            266AD734                                                                =001294=
00052 L8001 AP E161 EXEC   EXIT  LINK                  'CSQCAPX ' AT X'279F2A96','..Q........&...Y...............-y...' AT X =001295=
00052 L8001 AP E111 EISR   EXIT  TRACE_EXIT/OK                                                                                 =001296=
00052 L8001 AP 00E1 EIP    EXIT  LINK                  OK                          00F4,00000000 ....,00000E02 ....            =001297=


00052 L8001 AP A09A MQTRU EVENT CSQCPMGI               MESSAGE ID                                                              =001298=
00052 L8001 AP A091 MQTRU EXIT  APPLICATION-REQUEST    MQPUT 00000000,00000000                                                 =001299=
00052 L8001 AP 2523 ERM    EVENT REGAINING-CONTROL-FROM-OPENAPI-TRUE(MQM     )                                                 =001300=
00052 L8001 AP D500 UEH    EVENT LINK-TO-USER-EXIT-PROGRAM GENGEXIT AT EXIT POINT XRMIOUT                                      =001301=
00052 L8001 AP D501 UEH    EVENT RETURN-FROM-USER-EXIT-PROGRAM GENGEXIT WITH RETURN CODE 790568                                =001302=
00052 L8001 RM 0301 RMLN   ENTRY SET_LINK              01000009,26FEC31C , 0000000C , 00000008,YES,NECESSARY                   =001303=
00052 L8001 RM 0302 RMLN   EXIT  SET_LINK/OK           26FEC31C , 0000000C , 00000008,                                         =001304=
00052 L8001 AP 2521 ERM    EXIT  COBOL-APPLICATION-CALL-TO-TRUE(MQM     )                                                      =001305=
```

7.2.1 The API crossing exit (CSQCAPX)

Use care when changing the WMQ API crossing exit (CSQCAPX). It is possible to run CICS API commands. Amending this exit to make calls to nonthreadsafe CICS API commands can cause a switch to the QR TCB to run this command and then cause a switch back to the open TCB to continue with the WMQ call.

7.3 Threadsafe program definition considerations

Table 7-1 helps to find the most suitable API and CONCURRENCY definition for your threadsafe application programs. From a performance perspective, define the program by using the resource definition that efficiently avoids TCB Mode switching.

Table 7-1 Program definition considerations

API	Concurrency	Good candidate	Poor candidate
CICSAPI	**THREADSAFE**	Programs that use nonthreadsafe CICS API commands before the first call to open transaction environment (OTE) TRUE is issued	
	REQUIRED	Program using threadsafe CICS API commands; CICS SQL, WMQ, or DBCTL, or IP socket calls; or both	
OPENAPI	**REQUIRED**	CPU intensive applications	User key CICS DB2 applications
		Applications using APIs not supplied by CICS	Applications using many nonthreadsafe EXEC CICS commands
		Applications using threadsafe CICS commands only	
		CICS key applications using OTE TRUE (DB2, WMQ, DBCTL, or IP sockets	
	THREADSAFE	Same as required	Same as required

The following section describes an example of a user key OPENAPI program that forces extensive TCB Mode switches due to an inefficient program resource definition.

In CICS TS V4.2, the OTE is further enhanced. By using the CONCURRENCY REQUIRED option, you can move the CICS workload off the QR TCB without waiting until the first OTE TRUE call is issued to switch to a L8 TCB.

Sometimes it is difficult to find the most efficient API and CONCURRENCY definition if your application is logically threadsafe but issues several nonthreadsafe CICS commands. In such cases, use CICS System Management Facilities (SMF) performance records to monitor the number of TCB Mode Switch count for different program definitions.

7.3.1 OPENAPI programs and additional TCB switching

Starting in CICS TS V3, programs can be defined with API(OPENAPI) and, therefore, run almost independently of the QR TCB. Such programs run on an L8 or L9 open TCB, depending upon their EXECKEY value. For information about the OPENAPI definition, see 2.2.5, “CICS Transaction Server V3.1” on page 20. OPENAPI programs must be threadsafe and defined to CICS as such.

Because OPENAPI programs can potentially use APIs that are not supplied by CICS, the key of the TCB is important and must match the execution key. However, CICSAPI threadsafe programs can run in the CICS key or the user key irrespective of the TCB key. CICS services are implemented regardless of the TCB key under which they are running under, unlike MVS services for which the TCB key is important.

> **Important:** Use of APIs not supplied by CICS within CICS is at the risk of the user. IBM has not undertaken any testing of APIs not supplied by CICS in CICS, and use of such APIs is not supported by IBM Service.

The use of OPENAPI programs can increase TCB switching within CICS. If an OPENAPI program is defined to run with an execution key of user, it is given control under an L9 TCB rather than an L8 TCB. If the program issues a call to an OPENAPI TRUE, the task is switched to an L8 TCB for the duration of the call. The reason is that OPENAPI TRUEs must run in a CICS key under an L8 TCB. The following TRUEs are supplied by IBM:

- CICS DB2 Adapter
- CICS MQ Adapter
- CICS IMS Adapter, EXEC DLI TRUE
- CICS DBCTL Adapter
- DFHD2EX1
- DFHMQTRU
- DFHEDP
- DFHDBAT

- EZACIC01
- IP CICS Sockets Adapter

Upon completion of the call, CICS returns control to the application program on its L9 TCB.

Likewise, an OPENAPI program that starts nonthreadsafe EXEC CICS commands switches from its L8 or L9 TCB to the QR TCB for the duration of the CICS request. Then it switches back to the open TCB when returning control to the application program. This process occurs because, when a program is defined as being OPENAPI, it *must* run its application logic under an open L8 or L9 TCB. However, a CICSAPI threadsafe program does not have affinity to any one TCB and runs under any TCB that CICS determines is appropriate to use.

To avoid additional TCB switching, leave user key applications that make calls to OPENAPI enabled TRUEs defined as CICSAPI threadsafe programs. Other good candidates for threadsafe programs defined with API(CICSAPI) are those programs that start nonthreadsafe CICS API requests.

The following programs are good candidates to be defined as API(OPENAPI):

- Programs with an execution key of CICS that make calls to OPENAPI-enabled TRUEs
- Programs that start only threadsafe CICS API requests, API requests for APIs not supplied by CICS, or both
- CPU-intensive applications

EXECKEY program attribute: The EXECKEY program attribute determines the mode of open TCB that is assigned for an OPENAPI program to run under. User key programs run under an L9 TCB, and CICS key programs run under an L8 TCB. An exception is if a CICS system does not have storage protection active (that is, `STGPROT=NO` is specified). Then, all OPENAPI programs must run under L8 TCBs, regardless of their EXECKEY value. The reason is that `STGPROT=NO` makes CICS operate without any storage protection and run in a single storage key (key 8).

7.4 Function shipped commands

The temporary storage API commands (other equally valid commands such as file control commands) are threadsafe. They are threadsafe when the commands are performed against locally defined resources or against shared temporary storage queues in a coupling facility. However, if these commands are performed against remote resources, they must be function shipped to the remote region to run. Function shipping involves extra TCB switching due to multiregion operation

(MRO) and intersystem communication (ISC) CICS components not being threadsafe. The same concept is true for an **EXEC CICS LINK** command to a remote program (that is, a dynamic program link (DPL) call).

> **Request running on an open TCB:** In CICS TS V4.2, DFHMIRS is threadsafe. However, only the following requests that are function shipped over IPIC run on an open TCB:
>
> - File control requests
> - Temporary storage requests
> - Dynamic program link requests if the target program is defined as threadsafe and the mirror is already on an open TCB

Example 7-5 and Example 7-6 on page 164 show these commands being performed in both local and remote scenarios. Example 7-5 is a CICS trace of a threadsafe CICSAPI application program making a DB2 call on an open L8 TCB. It then uses the **EXEC CICS LINK** command to program DUMMY, which is defined as a local program. The **LINK** command is threadsafe. Therefore, no mode switch is made to the QR TCB, and the request is processed on the L8 TCB.

Example 7-5 CICS trace of link command on a local region

```
54728 L800I AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT                                                =000268=
54728 L800I AP 3250 D2D2  ENTRY DB2_API_CALL         22F76330                                                               =000269=
54728 L800I AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                             =000270=
54728 L800I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE -805 RETURNED ON EXEC SQL SELECT                               =000271=
54728 L800I MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME  DSNCSQL                                                                =000272=
54728 L800I MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                      =000273=
54728 L800I RM 0301 RMLN  ENTRY SET_LINK             01CF034D,22DE21EC , 0000000C , 00000008,YES,NECESSARY                  =000274=
54728 L800I RM 0302 RMLN  EXIT  SET_LINK/OK          22DE21EC , 0000000C , 00000008,                                        =000275=
54728 L800I AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                                =000276=
54728 L800I AP 00E1 EIP   ENTRY LINK                                            0004,226D1458 ._..,08000E02 ....            =000277=
54728 L800I AP E160 EXEC  ENTRY LINK                 'DUMMY   ' AT X'A43037D8',ASM                                          =000278=
54728 L800I PG 1101 PGLE  ENTRY LINK_EXEC            DUMMY,NO,NO                                                            =000279=
54728 L800I DD 0301 DDLO  ENTRY LOCATE               21C27B70,22B956DC,PPT,DUMMY                                            =000280=
54728 L800I DD 0302 DDLO  EXIT  LOCATE/OK            D7D7E3C5 , 2410F6B8                                                    =000281=
54728 L800I LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM      24199988                                                               =000282=
54728 L800I AP 00E1 EIP   ENTRY LINK                                            0004,226D1458 ._..,08000E02 ....            =000277=
54728 L800I AP E160 EXEC  ENTRY LINK                 'DUMMY   ' AT X'A43037D8',ASM                                          =000278=
54728 L800I PG 1101 PGLE  ENTRY LINK_EXEC            DUMMY,NO,NO                                                            =000279=
54728 L800I DD 0301 DDLO  ENTRY LOCATE               21C27B70,22B956DC,PPT,DUMMY                                            =000280=
54728 L800I DD 0302 DDLO  EXIT  LOCATE/OK            D7D7E3C5 , 2410F6B8                                                    =000281=
54728 L800I LD 0001 LDLD  ENTRY ACQUIRE_PROGRAM      24199988                                                               =000282=
54728 L800I LD 0002 LDLD  EXIT  ACQUIRE_PROGRAM/OK   A4303C30,24303C30,138,0,REUSABLE,ESDSA,OLD_COPY                        =000283=
54728 L800I AP 1940 APLI  ENTRY START_PROGRAM        DUMMY,CEDF,FULLAPI,EXEC,NO,2410D6B8,00000000 , 00000000,2,NO           =000284=
54728 L800I SM 0C01 SMMG  ENTRY GETMAIN              190,YES,00,TASK                                                        =000285=
54728 L800I SM 0C02 SMMG  EXIT  GETMAIN/OK           226D1788                                                               =000286=
54728 L800I AP 00E1 EIP   ENTRY RETURN                                          0004,226D1798 ._.q,08000E08 ....            =000287=
54728 L800I AP E160 EXEC  ENTRY RETURN               ASM                                                                    =000288=
54728 L800I SM 0301 SMGF  ENTRY FREEMAIN             226D1788,TASK                                                          =000289=
54728 L800I SM 0302 SMGF  EXIT  FREEMAIN/OK                                                                                 =000290=
54728 L800I AP 1941 APLI  EXIT  START_PROGRAM/OK     ....,NO,DUMMY                                                          =000291=
54728 L800I LD 0001 LDLD  ENTRY RELEASE_PROGRAM      24199988,A4303C30                                                      =000292=
54728 L800I LD 0002 LDLD  EXIT  RELEASE_PROGRAM/OK   24303C30,138,ESDSA                                                     =000293=
54728 L800I PG 1102 PGLE  EXIT  LINK_EXEC/OK         ,,,                                                                    =000294=
54728 L800I AP E161 EXEC  EXIT  LINK                 'DUMMY   ' AT X'A43037D8',0,0,ASM                                      =000295=
54728 L800I AP 00E1 EIP   EXIT  LINK                 OK                         00F4,00000000 ....,00000E02 ....            =000296=
```

Example 7-6 is a CICS trace of a threadsafe CICSAPI application program making a DB2 call on an open L8 TCB. It then does a DPL request to program DUMMY. Although the link command is threadsafe, a mode switch is made to the QR TCB to ship the request to the remote region. When the link to program DUMMY returns, the application continues to run on the QR TCB and does not switch back to the L8 TCB. The application does not switch to the L8 TCB until another DB2 request is made.

Example 7-6 CICS trace of a DPL

```
54734 L800I AP 3180 D2EX1 ENTRY APPLICATION         REQUEST EXEC SQL SELECT                                              =000262=
54734 L800I AP 3250 D2D2  ENTRY DB2_API_CALL        22F76330                                                             =000263=
54734 L800I AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                          =000264=
54734 L800I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE -805 RETURNED ON EXEC SQL SELECT                           =000265=
54734 L800I MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                            =000266=
54734 L800I MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                   =000267=
54734 L800I RM 0301 RMLN  ENTRY SET_LINK            01020035,22DE21EC , 0000000C , 00000008,YES,NECESSARY                =000268=
54734 L800I RM 0302 RMLN  EXIT  SET_LINK/OK         22DE21EC , 0000000C , 00000008,                                      =000269=
54734 L800I AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                             =000270=
54734 L800I AP 00E1 EIP   ENTRY LINK                                            0004,226D1458 ._..,08000E02 ....         =000271=
54734 L800I AP E160 EXEC  ENTRY LINK                'DUMMY   ' AT X'A43037D8',ASM                                        =000272=
54734 L800I PG 1101 PGLE  ENTRY LINK_EXEC           DUMMY,NO,NO                                                          =000273=
54734 L800I DD 0301 DDLO  ENTRY LOCATE              21C27B70,22B956DC,PPT,DUMMY                                          =000274=
54734 L800I DD 0302 DDLO  EXIT  LOCATE/OK           D7D7E3C5 , 2410F6B8                                                  =000275=
54734 L800I PG 1102 PGLE  EXIT  LINK_EXEC/EXCEPTION   REMOTE_PROGRAM,PJA7,DUMMY,,                                        =000276=
54734 L800I DS 0002 DSAT  ENTRY CHANGE_MODE         QR                                                                   =000277=
54734 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                           =000278=
54734 QR    AP 00DF ISP   ENTRY CONVERSE                                        0003,04000000 ....,D7D1C1F7 PJA7         =000279=
54734 QR    AP D900 XFP   ENTRY TRANSFORMER_1       226D14C0,0E0200000000000000000003C090000                             =000280=
54734 QR    PG 0500 PGIS  ENTRY INQUIRE_CURRENT_PROGRAM                                                                  =000281=
54734 QR    PG 0501 PGIS  EXIT  INQUIRE_CURRENT_PROGRAM/OK FUNCSHIP                                                      =000282=
54734 QR    AP D901 XFP   EXIT  TRANSFORMER_1       226D14C0,0E024C6E00580010FD016054734C00D5                            =000283=
54734 QR    AP FD01 ZARQ  ENTRY APPL_REQ            22F077F0,WRITE,READ,WAIT,FMH                                         =000284=
54734 QR    AP FD0D ZIS2  ENTRY IRC                 22F077F0,IOR,WRITE,WAIT,READ                                         =000285=
54734 QR    AP DD21 ZIS2  EVENT IRC                 SWITCH SUBSEQUENT TO SYSTEM (SCSCPJA7) RETURN CODE WAS 00000000      =000286=
54734 QR    AP DD22 ZIS2  EVENT IRC                 OUTBOUND REQUEST HEADER: FMH RQE CD , 12                             =000287=
54734 QR    DS 0004 DSSR  ENTRY WAIT_MVS            IRLINK,7F656CC0,YES,INHIBIT,YES,CONV,PJA7>ALA                        =000288=
54734 QR    DS 0005 DSSR  EXIT  WAIT_MVS/OK                                                                              =000289=
54734 QR    AP DD24 ZIS2  EVENT IRC                 INBOUND REQUEST HEADER: FMH RQE CD , 12                              =000290=
54734 QR    AP FD8D ZIS2  EXIT  IRC                 22F077F0,NORMAL                                                      =000291=
54734 QR    AP FC01 ZARQ  EVENT MRO/LU6.1           STATE SETTING TO SEND                                                =000292=
54734 QR    AP FD81 ZARQ  EXIT  APPL_REQ                                                                                 =000293=
54734 QR    AP D900 XFP   ENTRY TRANSFORMER_4       226D14C0,0E024C6E00330010D9016054734C00D5                            =000294=
54734 QR    AP D901 XFP   EXIT  TRANSFORMER_4       226D14C0,0E024C6E003C001000DF6054734C00D5                            =000295=
54734 QR    AP 00DF ISP   EXIT  CONVERSE                                        0005,04000000 ....,D7D1C1F7 PJA7         =000296=
54734 QR    AP E161 EXEC  EXIT  LINK                'DUMMY   ' AT X'A43037D8',0,0,ASM                                    =000297=
54734 QR    AP 00E1 EIP   EXIT  LINK                OK                          00F4,00000000 ....,00000E02 ....         =000298=
54734 QR    AP 00E1 EIP   ENTRY SEND-TEXT                                       0004,226D1458 ._..,08001806 ....         =000299=
54734 QR    AP E160 EXEC  ENTRY SEND                TEXT 'TRANSACTION COMPLETE          ' AT X'24303714',30 AT X'A4303802 =000300=
54734 QR    SM 0C01 SMMG  ENTRY GETMAIN             22,YES,00,CICS24_SAA                                                 =000301=
54734 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK          00041008                                                             =000302=
54734 QR    SM 0D01 SMMF  ENTRY FREEMAIN            22FA8020,22CBD6F0                                                    =000303=
54734 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK         TERMINAL storage at 22FA8020                                         =000304=
54734 QR    AP 00FA BMS   ENTRY SEND-OUT            CTRL                        0003,00000800 ....,04000020 ....         =000305=
54734 QR    SM 0301 SMGF  ENTRY GETMAIN             464,YES,00,MCPOSPWA,CICS                                             =000306=
54734 QR    SM 0302 SMGF  EXIT  GETMAIN/OK          22FA9008                                                             =000307=
54734 QR    PG 0500 PGIS  ENTRY INQUIRE_CURRENT_PROGRAM                                                                  =000308=
```

Example 7-7 is a CICS trace of a threadsafe CICSAPI application program making a DB2 call on an open L8 TCB. It then issues a WRITEQ-TS request to temporary storage queue TCBTEST, which is defined as a local queue. The WRITEQ-TS command is threadsafe, so that no mode switch is made to the QR TCB, and the request is processed on the L8 TCB.

Example 7-7 CICS trace of WRITEQ-TS command on local region

```
54910 L800I AP 3250 D2D2  ENTRY DB2_API_CALL        22F76330                                                    =000257=
54910 L800I AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                 =000258=
54910 L800I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE -805 RETURNED ON EXEC SQL SELECT                  =000259=
54910 L800I MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                   =000260=
54910 L800I MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                          =000261=
54910 L800I RM 0301 RMLN  ENTRY SET_LINK            01020037,22DE21EC , 0000000C , 00000008,YES,NECESSARY       =000262=
54910 L800I RM 0302 RMLN  EXIT  SET_LINK/OK         22DE21EC , 0000000C , 00000008,                             =000263=
54910 L800I AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                    =000264=
54910 L800I AP 00E1 EIP   ENTRY WRITEQ-TS                                    0004,23C41458 .D..,08000A02 ....   =000265=
54910 L800I AP E160 EXEC  ENTRY WRITEQ              TS 'TCBTEST ' AT X'24303850','THIS IS THE POST-SQL WRITEQ   =000266=
54910 L800I TS 0C01 TSMB  ENTRY MATCH               TCBTEST                                                     =000267=
54910 L800I TS 0C02 TSMB  EXIT  MATCH/OK            ,,,TCBTEST,,00000000,,ANY,NO,NO                             =000268=
54910 L800I TS 0201 TSQR  ENTRY WRITE               TCBTEST,23C41660 , 00000050,YES,AUXILIARY,EXEC              =000269=
54910 L800I TS 0901 TSAM  ENTRY WRITE_AUX_DATA      23C41660 , 00000050,TCBTEST,BB1A867EE0360C42,8,1,NO,NO,YES  =000270=
54910 L800I TS 0902 TSAM  EXIT  WRITE_AUX_DATA/OK   1,00000001                                                  =000271=
54910 L800I TS 0202 TSQR  EXIT  WRITE/OK            8                                                           =000272=
54910 L800I AP E161 EXEC  EXIT  WRITEQ              TS 'TCBTEST ' AT X'24303850','THIS IS THE POST-SQL WRITEQ   =000273=
54910 L800I AP 00E1 EIP   EXIT  WRITEQ-TS           OK                       00F4,00000000 ....,00000A02 ....   =000274=
```

Example 7-8 is a CICS trace of a threadsafe CICSAPI application program making a DB2 call on an open L8 TCB. It then issues a WRITEQ-TS request to temporary storage queue TCBTEST, which is defined as remote. Although the WRITEQ-TS command is threadsafe, a mode switch is made to the QR TCB to function ship the request to the remote region. When the WRITEQ-TS returns, the application continues to run on the QR TCB and does not switch back to the L8 TCB. The application is not switched to the L8 TCB until another DB2 request is made.

Example 7-8 CICS trace of WRITEQ-TS command being function shipped

```
54915 L800I AP 3180 D2EX1 ENTRY APPLICATION         REQUEST EXEC SQL SELECT                                     =000381=
54915 L800I AP 3250 D2D2  ENTRY DB2_API_CALL        22F76330                                                    =000382=
54915 L800I AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                 =000383=
54915 L800I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE -805 RETURNED ON EXEC SQL SELECT                  =000384=
54915 L800I MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                   =000385=
54915 L800I MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                          =000386=
54915 L800I RM 0301 RMLN  ENTRY SET_LINK            01CF034F,22DE21EC , 0000000C , 00000008,YES,NECESSARY       =000387=
54915 L800I RM 0302 RMLN  EXIT  SET_LINK/OK         22DE21EC , 0000000C , 00000008,                             =000388=
54915 L800I AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                    =000389=
54915 L800I AP 00E1 EIP   ENTRY WRITEQ-TS                                    0004,23C41458 .D..,08000A02 ....   =000390=
54915 L800I AP E160 EXEC  ENTRY WRITEQ              TS 'TCBTEST ' AT X'24303850','THIS IS THE POST-SQL WRITEQ   =000391=
54915 L800I TS 0C01 TSMB  ENTRY MATCH               TCBTEST                                                     =000392=
54915 L800I TS 0C02 TSMB  EXIT  MATCH/OK            TCBTEST,TCBTEST,,TCBTEST,,00000000,PJA7,NO,NO,NO            =000393=
54915 L800I DS 0002 DSAT  ENTRY CHANGE_MODE         QR                                                          =000394=
54915 QR    DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                  =000395=
54915 QR    AP 00DF ISP   ENTRY CONVERSE                                 0003,04000000 ....,D7D1C1F7 PJA7       =000396=
54915 QR    AP D902 XFX   ENTRY TRANSFORMER_1       23C414C0,0A024C6E00330010D9036054915C00CD                   =000397=
54915 QR    AP D903 XFX   EXIT  TRANSFORMER_1       23C414C0,0A024C6E00580010FD016054915C00CD                   =000398=
54915 QR    AP FD01 ZARQ  ENTRY APPL_REQ            22F077F0,WRITE,READ,WAIT,FMH                                =000399=
54915 QR    AP FD0D ZIS2  ENTRY IRC                 22F077F0,IOR,WRITE,WAIT,READ                                =000400=
54915 QR    AP DD21 ZIS2  EVENT IRC                 SWITCH SUBSEQUENT TO SYSTEM (SCSCPJA7) RETURN CODE WAS 00000000  =000401=
54915 QR    AP DD22 ZIS2  EVENT IRC                 OUTBOUND REQUEST HEADER: FMH RQE CD , 15                    =000402=
54915 QR    DS 0004 DSSR  ENTRY WAIT_MVS            IRLINK,7F656CC0,YES,INHIBIT,YES,CONV,PJA7>ALA               =000403=
54915 QR    DS 0005 DSSR  EXIT  WAIT_MVS/OK                                                                     =000404=
54915 QR    AP DD24 ZIS2  EVENT IRC                 INBOUND REQUEST HEADER: FMH RQE CD , 15                     =000405=
54915 QR    AP FD8D ZIS2  EXIT  IRC                 22F077F0,NORMAL                                             =000406=
```

```
54915 QR    AP FC01 ZARQ  EVENT MRO/LU6.1           STATE SETTING TO SEND                                                    =000407=
54915 QR    AP FD81 ZARQ  EXIT  APPL_REQ                                                                                     =000408=
54915 QR    AP D902 XFX   ENTRY TRANSFORMER_4       23C414C0,0A024C6E00330010D9036054915C00CD                                =000409=
54915 QR    AP D903 XFX   EXIT  TRANSFORMER_4       23C414C0,0A024C6E003C001000DF6054915C00CD                                =000410=
54915 QR    AP 00DF ISP   EXIT  CONVERSE                                           0005,04000000 ....,D7D1C1F7 PJA7          =000411=
54915 QR    AP E161 EXEC  EXIT  WRITEQ              TS 'TCBTEST ' AT X'24303850','THIS IS THE POST-SQL WRITEQ                =000412=
54915 QR    AP 00E1 EIP   EXIT  WRITEQ-TS           OK                             00F4,00000000 ....,00000A02 ....          =000413=
54915 QR    AP 00E1 EIP   ENTRY SEND-TEXT                                          0004,23C41458 .D..,08001806 ....          =000414=
54915 QR    AP E160 EXEC  ENTRY SEND                TEXT 'TRANSACTION COMPLETE          ' AT X'24303738',30 AT X'A43038CC =000415=
54915 QR    SM 0C01 SMMG  ENTRY GETMAIN             22,YES,00,CICS24_SAA                                                     =000416=
54915 QR    SM 0C02 SMMG  EXIT  GETMAIN/OK          0004C008                                                                 =000417=
54915 QR    SM 0D01 SMMF  ENTRY FREEMAIN            22FA88C0,22CBD6F0                                                        =000418=
54915 QR    SM 0D02 SMMF  EXIT  FREEMAIN/OK         TERMINAL storage at 22FA88C0                                             =000419=
54915 QR    AP 00FA BMS   ENTRY SEND-OUT            CTRL                           0003,00000800 ....,04000020 ....          =000420=
54915 QR    SM 0301 SMGF  ENTRY GETMAIN             464,YES,00,MCPOSPWA,CICS                                                 =000421=
54915 QR    SM 0302 SMGF  EXIT  GETMAIN/OK          22FF3008                                                                 =000422=
```

Example 7-9 is a CICS trace of a threadsafe CICSAPI application program making a DB2 call on an open L8 TCB. It then issues a WRITEQ-TS request to a shared temporary storage queue TCBTEST, which is in a coupling facility. In this scenario, the WRITEQ-TS request has no need to function ship. The application continues to run on the L8 TCB with no additional TCB switches to the QR TCB. In a threadsafe environment, convert remote temporary storage queues to shared temporary storage queues within a coupling facility.

> **Tip:** The initial call to the shared temporary storage server is always issued from the QR TCB, regardless of which TCB the program is currently on

Example 7-9 CICS trace of WRITEQ-TS request to shared temporary storage queue

```
00300 QR    AP 2520 ERM   ENTRY ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                             =000219=
00300 QR    US 0401 USXM  ENTRY INQUIRE_TRANSACTION_USER                                                                     =000220=
00300 QR    US 0402 USXM  EXIT  INQUIRE_TRANSACTION_USER/OK 00000000                                                         =000221=
00300 QR    RM 0301 RMLN  ENTRY ADD_LINK            RMI,22F914A4 , 22C1D640 , 00000008,000949D0 , 21B06F30 , 00000008,22F =000222=
00300 QR    RM 0302 RMLN  EXIT  ADD_LINK/OK         01C8000F,22F914A4 , 22C1D640 , 00000008,000949D0 , 21B06F30 , 0000000 =000223=
00300 QR    DS 0002 DSAT  ENTRY CHANGE_MODE         0000000C                                                                 =000224=
00300 L8000 DS 0003 DSAT  EXIT  CHANGE_MODE/OK                                                                               =000225=
00300 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIIN                                        =000226=
00300 L8000 SM 0C01 SMMG  ENTRY GETMAIN             198,YES,00,TASK                                                          =000227=
00300 L8000 SM 0C02 SMMG  EXIT  GETMAIN/OK          22661788                                                                 =000228=
CICS - AUXILIARY TRACE FROM 04/26/04 - APPLID SCSCPJA6 - TIME OF FIRST ENTRY ON THIS PAGE 09:58:10.4188565405            PAGE 00003

00300 L8000 SM 0D01 SMMF  ENTRY FREEMAIN            22661788                                                                 =000229=
00300 L8000 SM 0D02 SMMF  EXIT  FREEMAIN/OK         USER storage at 22661788                                                 =000230=
00300 L8000 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                      =000231=
00300 L8000 AP 3180 D2EX1 ENTRY APPLICATION         REQUEST EXEC SQL SELECT                                                  =000232=
00300 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL        230D7330                                                                 =000233=
00300 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK                                                                              =000234=
00300 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXEC SQL SELECT                                  =000235=
00300 L8000 MN 0201 MNMN  ENTRY ACCUMULATE_RMI_TIME   DSNCSQL                                                                =000236=
00300 L8000 MN 0202 MNMN  EXIT  ACCUMULATE_RMI_TIME/OK                                                                       =000237=
00300 L8000 AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT XRMIOUT                                       =000238=
00300 L8000 SM 0C01 SMMG  ENTRY GETMAIN             198,YES,00,TASK                                                          =000239=
00300 L8000 SM 0C02 SMMG  EXIT  GETMAIN/OK          22661788                                                                 =000240=
00300 L8000 SM 0D01 SMMF  ENTRY FREEMAIN            22661788                                                                 =000241=
00300 L8000 SM 0D02 SMMF  EXIT  FREEMAIN/OK         USER storage at 22661788                                                 =000242=
00300 L8000 AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETURN CODE 0                                      =000243=
00300 L8000 RM 0301 RMLN  ENTRY SET_LINK            01C8000F,22F914AC , 0000000C , 00000008,YES,NECESSARY                    =000244=
00300 L8000 RM 0302 RMLN  EXIT  SET_LINK/OK         22F914AC , 0000000C , 00000008,                                          =000245=
00300 L8000 AP 2521 ERM   EXIT  ASSEMBLER-APPLICATION-CALL-TO-TRUE(DSNCSQL )                                                 =000246=
00300 L8000 AP 00E1 EIP   ENTRY WRITEQ-TS                                          0004,22661458 ....,08000A02 ....          =000247=
00300 L8000 AP E160 EXEC  ENTRY WRITEQ              TS 'TCBTEST ' AT X'24300730','THIS IS THE POST-SQL WRITEQ                =000248=
```

```
00300 L8000 TS 0C01 TSMB  ENTRY MATCH                TCBTEST                                                                 =000249=
00300 L8000 TS 0C02 TSMB  EXIT  MATCH/OK             TCBTEST,TCBTEST,,TCBTEST,TSQSPQA1,22571FE0,,NO,NO,NO                    =000250=
00300 L8000 TS 0A01 TSSH  ENTRY WRITE                TCBTEST,22661660 , 00000050,22571FE0,YES,NO                             =000251=
00300 L8000 TS 0A0B TSSH  EVENT Before_server_request WRITE,TCBTEST,22661660 , 00000050,22571FE0,FUNC,YES,NO,0000300C       =000252=
00300 L8000 TS 0A0C TSSH  EVENT After_server_request  WRITE,OK,4                                                             =000253=
00300 L8000 TS 0A02 TSSH  EXIT  WRITE/OK             4                                                                       =000254=
00300 L8000 AP E161 EXEC  EXIT  WRITEQ               TS 'TCBTEST ' AT X'24300730','THIS IS THE POST-SQL WRITEQ               =000255=
00300 L8000 AP 00E1 EIP   EXIT  WRITEQ-TS            OK                        00F4,00000000 ....,00000A02 ....              =000256=
```

7.5 COBOL calls

If your application uses COBOL calls to start subprograms, be aware that the concurrency value used is the value set for the program at the calling level. Therefore, if PROGA is defined as **CONCURRENCY(QUASIRENT)** and PROGB is defined as **CONCURRENCY(THREADSAFE)**, the concurrency attribute that is honored is QUASIRENT when calling PROGB from PROGA. The following two trace examples demonstrate this behavior, which occurs when using dynamic COBOL calls, static COBOL calls, or both.

7.5.1 PROGA (Quasirent) calling PROGB (threadsafe)

In Example 7-10, PROGA is defined as QUASIRENT, and PROGB is defined as THREADSAFE. The trace shows that, after all the DB2 calls are completed (including the call in PROGB, which is defined as THREADSAFE), the program returns to the QR TCB.

Example 7-10 PROGA defined as Quasirent and PROGB defined as threadsafe

```
11281 QR    AP 1940 APLI  ENTRY START_PROGRAM        PROGA ,CEDF,FULLAPI,EXE
11281 QR    AP 1948 APLI  EVENT CALL-TO-LE/370       Thread_Initialization CAL
11281 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370   Thread_Initialization OK
11281 QR    AP 1948 APLI  EVENT CALL-TO-LE/370       Rununit_Init_&_Begin_Invo
11281 QR    AP 00E1 EIP   ENTRY DELETEQ-TS
11281 QR    AP E160 EXEC  ENTRY DELETEQ              TS 'TONYQ   ' AT X'A266AB
11281 QR    AP E161 EXEC  EXIT  DELETEQ              TS 'TONYQ   ' AT X'A266AB
11281 QR    AP 00E1 EIP   EXIT  DELETEQ-TS           OK
11281 QR    AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
                ****
                **** First SQL Call in PROGA            *****
                ****
11281 L802I AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT
11281 L802I AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETUR
11281 L802I AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
11281 L802I AP 3250 D2D2  ENTRY DB2_API_CALL         230D7330
11281 L802I AP 3251 D2D2  EXIT  DB2_API_CALL/OK
11281 L802I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXE
11281 L802I AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT
11281 L802I AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETUR
```

```
                ****
                **** First SQL Complete - back to QR    *****
                ****
11281 QR    AP 2521 ERM   EXIT  COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
11281 QR    AP 00E1 EIP   ENTRY WRITEQ-TS
11281 QR    AP E160 EXEC  ENTRY WRITEQ               TS 'TONYQ   ' AT X'2266AB
11281 QR    AP E161 EXEC  EXIT  WRITEQ               TS 'TONYQ   ' AT X'2266AB
11281 QR    AP 00E1 EIP   EXIT  WRITEQ-TS            OK
11281 QR    AP 00E1 EIP   ENTRY GETMAIN
11281 QR    AP E160 EXEC  ENTRY GETMAIN              AT X'226600F8',4080 AT X'
11281 QR    AP E161 EXEC  EXIT  GETMAIN              X'2266C948' AT X'226600F8
11281 QR    AP 00E1 EIP   EXIT  GETMAIN              OK
11281 QR    AP 00E1 EIP   ENTRY ADDRESS
11281 QR    AP E160 EXEC  ENTRY ADDRESS              AT X'A266D23C',SYSEIB,ASM
11281 QR    AP E161 EXEC  EXIT  ADDRESS              X'0005D494' AT X'A266D23C
11281 QR    AP 00E1 EIP   EXIT  ADDRESS              OK
                ****
                **** About to start PROGB               *****
                ****
11281 QR    AP 00E1 EIP   ENTRY LOAD
11281 QR    AP E160 EXEC  ENTRY LOAD                 'PROGB' AT X'2266D380'
11281 QR    AP E161 EXEC  EXIT  LOAD                 'PROGB' AT X'2266D380'
11281 QR    AP 00E1 EIP   EXIT  LOAD                 OK
11281 QR    AP 00E1 EIP   ENTRY PUSH
11281 QR    AP E160 EXEC  ENTRY PUSH                 HANDLE SYSEIB NOHANDLE AS
11281 QR    AP E161 EXEC  EXIT  PUSH                 HANDLE 0,0,SYSEIB,NOHANDL
11281 QR    AP 00E1 EIP   EXIT  PUSH                 OK
11281 QR    AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
                ****
                **** SQL Call in PROGB - switch to L8   *****
                ****
11281 L802I AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT
11281 L802I AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETUR
11281 L802I AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
11281 L802I AP 3250 D2D2  ENTRY DB2_API_CALL         230D7330
11281 L802I AP 3251 D2D2  EXIT  DB2_API_CALL/OK
11281 L802I AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE 0 RETURNED ON EXE
11281 L802I AP D500 UEH   EVENT LINK-TO-USER-EXIT-PROGRAM XXXRMI AT EXIT POINT
11281 L802I AP D501 UEH   EVENT RETURN-FROM-USER-EXIT-PROGRAM XXXRMI WITH RETUR
                ****
                **** First SQL Complete - back to QR    *****
                ****
11281 QR    AP 2521 ERM   EXIT  COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
11281 QR    AP 00E1 EIP   ENTRY WRITEQ-TS
11281 QR    AP E160 EXEC  ENTRY WRITEQ               TS 'TONYQ   ' AT X'2266AE
11281 QR    AP E161 EXEC  EXIT  WRITEQ               TS 'TONYQ   ' AT X'2266AE
11281 QR    AP 00E1 EIP   EXIT  WRITEQ-TS            OK
11281 QR    AP 00E1 EIP   ENTRY POP
11281 QR    AP E160 EXEC  ENTRY POP                  HANDLE SYSEIB NOHANDLE AS
```

```
11281 QR    AP E161 EXEC  EXIT  POP                     HANDLE 0,0,SYSEIB,NOHANDL
11281 QR    AP 00E1 EIP   EXIT  POP                     OK
11281 QR    AP 00E1 EIP   ENTRY WRITEQ-TS
11281 QR    AP E160 EXEC  ENTRY WRITEQ                  TS 'TONYQ   ' AT X'2266AB
11281 QR    AP E161 EXEC  EXIT  WRITEQ                  TS 'TONYQ   ' AT X'2266AB
11281 QR    AP 00E1 EIP   EXIT  WRITEQ-TS               OK
11281 QR    AP 00E1 EIP   ENTRY RETURN
11281 QR    AP E160 EXEC  ENTRY RETURN                  COBOLII 00008
11281 QR    AP 1948 APLI  EVENT CALL-TO-LE/370          Rununit_End_Invocation CA
11281 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370      Rununit_End_Invocation OK
11281 QR    AP 1948 APLI  EVENT CALL-TO-LE/370          Rununit_Termination CALLP
11281 QR    AP 00E1 EIP   ENTRY ADDRESS
11281 QR    AP E160 EXEC  ENTRY ADDRESS                 AT X'A2669800',SYSEIB,ASM
11281 QR    AP E161 EXEC  EXIT  ADDRESS                 X'0005D494' AT X'A2669800
11281 QR    AP 00E1 EIP   EXIT  ADDRESS                 OK
11281 QR    AP 00E1 EIP   ENTRY RELEASE
11281 QR    AP E160 EXEC  ENTRY RELEASE                 'PROGB' AT X'A2669948'
11281 QR    AP E161 EXEC  EXIT  RELEASE                 'PROGB' AT X'A2669948'
11281 QR    AP 00E1 EIP   EXIT  RELEASE                 OK
11281 QR    AP 00E1 EIP   ENTRY FREEMAIN
11281 QR    AP E160 EXEC  ENTRY FREEMAIN                AT X'A266C948',SYSEIB,NOH
11281 QR    AP E161 EXEC  EXIT  FREEMAIN                AT X'A266C948',0,0,SYSEIB
11281 QR    AP 00E1 EIP   EXIT  FREEMAIN                OK
11281 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370      Rununit_Termination OK CA
11281 QR    AP 1948 APLI  EVENT CALL-TO-LE/370          Thread_Termination
11281 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370      Thread_Termination OK
11281 QR    AP 1941 APLI  EXIT  START_PROGRAM/OK        ....,NO,PROGA
11281 QR    AP 2500 ERMSP ENTRY PERFORM_PREPARE         NO,0005D264
11281 QR    AP 2501 ERMSP EXIT  PERFORM_PREPARE/OK      READ_ONLY
11281 QR    AP 1760 LTRC  ENTRY PERFORM_PREPARE         NO,22CD04B0
11281 QR    AP 1761 LTRC  EXIT  PERFORM_PREPARE/OK      READ_ONLY
11281 QR    AP 05A8 APRC  ENTRY PERFORM_PREPARE         NO,00000001
11281 QR    AP 05A9 APRC  EXIT  PERFORM_PREPARE/OK      READ_ONLY
11281 QR    AP 2500 ERMSP ENTRY SEND_DO_COMMIT          241FC030,NO,YES,01380036,
11281 QR    AP 2520 ERM   ENTRY SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL )
                  ****
                  **** Return briefly to the L8 for commit processing  *****
                  ****
11281 L802I AP 3180 D2EX1 ENTRY SYNCPOINT-MANAGER       REQUEST
11281 L802I AP 3250 D2D2  ENTRY SINGLE_PHASE_COMMIT     230D7330
11281 L802I AP 3251 D2D2  EXIT  SINGLE_PHASE_COMMIT/OK
11281 L802I AP 3181 D2EX1 EXIT  SYNCPOINT-MANAGER       REQUEST
11281 QR    AP 2521 ERM   EXIT  SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL )
11281 QR    AP 2501 ERMSP EXIT  SEND_DO_COMMIT/OK       YES,YES,DSNCSQL
11281 QR    AP 2500 ERMSP ENTRY PERFORM_COMMIT          241FC030,NO,YES,YES,NO,NO
11281 QR    AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK       YES,YES,YES,NO,UNNECESSAR
11281 QR    AP 2500 ERMSP ENTRY PERFORM_COMMIT          NO,FORWARD,0005D264
11281 QR    AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-END
11281 QR    AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-END
```

```
11281 QR    AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK     YES
11281 QR    AP 1760 LTRC  ENTRY PERFORM_COMMIT        NO,FORWARD,22CD04B0
11281 QR    AP 1710 TFRF  ENTRY RELEASE_FACILITY      NO,NORMAL,22CD04B0,TC51
11281 QR    AP FD0B ZISP  ENTRY FACILITY_REQ          22CD04B0,FREE_DETACH,IMPL
11281 QR    AP FD03 ZDET  ENTRY DETACH                22CD04B0,TC51
11281 QR    AP FD18 ZSDS  ENTRY SEND_DFSYN            22CD04B0,TC51
11281 QR    AP FD1D ZSDR  ENTRY SEND_DFSYN_RESP       22CD04B0,TC51
11281 QR    AP FC90 VIO   EVENT TCTTE(22CD04B0)       SC38TC51,01CA,SEND,DATA,0
11281 QR    AP FD8B ZISP  EXIT  FACILITY_REQ
11281 QR    AP 1711 TFRF  EXIT  RELEASE_FACILITY/OK   TC51
11281 QR    AP 1761 LTRC  EXIT  PERFORM_COMMIT/OK     NO
11281 QR    AP 05A8 APRC  ENTRY PERFORM_COMMIT        NO,FORWARD,00000001
11281 QR    AP 05A9 APRC  EXIT  PERFORM_COMMIT/OK     NO
11281 QR    AP 0590 APXM  ENTRY RELEASE_XM_CLIENT     NORMAL
```

7.5.2 PROGA (threadsafe) calling PROGB (Quasirent)

If the definitions of PROGA and PROGB are swapped so that PROGA is now THREADSAFE and PROGB is QUASIRENT, the opposite effect occurs. PROGB remains on the L8 TCB after any DB2 calls due to the definition of PROGA are set to THREADSAFE. PROGB starts on the L8 TCB and continues there until completion as shown in Example 7-11.

Example 7-11 PROGA defined as threadsafe and PROGB defined as Quasirent

```
00108 QR    AP 1940 APLI  ENTRY START_PROGRAM         PROGA,CEDF,FULLAPI,EXE
00108 QR    AP 1948 APLI  EVENT CALL-TO-LE/370        Thread_Initialization CAL
00108 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370    Thread_Initialization OK
00108 QR    AP 1948 APLI  EVENT CALL-TO-LE/370        Rununit_Init_&_Begin_Invo
00108 QR    AP 00E1 EIP   ENTRY DELETEQ-TS
00108 QR    AP 00E1 EIP   EXIT  DELETEQ-TS            OK
00108 QR    AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
                ****
                **** First SQL Call in PROGA            *****
                ****
00108 L8000 AP 3180 D2EX1 ENTRY APPLICATION           REQUEST EXEC SQL SELECT
00108 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL          22DDF030
00108 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK
00108 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST   SQLCODE -805 RETURNED ON
00108 L8000 AP 2521 ERM   EXIT  COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
                ****
                **** PROGA continues on L8 TCB          *****
                ****
00108 L8000 AP 00E1 EIP   ENTRY WRITEQ-TS
00108 L8000 AP 00E1 EIP   EXIT  WRITEQ-TS             OK
00108 L8000 AP 00E1 EIP   ENTRY GETMAIN
00108 L8000 AP 00E1 EIP   EXIT  GETMAIN               OK
00108 L8000 AP 00E1 EIP   ENTRY ADDRESS
```

```
00108 L8000 AP 00E1 EIP   EXIT  ADDRESS              OK
                ****
                **** Here we are about to call PROGB    *****
                ****   remaining on the L8 TCB          *****
                ****
00108 L8000 AP 00E1 EIP   ENTRY LOAD
00108 L8000 AP 00E1 EIP   EXIT  LOAD                 OK
00108 L8000 AP 00E1 EIP   ENTRY PUSH
00108 L8000 AP 00E1 EIP   EXIT  PUSH                 OK
00108 L8000 AP 2520 ERM   ENTRY COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
00108 L8000 AP 3180 D2EX1 ENTRY APPLICATION          REQUEST EXEC SQL SELECT
00108 L8000 AP 3250 D2D2  ENTRY DB2_API_CALL         22DDF030
00108 L8000 AP 3251 D2D2  EXIT  DB2_API_CALL/OK
00108 L8000 AP 3181 D2EX1 EXIT  APPLICATION-REQUEST  SQLCODE -805 RETURNED ON
00108 L8000 AP 2521 ERM   EXIT  COBOL-APPLICATION-CALL-TO-TRUE(DSNCSQL )
00108 L8000 AP 00E1 EIP   ENTRY WRITEQ-TS
00108 L8000 AP 00E1 EIP   EXIT  WRITEQ-TS            OK
00108 L8000 AP 00E1 EIP   ENTRY POP
00108 L8000 AP 00E1 EIP   EXIT  POP                  OK
00108 L8000 AP 00E1 EIP   ENTRY WRITEQ-TS
00108 L8000 AP 00E1 EIP   EXIT  WRITEQ-TS            OK
00108 L8000 AP 00E1 EIP   ENTRY RETURN
00108 L8000 AP 1948 APLI  EVENT CALL-TO-LE/370       Rununit_End_Invocation CA
00108 L8000 AP 1949 APLI  EVENT RETURN-FROM-LE/370   Rununit_End_Invocation OK
00108 L8000 AP 1948 APLI  EVENT CALL-TO-LE/370       Rununit_Termination CALLP
00108 L8000 AP 00E1 EIP   ENTRY ADDRESS
00108 L8000 AP 00E1 EIP   EXIT  ADDRESS              OK
00108 L8000 AP 00E1 EIP   ENTRY RELEASE
00108 L8000 AP 00E1 EIP   EXIT  RELEASE              OK
00108 L8000 AP 00E1 EIP   ENTRY FREEMAIN
00108 L8000 AP 00E1 EIP   EXIT  FREEMAIN             OK
00108 L8000 AP 1949 APLI  EVENT RETURN-FROM-LE/370   Rununit_Termination OK CA
00108 L8000 AP 1948 APLI  EVENT CALL-TO-LE/370       Thread_Termination
00108 L8000 AP 1949 APLI  EVENT RETURN-FROM-LE/370   Thread_Termination OK
00108 L8000 AP 1941 APLI  EXIT  START_PROGRAM/OK     ....,NO,PROGA
                ****
                **** Program End Return to QR TCB        *****
                ****
00108 QR    AP 2500 ERMSP ENTRY PERFORM_PREPARE      NO,0005B264
00108 QR    AP 2501 ERMSP EXIT  PERFORM_PREPARE/OK   READ_ONLY
00108 QR    AP 1760 LTRC  ENTRY PERFORM_PREPARE      NO,22CD04B0
00108 QR    AP 1761 LTRC  EXIT  PERFORM_PREPARE/OK   READ_ONLY
00108 QR    AP 05A8 APRC  ENTRY PERFORM_PREPARE      NO,00000001
00108 QR    AP 05A9 APRC  EXIT  PERFORM_PREPARE/OK   READ_ONLY
00108 QR    AP 2500 ERMSP ENTRY SEND_DO_COMMIT       22F91450,NO,YES,01030001,
00108 QR    AP 2520 ERM   ENTRY SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL )
                ****
                **** Return briefly to the L8 for commit processing  *****
                ****
```

```
00108 L8000 AP 3180 D2EX1 ENTRY SYNCPOINT-MANAGER     REQUEST
00108 L8000 AP 3250 D2D2  ENTRY SINGLE_PHASE_COMMIT   22DDF030
00108 L8000 AP 3251 D2D2  EXIT  SINGLE_PHASE_COMMIT/OK
00108 L8000 AP 3181 D2EX1 EXIT  SYNCPOINT-MANAGER     REQUEST
00108 QR    AP 2521 ERM   EXIT  SYNCPOINT-MANAGER-CALL-TO-TRUE(DSNCSQL )
00108 QR    AP 2501 ERMSP EXIT  SEND_DO_COMMIT/OK     YES,YES,DSNCSQL
00108 QR    AP 2500 ERMSP ENTRY PERFORM_COMMIT        22F91450,NO,YES,YES,NO,NO
00108 QR    AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK     YES,YES,YES,NO,UNNECESSAR
00108 QR    AP 2500 ERMSP ENTRY PERFORM_COMMIT        NO,FORWARD,0005B264
00108 QR    AP 2520 ERM   ENTRY CALL-TRUES-FOR-TASK-END
00108 QR    AP 2521 ERM   EXIT  CALL-TRUES-FOR-TASK-END
00108 QR    AP 2501 ERMSP EXIT  PERFORM_COMMIT/OK     YES
00108 QR    AP 1760 LTRC  ENTRY PERFORM_COMMIT        NO,FORWARD,22CD04B0
00108 QR    AP 1710 TFRF  ENTRY RELEASE_FACILITY      NO,NORMAL,22CD04B0,TC3F
00108 QR    AP 1711 TFRF  EXIT  RELEASE_FACILITY/OK   TC3F
00108 QR    AP 1761 LTRC  EXIT  PERFORM_COMMIT/OK     NO
00108 QR    AP 05A8 APRC  ENTRY PERFORM_COMMIT        NO,FORWARD,00000001
00108 QR    AP 05A9 APRC  EXIT  PERFORM_COMMIT/OK     NO
00108 QR    AP 0590 APXM  ENTRY RELEASE_XM_CLIENT     NORMAL
```

With CICS TS V3 and later, the API attribute of the calling program is also inherited by the called program. For example, if PROGA was defined with API(OPENAPI) and EXECKEY(USER), it was started under an L9 TCB and was called PROGB under the L9 TCB.

7.6 The CSACDTA field

Historically, the CSACDTA field provided the address of the task control area (TCA) for the currently dispatched task running within CICS. Before OTE was introduced, all tasks ran under the control of the QR TCB. This behavior guaranteed a running task to retrieve the address of its own TCA if it accessed the CSACDTA field.

Considering the introduction of OTE, it is *no longer safe* to assume that the TCA address held within CSACDTA is the TCA of the task that is accessing the CSA. CSACDTA contains the address of the task that is currently dispatched on the *QR* TCB. The program that references CSACDTA might be running under an open TCB. In this case, the wrong TCA address is used by the program, leading to unpredictable results.

Since the release of CICS/ESA V4.1, direct access to CICS control blocks is not supported. Use the CICS system programming interface (SPI) for programs that need to access state information about a task.

The CSACDTA field has the following history in CICS TS:

- In releases before CICS TS V3.1, the CSACDTA field returns the address of the currently dispatched task running under the QR TCB.
- In CICS TS V3.1, CSACDTA is renamed to CSAQRTCA to further discourage its use.
- In CICS TS V3.2, IBM *withdrew* the ability to reference a TCA by using the CSAQRTCA field by loading it with the address of an area of fetch-protected storage. The result is an abend ASRD with message DFHSR0618 if it is referenced.

7.7 Ensuring CICS performance and capacity

This section explains how to investigate CICS performance issues. First, you ensure that, by using the current settings, the introduction of threadsafe applications saves CPU cycles and provides more parallelism. Then, you collect the performance data. Second, you ensure that you have sufficient CPU capacity for threadsafe regions.

IBM CICS Performance Analyzer (CICS PA) is shipped with sample reports that you can use to analyze your transaction performance. For details, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245.

7.7.1 Analyzing the current settings

Before you start to run threadsafe-enabled application programs, ensure that your CICS regions are not CPU constraint. Also ensure that your CICS DB2 attachment facility, the CICS WMQ adapter, and the CICS DBCTL interface are set up well.

You must complete the following tasks before and after your threadsafe migration project:

1. Analyze the CICS QR TCB CPU/Dispatch Ratio. Perform this step for all regions before your migration project.
2. Analyze the performance of your CICS DB2 attachment. We show the necessary actions to analyze POOL and ENTRY thread usage, thread reuse, and thread wait overflows.
3. Analyze the CICS OPEN TCB performance. Investigate delays due to limits or storage.

4. Analyze the CICS performance fields that are relevant for your threadsafe project:
   – QR TCB delays
   – First dispatch delays
   – Number of TCB switches
   – Number of DB2, WMQ, and DBCTL requests
5. Analyze the IBM Resource Measurement Facility™ (RMF™) Workload Activity Report to investigate processor constraints from the z/OS perspective.

Collecting the CICS performance data

To perform the tasks outlined in the previous section, collect the necessary performance data. For this book, we need CICS 110 subtype 1 records to calculate the QR CPU-to-dispatch Ratio and to analyze the threadsafe relevant performance fields, such as QR TCB delays and the number of TCB switches. We also need CICS SMF 110 subtype 2 records to analyze the performance of the CICS DB2 attachment. CICS SMF subtype 2 records contain the CICS statistics. To format the RMF Workload Activity Report, we need to collect RMF-SMF records type 70-78.

You must ensure that the required SMF record types are enabled as shown in Example 7-12, which shows the active parmlib member. You can also issue a **`/D SMF,O`** command to see the active values. The `SYS NOTYPE` statement shows that record types 110 and 70-78 are not excluded and, therefore, will be written to the SMF data sets.

Example 7-12 SMFPRM parmlib member

```
******************************** Top of Data **********************************
 ACTIVE                         /* ACTIVE SMF RECORDING            */
 DSNAME(SYS1.&SYSNAME..MAN1,
        SYS1.&SYSNAME..MAN2,
        SYS1.&SYSNAME..MAN3)
 SID(&SMFID.)
 NOPROMPT                       /* DO NOT PROMPT OPERATOR           */
 REC(PERM)                      /* TYPE 17 PERM RECORDS ONLY        */
 MAXDORM(3000)                  /* WRITE IDLE BUFFER AFTER 30 MIN   */
 STATUS(010000)                 /* WRITE SMF STATS AFTER 1 HOUR     */
 JWT(0030)                      /* 522 AFTER 30 MINUTES             */
 MEMLIMIT(4G)
 LISTDSN                        /* LIST DATA SET STATUS AT IPL      */
 SYS(NOTYPE(16:19,62:63,65:69,99),EXITS(IEFU83,IEFU84,IEFACTRT,
               IEFUSI,IEFUJI,IEFU29),NOINTERVAL,NODETAIL)
 SUBSYS(STC,EXITS(IEFU29,IEFU83,IEFU84,IEFUJP,IEFUSO,IEFACTRT),
```

```
              INTERVAL(SMF,SYNC))
 SUBSYS(JES2,EXITS(IEFUJI,IEFU83,IEFU84,IEFACTRT))
****************************** Bottom of Data ******************************
```

Next, activate performance and statistic monitoring in all regions that are to be analyzed. Figure 7-6 shows how CICS performance monitoring is activated. SMF 110 subtype 1 records are now written to the SMF data sets.

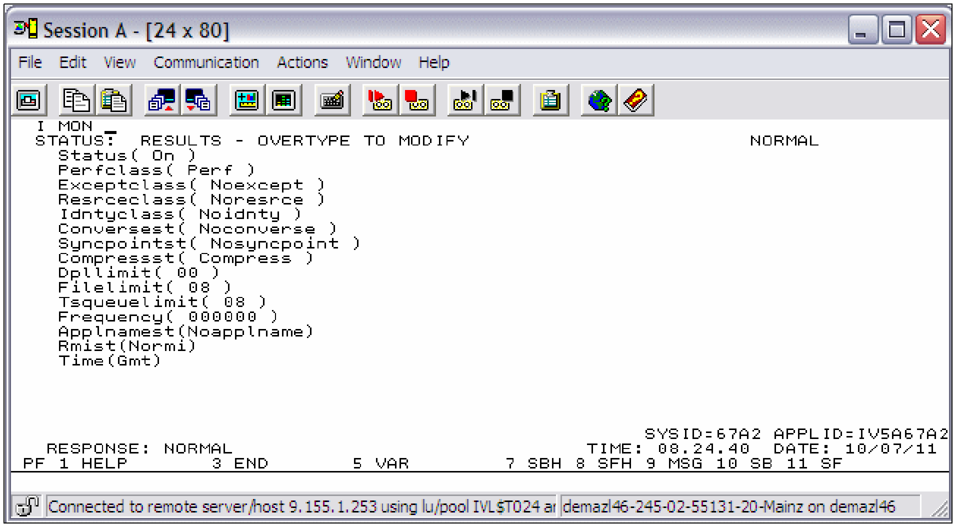


Figure 7-6   CEMT I MON command

Figure 7-7 shows how to activate the CICS interval statistics. For example, it the observation interval is 8 hours, specify 4 intervals per hour, meaning that, every 15 minutes, the CICS statistics are written to the SMF data sets.

> **Tip:** Use 15-minute statistic intervals to investigate performance trends at specific times of interest, such as during peak times.

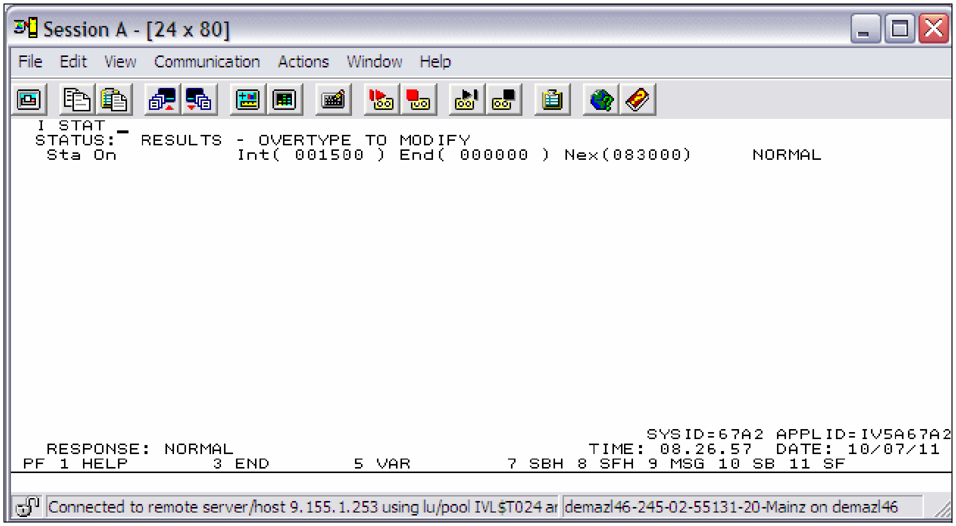


Figure 7-7 CEMT I STAT command

Collect meaningful performance data. The observation interval is sufficient to represent your typical workload during peak time. You can also collect performance data for more than one observation interval to ensure that all applications are included.

7.7.2 Ensuring the availability of sufficient CPU capacity for threadsafe regions

When you convert your CICS applications to run as threadsafe applications, you must ensure that your CICS regions are not CPU constraint. Application code remains on open TCBs after processing DB2, WMQ, DBCTL, or TCP/IP socket calls. Therefore, multiple open TCBs that run your applications are now dispatched at the same time. Therefore, open TCB delays affect your workload throughput.

To analyze the CPU availability of your region, start by investigating the QR CPU-to-dispatch ratio. The QR CPU-to-dispatch ratio indicates how often a dispatchable unit of work must wait because no processor capacity was available. For the assessment of this calculated value, we use the following scheme:

Good	80% and higher
Satisfactory	70 -79%
Unsatisfactory	50-69%
Poor	50% and lower

Investigate the low CPU-to-dispatch ratio for the following reasons:

- Involuntary MVS wait time
  - Extra-partition requests
  - MVS waits issued outside CICS control
- Preemption by other MVS tasks or other logical partitions (LPARs)
  - Check MVS priorities and WLM parameters.
  - Check DB2CONN / DB2ENTRY priority.
  - Check LPAR weights and fair share.

To investigate the QR CPU-to-dispatch ratio, we used CICS PA to format SMF 110 subtype 1 records. We used SMF records that represent the workload of an entire day. Therefore, we formatted average values of the records in 15-minute intervals. We included all tasks for a specific CICS region and the performance fields to calculate the QR CPU-to-dispatch ratio. The following fields are needed:

- QR TCB CPU time
- QR TCB Dispatch time
- Task Stop Time

We exported the formatted records to a sequential data set in the comma-separated values (CSV) format. Then, we downloaded the CSV file to a workstation to manage the data in a spreadsheet.

To calculate the CPU-to-dispatch ratio, use the following formula:

```
QR TCB to dispatch ratio = QR CPU Time / QR dispatch time
```

You can use the spreadsheet to calculate the QR TCB CPU-to-dispatch ratio per interval and to format the resulting value as a percentage.

Good practice: Illustrate the QR TCB CPU-to-dispatch ratio on a time line so that you can visually check the values throughout the day.

Next, you can create a chart that looks similar to the example in Figure 7-8.

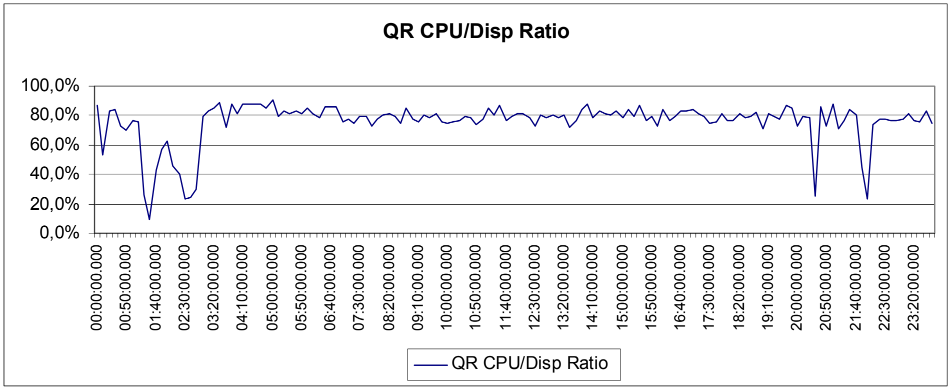


Figure 7-8 QR CPU-to-dispatch ratio

The chart in Figure 7-8 shows that the QR TCB CPU-to-dispatch ratio for the region is stable during its online service hours. The ratio is far below 80% when the region is less busy during the night. z/OS Workload Manager (WLM) reduces the processor availability for the region when CICS does not run enough transactions. The region must show a QR CPU-to-dispatch ratio of 80% during the peak times of your regions. The ratio is expected to be less than 80% if your regions are not busy or if the goals of WLM are not aggressive enough.

L8 TCBs: The CPU-to-dispatch ratio cannot be calculated for L8 TCBs.

Analyzing CPU utilization using RMF workload activity reports

In the previous section, we investigated the QR CPU-to-dispatch ratio to ensure that the CICS main TCB uses sufficient processor capacity. If the QR CPU-to-dispatch ratio is not at or greater than 80% during peak time, provide an RMF workload activity report to investigate the performance class that your region is running in. If the QR TCB is unable to obtain sufficient CP capacity, it is likely to get CPU delays for open TCBs that run threadsafe applications.

Earlier we collected SMF records 110 and RMF-SMF records 70-78. To create a WLM activity report, we used the job control language (JCL) shown in Example 7-13.

Example 7-13 RMF post processor JCL

```
//RMFPP EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DISP=SHR,DSN=KLEIN4.UNTERSED.VS6.D279.T0934.TERSED
//* IF OMITTED USES SMF BUFFER
```

```
//MFPMSGDS DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 SYSID(VS6)
 REPORTS(CPU)              /* CPU REPORT */
 SYSRPTS(WLMGL(SCPER))     /* WLM ACTIVITY REPORT */
 RTOD(0001,2359)           /* HHMM TO HHMM */
 SYSOUT(O)                 /* SYSOUT CLASS */
```

We used an existing service class, CICSSLOW, for one of the CICS regions. We started a nonthreadsafe workload that intensely uses CPU time. Now we can use the workload activity report in Example 7-14 to investigate the CPU utilization for the service class. The APPL % section shows CPU utilization based on uniprocessor capacity, meaning that the values can exceed 100% in systems with more than one processor. To get the system utilization in a sysplex, this value must be divided by the number of processors.

We can also use the workload activity report to investigate how the system meets its goals. If goals are not aggressive enough, you might see a QR CPU-to-dispatch ratio less than 80% during peak time.

After your threadsafe conversion project, investigate the workload activity report again. Make sure that the values for EXEC DELAY % and APPL % do not show any constraints and that the system still meets your response time goals.

Example 7-14 Workload activity report

```
REPORT BY: POLICY=STANDARD   WORKLOAD=CICS       SERVICE CLASS=CICSSLOW   RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
AVG       0.99  ACTUAL                 0  SSCHRT   0.0  IOC         0   CPU 1707.254  CP     94.84
MPL       0.99  EXECUTION              0  RESP     0.0  CPU    47016K   SRB    0.008  AAPCP   0.00
ENDED        0  QUEUED                 0  CONN     0.0  MSO         0   RCT    0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN              0  DISC     0.0  SRB       232   IIT    0.000
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.0  TOT    47016K   HST    0.000  AAP     0.00
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    26119   AAP    0.000  IIP     0.00
AVG ENC   0.00  STD DEV                0                                IIP    0.000
REM ENC   0.00                                          ABSRPTN    27K
MS ENC    0.00                                          TRX SERV   27K

GOAL: RESPONSE TIME 000.00.02.000 AVG

         RESPONSE TIME EX   PERF  AVG   --EXEC USING%--  -------------- EXEC DELAYS % -----------
SYSTEM   HHH.MM.SS.TTT VEL% INDX ADRSP  CPU AAP IIP I/O  TOT CPU

VS6      000.00.00.000 99.5  N/A   1.0   99 0.0 0.0 0.0  0.5 0.5
```

Investigating QR delays

Before you convert your applications to run threadsafe enabled, analyze the values for QR TCB delays. We expect to reduce the values for QR TCB delays when application programs are converted to run threadsafe. Therefore, we need to understand any QR contention before we start the project.

Figure 7-9 shows the values of QR TCB delays on a timeline throughout the day. To understand why the values increase from 8:00 a.m. until 17:00 p.m., we draw a second graph that shows the number of tasks per interval. The graphs for QR TCB delays and the number of tasks almost match. Therefore, we can say that, in this case, the more tasks we run, the higher QR TCB delays get. The values for QR TCB delays are in the low milliseconds, which is not critical. Because we used average values per interval, we have considerable peak values for QR TCB delays. If you have many peak values of more than 150 ms during peak time, consider enabling the option to run applications as threadsafe or to mirror your CICS region to use another QR TCB.

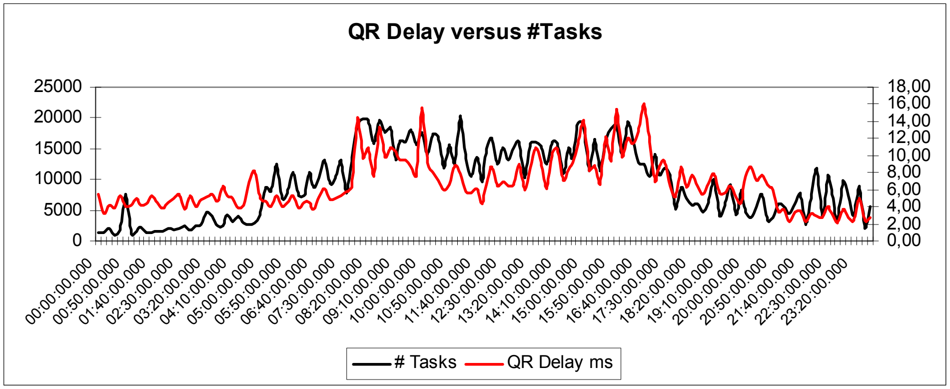


Figure 7-9 QR delay versus the number of tasks

Figure 7-10 is a comparison chart that shows how QR TCB delays are influenced by the number of TCB mode switches. This situation is more likely. The more QR-L8-QR TCB mode switches occur, the more tasks wait to get dispatched on the QR TCB.

We expect the number of TCB mode switches to come down drastically after the threadsafe conversion. Therefore, analyze the values for QR delays again after the project is completed.

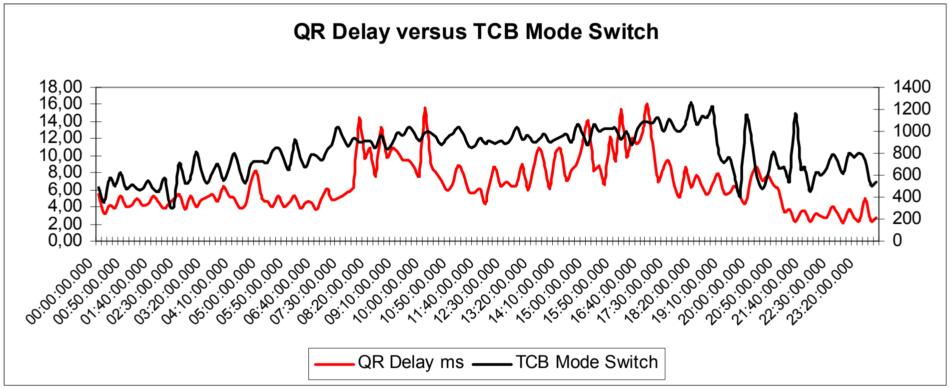


Figure 7-10 QR delay versus the number of TCB mode switches

7.7.3 Analyzing the DB2 attachment facility

Review the DB2 attachment settings before and after conducting a threadsafe conversion project.

Priority parameter

Review the priority for pool and db2entry threads. The current setting might be lower than the default of HIGH to EQUAL or LOW to reduce QR TCB contention. In a threadsafe environment, the **PRIORITY** parameter can be set back to the default value when QR TCB delays are in the lower milliseconds range.

DPMODE=HIGH assigns a higher scheduling priority to the DB2 TCB than to the CICS main TCB. In nonthreadsafe environments, this setting can lead to delays in the scheduling of the QR TCB. In a CPU constraint environment, CICS cannot accept the answers from the DB2 subtask fast enough, because the scheduling of another L8 TCB takes precedence over the scheduling of the QR TCB.

By using **DPMODE=EQUAL**, the thread TCBs have an MVS dispatching priority that is slightly lower than the CICS main task TCB.

Tip: Review the **PRIORITY** parameter for pool and entry threads when applications are converted to run threadsafe.

In general, use the following settings:

HIGH For high-priority and high-volume transactions

EQUAL For transactions that are more intensive for CICS than for DB2 (such as short, simple SQL statements)

LOW For low-priority, short SQL transactions, especially non-terminal-driven transactions

DB2 pool and entry threads

After conducting your threadsafe conversion project, review the settings for pool and entry threads.

You can use the SMF 110 subtype 2 records to review the performance of DB2 threads. SMF 110 subtype 2 records contain end of day (EOD) or interval (IN) statistics that can be formatted using CICS PA or the CICS supplied **DEFUSED** utility. First, verify that the defined number of threads is consistent. The number of thread TCBs cannot be less than the number of pool threads plus the sum of all DB2ENTRY threads.

Figure 7-11 shows the value that we defined on the **TCB LIMIT** parameter, which is the number of thread TCBs that the DB2 attachment can use. We defined 50 pool threads on the **THREADINESS** parameter, and therefore, allow 50 threads to be used by all DB2ENTRY definitions.

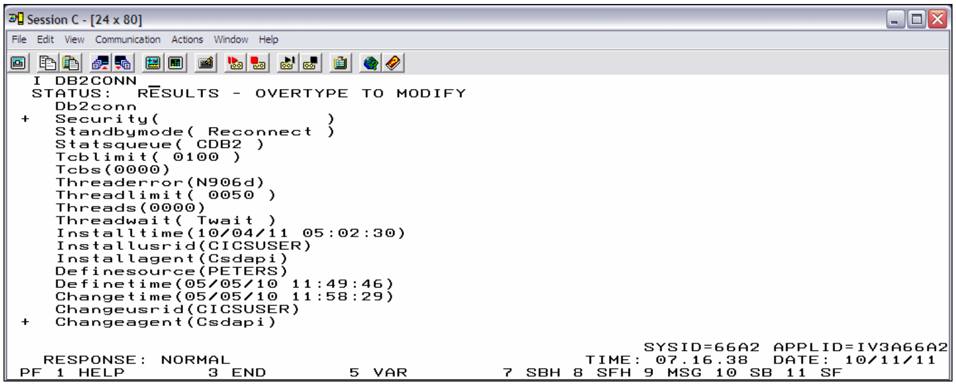


Figure 7-11 DB2 connection definition

Figure 7-12 shows the number of DB2 entry threads that we use per CICS region.

DB2 Entry Data		APPLIDs CIAA2	CIAA3	CIAA5	CIAA6	CIAA7	CIAA8	CIAAB	CIAAC	CIAAD	CIAAE	CIAAF
C102	ThreadLim		15									
	TreadHWM		4									
	PThreadLim		15									
	PThreadHWM		4									
	TaskTotal		2066									
C100	ThreadLim				20				20			20
	TreadHWM				6				18			17
	PThreadLim				12				12			12
	PThreadHWM				5				12			12
	TaskTotal				194				1131			26720

Figure 7-12 DB2 entry threads

DB2 thread TCBs are served by CICS open L8 TCBs. Therefore, we must define enough TCBs on the CICS **MAXOPENTCBS** parameter. We must also consider that we might need additional open TCBs for the CICS DBCTL interface, WMQ, or TCP/IP sockets.

Thread reuse and thread wait overflows

Review the values for thread wait overflows and thread reuse for protected threads. If **THREADWAIT=POOL** is specified, requests for threads are transferred to the pool when the value of **THREADLIMIT** is exceeded. When a transaction overflows to the pool, the transaction is controlled by the PRIORITY, THREADLIMIT, and THREADWAIT attributes that are specified for pool threads in the DB2CONN definition. These attributes in the DB2ENTRY definition are ignored. Avoid thread wait overflows.

Transactions that use the same DB2ENTRY and protected threads can reuse a thread if it is available. This reuse is because the threads remain active for about 45 seconds after being released, depending on the PURGECYCLE value. Check that protected threads are reused frequently.

Checking and limiting thread reuse: CICS TS V4.2 provides new facilities for you to check and limit the number of times a thread can be reused. When a thread reaches its reuse limit, CICS terminates it to free DB2 resources. Long-running CICS DB2 threads can cause resource issues in DB2, particularly in storage. You can now set a reuse limit on the DB2CONN definition to specify this reuse limit.

Figure 7-13 shows the DB2 entries that we use. No thread wait overflow is indicated, and DB2 entries show thread reuse.

		APPLIDs										
DB2 Entry Data		CIAA2	CIAA3	CIAA5	CIAA6	CIAA7	CIAA8	CIAAB	CIAAC	CIAAD	CIAAE	CIAAF
C102	ThreadReuse ThreadWTOverfl		2051 0									
C100	ThreadReuse ThreadWTOverfl				187 0				1076 0			24905 0

Figure 7-13 Thread reuse and thread wait overflows

7.8 Diagnosing performance problems

You can diagnose performance problems by using the following data sources:

- Message IEF374I
- SMF Type 30
- RMF workload activity reports
- CICS statistics
- CICS monitoring

This section reviews two types of performance problems:

- Increased response time
- CPU increase

7.8.1 Defining the problem

Usually, performance problems fall into two categories:

- Poor response time

  Users start complaining that their response time increased. The increase is usually an indication of a resource constraint somewhere in the system. Examples include waits of various types (enqueues, locks, slow I/O, string, or buffer waits); file, journal, or logger bottlenecks; slowdowns in DB2, DBCTL, or another subsystem; and not enough CPU cycles to support the workload.

  The bottleneck can be within the CICS region. For example, applications are doing more GETMAINs or FREEMAINs after implementing new functions in the Language Environment. Applications might be spending more time in DB2, IMS, or another database product after implementing a new release. The bottleneck can also be outside the CICS region. For example, DASD contention is slowing writes and causing applications to wait.

- Increased CPU time

  For example, a user paying the bill is complaining about increased CPU costs, or someone notices that another user is using more CPU than before. In such situations, determine how the CPU increase is being measured. For example, is it from SMF records or from a report from a vendor product:

  If the reported increase is from a vendor product, how is the CPU usage determined? Is it calculated from a formula based on the hardware type, or is it calculated from data reported by SMF or RMF?

  If the increase is reported following a change in hardware, was the formula used in the vendor product that was updated to reflect the new hardware? Is there a real increase, or are reports erroneous reporting an increase?

Each problem requires slightly different approaches and often different data to diagnose the cause and resolve the problem. You might consider also exploring the following questions:

- What is the scope?

  Identify the scope of the problem. Is it an overall slowdown. For example, are all transactions affected, or are only a few transactions affected, such as a single application? The scope of the symptoms helps to determine where to look next. Because you are trying to identify the resources that are adversely affecting response time, look at information that tells you about resource usage in the system:

  - System-wide resource usage (CICS statistics)
  - Usage by individual transactions (CICS monitoring data)
  - MVS data in RMF reports

- When did the problem start?

  Can you associate the onset of the problem with a change or a specific time?

- What changed?

  - Was maintenance applied (such as CICS, MVS, VTAM, Language Environment, and OEM products)?
  - Were application changes or hardware changes (such as processor, DASD, LPAR configuration, and new NCP or I/O configuration) made? Were any new releases of software installed?

7.8.2 Performance hierarchy

Performance problems represent a class of problems that are often difficult and time consuming to resolve. Similar to learning how to diagnose an 0C1 program check, you must learn to use and understand the tools and methods that can assist in problem resolution.

Figure 7-14 illustrates a performance hierarchy.

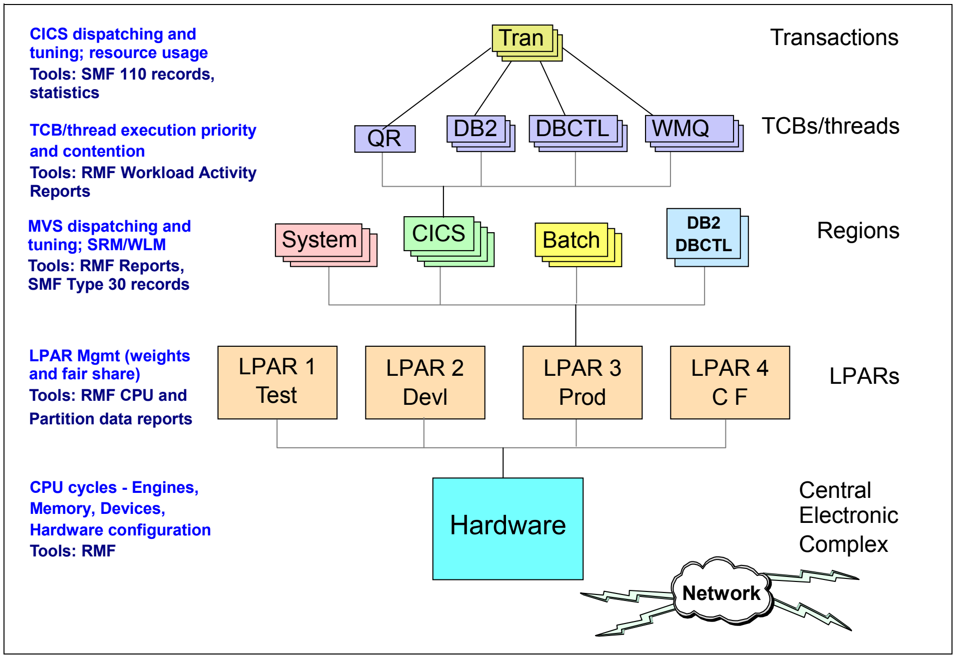


Figure 7-14 Performance hierarchy diagram

The *central processor complex* (CPC) is a physical collection of hardware that consists of main storage, one or more central processors (CPs), timers, and channels. The hardware resources of a central electronic complex (central processors, main storage, expanded storage, and channel paths) can be divided into LPARs. Each LPAR runs a separate copy of an operating system (such as z/OS, MVS, VM, VSE, and Linux).

Each layer shown in Figure 7-14 builds on the resources of the lower levels. For example, for CICS to dispatch a task, an engine (CP) must be made available (allocated) to the z/OS image, which in turn, dispatches the CICS region (that is, assigns a CP to the CICS region). The CICS region can then dispatch CICS tasks by using the CP allocated by z/OS.

Difficulty in solving a performance problem is reduced by a better understanding of the layers that provide the execution environment. The underlying resources allocated to CICS are provided by the hardware. If hardware resources are insufficient or poorly configured, CICS performance is affected. Tuning and application changes can reduce resource demand.

The MVS (z/OS) dispatcher, with assistance from WLM, allocates the available LPAR resources between regions (address spaces). MVS tuning can be performed to increase the share of these resources for a region, within the scope of the resources available to the LPAR. If an LPAR has insufficient resources, consider investigating the reports that detail the LPAR management data.

> **Important:** A lack of underlying hardware resources is nearly impossible to tune by using software.

The following sources can cause problems:

- Transactions
  - Application design
    - ENQ, locks
    - Resource usage
  - Routing
  - Priority
  - Classing
- TCBs/Threads
  - TCB usage
    - Open TCBs
    - Subtasking
    - Java
  - DB2
    - DPMODE
- Region or operating system
  - Region priority
  - Resource allocation
    - SRM
    - WLM goals, definitions, service level agreement (SLA)
    - Strings, buffers, and so on
- LPAR
  - Configuration
    - Weights, fair share, CPs
- Hardware
  - DASD
  - CPU
  - Timers

7.8.3 Key performance indicators

This section highlights key performance indicators (KPIs).

Indicators from System Management Facilities

CICS performance records (SMF 110 subtype 1) can generate the following KPIs:

- Wait for redispatch time (DISPWAIT), which is a measure of the length of time following the posting of the ECB until CICS redispatches the task
- Wait for QR Mode TCB time (QRMODDLY), which is the elapsed time a task waited for redispatch on the CICS QR TCB

  QRMODDLY is a subset of DISPWAIT time.
- Average CPU per task
- Average response time
- Wait times associated with resources, such as the following examples:
  - File wait
  - MRO link wait time
  - RMI time
  - RMI suspend time
  - TCLASS delay
  - First dispatch delay
- QR TCB CPU-to-dispatch ratio
- Transaction rate
- Log writes per second

Indicators from Resource Management Facility

RMF can generate the following KPIs:

- Workload activity reports:
  - TCB CPU seconds in the interval
  - APPL% (the percent an engine (CP) used in the interval)
  - Average CPU per task
  - Divide APPL% by RMF transaction rate
  - MVS busy
- RMF transaction-level report classes:
  - Average response time
  - Transaction rate

- Partition data report:
  - LPAR logical and physical busy
  - Central processor complex busy

7.8.4 Performance data sources

The following data sources provide performance information.

Message IEF374I

Message IEF3741 is written to the JESYSMSG log for the job during step termination. It shows the virtual storage above and below the 16-MB line. The sample in Figure 7-15 shows a one-step CICS job, where the job and step numbers are the same. It also includes all TCB and service request block (SRB) time in the region.

```
IEF373I STEP/CICS/START 2002349.1112
IEF374I STEP/CICS/STOP  2002349.1310 CPU 62MIN 37.76SEC SRB 10MIN 28SEC VIRT 5420K SYS 344K EXT 116612K SYS 16896K

IEF375I  JOB/IYOT122/START 2002349.1112
IEF376I  JOB/IYOT122/STOP  2002349.1310 CPU   62MIN 37.76SEC SRB   10MIN 28.05SEC
```

Figure 7-15 IEF374I message

SMF records

Performance data is captured in many places within a z/OS system. During step termination, information is collected as SMF type 30 records. The processor time used by the collective TCBs and SRBs in the address space is written to the JES log (JESYSMSG) during step termination, as message IEF374I.

The data from these records can be used to define the overall processor time associated with an address space. From a CICS perspective, these numbers include all time that is not associated with the actual transactions processed.

Dividing the total processor time by the number of transactions processed does not give a true representation of resources used. For example, suppose 50% of the transactions read a record from a VSAM file and show a message on the terminal and 50% of the transactions issue 100 EXEC SQL calls. Dividing the processor time by the total transactions does not provide a true picture of resource usage. In a case where the CPU per transaction suddenly increases, it is difficult to understand the root cause.

z/OS collects statistical information about the SMF interval. The interval is defined by using the INTVAL(tt) option in the SMFPRMxx member of SYS1.PARMLIB. To view the status of the SMF data sets, use the **D SMF** command. The SMF options in use can be displayed by using a **D SMF,O** command, as shown in Figure 7-16 on page 190.

```
COMMAND INPUT ===> /d smf,o
 IEE967I 13.10.10 SMF PARAMETERS 330
     MEMBER = SMFPRMZI    <------------------------------------- SYS1.PARMLIB member
         MULCFUNC -- DEFAULT
          . . . . .
         SUBSYS(OMVS,TYPE(0,30,70:79,90,88,89,99,101,110,245)) -- PARMLIB
         SUBSYS(OMVS,INTERVAL(SMF,SYNC)) -- PARMLIB
         SUBSYS(OMVS,NOEXITS) -- PARMLIB
         SUBSYS(STC,NODETAIL) -- SYS
         SUBSYS(STC,TYPE(0,30,70:79,88,89,90,99,101,110,245)) -- PARMLIB <- Record types collected
         SUBSYS(STC,INTERVAL(SMF,SYNC)) -- PARMLIB
         SUBSYS(STC,EXITS(IEFACTRT)) -- PARMLIB
        INTVAL(05) -- PARMLIB     <---------------------------------- SMF Interval
         NOPROMPT -- PARMLIB
         LISTDSN -- PARMLIB
         DSNAME(SYSD.MAN4) -- PARMLIB
         DSNAME(SYSD.MAN3) -- PARMLIB                                   SMF dataset names
         DSNAME(SYSD.MAN2) -- PARMLIB
         DSNAME(SYSD.MAN1) -- PARMLIB
```

Figure 7-16 The d smf, o command

The RMF function of z/OS gathers a large amount of information about resource usage that is written to SMF as record types 70 - 78. The information includes TCB and SRB times, DASD I/O counts, a breakdown of the response times, CP usage, and more.

It is also possible to obtain the number and rate of CICS transactions completed during the SMF interval. However, as in the case of the contents of the IEF374I message, this information is presented at a region level. For example, to calculate the CPU time per transaction, the total CPU used in the interval is divided by the number of transactions completing in the interval.

RMF reports are generated by using the RMF post processor (ERBRMFPP). For maximum granularity, each CICS region must be assigned to a separate reporting class. In addition, report classes must be defined to display the CICS transactions that complete during the interval. For example, a report class might be generated for all transactions that begin with JOR (such as JOR1, JOR2, and JOR3) with a second report class defined for transactions beginning with DB2 (such as DB21 and DB22). This way, transaction rates and response times can be reported for individual sets of transactions rather than the whole region. Up to 999 report classes can be in a sysplex. Report classes are defined by using the WLM facilities (**=WLM** in ISPF).

SMF Type 110 subtype 1 records

CICS collects performance data at the task level (activated through the **MNPER** system initialization table (SIT) parm, **CEMT SET MONITOR**, or **EXEC CICS SET MONITOR**). Three classes of performance monitoring can be selected:

- Performance class data (MNPER)
- Exception class data (MNEXE)
- New transaction resource class data (MNRES), with the addition of CICS TS V2.2 APAR PQ63143

This class data is present at the base code level in later releases of CICS.

Performance class data is detailed at the transaction level. It provides such information as response time, time spent waiting for a resource or I/O, and CPU time. At least one performance record is written for each transaction at task termination time. For long-running tasks, the MNFREQ option can be used to cause periodic records to be written.

Exception class monitoring data provides information about CICS resource shortages at the transaction level. This data can be used to identify system constraints that affect transaction performance. An exception record is written to SMF when the shortage is resolved. For a detailed description of exception records, see *CICS Performance Guide*, SC34-6247 (for CICS TS V2) or *CICS Transaction Server for z/OS Performance Guide*, SC34-6452 (for CICS TS V3).

With the addition of APAR PQ63143, transaction resource class data provides additional transaction-level information about file resources. To activate transaction resource collection for files, assemble a Monitor Control Table (MCT) with **FILE=parm**.

SMF Type 110 subtype 2 records

CICS interval statistics are collected for CICS resource usage at the expiration of each statistics recording interval and written to SMF as type 110 subtype 2 records. To specify the interval, use the STATINT SIT parameter, and specify **STATRCD=ON**. Otherwise, as is the case with older releases of CICS, the interval is set by using the CICS master terminal function **CEMT SET STATISTICS** or the **EXEC CICS SET STATISTICS** command.

The interval statistics can be considered as CICS region-level data, but at a more granular level than RMF data (for example, dataset level statistics versus the DASD activity in RMF).

SMF Type 30 records

The SMF 30 records contain *region-level statistics*. You can choose from several methods to view the records. The record shown in Figure 7-17 was selected by using a SORT with the control cards shown in Example 7-15 on page 192.

In the SMF type 30 record (offset 5 =x'1E'), subtype 5 (offset x'16'), a pointer to the processor section is found at offset x'38', with the section length at x'3C' and the number of processor sections at offset x'3E'.

```
000000 28E00000 DE1E0048  5D8D0102 349FE2E8  E2C4D1C5 E2F20005  000000B0 00260001 *........)SYSDJES2..........*
000020 000000D6 00B80001  0000018E 00400001  000001CE 00080001  000001D6 00540001 *...O..................O....*
000040 00000386 000A0002  0000022A 00C80001  000002F2 007C0001  0000036E 00180001 *...F..........2.@.....>....*
000060 00000390 001E0042  00000000 00000000  000003AE 00280000  00000386 00280000 *......................F....*
000080 00000B4C 00600019  00000000 000014AC  004C0043 00000000  00002890 00500001 *...<.-......<...........&..*
0000A0 00000000 00000000  00140000 00000000  00050000 F0F5E2D4  C6404040 4040E2D7 *...............05SMF     SP*
0000C0 F74BF04B F440E2E8  E2C44040 4040E6E2  C3E9D7D3 C5E7C9E8  D6E3F1F2 F2404040 *7.0.4 SYSD WSCZPLEXIYOT122 *
0000E0 40404040 40404040  40404040 4040C7D9  C1E4C5D3 4040D1D6  C2F1F6F5 F4F90001 *          GRAUEL  JOB16549..*
000100 C1000000 0000003D  8CEE003D 8D0D003D  8CEE0102 349F003D  8CD80102 349F003D *A...................Q......*
000120 8CE70102 349FD1C9  D440C7D9 C1E4C5D3  40404040 40404040  4040E2E8 E2F14040 *.X..JIM GRAUEL        SYS1  *
000140 4040C7D9 C1E4C5D3  40400000 00000000  00004040 40404040  40404040 40404040 *  GRAUEL  ....              *
000160 4040C140 40404040  40400000 00000000  00000000 00000000  00000000 00000000 *  A       .................*
000180 00000000 00000000  00000000 00000000  0053015B B9CF0000  00000000 00000000 *..............$............*
0001A0 0A38B1CB 00000000  00000000 00000A38  942A019C 71FB0[illegible]7D  1A93015B B4F70000 *...........M......'.L.$.7..*
0001C0 00000000 00000000  00000000 00000222  02000000 0000
                                                      0000  80000005 BBE00000 *..........................*
0001E0 F5550000 00050000  00000000 00000000  00000000 00000000  00000000 00000000 *5..........................*
000200 000F0000 44C90000  00000000 00000000  00000000 00000000  00000000 00000000 *.....I.....................*
000220 00000000 00000000  00000000 0080152C  01580000 00000000  00000000 00000000 *..........................*
000240 00000000 00000000  00000000 00000000  00000000 00000000  00000000 00000000 *..........................*
000260 00000000 00000000  00000000 00085891  2D9C008F A0006D40  00000005 60000108 *..........J......_ ....-...*
000280 00000054 B000071E  10000000 23E80000  00000000 00000000  00000000 00000000 *........Y..................*
```

```
 + 4  0005BBE0 TCB CPU Time     = 3757.76 seconds   = 62 Min. 37.76 seconds
 + 8  0000F555 SRB CPU Time     =  628.05 seconds   = 10 Min. 28.05 seconds
 +2C  000044C9 CPU for I/O      =  176.09 seconds   =  2 Min. 56.09 seconds
```

Figure 7-17 SMF type 30 record

Example 7-15 shows the control card that was used.

Example 7-15 Sort example

```
//SYSIN    DD *
      SORT FIELDS=(47,8,CH,A,11,4,PD,A,7,4,BI,A)
       INCLUDE COND=(6,1,FI,EQ,30)
```

Figure 7-18 shows the SMF Type 30 layout.

```
Processor Section:

Offsets     Name       Length   Format      Description
0      0    SMF30PTY   2        binary      Address space dispatching priority
                                              (note this field is not valid in goal mode)
4      4    SMF30CPT   4        binary      Step CPU time under the task control block (TCB), in
                                             hundredths of a second (including enclave time,
                                             preemptable class SRB time, and client SRB time).

8      8    SMF30CPS   4        binary      Step CPU time under the service request block (SRB),
                                             in hundredths of a second.


44     2C   SMF30IIP   4        binary      Amount of CPU time used to process I/O interrupts, in
                                            hundredths of a second.

52     34   SMF30HPT   4        binary      CPU time consumed for the step, in hundredths of a
                                            second, to support requests for data to be transferred
                                            between a hiperspace and an address space, when the
                                            hiperspace is backed by expanded storage.  The CPU
                                            time may vary depending on the availability of
                                            expanded storage.


68     44   SMF30ASR   4        binary      Additional CPU time accumulated by the preemptable
                                            SRBs and client SRBs for this job, in hundredths of a
                                            second.  This value is also included in the value in
                                            SMF30CPT.

Refer to OS/390 V2R4.0 MVS System Management Facilities (SMF) - SMF30COF
```

Figure 7-18 SMF Type 30 record layout notes

DFHJUP was then used to print the records in hex.

SMF 30 records consist of a header and several sections, including processor, performance, and I/O. The Processor section contains such information as the TCB (+4) and SRB (+8) times, which are reported in the IEF374I message.

An SMF 30 subtype 2 record is written at the completion of each SMF interval. A subtype 5 record is written at job termination.

7.8.5 RMF Workload Activity reports

RMF provides a wealth of information that is invaluable in the resolution of performance problems. You can use this information to help you understand how changes affect CPU, storage, and DASD usage.

Figure 7-20 on page 195 shows a WLM Workload Activity Report that presents data collected for report classes RIYOT122 and RJORIY1. Report class RIYOT122 provides RMF information about a CICS region called IYOT122.

Report class RJORIY1 was defined to show the number of transactions beginning with JOR, which ended in the SMF interval.

Report classes are defined by using the WLM ISPF panels (**=WLM** option 2.6, then enter a 3 beside CICS). Figure 7-19 shows report classes for TRANIDs starting with JOR (report class RJORIY1), and a second report class (RCICSIY1) for all transactions starting with C.

```
 Subsystem-Type  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type      Row 1 to 8 of 10
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : CICS        Fold qualifier names?   Y  (Y or N)
Description  . . . CICS transaction level rules

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------               -------Class--------
Action    Type       Name     Start                Service     Report
                                        DEFAULTS: ________    ________
 ____  1    SI      IYOT1     ___                  ________    ________
 ____  2    TN        JOR*    ___                  ________    RJORIY1
 ____  2    TN        C*      ___                  ________    RCICSIY1
```

Figure 7-19 RMF report classes

The report interval is listed in the start and end times at the top of the page. In Figure 7-20 on page 195, the minimum interval is defined by the INTVAL() parm in the SMFPRMxx member of SYS1.PARMLIB. In the samples collected, the interval was set to 5 minutes:

```
INTVAL(05)                  /* SMF GLOBAL RECORDING INTERVAL */
```

Ensure that the SMF 70 - 79 records are being collected with the CICS 110 records.

The records to be collected are also defined in the SMFPRMxx member:

```
SUBSYS(STC,EXITS(IEFACTRT),INTERVAL(SMF,SYNC),
                                TYPE(0,30,70:79,88,89,90,99,110,245))
          SUBSYS(OMVS,NOEXITS,INTERVAL(SMF,SYNC),
                                TYPE(0,30,70:79,90,88,89,99,110,245))
```

When the reports are formatted, a larger interval can be report than was specified in the SMFPRMxx member by using the DINTV parm. However, the length of the minimum interval is the value specified for INTVAL.

SMF88 data: SMF88 data that is formatted using IXGRPT1 cannot summarize at a larger interval than the interval used for data collection (the INTVAL value specified in the current SMFPRMxx).

- **Report Class by Transactions**

```
REPORT BY: POLICY=WLMPOL                        REPORT CLASS=RJORIY1
                                              DESCRIPTION =Report class -JOR trans IYOT1

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT
AVG       0.00  ACTUAL                 161
MPL       0.00  EXECUTION                0
ENDED    27713  QUEUED                   0
END/S    92.39  R/S AFFINITY             0
#SWAPS       0  INELIGIBLE               0
EXCTD        0  CONVERSION               0
AVG ENC   0.00  STD DEV                 56
REM ENC   0.00
MS ENC    0.00
```

- **27713 JOR* tasks completed in the interval**
- **92.39 transactions ended per second**
- **Average transaction response time is .161 seconds**

- **Report Class by Region**

```
REPORT BY: POLICY=WLMPOL                                REPORT CLASS=RIYOT122
                                                        DESCRIPTION =

TRANSACTIONS  TRANS.-TIME  HHH.MM.SS.TTT --DASD I/O-- ---SERVICE---- --SERVICE RATES-- PAGE-IN RATES  ----STORAGE----
AVG      1.00 ACTUAL                   0 SSCHRT  3881 IOC    116525  ABSRPTN    123986 SINGLE    0.0 AVG     7756.33
MPL      1.00 EXECUTION                0 RESP     1.2 CPU     2048K  TRX SERV   123986 BLOCK     0.0 TOTAL   7755.58
ENDED       0 QUEUED                   0 CONN     0.9 MSO    34897K  TCB         156.5 SHARED    0.0 CENTRAL 7755.58
END/S    0.00 R/S AFFINITY             0 DISC     0.1 SRB    127300  SRB           9.7 HSP       0.0 EXPAND     0.00
#SWAPS      0 INELIGIBLE               0 Q+PEND   0.3 TOT    37189K  RCT           0.0 HSP MISS  0.0
EXCTD       0 CONVERSION               0 IOSQ     0.0 /SEC   123974  IIT           7.4 EXP SNGL  0.0 SHARED     1.00
AVG ENC  0.00 STD DEV                  0                             HST           0.0 EXP BLK   0.0
REM ENC  0.00                                                        APPL %       57.9 EXP SHR   0.0
MS ENC   0.00
```

Figure 7-20 Workload Activity Reports by class

In Figure 7-20, note the following fields:

MPL Multiprogramming level. The number of address spaces active in this service or report class during the interval.

TCB Provides the CPU seconds accumulated in TCB mode during the collection interval.

SRB Provides the CPU seconds accumulated in SRB mode during the collection interval.

APPL% The percentage of an engine (CP) used during the collection interval.

AVG The average number of central and expanded storage frames allocated to address space identifiers (ASIDs) in the report class.

SINGLE The average rate at which pages are read into main storage from auxiliary storage (DASD).

SSCHRT The number of start subchannels (SSCHs) per second in the reported interval.

RESP Average DASD response time (in milliseconds).

ENDED Reports the number of CICS transactions that ended during the SMF interval.

END/S Provides the transaction rate for those transactions reported, as defined in the report class.

TRANS. TIME The transaction response time.

7.8.6 CICS PA reports

CICS PA can provide a report on transaction response time as reported by WLM. It provides a slightly different perspective. For example, in Figure 7-21 on page 197, it reports by both service class (which is not being used) and report class. In addition to the information provided in the RMF report, CICS PA provides data showing the standard deviation and 90% peak. For further examples of CICS PA reports that can be used during threadsafe analysis, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245.

```
 V3R2M0                                                  CICS Performance Analyzer
                                                  Workload Manager Activity Summary by Service Class

WKLD0001 Printed at 12:03:45  3/15/2011 Data from 15:47:53  6/01/2004 to 15:58:53  6/01/2004    Page 1

Service                                      -------------- Response Time --------------
Class      APPLID      Phase         #Tasks     Average     Std Dev     90% Peak     Maximum

FINSCLAS   CICPTOR1    BTE              176       .5665       .4369        .8753      1.3745
           CICPAOR1    EXE              169       .5239       .4564        .8280      1.1684
STOSCLAS   CICPTOR1    BTE             2123       .9265       .3981       1.2675      2.0246
           CICPAOR1    EXE             2078       .8639       .3627       1.1927      1.8327
QUIKSERV   CICPAOR1    BTE             5476       .3846       .1976        .4673       .6571
LONGSERV   CICPAOR1    BTE             1958      1.5861       .8392       2.2179      5.5094
*  Grand Total   *     BTE             9733       .6853       .4812       1.3718      2.0246
*  Grand Total   *     EXE             2247       .8047       .3927       0.9201      5.5094

 V3R2M0                                                   CICS Performance Analyzer
                                                  Workload Manager Activity Summary by Report Class

WKLD0001 Printed at 12:03:45  3/15/2011 Data from 15:47:53  6/01/2004 to 15:58:53  6/01/2004     Page     2

Report                                       -------------- Response Time --------------
Class      APPLID      Phase         #Tasks     Average     Std Dev     90% Peak     Maximum

FINSCLAS   CICPTOR1    BTE              176       .5665       .4369        .8753      1.3745
           CICPAOR1    EXE              169       .5239       .4564        .8280      1.1684
STOSCLAS   CICPTOR1    BTE             2123       .9265       .3981       1.2675      2.0246
           CICPAOR1    EXE             2078       .8639       .3627       1.1927      1.8327
QUIKSERV   CICPAOR1    BTE             5476       .3846       .1976        .4673       .6571
LONGSERV   CICPAOR1    BTE             1958      1.5861       .8392       2.2179      5.5094
*  Grand Total   *     BTE             9733       .6853       .4812       1.3718      2.0246
*  Grand Total   *     EXE             2247       .8047       .3927       0.9201      5.5094
```

Figure 7-21 CICS PA workload report

CICS PA provides extensive reports and analysis of the CICS performance monitoring record. CICS writes a performance monitoring record to SMF as 110 record subtype 1 when each task completes. The records contain an extensive amount of information about the task showing everything from response time, CPU used, to suspend/wait times. Each segment of response time is reported. For example, if the task issues 100 file control calls, the calls are detailed regarding the type (such as read, read/update, and rewrite). The total file I/O wait time is recorded. Figure 7-22 shows an example CICS PA report.

```
 V3R2M0                                             CICS Performance Analyzer
                                                       Performance Summary

SUMM0001 Printed at 12:03:45  3/15/2011     Data from 11:10:51  2/14/2005 to 11:34:13  2/14/2005                  Page      1

                            Avg      Avg      Max      Avg      Avg      Avg      Avg      Avg      Avg      Avg      Avg
Tran     APPLID    #Tasks  Int Resp Response Response Dispatch User CPU  Suspend DispWait RMI Susp  FC Wait  IR Wait  TC Wait
                              Time     Time     Time     Time     Time     Time     Time     Time     Time     Time     Time
AADD     IYK2Z1V1       5    .0020    .0020    .0023    .0019    .0012    .0001    .0000    .0000    .0000    .0000    .0000
AADD     IYK2Z1V3       5    .0330    .0330    .0945    .0303    .0035    .0028    .0027    .0000    .0000    .0000    .0000
AADD                   10    .0175    .0175    .0945    .0161    .0024    .0014    .0013    .0000    .0000    .0000    .0000
```

Figure 7-22 CICS PA performance summary

7.8.7 The DFH0STAT utility

The **DFH0STAT** utility is supplied as a compiled sample program. The source continues to be available as a sample COBOL program in SDFHSAMP. You need to install CSD group DFH$STAT. You can run the utility as the STAT transaction to collect CICS statistics and write them to the JES spool. You can then view the output under TSO (Figure 7-23). The SIT parm **SPOOL=YES** is required.

The information provided through the **DFH0STAT** utility is also available in the CICS shutdown statistics.

> **Important:** Do not use the **DFH0STAT** utility in place of the shutdown statistics. It reports only information that is provided in the current statistics interval, and the unsolicited records are lost.

```
Applid IV5A67A2  Sysid 67A2  Jobname CICS67A2    Date 10/14/2011  Time 08:41:37            CICS 6.7.0            PAGE
______________________________________________________________________________________________________________________________
System Status
_____________
  MVS Product Name. . . . . . . :  MVS/SP7.1.2                          CICS Transaction Server Level . . :  04.02.00
  CICS Startup. . . . . . . . . :  INITIAL                              MVS Workload Manager (WLM) Mode . :  Goal
  CICS Status . . . . . . . . . :  ACTIVE                               WLM Server. . . . . . . . . . . . :  No
                                                                        WLM Workload Name . . . . . . . . :  PRDBAT
  VTAM Open Status. . . . . . . :  OPEN                                 WLM Service Class . . . . . . . . :  PRDBATL0
  IRC Status. . . . . . . . . . :  CLOSED                               WLM Report Class. . . . . . . . . :
  IRC XCF Group Name. . . . . . :  DFHIR000                             WLM Resource Group. . . . . . . . :
                                                                        WLM Goal Type . . . . . . . . . . :  Discretionary
  Storage Protection. . . . . . :  INACTIVE                             WLM Goal Value. . . . . . . . . . :  N/A
  Transaction Isolation . . . . :  INACTIVE                             WLM Goal Importance . . . . . . . :         0
  Reentrant Programs. . . . . . :  PROTECT                              WLM CPU Critical. . . . . . . . . :  No
  Exec storage command checking :  ACTIVE                               WLM Storage Critical. . . . . . . :  No

  Force Quasi-Reentrant . . . . :  No                                   RLS Status. . . . . . . . . . . . :  RLS=NO
                                                                        RRMS/MVS Status . . . . . . . . . :  RRMS=NO
  Program Autoinstall . . . . . :  INACTIVE
  Terminal Autoinstall. . . . . :  ENABLED                              TCP/IP Status . . . . . . . . . . :  OPEN
```

Figure 7-23 DFH0STAT sample output

The dispatcher summary is used to track information, such as the TCB and SRB time accumulated in the address space. For the QR TCB, the CPU-to-dispatch ratio is calculated. CPU and dispatch time information is provided for all TCB modes, but the ratio is only calculated for the QR TCB. For open TCB modes, such as J8 and L8, the TCBs are not necessarily permanent. That is, they can be attached and detached during the CICS run or within a statistics interval.

Figure 7-24 shows the dispatcher report.

```
Dispatcher
__________
  Current ICV time . . . . . . . . . . . . . . :          1,000ms
  Current ICVR time. . . . . . . . . . . . . . :          5,000ms
  Current ICVTSD time. . . . . . . . . . . . . :            500ms
  Current PRTYAGING time . . . . . . . . . . . :         32,768ms
  MRO (QR) Batching (MROBTCH) value. . . . . . :              1
  Concurrent Subtasking (SUBTSKS) value. . . . :              0
  Current number of CICS Dispatcher tasks. . . :             24
  Peak number of CICS Dispatcher tasks . . . . :             25
  Current number of TCBs attached. . . . . . . :              9
  Current number of TCBs in use. . . . . . . . :              8
  Number of Excess TCB Scans . . . . . . . . . :              2
  Excess TCB Scans - No TCB Detached . . . . . :              2
  Number of Excess TCBs Detached . . . . . . . :              0
  Average Excess TCBs Detached per Scan. . . . :              0
  Number of CICS TCB MODEs . . . . . . . . . . :             21
  Number of CICS TCB POOLs . . . . . . . . . . :              5
Dispatcher TCB Modes
____________________
  Dispatcher Start Time and Date . . . . . . . :    05:03:31.052363  10/05/2011
  Address Space Accumulated CPU Time . . . . . :  0000:03:19.335565  (Not Reset)
  Address Space Accumulated SRB Time . . . . . :  0000:00:26.255088  (Not Reset)
  Address Space CPU Time (Since Reset) . . . . :  0000:00:00.190212
  Address Space SRB Time (Since Reset) . . . . :  0000:00:00.022941

   TCB    TCBs Attached    Op. System       Op. System        Total TCB          Total TCB           DS TCB       TCB CPU/Disp
   Mode  Current    Peak     Waits           Wait Time       Dispatch Time        CPU Time          CPU Time          Ratio
  ____________________________________________________________________________________________________________________________
    QR        1        1         958   0000:11:37.153064  0000:00:00.212077  0000:00:00.188448  0000:00:00.059050       88.8%
    RO        1        1           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    CO        0        0           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    SZ        0        0           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    RP        0        0           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    FO        1        1           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    SL        1        1           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    SO        1        1           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    SP        1        1           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    EP        2        2           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
    TP        0        0           0   0000:00:00.000000  0000:00:00.000000  0000:00:00.000000  0000:00:00.000000
  F1=HELP   F2=SPLIT   F3=END   F4=RETURN   F5=IFIND   F6=BOOK   F7=UP   F8=DOWN   F9=SWAP  F10=LEFT  F11=RIGHT  F12=RETRIEVE
```

Figure 7-24   DFH0STAT dispatcher report

CICS TS V2.2 introduced the ability to collect additional TCB information for the CICS address space. This function is present at the base code level in later releases of CICS. The **DFH0STAT** utility changed to show the TCB structure with CPU and storage information for each TCB.

Similar to all DFH0STAT displays, this display in Figure 7-24 is a snapshot captured at a particular time. It shows the TCBs and their status as they were when the STAT transaction was run. The number of TCBs can change over the course of a CICS run. Open TCBs (S8, L8, L9, J8, J9, X8, and X9) can be detached and a new TCB attached at a later time, which might be at the same address. Therefore, the CPU values might seem incorrect over an extended time, because more than a single TCB is being reported. Multiple displays provide a trend, but they must not be used as a substitute for the dispatcher shutdown statistics and RMF data produced for the region.

The address space accumulated TCB and SRB CPU time is displayed. Storage allocated information is provided at both address space and TCB levels.

Figure 7-25 shows the DFH0STAT Dispatcher MVS TCBs report.

```
Dispatcher - MVS TCBs

 Dispatcher Start Time and Date . . . . . . :    05:03:31.052363  10/05/2011
 Address Space Accumulated CPU Time . . . . :  0000:03:20.334982  (Not Reset)
 Address Space Accumulated SRB Time . . . . :  0000:00:26.363005  (Not Reset)
 Address Space CPU Time (Since Reset) . . . :  0000:00:00.216289
 Address Space SRB Time (Since Reset) . . . :  0000:00:00.012938

 Current number of CICS TCBs . . . . . . . . . . . . :            13
 Current CICS TCB CPU time . . . . . . . . . . . . . :  00:03:20.34198
 Current CICS TCB Private Stg below 16MB . . . . . . :         5,464K
 Current CICS TCB Private Stg below 16MB in use  . . :         5,383K
 Current CICS TCB Private Stg above 16MB . . . . . . :        71,880K
 Current CICS TCB Private Stg above 16MB in use  . . :        71,670K
 Current number of non-CICS TCBs . . . . . . . . . . :             2
 Current non-CICS TCB CPU time . . . . . . . . . . . :  00:00:26.73940
 Current non-CICS TCB Private Stg below 16MB . . . . :           80K
 Current non-CICS TCB Private Stg below 16MB in use  :           54K
 Current non-CICS TCB Private Stg above 16MB . . . . :         3,928K
 Current non-CICS TCB Private Stg above 16MB in use  :         3,885K
```

TCB Address	TCB Name	CICS TCB	Current TCB <--- TCB CPU Time --->		-Private Stg Below 16MB- Allocated	In Use	-Private Stg Above 16MB- Allocated	In Use	Task Number	Tran ID	Task Status
007E5E88	non-cics	No	00:00:26.73854	99.9%	80K	54K	3,648K	3,648K	None		
007E5858	JS	Yes	00:00:00.03223	0.0%	5,388K	5,351K	71,524K	71,524K	None		
007F8390	FO	Yes	00:00:00.18431	0.0%	16K	1K	96K	96K	None		
007F8160	RO	Yes	00:00:00.37259	0.1%	36K	18K	28K	28K	None		
007E5480	DFHTRTCB	Yes	00:00:00.00035	0.0%	0K	0K	4K	4K	None		
007C5E88	QR	Yes	00:03:19.69076	99.6%	16K	12K	60K	60K	987	STAT	Run
007B4688	DFHSKTSK	Yes	00:00:00.04178	0.0%	8K	0K	8K	8K	None		
007B4988	EP001	Yes	00:00:00.00005	0.0%	0K	0K	0K	0K	None		
007B9388	EP000	Yes	00:00:00.00004	0.0%	0K	0K	0K	0K	None		
007B9A88	SP	Yes	00:00:00.00009	0.0%	0K	0K	0K	0K	None		
007B9788	non-cics	No	00:00:00.00086	0.0%	0K	0K	280K	280K	None		
007C2340	SO	Yes	00:00:00.00254	0.0%	0K	0K	144K	144K	None		
007C25E8	SL	Yes	00:00:00.01047	0.0%	0K	0K	8K	8K	None		
007C2890	L8000	Yes	00:00:00.00641	0.0%	0K	0K	8K	8K	None		
007C52F8	CQ	Yes	00:00:00.00098	0.0%	0K	0K	0K	0K	None		

Figure 7-25 DFH0STAT Dispatcher MVS TCB support

Each TCB entry in the display in Figure 7-25 shows the TCB address, the TCB name, and the current TCB accumulated CPU time. Also, the storage allocation is provided. The TCB name is taken from the PRB Contents Directory Entry (CDE) for TCBs (not provided by CICS) or the CICS name in the KTCB (in the Kernel Domain).

In addition, the active task (if the TCB is running when the inquiry is issued), the mother (attaching) TCB, sister (attached by the same mother) TCB, and daughter (attached by this TCB) TCBs are shown (Figure 7-26).

```
 TCB                CICS   Mother     Sister    Daughter
 Address   TCB Name  TCB      TCB        TCB        TCB
 ______________________________________________________
 007E5E88  non-cics  No     007FF890             007E5858
 007E5858  JS        Yes    007E5E88             007F8390
 007F8390  FO        Yes    007E5858  007E5480  007F8160
 007F8160  RO        Yes    007F8390             007C5E88
 007E5480  DFHTRTCB  Yes    007E5858
 007C5E88  QR        Yes    007F8160             007B4688
 007B4688  DFHSKTSK  Yes    007C5E88  007B4988
 007B4988  EP001     Yes    007C5E88  007B9388
 007B9388  EP000     Yes    007C5E88  007B9A88
 007B9A88  SP        Yes    007C5E88  007C2340  007B9788
 007B9788  non-cics  No     007B9A88
 007C2340  SO        Yes    007C5E88  007C25E8
 007C25E8  SL        Yes    007C5E88  007C2890
 007C2890  L8000     Yes    007C5E88  007C52F8
 007C52F8  CQ        Yes    007C5E88
```

Figure 7-26 DFH0STAT MVS TCB report

By looking at the QR TCB in Figure 7-26, you see the TCB at 007C5E88. It was attached by the TCB at 007F8160 (the RO), and it attached several daughter TCBs. The daughter TCBs are listed in reverse order starting with DFHSKTSK at location 007A4688. Following the sister TCB chain, it is observed that all TCBs in the chain are all daughters of the QR TCB.

Also notice the percentage of CPU Time that the QR TCB accumulated of the address space CPU time.

7.8.8 Review

You must consider many different areas when reviewing performance. Many tools can help, but no single tool can be used alone to capture a picture of your overall performance. The steps and guidelines listed in the previous sections provide a clearer picture of the performance of your systems.

As a review, you must perform the following tasks:

- Define the problem.
- Understand the workload.
- Understand the physical configuration.
- If CPU increases, establish a baseline before implementing a new function:
  - CICS Performance data (SMF 110 records)
  - SMF 70-78, SMF 30
  - CICS trace
  - CPU-to-Dispatch ratio
  - Measure with the following three data collectors:
    - DFH0STAT
    - Shutdown statistics
    - CICS PA
- If response time increases, establish a baseline using the following information:
  - CICS Performance data (SMF 110 records)
  - SMF 70-78, SMF 30 records

Remember that capacity problems usually manifest themselves as response time problems.

8

Migration scenario

This chapter provides an end-to-end threadsafe migration scenario that uses a migration plan based on the concepts in Chapter 4, “Threadsafe tasks” on page 71. Using this plan, we converted a sample CICS DB2 application, running as quasi-reentrant (QR) under CICS Transaction Server for z/OS (CICS TS) V1.3, to threadsafe running under CICS TS V2.3. This chapter highlights each step of the migration process, including the displays of the required system and the application changes.

DB2 is an example of a user of the CICS Resource Manager Interface (RMI) that was enhanced to use the open transaction environment (OTE). In addition, you can configure the z/OS Communications Server IP CICS Sockets V1R7 and later to use OTE. Also, starting in CICS TS V3.2, the WebSphere MQ CICS adapter also uses OTE.

In this scenario, CICS TS V2.3 is the target release. The migration process is the same if we take any QR program and convert it to a threadsafe program. However, additional considerations apply when migrating the program to be an OPENAPI program in CICS TS V3. At the end of the chapter, we explain why RMI users who use OTE (CICS DB2, IP CICS Sockets, or WebSphere MQ applications) cannot perform this type of migration.

The chapter includes the following sections:

- Description of the sample application
- Migration plan

© Copyright IBM Corp. 2004, 2012. All rights reserved.

- Migration part 1
- Migration part 2
- Performance measurement
- Additional considerations for OPENAPI programs

8.1 Description of the sample application

The sample application is not realistic because it generates a large volume of CICS DB2 tasks and does not serve any useful business purpose. However, the profile of the individual tasks is similar to some of the typical CICS transactions in large DB2 applications.

The application generates a large volume of CICS DB2 transactions. It consists of a driver transaction, which asynchronously starts 10 daughter transactions. Upon completion, each daughter transaction restarts a finite number of times.

Eleven application programs correspond to the eleven CICS DB2 transactions described previously. Each program issues many EXEC SQL requests. The 10 daughter programs are similar, but not identical. In addition to the EXEC SQL requests, various EXEC CICS commands are issued, and some updates are made to shared resources.

In addition to the application programs, this scenario involves three global user exit (GLUE) programs, a PLTPI program, and a dynamic plan exit (DPE) program.

8.2 Migration plan

We followed a migration plan to migrate the application from running as quasi-reentrant in CICS TS V1.3 to running as threadsafe in CICS TS V2.3.

To begin, we decided that we did not want to simultaneously implement the following changes:

- Migrating the application from a CICS TS V1.3 region to a CICS TS V2.3 region
- Migrating the application to be fully threadsafe

At first glance, the first of the two changes might not seem relevant to a threadsafe migration, but it *is* as explained previously in this book. Therefore, we divided our migration plan (Table 8-1) into two parts to reflect the fact that most organizations migrate to threadsafe in two stages.

Table 8-1 Migration plan

Threadsafe migration plan	
Part 1: Migrating applications from CICS TS V1.3 to V2.3	
Step 1	Identifying the in-scope exits for part 1
Step 2	Converting the in-scope exits to threadsafe programs
Step 3	Addressing nonthreadsafe commands
Step 4	Confirming performance after migration to CICS Transaction Server V2.3
Part 2: Migrating applications to be fully threadsafe	
Step 1	Identifying programs in scope for part 2
Step 2	Converting user exits to be threadsafe
Step 3	Converting application programs to be threadsafe
Step 4	Addressing nonthreadsafe commands
Step 5	Making changes to the CICS Transaction Server region

8.3 Migration part 1

This part of the migration plan entails converting a quasi-reentrant DB2 application running under CICS TS V1.3 to run as largely quasi-reentrant under CICS TS V2.3, with a minimum of threadsafe-related changes. It focuses on ensuring that the sample application can run as quasi-reentrant under CICS TS V2.3 and can incur the same number of task control block (TCB) switches as it did when running under CICS TS V1.3.

Not in this book: This book does not describe how to upgrade a CICS region from release to release, and therefore, does not address this aspect of the migration.

8.3.1 Identifying the in-scope exits for part 1

For the first part of the migration, we are interested in only programs, user exits, and commands that can be started on the call path of an EXEC SQL statement. These entities can cause an increase in TCB switches in CICS TS V2 or later.

Only two exit points, XRMIIN and XRMIOUT, are started as a result of an EXEC SQL statement. Also one user-replaceable module (URM), the DPE program, can be started on the first SQL call of each unit of work.

XRMIIN and XRMIOUT

To determine whether this CICS region has programs running at the XRMIIN and XRMIOUT exit points, we ran the sample statistics program, **DFH0STAT**, which is supplied by CICS, to request a Global User Exits report (Figure 8-1).

```
Global User Exits
_________________
    Exit     Program    Entry    <------- Global Area ------->  Number   Program
    Name       Name      Name    Entry Name  Length  Use Count  of Exits  Status
  ________________________________________________________________________________
  XTSQRIN   XXXTS     XXXTS      XXXTS         64          1         1   Started
  XEIIN     XXXEI     XXXEI                     0          0         2   Started
  XEIOUT    XXXEI     XXXEI                     0          0         2   Started
  XRMIIN    XXXRMI    XXXRMI                    0          0         2   Started
  XRMIOUT   XXXRMI    XXXRMI                    0          0         2   Started
```

Figure 8-1 Global User Exits section of the DFH0STAT report

As shown in Figure 8-1, program XXXRMI is enabled at both exit points. Therefore, at least one program is in scope for part 1. Next we must determine whether it is possible for any of the other enabled exit points (XEIIN, XEIOUT, or XTSQRIN) to be started by using XXXRMI. The only accurate method to find our answer is to examine the source code.

An examination of the XXXRMI source code shows that it does *not* contain any code that causes the other exit points to be started. Therefore, the scope of this step is limited to XXXRMI.

Dynamic plan exit program

To determine whether any DPE programs exist, we ran the **DFH0STAT** utility again, this time requesting reports for DB2 connection and DB2 entries.

As shown in Figure 8-2 on page 207, a single DPE program, named PLANEXIT, is in use that can be added to the list of in-scope programs. We must also determine whether PLANEXIT calls any other programs or starts any user exits. The best method to find our answer is to examine the source code.

```
DB2 Connection
______________
  DB2 Connection Name. . . . . . . . . . . . :  DB2CON
  DB2 Group Id . . . . . . . . . . . . . . . :
  DB2 Sysid. . . . . . . . . . . . . . . . . :  D7Q2
  DB2 Release. . . . . . . . . . . . . . . . :   7.1.0
  DB2 Connection Status. . . . . . . . . . . :  CONNECTED
  DB2 Connection Error . . . . . . . . . . . :  SQLCODE
  DB2 Standby Mode . . . . . . . . . . . . . :  RECONNECT
  DB2 Pool Thread Plan Name. . . . . . . . . :
  DB2 Pool Thread Dynamic Plan Exit Name . . :  PLANEXIT
  Pool Thread Authtype . . . . . . . . . . . :  SIGNID

DB2 Entries
___________
  DB2Entry Name. . . . . . . . . . . . . . . :  MIG
  DB2Entry Static Plan Name. . . . . . . . . :
  DB2Entry Dynamic Plan Exit Name. . . . . . :  PLANEXIT
  DB2Entry Authtype. . . . . . . . . . . . . :  SIGNID
  DB2Entry Authid. . . . . . . . . . . . . . :
```

Figure 8-2 DB2 section of the DFH0STAT report

As shown in “PLANEXIT” on page 385, the PLANEXIT source code does *not* call any other programs, but it does issue an **EXEC CICS ASSIGN** command. Because we already established the list of active exit points (Figure 8-1 on page 206), we can conclude that PLANEXIT causes program XXXEI to start at the XEIIN and XEIOUT exit points.

We now know the full scope of the programs that we must address in the first part of the migration:

- User exit program XXXRMI, which is started twice (at XRMIIN and XRMIOUT) on every SQL call
- URM PLANEXIT, which is started on the first SQL call of each unit of work
- User exit program XXXEI, which is started twice (at XEIIN and XEIOUT) every time PLANEXIT runs

8.3.2 Converting the in-scope exits to threadsafe programs

After establishing the list of in-scope exit programs for part 1, we must now determine whether we can redefine these programs as threadsafe. This procedure entails identifying any instances of updates to shared resources and removing or serializing access if they exist. When completed, it is safe to redefine the programs as threadsafe.

This procedure includes the following tasks:

1. Running the DFH0STAT utility to find shared program storage
2. Running the DFH0STAT utility to find global work areas
3. Running the DFHEISUP utility to find potential shared resources
4. Examining the source code
5. Redefining programs as threadsafe

Running the DFH0STAT utility to find shared program storage

The supplied sample statistics program, **DFH0STAT**, can provide useful information about the use of shared storage. First, the System Status section shows whether reentrant programs are in read-only storage (Figure 8-3).

```
System Status
_____________
  MVS Product Name. . . . . . . :  MVS/SP7.0.4
  CICS Startup. . . . . . . . . :  INITIAL
  CICS Status . . . . . . . . . :  ACTIVE
Storage Protection. . . . . . :  INACTIVE
Transaction Isolation . . . . :  INACTIVE
Reentrant Programs. . . . . . : PROTECT
```

Figure 8-3 System Status section of the DFH0STAT report

Second, the Programs section shows where each program resides (Figure 8-4).

```
Programs
________
  Program   Data   Exec                    Program    Program
    Name     Loc   Key   Times Used  ...    Size      Location
  ____________________________________________________________
  DB2MANY    Any   USER           1          1,536     ERDSA
  DB2PROGA   Any   USER          12          1,312     ERDSA
  DB2PROG1   Any   USER          12          1,256     ERDSA
  DB2PROG2   Any   USER          12          1,256     ERDSA
  DB2PROG3   Any   USER          12          1,256     ERDSA
  DB2PROG4   Any   USER          12          1,216     ERDSA
  DB2PROG5   Any   USER          12          1,216     ERDSA
  DB2PROG6   Any   USER          12          1,216     ERDSA
  DB2PROG7   Any   USER          12          1,216     ERDSA
  DB2PROG8   Any   USER          12          1,304     ERDSA
  DB2PROG9   Any   USER          12          1,304     ERDSA
  EXITENBL   Any   USER           1            432     ERDSA
  PLANEXIT   Any   USER         121            208     ERDSA
  XXXEI      Any   USER           1            184     ERDSA
  XXXRMI     Any   USER           1            184     ERDSA
  XXXTS      Any   USER           1            104     ERDSA
```

Figure 8-4 Programs section of the DFH0STAT report

By looking at the information in Figure 8-3 on page 208 and Figure 8-4, we can conclude that all application programs and exits are reentrant and are in protected ERDSA storage. Therefore, we cannot use the program as a form of shared storage.

Running the DFH0STAT utility to find global work areas

We ran the **DFH0STAT** utility to determine which user exit programs are enabled for the application. In addition to listing the exit programs, the **DFH0STAT** utility also shows whether each one has a global work area (GWA).

The Exit Programs section of the report (Figure 8-5 on page 210) shows that both user exit programs in scope for part 1 of the migration, XXXRMI and XXXEI, have a GWA length of zero. Also the user exit programs do not share a GWA owned by another exit program, because the Entry Name column is blank for these programs, and the programs that own a GWA have a use count of 1.

```
Exit Programs
_____________
                      <---- Global Area ---->   No.
  Program    Entry     Entry           Use    of    Program
    Name      Name      Name   Length  Count  Exits  Status
  ____________________________________________________________
  DFHEDP    DLI                     0      0      0  Started
  DFHD2EX1  DSNCSQL   DSNCSQL      16      1      0  Started
  XXXEI     XXXEI                   0      0      2  Started
  XXXRMI    XXXRMI                  0      0      2  Started
  XXXTS     XXXTS     XXXTS        64      1      1  Started
```

Figure 8-5 Exit Programs section of the DFH0STAT report

Running the DFHEISUP utility to find potential shared resources

Next, we ran the load module scanner (the **DFHEISUP** utility), which is supplied by CICS, against *all* of the programs and exits, using our own modified version of the supplied threadsafe inhibitors table, DFHEIDTH.

> **Attention:** Scanning the in-scope programs alone might not be sufficient. The reason is that the commands to create or address a shared resource might not be confined to the programs that access or update it.

Figure 8-6 shows the changes that we made to the DFHEIDTH table.

```
EXTRACT EXIT GASET *
GETMAIN SHARED *
ADDRESS CWA *
LOAD SET *          LOAD SET * command added to the supplied list
```

Figure 8-6 Modified DFHEIDTH threadsafe inhibitors table

Figure 8-7 on page 211 shows the output from the **DFHEISUP** utility, which does not mention XXXRMI, XXXEI, and PLANEXIT, meaning that they do not issue any of the threadsafe inhibitor commands. The report also shows no instance in the application of the **LOAD**, **GETMAIN SHARED**, and **EXTRACT EXIT** commands. However, the application does use the common work area (CWA), whose address can be passed to other programs as a parameter.

```
Module Name       'CICSRS4.MIG.LOAD(DB2PROG4)'
Module Language   Assembler
Offset/EDF        Command
-----------------  ------------------------------------------
00001387/no-edf    ADDRESS CWA

Module Name       'CICSRS4.MIG.LOAD(DB2PROG5)'
Module Language   Assembler
Offset/EDF        Command
-----------------  ------------------------------------------
00001387/no-edf    ADDRESS CWA

Module Name       'CICSRS4.MIG.LOAD(DB2PROG6)'
Module Language   Assembler
Offset/EDF        Command
-----------------  ------------------------------------------
00001387/no-edf    ADDRESS CWA

Module Name       'CICSRS4.MIG.LOAD(DB2PROG7)'
Module Language   Assembler
Offset/EDF        Command
-----------------  ------------------------------------------
00001387/no-edf    ADDRESS CWA

Total possible commands located = 4
```

Figure 8-7 DFHEISUP detailed report using the DFHEIDTH filter table

Examining the source code

Running utilities, such as **DFHOSTAT** and **DFHEISUP**, can help determine whether a program is likely to be threadsafe, but is no substitute for a full understanding of the application.

By following the previous steps, we can make the following conclusions:

- All programs are reentrant and are in read-only storage.
- XXXRMI and XXXEI do not use GWAs.
- The application indicates no use of the **EXEC CICS SHARED GETMAIN** command.
- The application indicates no use of the **EXEC CICS EXTRACT EXIT** command.
- The application indicates no use of the **EXEC CICS LOAD** command.
- The application uses the CWA, but not necessarily in the in-scope programs.

Now we must examine the source code of the in-scope programs for evidence of CWA access and any nonstandard programming techniques that can result in access to a shared resource. By looking at the source code for XXXRMI, PLANEXIT, and XXXEI in Appendix C, “Assembler routines” on page 371, we see no evidence of this concern.

Redefining programs as threadsafe

After establishing that XXXRMI, PLANEXIT, and XXXEI are threadsafe, we can redefine them as such. Figure 8-8 shows the program definition for PLANEXIT after it is redefined as threadsafe. The same change was made to the XXXRMI and XXXEI definitions.

```
CEDA  View PROGram( PLANEXIT )
 PROGram        : PLANEXIT
 Group          : THDSAFE
 DEscription    :
 Language       :
 RELoad         : No
 RESident       : No
 USAge          : Normal
 USElpacopy     : No
 Status         : Enabled
 RSl            : 00
 CEdf           : Yes
 DAtalocation   : Any
 EXECKey        : User
 COncurrency    : Threadsafe
REMOTE ATTRIBUTES
```

Figure 8-8 CEDA VIEW PROGRAM display

Figure 8-9 shows how we can use CEMT to confirm that these programs are threadsafe.

```
I PROG(XXX*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Prog(XXXEI   ) Leng(0000000184) Ass Pro Ena Pri     Ced
    Res(001) Use(0000000001) Any Uex Ful Thr          Nat
 Prog(XXXRMI  ) Leng(0000000184) Ass Pro Ena Pri     Ced
    Res(001) Use(0000000001) Any Uex Ful Thr          Nat
 Prog(XXXTS   ) Leng(0000000104) Ass Pro Ena Pri     Ced
    Res(001) Use(0000000001) Any Uex Ful Qua          Nat
```

Figure 8-9 CEMT INQUIRE PROGRAM display

8.3.3 Addressing nonthreadsafe commands

After successfully converting each in-scope program for part 1 to be threadsafe, determine whether any of these programs issues nonthreadsafe commands. These programs are started on the SQL call path, and, therefore, are critical to performance. Any nonthreadsafe commands issued within an SQL flow cause a TCB switch from L8 to QR and back again. For information, see Chapter 2, “Overview of an open transaction environment and threadsafe applications” on page 11.

To determine which commands are issued by XXXRMI, PLANEXIT, and XXXEI, we ran the load module scanner utility, **DFHEISUP**, with the supplied nonthreadsafe command table, DFHEIDNT. Figure 8-10 shows the DFHEISUP summary report when run against XXXRMI, PLANEXIT, and XXXEI. It also shows that the number of nonthreadsafe commands in these three programs is zero.

```
CICS LOAD MODULE SCANNER UTILITY
SCAN PERFORMED ON Fri May  7 11:21:25 2004 USING TABLE RSTABLE2.3

SUMMARY LISTING OF CICSRS4.MIG.LOAD.PART1
=====================================
Module Name   Commands Found  Language

LOAD LIBRARY STATISTICS
=======================
Total modules in library                            =      3
Total modules Scanned                               =      3
Total CICS modules/tables not scanned               =      0
Total modules possibly containing requested commands =      0
```

Figure 8-10 DFHEISUP summary report using the DFHEIDNT filter table

Therefore, we can conclude that no further work is necessary to address nonthreadsafe commands in part 1 of the migration.

8.3.4 Confirming performance after migration to CICS Transaction Server V2.3

> **Important:** The results in this section are specific to the sample application and the system it was running on. The purpose of this section is to show the importance of converting user exits on the SQL call path to be threadsafe when upgrading to CICS TS V2 or later. Do not use these results as a benchmark for any other applications or systems.

To confirm that the application does not incur extra TCB switches in CICS TS V2.3 and has comparable performance with CICS TS V1.3, we used CICS Performance Analyzer (CICS PA) V1.3 to investigate the System Management Facilities (SMF) type 110 records. Figure 8-11 shows the selection criteria that we used to generate the reports. We used 5-minute intervals (the difference between SMFSTART and SMFSTOP) in each report.

```
CICSPA IN(SMFIN001),
       SMFSTART(yyyy/mm/dd,hh:mm:ss.00),
       SMFSTOP(yyyy/mm/dd,hh:mm:ss.00),
       APPLID(cicsapplid),
       LINECNT(60),
       FORMAT(':','/'),
   SUMMARY(OUTPUT(TESTSUM),
       BY(TRAN),
       SELECT(PERFORMANCE(
       INC(TRAN(DB21,DB22,DB23,DB24,DB25,
       DB26,DB27,DB28,DB29,DB2A)))),
       FIELDS(TRAN,
              TASKCNT,
              DB2REQCT(TOTAL),
              CHMODECT(TOTAL)))
```

Figure 8-11 Selection criteria for the CICS PA report

First, we measured our baseline. Figure 8-12 shows the result of running CICS PA before part 1 of the migration, when the application was running under CICS TS V1.3.

```
 V1R3M0      CICS Performance Analyzer
                Performance Summary
 ______________________________________________________
 Data from 02:44:58  5/13/2004 to 02:49:59  5/13/200


                      Total    Total
Tran       #Tasks   DB2 Reqs ChngMode

DB2A          482     482000        0
DB21          470     470000        0
DB22          479     479000        0
DB23          483     483000        0
DB24          484     484000        0
DB25          461     461000        0
DB26          481     481000        0
DB27          494     494000        0
DB28          482     482000        0
DB29          490     490000        0
```

Figure 8-12 CICS PA report showing SQL calls in CICS TS V1.3

Number of switches: The number of switches between the QR TCB and DB2 subtask thread TCBs is not captured in SMF type 110 records for CICS TS V1.3. However, we can calculate this value from the number of SQL calls by using the following formula:

```
TCB switches = (SQL calls x 2) + (sync points x 4) - (read-only sync points x 2)
```

Units of work with no DB2 updates perform a single-phase commit rather than a two-phase commit. Therefore, two switches occur during sync point instead of four.

As expected, the ChngMode field is zero for our CICS TS V1.3 transactions (see Figure 8-12 on page 215). However, we can calculate the number of TCB switches by using the formula in the “Number of switches” note. From our knowledge of the sample application, we have a read-only workload with only one sync point per task. Therefore, we can calculate the total TCB switches for each transaction by using the following modified formula:

```
TCB switches = (2 x DB2 Reqs) + (2 x number of tasks)
```

After determining our baseline, we measured application performance in CICS TS V2.3. Figure 8-13 shows the result of running the same CICS PA report after completing part 1 of the migration.

```
 V1R3M0     CICS Performance Analyzer
               Performance Summary
__________________________________________________
Data from 15:09:58  5/13/2004 to 15:14:59  5/13/2004


                          Total    Total
Tran       #Tasks   DB2 Reqs ChngMode

DB2A          498     498000   996996
DB21          499     499000   998998
DB22          500     500000   1001E3
DB23          498     498000   996996
DB24          498     498000   996996
DB25          498     498000   996996
DB26          499     499000   998998
DB27          499     499000   998998
DB28          498     498000   996996
DB29          498     498000   996996
```

Figure 8-13 CICS PA report showing TCB switches in CICS TS V2.3

To compare the numbers from CICS TS V1.3 and CICS TS V2.3, we calculated the averages across all transactions and tabulated the results (Table 8-2).

Table 8-2 CICS TS V1.3 versus CICS TS V2.3

	CICS TS 1.3	**CICS TS 2.3**
Average SQL calls per task	1000	1000
Average TCB switches per task	2002	2002
Transaction throughput	15.97 tps	16.56 tps

The numbers shown in Table 8-2 confirm that we achieved the goal of part 1 of the migration plan. The application is now running as quasi-reentrant under CICS TS V2.3, without extra TCB switches, and with a transaction throughput that is similar to CICS TS V1.3. In fact, we measured a slight improvement in throughput.

CICS PA: For information about CICS PA, see Chapter 9, “Threadsafe enablement using CICS Tools” on page 245, or the IBM Redbooks publication *CICS Performance Analyzer*, SG24-6063.

Application performance when upgrading without converting user exit programs to threadsafe

Throughout this book, we highlight the benefit of converting all user exit programs on the SQL call path to threadsafe when upgrading to CICS TS V2 or later, even if the intention is to leave application code as quasi-reentrant. For this reason, we split the migration plan into two parts in this chapter.

To further illustrate this point, we decided to measure the performance of the sample application if we upgraded to CICS TS V2.3 without converting the user exit programs on the SQL call path to threadsafe. Therefore, we redefined XXXRMI, PLANEXIT, and XXXEI as quasi-reentrant on CICS TS V2.3. We also generated the same CICS PA report that we produced with the programs defined as threadsafe. To differentiate these tasks from the tasks done in part 1 of the migration, we call this approach the *simplistic conversion* to CICS TS V2.3. Figure 8-14 shows the results.

```
 V1R3M0     CICS Performance Analyzer
               Performance Summary
__________________________________________________
Data from 16:39:58  5/13/2004 to 16:44:59  5/13/2004


                       Total    Total
Tran       #Tasks   DB2 Reqs ChngMode

DB2A          368     368000   2209E3
DB21          368     368000   2209E3
DB22          368     368000   2209E3
DB23          367     367000   2203E3
DB24          368     368000   2209E3
DB25          368     368000   2209E3
DB26          367     367000   2203E3
DB27          367     367000   2203E3
DB28          368     368000   2209E3
DB29          369     369000   2215E3
```

Figure 8-14 CICS PA report showing TCB switches after a simplistic conversion

To compare the simplistic conversion values with CICS TS V1.3 and our actual CICS TS V2.3 migration, we again calculated the averages across all transactions and added the results to our table (Table 8-3).

Table 8-3 CICS TS V1.3 versus CICS TS V2.3 actual and simplistic conversions

	CICS TS V1.3	**CICS TS V2.3 (actual conversion)**	**CICS TS V2.3 (simplistic conversion)**
Average SQL calls per task	1000	1000	1000
Average TCB switches per task	2002	2002	**6003**
Transaction throughput	15.97 tps	16.56 tps	**12.26 tps**

Table 8-3 shows that not defining all user exit programs on the SQL call path in the sample application as threadsafe significantly increases TCB switches after upgrading from CICS TS V1.3 to V2.3 (Figure 8-15).

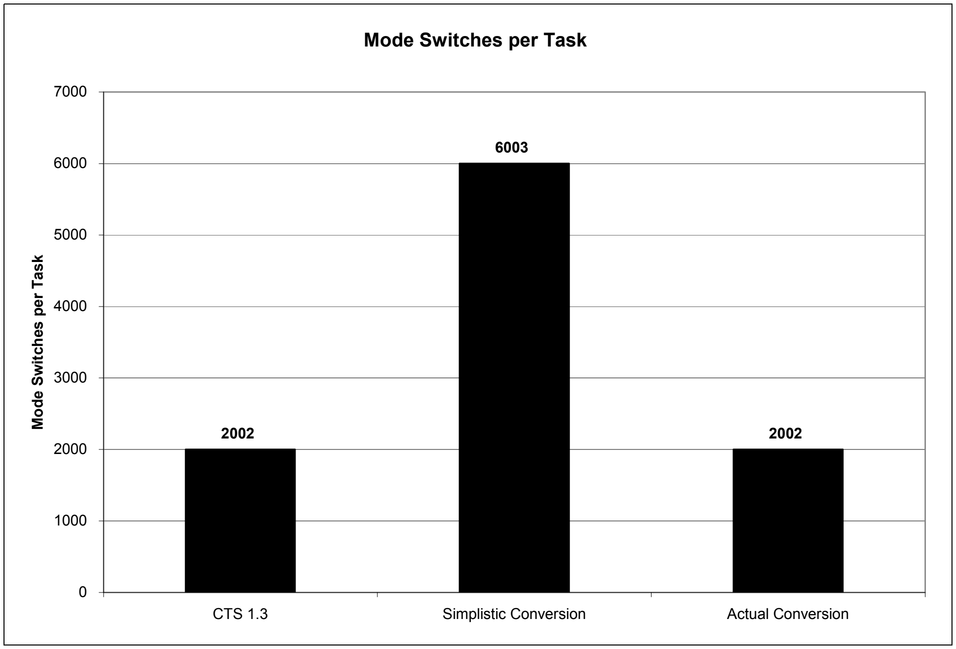


Figure 8-15 Mode switches per task

It also results in a corresponding decline in transaction throughput (Figure 8-16).

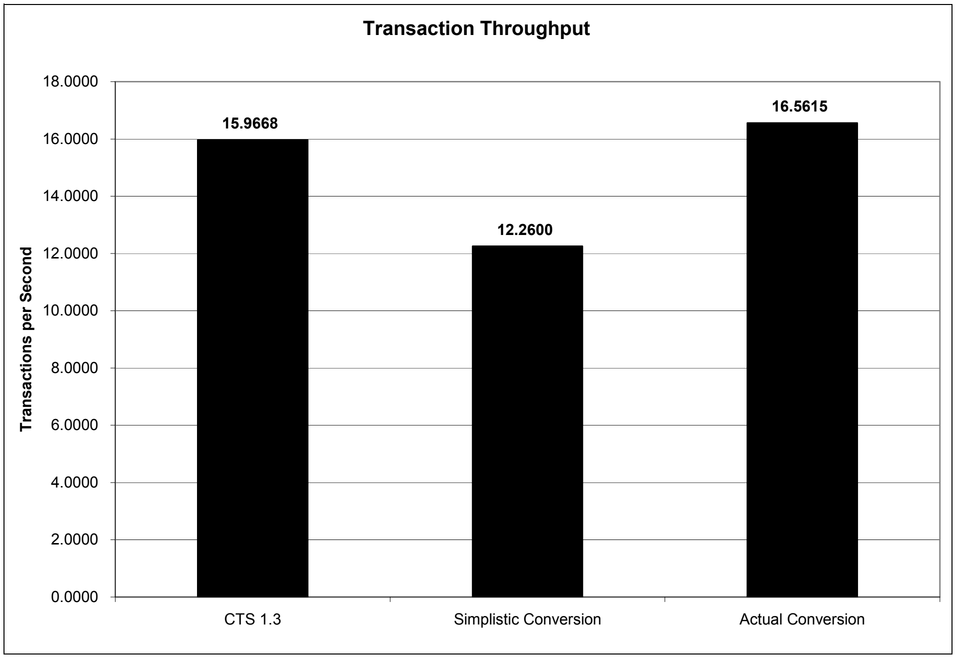


Figure 8-16 Transaction throughput

8.4 Migration part 2

Part 2 of the migration plan entails converting the application to be threadsafe.

8.4.1 Identifying programs in scope for part 2

First, we identify the application programs (including PLT programs), user exits, and URMs that are defined as quasi-reentrant. If program autoinstall is in operation, it is not sufficient to use a list of programs defined to CICS. We must start with the application load libraries concatenated within DFHRPL.

Figure 8-17 shows the list of modules in the sample application load library.

```
Name            Size       TTR      AC   AM   RM   Attributes
DB2MANY        00000558   001D46   00    31  ANY     RN RU
DB2PROGA       00000480   001E06   00    31  ANY     RN RU
DB2PROG1       00000448   001E0F   00    31  ANY     RN RU
DB2PROG2       00000448   001E18   00    31  ANY     RN RU
DB2PROG3       00000448   001E21   00    31  ANY     RN RU
DB2PROG4       00000420   001E2A   00    31  ANY     RN RU
DB2PROG5       00000420   001E33   00    31  ANY     RN RU
DB2PROG6       00000420   001E3C   00    31  ANY     RN RU
DB2PROG7       00000420   001F02   00    31  ANY     RN RU
DB2PROG8       00000478   001F0B   00    31  ANY     RN RU
DB2PROG9       00000478   001F14   00    31  ANY     RN RU
EXITENBL       000001B0   001D1A   00    31  ANY     RN RU
PLANEXIT       000000D0   001D23   00    31  ANY     RN RU
XXXEI          000000B8   001D2C   00    31  ANY     RN RU
XXXRMI         000000B8   001D35   00    31  ANY     RN RU
XXXTS          00000068   001D3E   00    31  ANY     RN RU
```

Figure 8-17 Application load library member list

Figure 8-18 shows the corresponding entries from the Programs by DSA and LPA section of a DFH0STAT report.

```
Programs by DSA and LPA
_______________________
  Program   Concurrency                 Program    Program
   Name       Status    Times Used  ...  Size      Location
  _____________________________________________________
  DB2MANY   Quasi Rent           3         1,368    ERDSA
  DB2PROGA  Quasi Rent          36         1,152    ERDSA
  DB2PROG1  Quasi Rent          36         1,096    ERDSA
  DB2PROG2  Quasi Rent          36         1,096    ERDSA
  DB2PROG3  Quasi Rent          36         1,096    ERDSA
  DB2PROG4  Quasi Rent          36         1,056    ERDSA
  DB2PROG5  Quasi Rent          36         1,056    ERDSA
  DB2PROG6  Quasi Rent          36         1,056    ERDSA
  DB2PROG7  Quasi Rent          36         1,056    ERDSA
  DB2PROG8  Quasi Rent          36         1,144    ERDSA
  DB2PROG9  Quasi Rent          36         1,144    ERDSA
  EXITENBL  Quasi Rent           1           432    ERDSA
  PLANEXIT  Thread Safe        363           208    ERDSA
  XXXEI     Thread Safe          1           184    ERDSA
  XXXRMI    Thread Safe          1           184    ERDSA
  XXXTS     Quasi Rent           1           104    ERDSA
```

Figure 8-18 Programs by DSA section of DFH0STAT report

By using the information in Figure 8-17 on page 220 and Figure 8-18, we now have a list of the application programs and exits that are defined as quasi-reentrant:

- DB2MANY
- DB2PROG1
- DB2PROG2
- DB2PROG3
- DB2PROG4
- DB2PROG5
- DB2PROG6
- DB2PROG7
- DB2PROG8
- DB2PROG9
- DB2PROGA
- EXITENBL
- XXXTS

These programs constitute the full scope of part 2 of the migration.

8.4.2 Converting user exits to be threadsafe

This procedure entails such tasks as gathering information, examining code, and other tasks.

Gathering information using DFH0STAT

To determine whether any user exits are in scope for part 2, we again look at the Exit Programs section of the **DFH0STAT** utility.

Figure 8-19 shows one GLUE in scope for migration. Program XXXTS is enabled at the XTSQRIN exit point. According to the *CICS Customization Guide*, SC34-6227, XTSQRIN is started before each user temporary storage request.

```
User Exit Programs
__________________
                      <---- Global Area ---->   No.
  Program    Entry     Entry             Use    of    Program    Program
    Name      Name      Name    Length  Count  Exits  Status     Concurrency
  ___________________________________________________________________________
  XXXEI     XXXEI                    0      0      2  Started   Thread Safe
  XXXRMI    XXXRMI                   0      0      2  Started   Thread Safe
  XXXTS     XXXTS     XXXTS         64      1      1  Started   Quasi Rent

Global User Exits
_________________
    Exit    Program    Entry    <------- Global Area ------->   Number   Program
    Name      Name      Name    Entry Name  Length  Use Count  of Exits  Status
  _____________________________________________________________________________
  XTSQRIN   XXXTS     XXXTS      XXXTS         64          1         1   Started
  XEIIN     XXXEI     XXXEI                     0          0         2   Started
  XEIOUT    XXXEI     XXXEI                     0          0         2   Started
  XRMIIN    XXXRMI    XXXRMI                    0          0         2   Started
  XRMIOUT   XXXRMI    XXXRMI                    0          0         2   Started
```

Figure 8-19 User Exits section of the DFH0STAT report

The **DFH0STAT** report also shows that XXXTS owns a GWA (Figure 8-19). By definition, a GWA is a shared resource, and we now must determine which programs access it. As owner of the GWA, the XXXTS code might not be threadsafe, but we must examine the source code to confirm whether it is.

Examining the source code

An examination of the source code for XXXTS (see “XXXTS exit” on page 389) confirms that this program is not threadsafe, because it updates a counter field in the GWA without serialization.

Gathering information by using the DFHEISUP utility

After discovering a shared resource (the XXXTS GWA), we now determine which other programs access this resource. The DFH0STAT report (Figure 8-19 on page 222) indicates that no other user exits programs share this GWA. However, we must also look for programs that address it by using the **EXEC CICS EXTRACT EXIT** command.

We ran the **DFHEISUP** utility against the entire application using a single filter table entry:

```
EXEC CICS EXTRACT EXIT GASET *
```

Figure 8-20 shows the report from the **DFHEISUP** utility.

```
CICS LOAD MODULE SCANNER UTILITY
SCAN PERFORMED ON Fri May  7 16:35:42 2004 USING TABLE RSTABLE2.3

SUMMARY LISTING OF CICSRS4.MIG.LOAD
=====================================
Module Name   Commands Found  Language

LOAD LIBRARY STATISTICS
=======================
Total modules in library                              =     16
Total modules Scanned                                 =     16
Total CICS modules/tables not scanned                 =      0
Total modules possibly containing requested commands =      0
```

*Figure 8-20 DFHEISUP summary report for EXTRACT EXIT * command*

The **DFHEISUP** utility indicates that no program issues the **EXEC CICS EXTRACT EXIT** command. Therefore, we conclude that access to this GWA is limited to the XXXTS program.

Important: We reached this conclusion because we know that the sample application always uses standard CICS interfaces to address GWAs. Applications that use other methods to address GWAs require further investigation before reaching this conclusion.

Serializing access to GWAs

After establishing that the XXXTS program is the only one to update the GWA, we ensure that this update is serialized. Figure 8-21 shows the appropriate extract from the source code.

```
GWAUPDT  EQU   *
         L     R6,GWACOUNT         GET THE COUNTER
         LA    R6,1(R6)            INCREMENT
         ST    R6,GWACOUNT           AND STORE
         B     RETURN              EXIT
```

Figure 8-21 XXXTS source code (quasi-reentrant)

As shown in Figure 8-21, the update is performed with a store (ST) instruction. We can use XPI enqueue and dequeue commands to serialize this update. However, because a single field is being updated, we replaced the store with a compare and swap (CS) routine. Figure 8-22 shows the changed code.

```
GWAUPDT  EQU   *
         L     R6,GWACOUNT         PUT ORIGINAL COUNTER IN R6
LOOP     LR    R7,R6               CREATE A COPY IN R7 TO MODIFY
         LA    R7,1(R7)            INCREMENT THE COPY IN R7
         CS    R6,R7,GWACOUNT      USE COMPARE & SWAP TO UPDATE
         BC    4,LOOP               AND REPEAT IF UNSUCCESFUL
         B     RETURN              EXIT
```

Figure 8-22 XXXTS source code (threadsafe)

After serializing access to the GWA, the XXXTS program now contains threadsafe code.

Redefining exits as threadsafe

We can now redefine all exits as threadsafe. Figure 8-23 shows the XXXTS program redefined as threadsafe.

```
OBJECT CHARACTERISTICS
  CEDA  View PROGram( XXXTS    )
   PROGram        : XXXTS
   Group          : MIGAPPL3
   DEscription    :
   Language       :
   RELoad         : No
   RESident       : No
   USAge          : Normal
   USElpacopy     : No
   Status         : Enabled
   RSl            : 00
   CEdf           : Yes
   DAtalocation   : Any
   EXECKey        : User
   COncurrency    : Threadsafe
  REMOTE ATTRIBUTES
   DYnamic        : No
 + REMOTESystem   :
```

Figure 8-23 CEDA view program display

8.4.3 Converting application programs to be threadsafe

For most applications, conversion to be threadsafe is the biggest step in a threadsafe migration. This step is also the most dependent on user application knowledge. The migration process described in this section is valid for the sample application because we know that this application uses standard CICS interfaces to create and address shared resources.

Running the DFH0STAT utility to find shared program storage

Although we performed this step in part 1 of the migration, we must repeat it now if the application changed. For example, in a real-life scenario, months might elapse between the implementation of parts 1 and 2 of the migration plan. This situation is not the case for the sample application. Therefore, the conclusion in part 1 is still valid: Program storage is not used as a shared resource within the application.

For details about the results of this step, see “Running the DFH0STAT utility to find shared program storage” on page 208.

Running the DFHEISUP utility to find potential shared resources

Although we performed this step in part 1 of the migration, we must repeat it if the application changed. For information about this step, see "Running the DFHEISUP utility to find potential shared resources" on page 210. Figure 8-7 on page 211 shows the DFHEISUP report using DFHEIDTH filter table.

Running the **DFHEISUP** utility with the DFHEIDTH threadsafe inhibitors table reveals that the programs that are listed all address the CWA (Figure 8-7 on page 211):

- DB2PROG4
- DB2PROG5
- DB2PROG6
- DB2PROG7

The **DFHEISUP** utility confirms the absence in the application of the **GETMAIN SHARED**, **EXTRACT EXIT**, and **LOAD SET** commands. Therefore, because we know that *the sample always uses standard CICS interfaces*, we can conclude that the CWA is the only remaining shared resource that we need to address.

Examining source code

We have one remaining shared resource to investigate, the CWA. As we have identified, the DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 access it.

By examining the source code (see Appendix C, "Assembler routines" on page 371), we see that each of the four programs access and update the data in the CWA by using the same sequence of instructions. Figure 8-24 shows an extract, in which all four programs take a counter from the CWA, increment it by 1 and then store it back. This code is not threadsafe, and unless it is changed, all four programs must remain defined as quasi-reentrant.

```
*****************************************************************
*                                    INCREMENT COUNTER IN CWA
         EXEC CICS ADDRESS CWA(R10)
         USING CWASTG,R10
         L     R9,CWACOUNT
         LA    R9,1(R9)
         ST    R9,CWACOUNT
*****************************************************************
```

Figure 8-24 DB2PROG4-7 source code (quasi-reentrant)

> **Serializing access:** For our test, we used the enqueue/dequeue technique to serialize access. The compare-and-swap technique is less costly than using the enqueue/dequeue technique.

Serializing access to shared resources

We have nonthreadsafe code in the application. Programs DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 all update a counter field in the CWA. To convert this code to be threadsafe, we serialize access to the CWA.

To serialize access, we must change all four programs to use an identical serialization technique. We chose the enqueue/dequeue technique for the address of the CWA. Figure 8-25 shows the appropriate extract of code after adding the **EXEC CICS ENQ** and **DEQ** commands. (Figure 8-24 on page 226 shows the code before the change.)

```
*****************************************************************
*                                    INCREMENT COUNTER IN CWA
         EXEC CICS ADDRESS CWA(R10)
         USING CWASTG,R10
         EXEC  CICS ENQ RESOURCE(CWASTG)
         L     R9,CWACOUNT           UPDATE CWA WHILE
         LA    R9,1(R9)               OWNING ENQ ON
         ST    R9,CWACOUNT             CWA ADDRESS
         EXEC  CICS DEQ RESOURCE(CWASTG)
*****************************************************************
```

Figure 8-25 DB2PROG4-7 source code (threadsafe)

The code in Figure 8-25 is threadsafe and enables programs DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 to be redefined as threadsafe.

Redefining application programs as threadsafe

After completing our analysis of all application programs, and identifying and serializing access to their shared resources, we can redefine all programs as threadsafe.

Figure 8-26 shows a CEMT display of the application programs after they are redefined as threadsafe.

```
I PROG(DB2*)
STATUS:  RESULTS - OVERTYPE TO MODIFY
 Prog(DB2MANY ) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROGA) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG1) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG2) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG3) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG4) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG5) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG6) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
 Prog(DB2PROG7) Leng(0000000000)     Pro Ena Pri     Ced
    Res(000) Use(0000000000) Any Uex Ful Thr
Prog(DB2PROG8) Leng(0000000000)     Pro Ena Pri     Ced
   Res(000) Use(0000000000) Any Uex Ful Thr
Prog(DB2PROG9) Leng(0000000000)     Pro Ena Pri     Ced
   Res(000) Use(0000000000) Any Uex Ful Thr
```

Figure 8-26 CEMT INQUIRE PROGRAM display

8.4.4 Addressing nonthreadsafe commands

After successfully converted and redefining all application programs and exits as threadsafe, the last task is to investigate the extent to which nonthreadsafe commands are used within the application. In particular, we are looking for commands that can be issued between the first and last SQL call within a CICS task. Nonthreadsafe commands that are issued before the first SQL call or after the last SQL call do not have a detrimental impact on performance.

> **OPENAPI application:** In CICS TS V3.1 if the application is an OPENAPI application, all nonthreadsafe commands, wherever they are in the program, cause two TCB switches per command.

To determine which commands are issued within the application, we ran the load module scanner utility, **DFHEISUP**, with the supplied nonthreadsafe command table, DFHEIDNT, against the whole application. Figure 8-27 shows the summary output from the **DFHEISUP** utility and highlights the programs that need further investigation. From our knowledge of the sample application, we know that each application program listed in the report is a self-contained CICS transaction. Therefore, we can examine the use of commands on a program-by-program basis.

```
SUMMARY LISTING OF CICSRS4.MIG.LOAD
=====================================
Module Name                    Commands Found  Language
'CICSRS4.MIG.LOAD(DB2MANY)'                 2  Assembler
'CICSRS4.MIG.LOAD(DB2PROGA)'                3  Assembler
'CICSRS4.MIG.LOAD(DB2PROG1)'                4  Assembler
'CICSRS4.MIG.LOAD(DB2PROG2)'                4  Assembler
'CICSRS4.MIG.LOAD(DB2PROG3)'                4  Assembler
'CICSRS4.MIG.LOAD(DB2PROG4)'                2  Assembler
'CICSRS4.MIG.LOAD(DB2PROG5)'                2  Assembler
'CICSRS4.MIG.LOAD(DB2PROG6)'                2  Assembler
'CICSRS4.MIG.LOAD(DB2PROG7)'                2  Assembler
'CICSRS4.MIG.LOAD(DB2PROG8)'                3  Assembler
'CICSRS4.MIG.LOAD(DB2PROG9)'                3  Assembler
'CICSRS4.MIG.LOAD(EXITENBL)'                2  Assembler

LOAD LIBRARY STATISTICS
=======================
Total modules in library                          =     16
Total modules Scanned                             =     16
Total CICS modules/tables not scanned             =      0
Total modules possibly containing requested commands =     12
```

Figure 8-27 DFHEISUP summary report using DFHEIDNT filter table

For the source code for the programs in the following sections, see Appendix C, “Assembler routines” on page 371.

DB2MANY program

Figure 8-28 shows the two nonthreadsafe commands, **EXEC CICS START** and **EXEC CICS SEND**, that are discovered by the **DFHEISUP** utility in the DB2MANY program.

```
Module Name         'CICSRS4.MIG.LOAD(DB2MANY)'
Module Language     Assembler
Offset/EDF          Command
-----------------   -------------------------------------------
00001650/no-edf     START TRANSID FROM LENGTH INTERVAL
00001659/no-edf     SEND TEXT FROM LENGTH FREEKB TERMINAL
```

Figure 8-28 DFHEISUP report for the DB2MANY program

The source code in “DB2MANY” on page 372 shows that the DB2MANY program contains EXEC SQL calls. However, both the **START** and **SEND** commands always run after the last call. Because neither command affects performance, no action is required.

The **ASKTIME** command, which runs between SQL calls, was made threadsafe in CICS TS V2.3.

DB2PROG1, DB2PROG2, and DB2PROG3 programs

Figure 8-29 shows the four nonthreadsafe commands that are discovered by the **DFHEISUP** utility in the DB2PROG1, DB2PROG2, and DB2PROG3 programs:

- **EXEC CICS RETRIEVE**
- **EXEC CICS POST**
- **EXEC CICS WAITCICS**
- **EXEC CICS START**

```
Module Name        'CICSRS4.MIG.LOAD(DB2PROG1)'
Module Language    Assembler
Offset/EDF         Command
-----------------  ------------------------------------------
00001435/no-edf    RETRIEVE LENGTH SET
00001444/no-edf    POST SET INTERVAL
00001453/no-edf    WAITCICS ECBLIST NUMEVENTS PURGEABILITY NAME
00001466/no-edf    START TRANSID FROM LENGTH INTERVAL

Module Name        'CICSRS4.MIG.LOAD(DB2PROG2)'
Module Language    Assembler
Offset/EDF         Command
-----------------  ------------------------------------------
00001435/no-edf    RETRIEVE LENGTH SET
00001444/no-edf    POST SET INTERVAL
00001453/no-edf    WAITCICS ECBLIST NUMEVENTS PURGEABILITY NAME
00001466/no-edf    START TRANSID FROM LENGTH INTERVAL

Module Name        'CICSRS4.MIG.LOAD(DB2PROG3)'
Module Language    Assembler
Offset/EDF         Command
-----------------  ------------------------------------------
00001435/no-edf    RETRIEVE LENGTH SET
00001444/no-edf    POST SET INTERVAL
00001453/no-edf    WAITCICS ECBLIST NUMEVENTS PURGEABILITY NAME
00001466/no-edf    START TRANSID FROM LENGTH INTERVAL
```

Figure 8-29 DFHEISUP report for programs DB2PROG1, DB2PROG2, and DB2PROG3

The source code in “DB2PROG1” on page 376 shows that these three programs contain EXEC SQL calls. However, both the **RETRIEVE** and **POST** commands will always run before the first call, and the **START** command will always be run after the last call. Therefore, these commands do not affect performance.

However, the **WAITCICS** command presents a problem, because it is nonthreadsafe and always runs between SQL calls. We have the following options:

- Leave the code unchanged and not gain the performance benefit from defining the programs as threadsafe.
- Redesign the code so that the **WAITCICS** command does not run between SQL calls.
- Change the code so that the **WAITCICS** command is no longer required.

A simple solution in this particular case makes the last option viable. We can substitute the **EXEC CICS WAIT EXTERNAL** command, which *is* threadsafe, for the **WAITCICS** command in the application. Figure 8-30 shows the code before the change.

```
         LA    R9,ECB1               WAIT UNTIL ECB POSTED
         EXEC  CICS WAITCICS                                          X
                    ECBLIST(R9)                                       X
                    NUMEVENTS(=F'1')                                  X
                    NAME(=C'APPLWAIT')                                X
                    PURGEABLE
```

Figure 8-30 Unchanged code contains a nonthreadsafe command

Figure 8-31 show the code after the change.

```
         LA    R9,ECB1               WAIT UNTIL ECB POSTED
         EXEC  CICS WAIT EXTERNAL                                     X
                    ECBLIST(R9)                                       X
                    NUMEVENTS(=F'1')                                  X
                    NAME(=C'APPLWAIT')                                X
                    PURGEABLE
```

Figure 8-31 Code changed to use a threadsafe command

DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 programs

Figure 8-32 shows the two nonthreadsafe commands discovered by the **DFHEISUP** utility in the DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 programs:

- **EXEC CICS RETRIEVE**
- **EXEC CICS START**

```
Module Name          'CICSRS4.MIG.LOAD(DB2PROG4)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------------
00001409/no-edf      RETRIEVE LENGTH SET
00001427/no-edf      START TRANSID FROM LENGTH INTERVAL

Module Name          'CICSRS4.MIG.LOAD(DB2PROG5)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------------
00001409/no-edf      RETRIEVE LENGTH SET
00001427/no-edf      START TRANSID FROM LENGTH INTERVAL

Module Name          'CICSRS4.MIG.LOAD(DB2PROG6)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------------
00001409/no-edf      RETRIEVE LENGTH SET
00001427/no-edf      START TRANSID FROM LENGTH INTERVAL

Module Name          'CICSRS4.MIG.LOAD(DB2PROG7)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------------
00001409/no-edf      RETRIEVE LENGTH SET
00001427/no-edf      START TRANSID FROM LENGTH INTERVAL
```

Figure 8-32 DFHEISUP report for the DB2PROG4, DB2PROG5, DB2PROG6, and DB2PROG7 programs

The source code in "DB2PROG4" on page 379 shows that these four programs contain EXEC SQL calls, but the **RETRIEVE** command always runs before the first call, and the **START** command always runs after the last call. Because neither command affects performance, no action is required.

The **ASKTIME** command, which runs between SQL calls, was made threadsafe in CICS TS V2.3.

DB2PROG8, DB2PROG9, and DB2PROGA programs

Figure 8-33 shows the three nonthreadsafe commands discovered by the **DFHEISUP** utility in the DB2PROG8, DB2PROG9, and DB2PROGA programs:

- **EXEC CICS RETRIEVE**
- **EXEC CICS WRITEQ TD**
- **EXEC CICS START**

```
Module Name          'CICSRS4.MIG.LOAD(DB2PROG8)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------
00001489/no-edf      RETRIEVE LENGTH SET
00001507/no-edf      WRITEQ TD QUEUE FROM LENGTH
00001516/no-edf      START TRANSID FROM LENGTH INTERVAL

Module Name          'CICSRS4.MIG.LOAD(DB2PROG9)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------
00001489/no-edf      RETRIEVE LENGTH SET
00001507/no-edf      WRITEQ TD QUEUE FROM LENGTH
00001516/no-edf      START TRANSID FROM LENGTH INTERVAL

Module Name          'CICSRS4.MIG.LOAD(DB2PROGA)'
Module Language      Assembler
Offset/EDF           Command
-----------------    ------------------------------------------
00001497/no-edf      RETRIEVE LENGTH SET
00001515/no-edf      WRITEQ TD QUEUE FROM LENGTH
00001524/no-edf      START TRANSID FROM LENGTH INTERVAL
```

Figure 8-33 DFHEISUP detailed report for programs DB2PROG8, 9 and A

The source code in "DB2PROG8" on page 382 shows that these three programs contain EXEC SQL calls, but the **RETRIEVE** command will always run before the first call, and both the **WRITEQ TD** and **START** command will always run after the last call. Because the commands do not affect performance, no action is required.

The **READQ TS** command, which runs between SQL calls, was made threadsafe in CICS TS V2.2.

EXITENBL program

Figure 8-34 shows the two nonthreadsafe commands discovered by the **DFHEISUP** utility in the EXITENBL program. Both were commands were the **EXEC CICS ENABLE** command.

```
Module Name         'CICSRS4.MIG.LOAD(EXITENBL)'
Module Language     Assembler
Offset/EDF          Command
-----------------   ------------------------------------------
00000824/no-edf     ENABLE PROGRAM EXIT START
00000833/no-edf     ENABLE PROGRAM GALENGTH EXIT START
```

Figure 8-34 DFHEISUP report for the EXITENBL program

The source code in "EXITENBL" on page 386 shows that this program does not contain EXEC SQL calls. Based on our knowledge of the application, EXITENBL is in the PLTPI and runs at CICS startup. Therefore, no action is required.

Our investigation of the EXEC CICS commands in the application is now completed. We confirmed that all nonthreadsafe commands run either before the first SQL call or after the last SQL call in every CICS program, with one exception. Moreover, we addressed the one instance of a nonthreadsafe command running between SQL calls by substituting it with a similar command that is threadsafe.

8.4.5 Making changes to the CICS Transaction Server region

We have a fully threadsafe application that does not issue nonthreadsafe commands between SQL calls. Next, we must make appropriate changes to the CICS region so that the application can use the OTE.

Table 8-4 shows the parameter values that we implemented in the CICS TS V2.3 region. The CICS TS V1.3 region values are shown for comparison.

Table 8-4 CICS system parameters pre- and post-migration

Parameter	Pre-migration (CICS TS V1.3)	Post-migration (CICS TS V2.3)
SIT		
MXT	110	110
DSA	4 M	4 M
MAXOPENTCBS		130
FORCEQR		NO

Parameter	Pre-migration (CICS TS V1.3)	Post-migration (CICS TS V2.3)
DB2CONN		
TCBLIMIT	130	130
DB2ENTRY		
THREADLIMIT	120	120
PRIORITY	LOW	LOW

As shown in Table 8-4, the only changes required in our CICS TS V2.3 region are to set **FORCEQR** to NO (the default) and to set **MAXOPENTCBS** to be the same value as **TCBLIMIT**. The **MAXOPENTCBS**, **TCBLIMIT**, and **THREADLIMIT** parameters are all higher than the MXT parameter, because we chose the MXT parameter to throttle the CICS workload if throttling is required.

Our migration of the application to threadsafe is now completed. In the last step of the migration plan, we confirm that we achieved our goal: to improve application performance.

8.5 Performance measurement

We measured the performance of the sample application after it was fully converted to a threadsafe application. We show and compare the results with the corresponding values measured when the application was quasi-reentrant.

Important: The results in this section are specific to the sample application and the system it was running on. The purpose of this section is to show that threadsafe migrations improve application performance. Do not use these results as a benchmark for any other applications or systems.

8.5.1 Reports

We used SMF type 110 records to gather the following measurements for each transaction:

- The number of SQL calls
- The number of TCB switches
- The response time
- The CPU time
- The throughput (tasks per second)

We used CICS PA V1.3 to report against the SMF data. Figure 8-35 shows the selection criteria we used to generate the reports. We used 5-minute intervals (the difference between SMFSTART and SMFSTOP) in all our reports.

```
CICSPA IN(SMFIN001),
       SMFSTART(yyyy/mm/dd,hh:mm:ss.nn),
       SMFSTOP(yyyy/mm/dd,hh:mm:ss.nn),
       APPLID(cicsapplid),
       LINECNT(60),
       FORMAT(':','/'),
   SUMMARY(OUTPUT(TESTSUM),
       BY(TRAN),
       SELECT(PERFORMANCE(
       INC(TRAN(DB21,DB22,DB23,DB24,DB25,
       DB26,DB27,DB28,DB29,DB2A)))),
       FIELDS(TRAN,
              TASKCNT,
              RESPONSE(TOTAL),
              DB2REQCT(TOTAL),
              CHMODECT(TOTAL),
              CPU(TIME(TOT)),
              QRCPU(TIME(TOT)),
              L8CPU(TIME(TOT))))
```

Figure 8-35 CICS PA report selection criteria

Totals versus averages: In the next few CICS PA performance reports, we use totals, not averages. If averages are required, divide the number of tasks by the total you are interested in.

Figure 8-36 shows the report generated for the application after part 1 of the migration was completed (quasi-reentrant application, with threadsafe exits on the SQL call path).

```
 V1R3M0                    CICS Performance Analyzer
                              Performance Summary
             __________________________________________________
             Data from 15:09:58  5/13/2004 to 15:14:59  5/13/2004


                     Total     Total    Total    Total    Total    Total
Tran       #Tasks Response DB2 Reqs ChngMode User CPU   QR CPU   L8 CPU
                      Time                       Time     Time     Time
DB2A          498  300.471   498000   996996  37.8468   8.2547  29.5920
DB21          499  301.155   499000   998998  37.0994   7.4055  29.6939
DB22          500  301.048   500000   1001E3  36.6220   7.3962  29.2258
DB23          498  300.426   498000   996996  36.7766   7.3603  29.4164
DB24          498  300.475   498000   996996  37.4116   7.8154  29.5961
DB25          498  300.414   498000   996996  37.1906   7.8236  29.3670
DB26          499  300.977   499000   998998  37.6047   7.9280  29.6767
DB27          499  301.023   499000   998998  37.4565   7.8656  29.5908
DB28          498  300.599   498000   996996  37.8501   8.2476  29.6025
DB29          498  300.456   498000   996996  37.7188   8.2497  29.4691
```

Figure 8-36 Performance report before full migration to threadsafe

Figure 8-37 shows the corresponding report after part 2 was completed (fully threadsafe application).

```
 V1R3M0                  CICS Performance Analyzer
                           Performance Summary
             ____________________________________________________
             Data from 17:19:59  5/13/2004 to 17:24:59  5/13/2004


                   Total     Total    Total    Total    Total    Total
Tran      #Tasks Response DB2 Reqs ChngMode User CPU   QR CPU   L8 CPU
                     Time                       Time     Time     Time
DB2A         813  303.524   813000     3252  49.1480    .2283  48.9198
DB21         959  304.113   959000     3836  54.2789    .2374  54.0415
DB22         967  303.960   967000     3868  53.7371    .2400  53.4971
DB23         959  304.398   959000     3836  54.1888    .2366  53.9522
DB24         951  303.937   951000     3804  53.9875    .2169  53.7706
DB25         955  303.988   955000     3820  53.9665    .2187  53.7478
DB26         951  303.723   951000     3804  54.0771    .2176  53.8595
DB27         956  303.770   956000     3824  54.1474    .2174  53.9300
DB28         813  303.917   813000     3252  49.1606    .2285  48.9321
DB29         817  303.342   817000     3268  49.0784    .2247  48.8537
```

Figure 8-37 Performance report after full migration to threadsafe

The data in Figure 8-36 on page 237 and Figure 8-37 shows that the threadsafe migration reduced the TCB switches that we intended, which in turn, resulted in substantial improvements in all key performance indicators (KPIs):

- Transaction CPU time
- Transaction response time
- Transaction throughput (tasks per second)

We used the values in the CICS PA reports to calculate our KPIs and tabulated the results as shown in Table 8-5.

Table 8-5 KPIs: Quasi-reentrant versus threadsafe

	Quasi-reentrant	**Threadsafe**	**Improvement**
Average SQL calls per task	1000	1000	
Average TCB switches per task	2002	4	
Average CPU time per task	0.0749 sec	0.0575 sec	**23%**
Average response time	0.6032 sec	0.3324 sec	**45%**
Transaction throughput	16.62 tps	30.47 tps	**83%**

To illustrate the KPIs more clearly, we created charts for each one as shown in 8.5.2, “Charts” on page 239.

8.5.2 Charts

This section provides graphs that compare details of the KPIs before and after making the applications threadsafe. Figure 8-38 illustrates the mode switches per task.

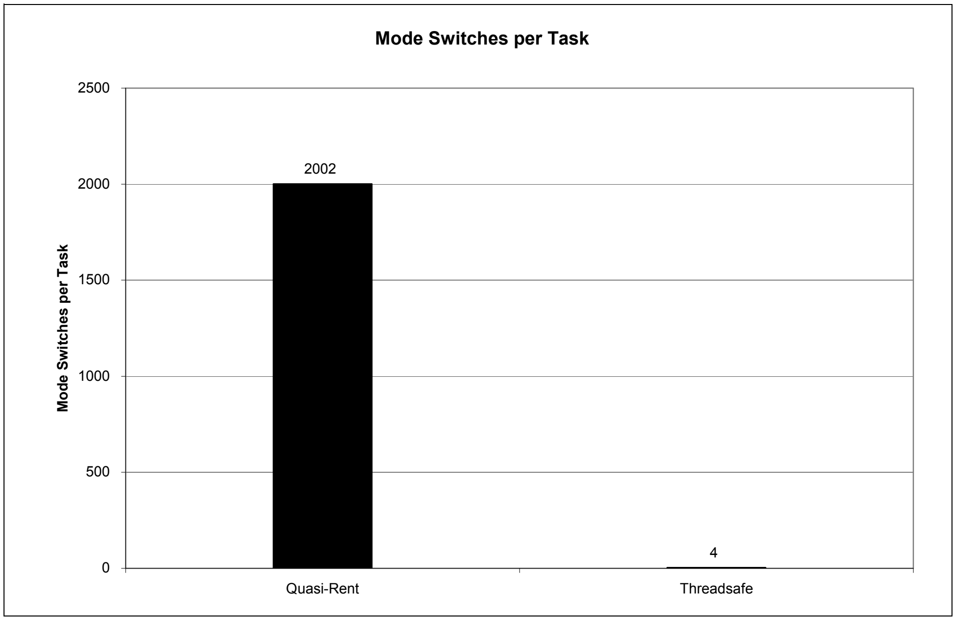


Figure 8-38 Mode switches per task CICS TS V2.3

Figure 8-39 shows average CPU per task. Notice that the average CPU is less after the application is made threadsafe.

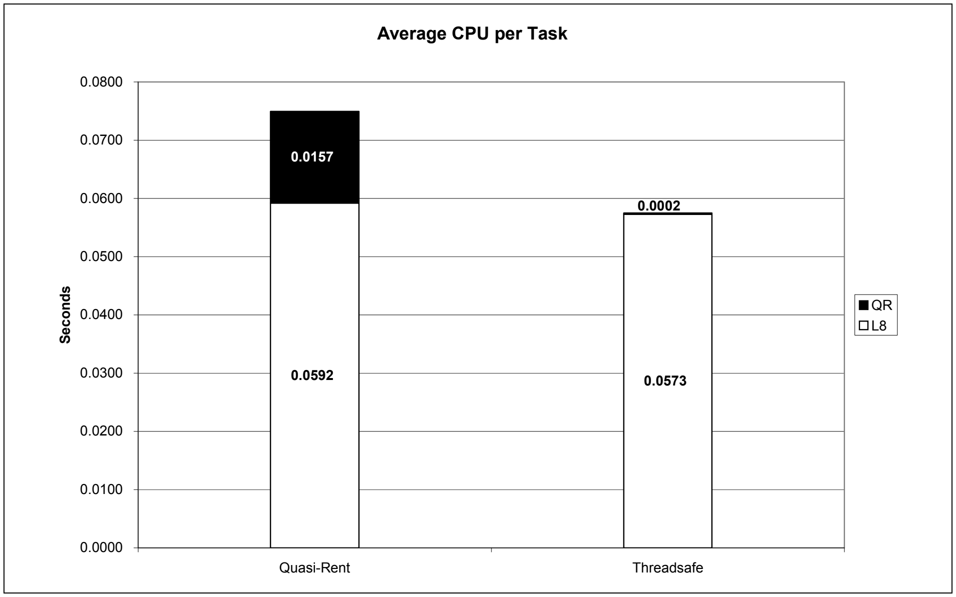


Figure 8-39 Average CPU per task

Figure 8-40 shows average response time. Notice that, after the application is made threadsafe, the average response time is much less.

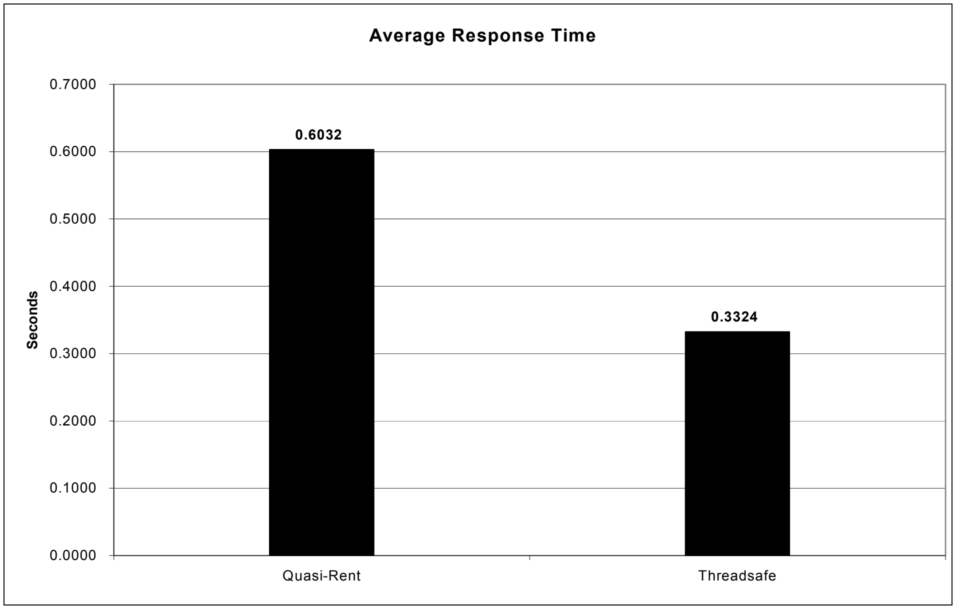


Figure 8-40   Average response time

Figure 8-41 shows transaction throughput. Notice that, after the application is made threadsafe, the result is far more transaction throughput.

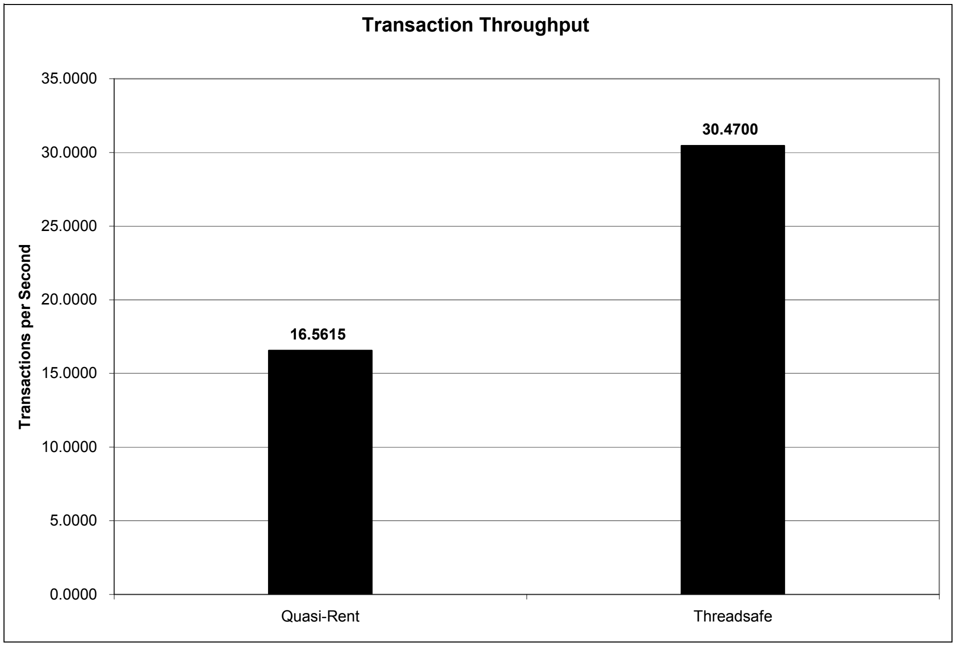


Figure 8-41 Transaction throughput

8.5.3 Conclusions

The performance measurements complete the migration of the sample application from a quasi-reentrant to threadsafe application. The reports (see 8.5.1, "Reports" on page 235) and charts (see 8.5.2, "Charts" on page 239) illustrate that we achieved our goal. By migrating the entire application to a threadsafe application, we delivered substantial improvements in each of the key performance indicators:

- 23% improvement in transaction CPU time
- 45% improvement in transaction response times
- 83% improvement in transaction throughput

8.6 Additional considerations for OPENAPI programs

CICS TS V3.1 extends OTE. By using it, applications can be defined as THREADSAFE and with an API attribute that takes the values of CICSAPI (the default) or OPENAPI. Therefore, a threadsafe application in CICS TS V2 is a threadsafe, CICSAPI program in CICS TS V3.1.

The OPENAPI attribute mandates the application to be defined also as THREADSAFE. It must be coded to threadsafe standards, because it will run on an open TCB. Therefore, an OPENAPI application is defined as THREADSAFE and OPENAPI.

THREADSAFE, OPENAPI applications differ from THREADSAFE, CICSAPI applications in that THREADSAFE, OPENAPI application *always* run on an open TCB. THREADSAFE, CICSAPI applications run on QR TCB or an open TCB, whichever is being used at the time. This way, THREADSAFE, OPENAPI applications can safely use APIs not supplied by CICS because they are guaranteed not to run on the QR TCB. Any API command (not a CICS API command) that halts the TCB halts just the open TCB and not CICS.

For APIs that are not CICS APIs to function correctly, the TCB key is important and must match the execution key. For example, an MVS GETMAIN determines the storage key required by examining the TCB key rather than the PSW execution key. For CICS API programs, the TCB key is irrelevant, because the CICS API works independently of the TCB key.

Therefore, the THREADSAFE, OPENAPI, EXECKEY(USER) programs *always* run on an L9 TCB. Also, THREADSAFE, OPENAPI, EXECKEY(CICS) applications *always* run on an L8 TCB (assuming storage protection is active).

Task-related user exits (TRUEs) always run in EXECKEY(CICS). OPENAPI TRUEs, such as the CICS DB2 TRUE or the IP CICS Sockets TRUE in z/OS Communications Server Version 1 Release 7 (if configured), always run on an L8 TCB. Therefore, a conflict exists between a EXECKEY(USER) application that is defined as OPENAPI that must run on an L9 TCB and an OPENAPI TRUE that must run on an L8 TCB. Two TCB switches occur for *every* call to an OPENAPI TRUE, L9 to L8 and L8 to L9 afterward.

Important: Keep EXECKEY(USER) CICS DB2 applications that were previously made threadsafe and defined to CICS as threadsafe as THREADSAFE, CICSAPI PROGRAMS. If storage protection is not used, do not keep these applications as THREADSAFE, CICSAPI PROGRAMS. The applications must not be defined as OPENAPI. Use this same guidance for EXECKEY(USER) IP CICS Sockets applications.

9

Threadsafe enablement using CICS Tools

This chapter highlights how the following CICS Tools can assist in enabling applications to be threadsafe:

- CICS Performance Analyzer (CICS PA) for z/OS
- CICS Interdependency Analyzer (CICS IA) for z/OS
- CICS Configuration Manager (CICS PM)

For an overview of the CICS Tools used, see Appendix A, “Overview of CICS Tools” on page 337.

The chapter includes the following sections:

- Four-step CICS Tools process
- Application case study using the CICS Tools process
- Identifying candidates and capturing a baseline
- Analyzing the program behavior
- Changing the program definitions
- Testing and benchmarking the results

© Copyright IBM Corp. 2004, 2012. All rights reserved.

9.1 Four-step CICS Tools process

The CICS Tools process for applications entails the following tasks:

1. Identifying candidates and capturing baseline by using CICS Performance Analyzer
2. Analyzing program behavior by using CICS IA
3. Changing CICS resource definitions by using CICS CM
4. Testing and benchmarking the results by using CICS PA and CICS IA

9.1.1 Identifying candidates and capturing baseline by using CICS Performance Analyzer

First, you must determine which applications and transactions are good candidates for threadsafe. To begin, start with the transactions that bring the largest benefit with the smallest amount of work. Ask the following questions to determine threadsafe candidates:

- Which transactions use large amounts of CPU because of task control block (TCB) switching?
- How many switches (change modes) occurred?
- What was the delay as the result?
- How much CPU time did they use?
- What is this costing my organization?

To answer these questions, use the following reports supplied by CICS PA, the historical database, CICS Explorer and Microsoft Excel charts and graphs, and comma-separated values (CSV) files:

- CPU Usage, Delays, Change Mode Delays
- TCB Analysis Report
- Excel Spreadsheet charts and graphs
- CICS Explorer Extracts
- Run test script with baseline data to use as input to the Transaction Profiling Report

For information about CICS PA, see “CICS Performance Analyzer for z/OS” on page 338.

CICS PA Explorer plug-in

The CICS PA Explorer plug-in, based on Eclipse, operates on top of the CICS Explorer to help you analyze CICS performance data. The CICS PA plug-in is used to analyze the performance class data of the CICS Monitoring Facility (CMF). Before the data is analyzed, it is stored in a database or formatted as CSV files using the CICS Performance Analyzer for z/OS.

By using the CICS PA plug-in, you can perform the following tasks:

- View and sort the CSV or database data in a spreadsheet viewer.
- Select a single transaction or multiple transactions for analysis.
- Perform CPU time analysis.
- Perform file analysis.
- Perform response time analysis.
- Perform storage analysis.
- Perform threadsafe analysis.
- Perform response time analysis.

CICS PA reports and extracts

CICS PA performance summary, performance list, and performance list extended reports answer the questions in 9.1.1, “Identifying candidates and capturing baseline by using CICS Performance Analyzer” on page 246. CICS PA provides extensive sample report forms (Figure 9-2) that show, for example, CPU and TCB usage, TCB delays, change mode delays.

```
 Report Forms
Command ===>                                                  Scroll ===> PAGE

 Report Forms Data Set . . : JAMESE.CICSPA.FORM

 /   Name    Type            Description                 Changed        ID
    BADCHMDS LISTX   Top 20 Worst Change TCB Modes    2010/08/30 07:42 JAMESE
    BADCPU   LISTX   Top 20 Worst CPU Times           2008/10/17 10:32 JAMESE
    BADWMQRQ LISTX   Top 20 Worst WMQ Requests        2008/10/10 00:00 CICSPA
    COMMWLST LIST    Transaction Comms Wait Analysis  2010/09/01 03:41 JAMESE
    COMMWSUM SUMMARY Transaction Comms Wait Analysis  2008/06/05 00:00 CICSPA
    CPULEXTR LIST    CPU Analysis and Extract         2011/06/01 00:00 CICSPA
    CPULST   LIST    Transaction CPU Analysis         2011/06/01 00:00 CICSPA
    CPULST1  LIST    Transaction CPU Analysis (1)     2011/06/01 00:00 CICSPA
    CPUSEXTR SUMMARY CPU Analysis and Extract         2011/06/01 00:00 CICSPA
    CPUSUM   SUMMARY Transaction CPU Analysis         2011/05/19 07:49 JAMESE
    CPUSUM1  SUMMARY Transaction CPU Analysis (1)     2008/06/05 00:00 CICSPA
    CPU3LEXT LIST    CPU Analysis and Extract (V3)    2011/06/01 00:00 CICSPA
    CPU3SEXT SUMMARY CPU Analysis and Extract (V3)    2011/06/01 00:00 CICSPA
    CPU4LEXT LIST    CPU Analysis and Extract (V4)    2011/06/01 00:00 CICSPA
    CPU4SEXT SUMMARY CPU Analysis and Extract (V4)    2011/06/01 00:00 CICSPA
    CPU8LST  LIST    Transaction CPU Analysis (Key 8) 2011/06/01 00:00 CICSPA
    CPU8SUM  SUMMARY Transaction CPU Analysis (Key 8) 2011/06/01 00:00 CICSPA
    CPU9LST  LIST    Transaction CPU Analysis (Key 9) 2011/06/01 00:00 CICSPA
    CPU9SUM  SUMMARY Transaction CPU Analysis (Key 9) 2011/06/01 00:00 CICSPA
    CSWANLST LIST    Cross-System Analysis List       2008/06/05 00:00 CICSPA
    CSWEXLST LIST    Cross-System Extract List Report 2008/06/05 00:00 CICSPA
    EXPLORE4 SUMMARY Explorer CSV for CICS TS V4      2009/01/01 00:00 CICSPA
```

Figure 9-1 CICS PA report forms

9.1.2 Analyzing program behavior by using CICS IA

After using CICS PA to determine candidate applications based on transaction performance characteristics, you can determine good candidate programs for threadsafe, based on program behavior. CICS IA has many features to help you to understand the behavior of your applications in regard to making them threadsafe. For information about CICS IA, see “CICS Interdependency Analyzer for z/OS” on page 343.

Analyzing the collected CICS IA data

After the data is collected in the CICS IA database, you can use the tools in CICS IA to analyze the data for threadsafe conformance. You can use the features of the tools to answer the following questions among others:

- What programs can be made threadsafe without program modification?
- Which programs contain only threadsafe commands?
- Which programs, and how many, have commands that need investigation to determine whether they have data integrity issues?
- Which commands need serialization wrapped around them?
- On which TCB does the command currently run?
- Which commands cause a TCB mode switch because the API is not threadsafe and must run on the QR TCB?
- Which transactions and programs use GETMAIN SHARED?
- Are transactions freeing shared storage (by using FREEMAIN)?
- Which commands cause a TCB swap?

Threadsafe Dynamic Analysis report

The Threadsafe Dynamic Analysis report provides information, by program, that can help to evaluate whether a program is a good candidate to be made threadsafe. For each program, the report provides the following information:

- All DB2, MQ, and IMS calls
- All Dynamic COBOL calls
- All EXEC CICS calls

  For CICS calls, it breaks down the calls as follows:

  - EXEC CICS commands that do not cause a TCB swap
  - EXEC CICS commands that do cause a TCB swap
  - EXEC CICS commands where insufficient information is available to determine whether they will cause a TCB swap

- All EXEC CICS calls that might be threadsafe inhibitors, such as the following commands that might cause data integrity issues and need further investigation
  - **ADDRESS CWA**
  - **EXTRACT EXIT**
  - **GETMAIN SHARED**
  - **LOAD**

You can run this report in SUMMARY or DETAIL mode. SUMMARY mode provides a count of each type of command for each program. For example, it includes the number of MQ calls, DB2 calls, threadsafe CICS calls, and inhibitor calls. The DETAIL report lists all of the commands.

You can also run the report against your current CICS TS release, which is the one you collected CICS IA data for. Alternatively, you can choose to run the report against a future release to which you are looking to upgrade. This option is useful to see how many of your CICS commands that currently cause a TCB swap will not cause the swap in future releases.

CICS IA scanners

CICS IA has the following load module scanners:

- The original load module scanner that reports about possible affinities and dependencies in a program

  This scanner also reports the program language, produces a batch report, and populates the following DB2 tables:

  - CIU_SCAN_SUMMARY
  - CIU_SCAN_DETAIL
- The additional CSECT scanner that reports on linkage and compiler attributes of all CSECTs within a program
- This scanner produces a batch report and populates the following DB2 tables:
  - CIU_CSECT_INFO
  - CIU_PROGRAM_INFO

You can use the CIU_SCAN_DETAIL table to query for possible CICS commands such as the **ADDRESS CWA** or **EXRACT EXIT** commands.

In addition, you can use the CSECT scanner to query for link-edit information, such as the link-edit date or compiler name.

In CICS IA V3.2, all scanner tables are added to the CICS IA Explorer plug-in, which you can learn more about in “New in CICS IA V3.2” on page 351.

For information about how to run the scanner job, see *CICS IA User's Guide and Reference, Version 3 Release 2*, SC34-7211.

Command Flow feature

By using the CICS IA Command Flow feature, you can view CICS, SQL, IMS, and MQ commands issued by a transaction in chronological order. It captures commands before TCB mode and after TCB mode, showing you which commands are causing TCB swaps. In the **Command Flow** view in the CICS IA Explorer plug-in, you can right-click to find commands that cause TCB swaps. You can also use the Command Flow feature to pair up GETMAIN and FREMAIN addresses.

Threadsafe queries supplied by CICS IA Explorer

The following threadsafe queries are supplied with CICS IA Explorer:

- All programs that issue a `GETMAIN SHARED` command
- All programs that issue an `ADDRESS CWA` command
- All programs that issue an `EXTRACT EXIT` command
- All programs that issue a `LOAD` command
- All programs that might have threadsafe data integrity issues
- CICS commands by TCB mode and program
- DB2 commands by TCB mode and program
- IMS commands by TCB mode and program
- MQ commands by TCB mode and program

You can also copy and modify these sample queries or create your own.

9.1.3 Changing CICS resource definitions by using CICS CM

The CICS Configuration Management tool can assist with threadsafe enablement by providing a single point of control for modifying your program definitions in a controlled and secure way. It can provide an audit of all the changes you completed and can back out these changes in a controlled and secure way.

You can use the following features:

- Search for programs within a configuration, CICS system definition data set (CSD) list, or group that has CONCURRENCY set to QUASIRENTRANT.
- Change program definitions to THREADSAFE or REQUIRED.
- Create transformation rules for mass changes to THREADSAFE or REQUIRED. You can create these rules across multiple regions and environments.
- Package change, promote, and install across development, test, and production environments.

- Maintain an audit history of CICS resource modifications.
- Back out to a previous state if required.
- Decide who can change the CONCURRENCY attribute.
- Enforce rules that certain programs must remain QUASIRENTRANT until further investigation is done.

The CICS CM Explorer plug-in provides a reporting and searching capability. For example, you can search on resource definitions including several attributes such as CONCURRENCY, API, and LANGUAGE. This tool can assist you in reporting on programs that are already made THREADSAFE, but that you now want to change to concurrency REQUIRED.

For information about CICS CM, see “CICS Configuration Manager” on page 356.

9.1.4 Testing and benchmarking the results by using CICS PA and CICS IA

In the last part of the threadsafe enablement, you must ensure that you did not caused any data integrity issues nor a performance degradation.

To ensure data integrity, perform full regression tests. CICS IA can assist you with these tests by measuring how many of the transactions, programs, and commands issued by those programs have been collected. First, you must collect CICS IA data for your regression tests before and after performing any changes. You can then compare the number of CICS, MQ, DB2, or IMS commands that you see.

You can rerun the CICS PA reports used in 9.1.1, “Identifying candidates and capturing baseline by using CICS Performance Analyzer” on page 246, against System Management Facilities (SMF) data captured after threadsafe enablement. If a performance degradation is found, run additional reports to analyze the reason for the degradation. If an increase in TCB swaps is seen for a CICS transaction, run the CICS IA Command Flow feature to understand which commands are causing these swaps.

You can also use the CICS Transaction Profiling reports provided by CICS PA. The Transaction Profiling Reports provides a comparison of two sets of Performance Summary Reports: one for the report data and one for the baseline data.

9.2 Application case study using the CICS Tools process

For this case study, we run some CICS PA performance reports against a CICS TS V4.2 development region that is running various types of work. The development region has the following applications:

- An MQ mailing application that consists of one transaction called MAIL, which is one of the sample applications that ships with the MQ product
- An IMS DLI application that consists of six transactions, DLE1 to DLE6, issuing various EXEC DLI calls
- A DB2 and remote file application that is coded so that it highlights potential data integrity issues and a high number of TCB swaps
- A fourth application that drives several EXEC CICS calls and is on the QR TCB

All of the programs in these applications are initially defined with a CONCURRENCY of QUASIRENTRANT.

Before enabling any application program to be defined as threadsafe, we must review the application code for the following two reasons:

- To maintain application data integrity.

  In releases before CICS TS V2.2, user applications and exits run on the QR TCB, which is a restricted or closed environment. CICS provided the necessary serialization to ensure that application data integrity was never compromised.

  In this environment, programs are ensured that no more than one quasi-reentrant program can run at the same time. For applications that have DB2 calls, MQ calls, IMS calls, or that are enabled as concurrency REQUIRED, two or more programs can run concurrently on different open TCBs and the QR TCB. Therefore, shared resources used by an application are serialized to prevent any application integrity problems due to more than one program accessing the same resource at the same time.
- To ensure that, after CICS moves an application to an open TCB, it remains there as long as possible.

  CICS switches the application program back to the QR TCB to run CICS API or SPI commands that are nonthreadsafe. CICS must do this task to maintain the integrity of, for example, the CSA and other control blocks that the commands use.

This scenario follows the CICS Tools process explained in 9.1, “Four-step CICS Tools process” on page 246.

9.3 Identifying candidates and capturing a baseline

To begin, we use CICS PA to analyze SMF data and to select candidate transactions for our case study. We also position ourselves for benchmarking results to verify our anticipated savings.

9.3.1 Collecting SMF data for the applications

To collect SMF data for the applications:

1. Set up the CICS SMF data capture as documented in the *IBM CICS Performance Analyzer for z/OS User's Guide*, SC34-7153-01, at:

   http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.pa.doc/pdf/cpaug-pdf.pdf

2. Use a workload simulator to run the applications.
3. Copy the SMF110 data to a safe place for future use.

> **Important:** Do not activate the CICS IA collector when collecting CICS PA data for analysis because it can impact your results.

9.3.2 Running the CICS PA reports

We run the following reports:

- Wait time analysis report
- Worst top 20 transactions causing TCB swaps
- Transaction summary report
- TCB usage and wait delay report

Adding report forms

Before running these reports, we add the necessary report forms to our personal set of report forms:

1. In the CICS PA ISPF option, choose option 3 for "Report Forms."

   Figure 9-2 on page 255 shows CICS PA Report forms. Two of the reports that are required, BADCHMDS and CPUSUM1, are already selected.

   CICS PA comes with over 170 sample report forms. We must add the form for TCB usage and delays. The WAIT time analysis report does not have a report form.

```
 File  Confirm  Samples  Options  Help
                                 Report Forms                   Row 1 to 8 of 8
 Command ===>                                                  Scroll ===> PAGE

 Report Forms Data Set . . : JAMESE.CICSPA.FORM

 /   Name     Type             Description                Changed        ID
    BADCHMDS LISTX   Top 20 Worst Change TCB Modes    2010/08/30 07:42 JAMESE
    CPUSUM   SUMMARY Transaction CPU Analysis         2011/10/12 04:15 JAMESE
    CPUSUM1  SUMMARY Transaction CPU Analysis (1)     2011/10/12 04:20 JAMESE
    CSWANLST LIST    Cross-System Analysis List       2008/06/05 00:00 CICSPA
    CSWEXLST LIST    Cross-System Extract List Report 2008/06/05 00:00 CICSPA
    EXPLORE4 SUMMARY Explorer CSV for CICS TS V4      2009/01/01 00:00 CICSPA
    PGAPLSUM SUMMARY Transactions by Application Prog 2009/01/01 00:00 CICSPA
    TESTGRP  SUMMARY Summary Report Form              2010/04/22 12:20 JAMESE
 ******************************* Bottom of data ********************************
F1=Help      F3=Exit      F5=Rfind     F6=New       F7=Backward  F8=Forward
 F10=Actions  F12=Cancel
```

Figure 9-2   Report forms in CICS PA

2. To add the TCB usage form, on the command line, type **SAMPLES**.
3. In the window that opens showing the available samples (Figure 9-3), type **F TCB4SUM** to find the required report.

```
 Sample Report For   CHARS 'TCB4SUM' found
Command ===> F TCB4SUM                    Scroll ===> PAGE

Select one or more Sample Report Forms then press EXIT.

   Name      Type     Description
   TCB4SUM   SUMMARY  CICS TCB Usage and Delays (V4)
   TCLDLSUM  SUMMARY  Tclass Delays by Tranclass Name
   TCLST1    LIST     Terminal Control Activity (1)
   TCLST2    LIST     Terminal Control Activity (2)
   TCPIPSUM  SUMMARY  Transactions by TCP/IP Service
   TCPLST    LIST     CICS Support for TCP/IP Analysis
   TCPSUM    SUMMARY  CICS Support for TCP/IP Analysis
   TCSUM2    SUMMARY  Terminal Control Activity (2)
   TDLST     LIST     Transient Data Activity
```

Figure 9-3   Selecting a report form from the samples

Running the reports

Now that we selected our report forms, we must run our four reports. In CICS PA ISPF, select option 2 for "Report Sets". As Figure 9-4 shows, three of the report sets are created for the reports required in this scenario.

```
File  Systems  Confirm  Options  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                                 Report Sets            Member WORK03 deleted
Command ===>                                                 Scroll ===> PAGE

Report Sets Data Set . . : JAMESE.CICSPA.RSET

 /    Name             Description                Changed        ID
     ALERTS   CICS PA Report Set              2010/11/09 11:37 JAMESE
     CPUSUM   CICS PA Report Set              2011/10/17 05:47 JAMESE
     TCBSUMM  CICS PA Report Set              2011/10/17 05:48 JAMESE
     WAITREP  CICS PA Report Set              2011/10/17 05:55 JAMESE
```

Figure 9-4 Report Sets panel

Creating a report

To create a report, enter **NEW** on the command line, and you are taken through an ISPF dialog that helps you create the required report.

To create a report set for the "Worst top 20 TCB swapping" transactions:

1. On the command line, type **NEW**.
2. In the New Report Set panel (Figure 9-5), enter the name of the report set, and press Enter.

```
 New Report Set        Enter required field
Command ===>

Specify the name of the new Report Set and optional model.

Name . . . worsttcb

Model  . .

 F1=Help    F3=Exit    F6=Resize F12=Cancel
```

Figure 9-5 New Report Set panel

3. In the Report Set - WORSTTCB panel (Figure 9-6), select type **Extended List**, which is the worst TCB report form, and press Enter.

```
File  Systems  Confirm  Options  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT                     Report Set - WORSTTCB                    Row 1 of 44
Command ===>                                                 Scroll ===> PAGE

Description . . . CICS PA Report Set

Enter "/" to select action.

             ** Reports **                          Active
     -       Options                                  No
                  Global                              No
     -       Selection Criteria                       No
                  Performance                         No
                  Exception                           No
     -       Performance Reports                      No
                  List                                No
                  List Extended                       No
                  Summary                             No
                  Totals                              No
                  Wait Analysis                       No
                  Transaction Profiling               No
                  Cross-System Work                   No
                  Transaction Group                   No
                  BTS                                 No
                  Workload Activity                   No
                  Transaction Tracking List           No
                  Transaction Tracking Summary        No
     -       Exception Reports                        No
                  List                                No
                  Summary                             No
 F1=Help      F3=Exit      F7=Backward  F8=Forward  F10=Actions  F12=Cancel
```

Figure 9-6 Creating the WORSTTCB report set

4. In the next panel (Figure 9-7 on page 258), enter the APPLID and the form of the Report Format. In this example, we run the report against our CICS region with APPLID `IYDZEJ07`. Press PF4 to list the report sets available for type LISTX, and, for this example, select **BADCHMDS**.

```
Command ===>

System Selection:                       Report Output:
 APPLID . . IYDZEJ07  +                  DDname . . . . . . . . . . LSTX0001
 Image  . . MV2F      +                  Print Lines per Page . .      (1-255)
 Group  . .           +

Report Format:
 Form . . . BADCHMDS  +
 Title  . . Worst 2o transactions casuing TCB swaps.


Selection Criteria:
 s  Performance
```

Figure 9-7 Defining the run options for the WORSTTCB report

5. In the WORSTTCB - Performance Select Statement panel (Figure 9-8), exclude CICS transactions by selecting **Performance Criteria**. In this panel, the line `EXC TRAN C*` is highlighted to show how to exclude all CICS transactions.

```
File  Edit  Lists  Options  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                    WORSTTCB - Performance Select Statement Row 1 of 9 More: >
Command ===>                                                  Scroll ===> PAGE

         Active  sssssssssssssss Report Interval ssssssssssssssss
   Inc   Start   ssssssss From sssssssss  sssssssss To ssssssssss
   Exc   Stop    YYYY/MM/DD  HH:MM:SS.TH  YYYY/MM/DD  HH:MM:SS.TH


ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss

   Inc  Field            sss Value or Range sss
/  Exc  Name +    Type    Value/From   To          List +
   EXC  TRAN              C*



****************************** Bottom of data ********************************
```

*Figure 9-8 Excluding all transactions starting with C**

6. Press PF3 to return to the Report Set menu.

7. On the Report Set menu panel (Figure 9-9), enter **run** against the List Extended (LISTX).

```
 File  Systems  Confirm  Options  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT                       Report Set - WORSTTCB                    Row 1 of 44
Command ===>                                                   Scroll ===> PAGE

Description . . . CICS PA Report Set

Enter "/" to select action.

             ** Reports **                         Active
     -       Options                                 Yes
                  Global                             Yes
     -       Selection Criteria                      No
                  Performance                        No
                  Exception                          No
     -       Performance Reports                     Yes
                  List                               No
             run  List Extended                      Yes
                  Summary                            No
                  Totals                             No
                  Wait Analysis                      No
                  Transaction Profiling              No
                  Cross-System Work                  No
                  Transaction Group                  No
                  BTS                                No
                  Workload Activity                  No
                  Transaction Tracking List          No
                  Transaction Tracking Summary       No
     -       Exception Reports                       No
                  List                               No
                  Summary                            No
 F1=Help      F3=Exit      F7=Backward  F8=Forward  F10=Actions  F12=Cancel
```

Figure 9-9 Running the report

8. In the Run Report Set WORSTTCB panel (Figure 9-10), press Enter to view the job control language (JCL). Then type **sub** on a command line to submit this job.

```
File  Systems  Options  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                            Run Report Set WORSTTCB
Command ===>

Specify run options then press Enter to continue submit.

System Selection:
 CICS APPLID . . IYDZEJ07  +  Image . . MV2F      +  Group . .           +
 DB2 SSID  . . .       +      Image . .           +  Group . .           +
 MQ SSID . . . .       +      Image . .           +  Group . .           +
 Logger  . . . .           +  Image . .           +  Group . .           +

    Override System Selections specified in Report Set

                                        sssss Report Interval ssssss
Missing SMF Files Option:                    YYYY/MM/DD  HH:MM:SS.TH
1  1. Issue error message               From
   2. Leave DSN unresolved in JCL       To
   3. Disregard offending reports

Enter "/" to select option
/  Edit JCL before submit

 F1=Help      F3=Exit      F4=Prompt    F7=Backward  F8=Forward  F10=Actions
F12=Cancel
```

Figure 9-10   Submitting the CICS PA report

9. Repeat this process for each of the four required reports.

9.3.3 Output from the CICS PA reports

Next we view and analyze the output from the reports that we ran against region REDDVA42.

The first two reports, CICS Transaction Summary and TCB Usage and Delays, are used to set a base line for our performance criteria. After we make some changes, we can use these reports to compare results.

Figure 9-11 show the CICS Transaction Summary report.

```
 V3R2M0                                                  CICS Performance Analyzer
                                                            Performance Summary
                                            __________________________________________________
SUMM0001 Printed at 12:30:13 11/27/2011     Data from 11:31:47 11/27/2011 to 12:27:01 11/27/2011
CICS TCB CPU Analysis - Summary
```

Tran	#Tasks	Avg Response Time	Max Response Time	Avg Dispatch Time	Avg User CPU Time	Avg Suspend Time	Avg DispWait Time	Avg QR CPU Time	Avg MS CPU Time	Avg KY8 CPU Time	Avg KY9 CPU Time	Avg RO CPU Time
/FOR	4992	.1122	.9129	.0025	.0003	.1097	.0002	.0003	.0000	.0000	.0000	.0000
DE1	584	.1868	1.4725	.0044	.0009	.1824	.0388	.0009	.0000	.0000	.0000	.0000
DE20	**584**	**.2279**	**.7652**	**.0041**	**.0014**	**.2238**	**.0581**	**.0014**	**.0000**	**.0000**	**.0000**	**.0000**
DLE1	**6725**	**.2074**	**8.8308**	**.0219**	**.0027**	**.1855**	**.0870**	**.0007**	**.0000**	**.0019**	**.0000**	**.0000**
DLE2	**6726**	**.1481**	**8.7425**	**.0117**	**.0012**	**.1364**	**.0410**	**.0005**	**.0000**	**.0007**	**.0000**	**.0000**
DLE3	**6726**	**.1505**	**8.7287**	**.0134**	**.0019**	**.1371**	**.0477**	**.0005**	**.0000**	**.0013**	**.0000**	**.0000**
DLE4	**6724**	**.1715**	**8.8726**	**.0296**	**.0015**	**.1419**	**.0539**	**.0005**	**.0000**	**.0010**	**.0000**	**.0000**
DLE5	**6724**	**.1833**	**8.9314**	**.0314**	**.0019**	**.1519**	**.0640**	**.0006**	**.0000**	**.0013**	**.0000**	**.0000**
HR1	199	.1933	1.4525	.0049	.0009	.1884	.0407	.0008	.0001	.0000	.0000	.0000
HR2	198	.1476	.4309	.0037	.0008	.1438	.0197	.0008	.0000	.0000	.0000	.0000
IT1	188	.1209	1.2516	.0102	.0006	.1106	.0024	.0006	.0000	.0000	.0000	.0000
IT2	453	.2051	.9069	.0057	.0020	.1994	.0701	.0020	.0000	.0000	.0000	.0000
IT8	587	.3745	6.0793	.0070	.0024	.3675	.1427	.0023	.0001	.0000	.0000	.0000
MAIL	**2402**	**4.6797**	**11.3737**	**.0238**	**.0046**	**4.6560**	**.5643**	**.0028**	**.0000**	**.0017**	**.0000**	**.0000**
OE1	333	.1180	1.1376	.0043	.0008	.1138	.0058	.0007	.0001	.0000	.0000	.0000
OE2	333	.2175	.9151	.0067	.0011	.2107	.0496	.0011	.0000	.0000	.0000	.0000
OE4	331	.1186	1.1951	.0051	.0014	.1135	.0072	.0014	.0001	.0000	.0000	.0000
OE5	319	.6312	1.3584	.0101	.0037	.6212	.2483	.0037	.0000	.0000	.0000	.0000
PA2	555	.1130	.4703	.0025	.0003	.1105	.0007	.0003	.0000	.0000	.0000	.0000
PS2	735	.1149	.8537	.0044	.0015	.1105	.0049	.0014	.0000	.0000	.0000	.0000
PS3	667	.1209	1.7920	.0057	.0022	.1152	.0074	.0021	.0001	.0000	.0000	.0000
SC2	200	.5388	1.4708	.0085	.0029	.5302	.2033	.0028	.0001	.0000	.0000	.0000
SC6	746	.2296	.8648	.0047	.0010	.2249	.0562	.0010	.0000	.0000	.0000	.0000
TS1	294	.1090	.4252	.0027	.0004	.1064	.0007	.0003	.0000	.0000	.0000	.0000
TXM0	**3**	**.0123**	**.0149**	**.0120**	**.0024**	**.0003**	**.0003**	**.0006**	**.0018**	**.0000**	**.0000**	**.0018**
TXM1	**602**	**1.2916**	**20.5296**	**.0364**	**.0178**	**1.2551**	**.8727**	**.0126**	**.0000**	**.0051**	**.0000**	**.0000**
TXM2	**594**	**1.2754**	**2.3180**	**.0332**	**.0175**	**1.2423**	**.8912**	**.0123**	**.0000**	**.0051**	**.0000**	**.0000**
TXM3	**594**	**1.2594**	**2.1294**	**.0363**	**.0175**	**1.2231**	**.8735**	**.0123**	**.0000**	**.0051**	**.0000**	**.0000**
TXM4	**590**	**1.2736**	**2.2836**	**.0357**	**.0175**	**1.2379**	**.8887**	**.0123**	**.0000**	**.0051**	**.0000**	**.0000**
TXM5	**587**	**1.2655**	**2.0654**	**.0334**	**.0175**	**1.2321**	**.8799**	**.0123**	**.0000**	**.0051**	**.0000**	**.0000**
Total	51295	.4465	20.5296	.0183	.0027	.4282	.1233	.0015	.0000	.0012	.0000	.0000

Figure 9-11 TCB CPU Summary report

Figure 9-12 shows the TCB Usage and Delays report.

V3R2M0 CICS Performance Analyzer
Performance Summary

SUMM0001 Printed at 12:33:44 11/27/2011 Data from 11:31:47 11/27/2011 to 12:27:01 11/27/2011
CICS TCB Usage and Delays

Tran	#Tasks	Avg TCBAtach Count	Avg DSTCBHWM Count	Max DSTCBHWM Count	Avg DSCHMDLY Count	Max DSCHMDLY Count	Avg MaxJTDly Count	Avg MaxOTDly Count	Avg MAXSTDLY Count	Avg MAXTTDLY Count	Avg MAXXTDLY Count	Avg KY8 Disp Count
/FOR	4992	0	0	0	0	6	0	0	0	0	0	0
DE1	584	0	0	0	0	42	0	0	0	0	0	0
DE20	584	0	0	0	0	42	0	0	0	0	0	0
DLE1	6725	0	1	1	37	40	0	0	0	0	0	19
DLE2	6726	0	1	1	18	20	0	0	0	0	0	9
DLE3	6726	0	1	1	30	32	0	0	0	0	0	15
DLE4	6724	0	1	1	26	28	0	0	0	0	0	13
DLE5	6724	0	1	1	38	46	0	0	0	0	0	19
HR1	199	0	0	0	0	26	0	0	0	0	0	0
HR2	198	0	0	0	0	6	0	0	0	0	0	0
IT1	188	0	0	0	0	6	0	0	0	0	0	0
IT2	453	0	0	0	0	24	0	0	0	0	0	0
IT8	587	0	0	0	0	46	0	0	0	0	0	0
MAIL	2402	0	1	1	38	1152	0	0	0	0	0	19
OE1	333	0	0	0	0	26	0	0	0	0	0	0
OE2	333	0	0	0	0	6	0	0	0	0	0	0
OE4	331	0	0	0	0	26	0	0	0	0	0	0
OE5	319	0	0	0	0	6	0	0	0	0	0	0
PA2	555	0	0	0	0	6	0	0	0	0	0	0
PS2	735	0	0	0	0	26	0	0	0	0	0	0
PS3	667	0	0	0	0	64	0	0	0	0	0	0
SC2	200	0	0	0	0	26	0	0	0	0	0	0
SC6	746	0	0	0	0	26	0	0	0	0	0	0
TS1	294	0	0	0	0	6	0	0	0	0	0	0
TXM0	3	0	0	0	2	2	0	0	0	0	0	0
TXM1	602	0	0	1	204	330	0	0	0	0	0	102
TXM2	594	0	1	1	204	204	0	0	0	0	0	102
TXM3	594	0	1	1	204	204	0	0	0	0	0	102
TXM4	590	0	1	1	204	204	0	0	0	0	0	102
TXM5	587	0	1	1	204	204	0	0	0	0	0	102
Total	51295	0	0	1	33	1152	0	0	0	0	0	16

Figure 9-12 TCB Usage and Delays report

We use the Worst 20 transactions causing TCB swaps report (Figure 9-13 on page 264) and the Wait Time Analysis report (Figure 9-14 on page 265) to find candidate programs to configure with the THREADSAFE or REQUIRED concurrency attribute.

The report in Figure 9-13 shows that the MAIL transaction and the TXM* transactions are the worst TCB swapping candidates. The MAIL transaction is the MQ application. From the TCB CPU Summary (Figure 9-11 on page 262), we see that it spends some of its time on the Key 8 TCB. This behavior is expected because all EXEC MQ calls from CICS TS V3.2 and later run on the L8 open TCB.

```
 V3R2M0                                           CICS Performance Analyzer
                                                  Performance List Extended
                                            ________________________________________________
 LSTX0001 Printed at 12:34:23 11/27/2011 Data from 11:31:47 11/27/2011 to 12:27:01 11/27/2011
 Worst 20 transactions casuing TCB swaps.

Tran DSCHMDLY Userid TaskNo Stop         Response Dispatch Dispatch User CPU  Suspend  Suspend DispWait QRModDly DSCHMDL
     Count                  Time           Time     Time     Count    Time     Time     Count    Time     Time     Time
MAIL       46 CICSUSR   185 12:21:29.364  4.5030    .1847       59    .0093   4.3183       59    .0713    .0343    .0478
MAIL       46 CICSUSR   222 12:21:30.249  5.3723    .0507       58    .0071   5.3216       58    .0170    .0121    .0097
TXM1      204 CICSUSR   269 12:21:32.263   .0550    .0208      305    .0189    .0342      305    .0026    .0011    .0023
TXM1      204 CICSUSR   275 12:21:33.260   .0558    .0199      304    .0185    .0359      304    .0027    .0017    .0021
TXM2      204 CICSUSR   277 12:21:33.614   .0542    .0196      305    .0184    .0345      305    .0048    .0032    .0043
TXM2      204 CICSUSR   286 12:21:34.372   .0599    .0220      305    .0183    .0379      305    .0085    .0047    .0036
TXM3      204 CICSUSR   274 12:21:32.968   .0515    .0204      304    .0183    .0311      304    .0027    .0017    .0022
TXM3      204 CICSUSR   289 12:21:34.724   .0497    .0191      305    .0179    .0307      305    .0024    .0012    .0021
TXM4      204 CICSUSR   276 12:21:33.312   .0465    .0185      305    .0175    .0280      305    .0019    .0009    .0014
TXM4      204 CICSUSR   285 12:21:34.309   .0468    .0184      305    .0175    .0284      305    .0025    .0012    .0022
TXM5      204 CICSUSR   368 12:21:37.841   .2619    .0279      305    .0165    .2340      305    .1193    .0740    .0991
TXM5      204 CICSUSR   373 12:21:37.866   .1616    .0225      303    .0174    .1391      303    .0593    .0221    .0521
```

Figure 9-13 Worst 20 TCB swaps report

Similarly the TXM* transactions make up our DB2 application. By looking at the report in Figure 9-11 on page 262 again, we see that some of the work is on the Key 8 TCB. Again this behavior is expected because all EXEC DB2 calls now run on the L8 open TCB.

The WAIT time Analysis report (Figure 9-14 on page 265) shows that both the MAIL transaction and the TXM1 transaction spend a percentage of their time waiting for the QR TCB. This report also shows that the MAIL transaction spends a large amount of time waiting for terminal input. The transaction is a terminal-driven transaction and uses BMS maps to facilitate the transaction.

The TXM* application also spends a lot of time waiting for the MRO link, indicating that the files are remote.

The IMS application consists of DLE* transactions. From the various reports, we see that the IMS DLI calls are still running on both the QR TCB and L* TCB, indicating that we are running on IMS Version 12.

```
 Tran=TXM1
   Summary Data                                              -------- Time ---------    ------ Count ------
                                                                   Total     Average       Total    Average
     # Tasks                                                                                 602
     Response Time                                              777.5223      1.2916
     Dispatch Time                                               21.9301      0.0364      184421      306.3
     CPU Time                                                    10.7076      0.0178      184421      306.3
     Suspend Wait Time                                          755.5867      1.2551      184421      306.3
     Dispatch Wait Time                                         525.3696      0.8727      183819      305.3
       QR TCB Redispatch Wait Time                              497.7946      0.8269      122399      203.3
     Resource Manager Interface (RMI) elapsed time                8.7792      0.0146       62510      103.8
     Resource Manager Interface (RMI) suspend time                0.0467      0.0001         304        0.5

   Suspend Detail                                            ------------------- Suspend Time -------------
                                                                  Total     Average   %age  Graph
    IRIOWTT  MRO link wait time                                544.9295      0.9052  72.1% |**************
    DSCHMDLY Redispatch wait time caused by change-TCB mode    142.6871      0.2370  18.9% |***
    DSPDELAY First dispatch wait time                           46.8972      0.0779   6.2% |*
    N/A      Other Wait Time                                    20.5402      0.0341   2.7% |
    LMDELAY  Lock Manager (LM) wait time                         0.5325      0.0009   0.1% |
    GVUPWAIT Give up control wait time                           0.0001      0.0000   0.0% |
    TCIOWTT  Terminal wait for input time                        0.0001      0.0000   0.0% |

Tran=MAIL
   Summary Data                                              -------- Time ---------    ------ Count ------
                                                                   Total     Average       Total    Average
     # Tasks                                                                                2402
     Response Time                                            11240.7511      4.6797
     Dispatch Time                                               57.0863      0.0238      124943       52.0
     CPU Time                                                    10.9956      0.0046      124943       52.0
     Suspend Wait Time                                        11183.6607      4.6560      124943       52.0
     Dispatch Wait Time                                        1355.3384      0.5643      122541       51.0
       QR TCB Redispatch Wait Time                             1314.1879      0.5471       75758       31.5
     Resource Manager Interface (RMI) elapsed time               31.0560      0.0129       50809       21.2
     Resource Manager Interface (RMI) suspend time                0.2562      0.0001         147        0.1

   Suspend Detail                                            ------------------- Suspend Time -------------
                                                                   Total     Average   %age  Graph
     TCIOWTT  Terminal wait for input time                    10521.2620      4.3802  94.1% |**************
     DSPDELAY First dispatch wait time                          240.7927      0.1002   2.2% |
     DSCHMDLY Redispatch wait time caused by change-TCB mode    136.7795      0.0569   1.2% |
     N/A      Other Wait Time                                   134.5612      0.0560   1.2% |
     LMDELAY  Lock Manager (LM) wait time                       131.1596      0.0546   1.2% |
     GVUPWAIT Give up control wait time                          15.9092      0.0066   0.1% |
     TSIOWTT  VSAM TS I/O wait time                               3.1965      0.0013   0.0% |
     ENQDELAY Local Enqueue wait time                             0.0000      0.0000   0.0% |
```

Figure 9-14 Wait Time Analysis for the MAIL and TXM1 transactions

Now that we identified these transactions as possible candidates, we use CICS IA to investigate further.

9.3.4 Loading the data into a historical database and DB2

An alternative to running the reports is to extract the performance and statistics data into CICS PA historical databases (HDBs) and populate the CICS PA Explorer database. The CICS PA Explorer plug-in uses JDBC Type 4 to query the database.

Query your data:

1. Open the CICS PA perspective.
2. Open the **Records** view.
3. Select to view the **Performance Summary**.
4. Optional: Select your CICS region by APPLID.
5. In the Performance Summary tree (Figure 9-15), right-click and select **Performance history** → **Threadsafe**.

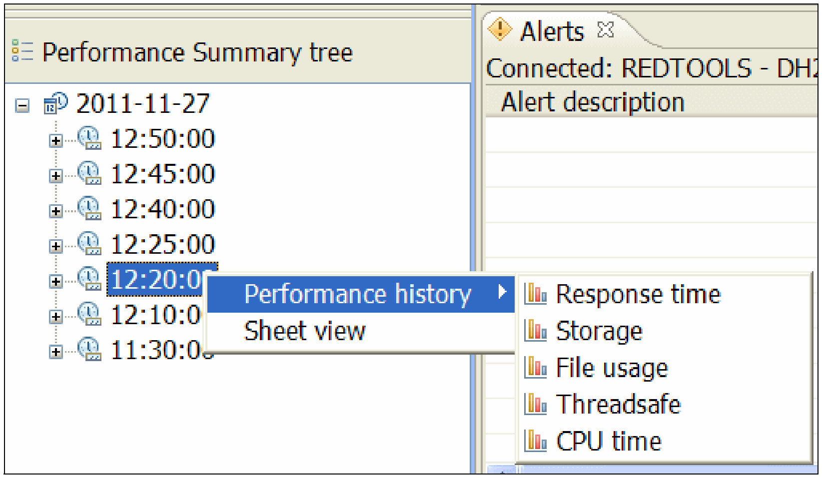


Figure 9-15 Performance Summary tree

We now see a chart from which you can choose the transactions with the highest number of change modes and that use the most CPU. In this scenario, we have only one transaction, TXM1, with 406 TCB change modes. A real-world environment has many transactions with a potential for savings.

The CICS PA Explorer - Threadsafe Chart in Figure 9-16 on page 267 shows that TXM1 is nonthreadsafe. This chart also shows the following information:

– The TXM* transactions ran 380 times on average.
– The MAIL transaction ran 1560 times.
– The TXM* transactions performed 204 TCB mode switches.
– The MAIL transaction performed 38 TCB mode switches.
– Transactions with the highest number of TCB mode switches are listed on the left.

Color guide: The colors in the bar chart indicate CPU on the QR TCB (light blue) and the L8 TCB (dark blue). You can hover over the bar to see the respective times for each TCB.

6. From this chart, double-click the bar for the **TXM1** transaction to view the Transaction Detail Analysis.

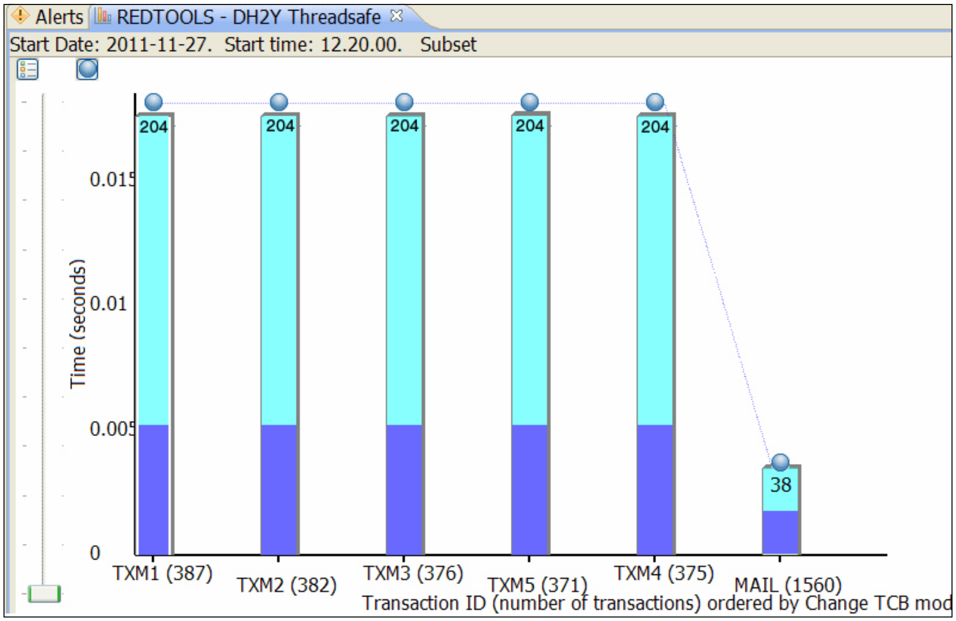


Figure 9-16 CICS PA Explorer: Threadsafe view

This view shows the CPU time breakdown by TCB type, as shown in Figure 9-17 on page 268.

The Transaction Detail for TXM1 shows that 387 transactions were run and 204 TCB mode switches occurred on average.

7. Click a color in the pie chart. The corresponding TCB time is highlighted.

   To access additional detail reports, double-click one of the report pies shown at the top of the view.

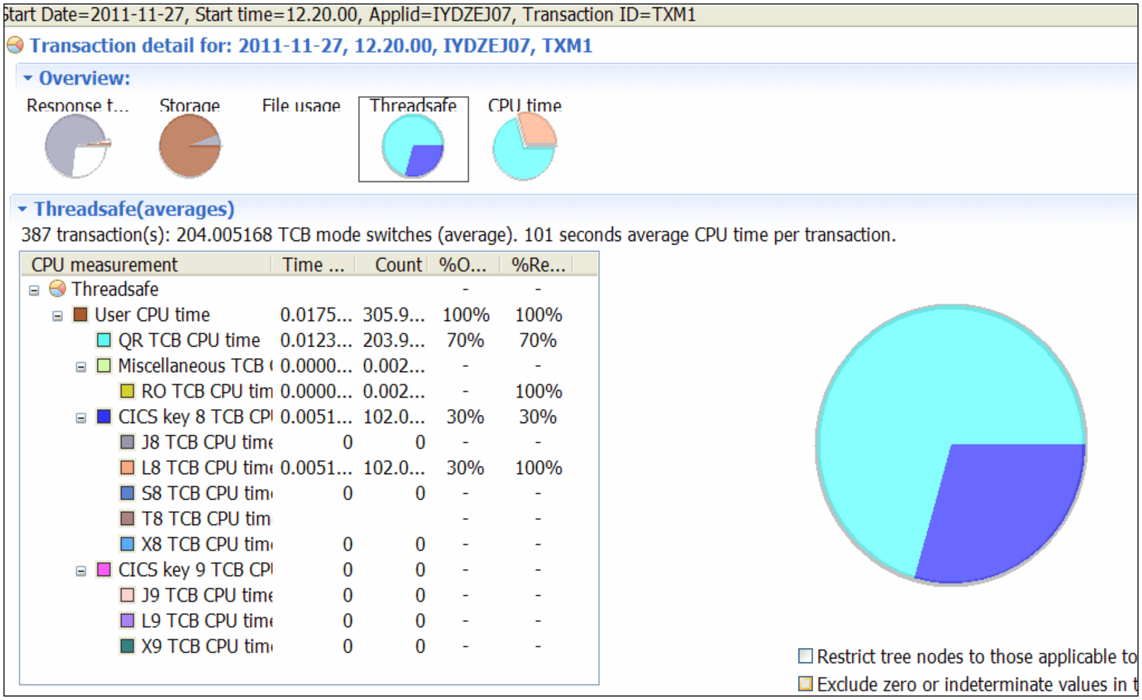


Figure 9-17 CICS PA Explorer: Threadsafe Detail View TXM1 is not Threadsafe

9.4 Analyzing the program behavior

We now use CICS IA to collect interdependency and command flow data for the MAIL transaction and the TXM* transactions.

9.4.1 Collecting interdependency data

To capture the required data for threadsafe analysis, we must first set the correct options in CICS IA. The collection of dependency data is controlled by the CINT transaction. We need to ensure that we collect all the required details.

To begin, in the CINT transaction, set the CICS API collection options to collect detailed information for the programs and files:

1. Select option 2 for "Configure Region Options."
2. Select either the Defaults or the required CICS region.
3. Select option 4 for "Options".

4. Select option 3 for “CICS API Options.”
5. Set the options as shown in Figure 9-18.

```
CIU240       CICS Interdependency Analyzer for z/OS - V3R2M0       2011/10/17
                        CICS Resources Options for                 09:02:55AM
          CICS Sysid   : DFTS     CICS Applid   : DEFAULTS

Modify the options and press Enter to update, or PF12 to Cancel.

Detect command types: Y=Yes, N=No
                      D=Yes+Detail ( Only for API types marked with * )
APIs
*Programs . . . D *Files. . . . . D *Transactions . D  Task Control . Y
 Presentation . Y *TS Queues  . . D *TD Queues  . . D  Journals . . . Y
 DTP  . . . . . Y  Counters . . . Y  FEPI . . . . . Y *WEB Services . D
*Exits  . . . . D  Others . . . . Y *EVENTS . . . . D  ATOMServices . Y
 XMLtransform . Y  WSAddressing . Y

CICS Sysid:  EJ07   CICS Applid:  IYDZEJ07   TermID:  TC58

F1=          F2=           F3=Exit      F4=          F5=          F6=
F7=          F8=           F9=          F10=         F11=         F12=Cancel
```

Figure 9-18 CICS IA - API Collection Options

We know that our applications involve DB2, MQ, and IMS calls. Therefore, we also must set the collection of the resources.

6. From the Resource Options menu, select option 5 for DB2, IMS, MQ, and True Options. Then activate the required resource collection as shown in Figure 9-19.

```
CIU250       CICS Interdependency Analyzer for z/OS - V3R2M0       2011/10/17
              DB2/MQ/IMS/RMI True   Resource Options for           09:07:03AM

          CICS Sysid   : DFTS     CICS Applid   : DEFAULTS

Modify the options and press Enter to update, or PF12 to Cancel.

Detect command types: Y=Yes, N=No

DB2 Options              MQ Options               IMS Options

 DB2 . . . . . . . Y       MQ  . . . . . . . Y       IMS . . . . . . . Y

RMI True Options

 RMI True. . . . N


CICS Sysid:  EJ07   CICS Applid:  IYDZEJ07   TermID:  TC58

F1=          F2=           F3=Exit      F4=          F5=          F6=
F7=          F8=           F9=          F10=         F11=         F12=Cancel
```

Figure 9-19 CICS IA RMI True options

7. Start the collector. From CINT, select option 1 (Figure 9-20).

```
CIU100        CICS Interdependency Analyzer for z/OS - V3R2M0        2011/10/17
                                Operations Menu                      09:09:43AM

Type action code then press ENTER.                                  More :

1= Start 2= Stop 3= Pause 4= Continue 5= Statistics 6= Refresh Run Options

     CICS      CICS                 Start       Start
 Act  Applid    Sysid Status         Date        Time         Collecting
      ALL       ALL
      IYDZEJOA  EJOA  UNCONNECTED
      IYDZEJ02  EJ02  STOPPED
  1   IYDZEJ07  EJ07  STOPPED



CICS Sysid:  EJ07   CICS Applid:  IYDZEJ07   TermID:  TC58
CIU2120I Press Enter to confirm Start with data restore or PF12 to cancel
F1=Help      F2=          F3=End        F4=          F5=Refresh  F6=
F7=Page Up   F8=Page Down  F9=          F10=         F11=         F12= CANCEL
```

Figure 9-20 CINT START option

As an alternative method, use the CICS IA Explorer plug-in to operate the Interdependency and Affinity collection as shown in Figure 9-21.

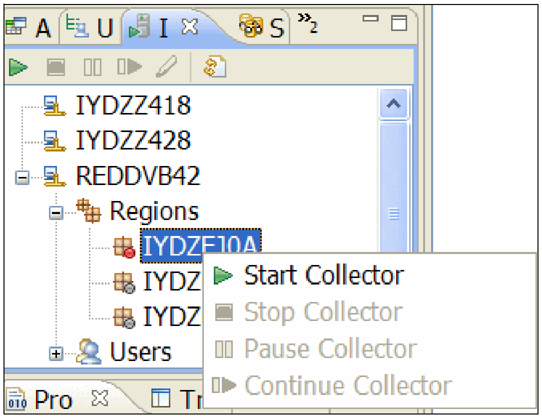


Figure 9-21 Explorer Operations

As explained in the next section, we now must run our workload simulator to capture the CICS IA dependencies. Then we need to stop the CICS IA collector by using the CINT transaction or from the CICS IA Explorer plug-in.

9.4.2 Loading CICS IA dependency data

Next we load the CICS IA dependency data into the DB2 database. We must run the sample job CIUUPDB, which loads the data for all resource types:

- CICS
- DB2
- MQ
- IMS
- Natural or ADABAS

The CIUUPDB job also repopulates the CIU_RESOURCE DB2 table that is used by the Explorer plug-in.

In CICS IA V3.2, you can associate a *Collection Identifier* with each collection that you load into DB2. The identifier can be up to 16 characters and can assist in distinguishing collections. In this case, the collection is done before we change the program definition to indicate that it can run on an open TCB. We use COLDVA42 as the collection ID in this load job (Figure 9-22).

```
//SYSTSIN  DD *
DSN SYSTEM(DH2Y)
RUN PROGRAM(CIUUREG) -
    PLAN(REDIADB) LIB('CICSIA32.RTC62.SCIULOAD') -
    PARMS('DEP,COLLID=COLDVA42')
END
/*
```

Figure 9-22 Collection Identifier

After we load the data, we can start using the CICS IA Explorer plug-in to analyze the data.

9.4.3 Analyzing interdependency data

To analyze interdependency data:

1. In CICS Explorer (Figure 9-23), select the CICS IA perspective. Select **Window** → **Open perspective** → **Other**.

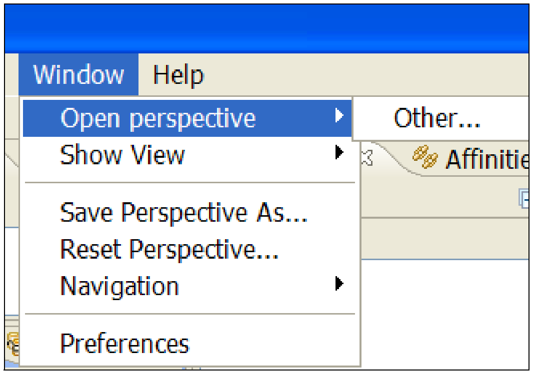


Figure 9-23 Selecting the CICS IA perspective

2. In the Open Perspective window (Figure 9-24), select **CICS IA**.

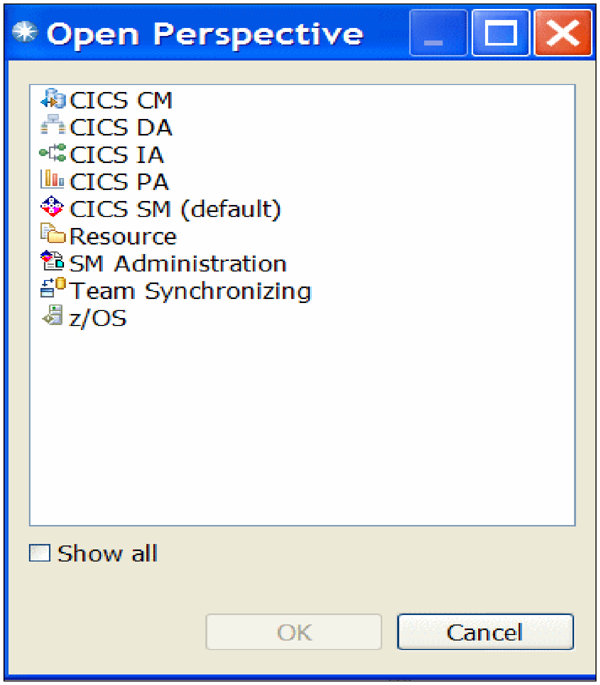


Figure 9-24 Selecting the CICS IA perspective

3. Restrict the analysis to your latest collection by setting the scope to your collection ID. Select your collection ID, right-click, and select **Set as current scope** (Figure 9-25).

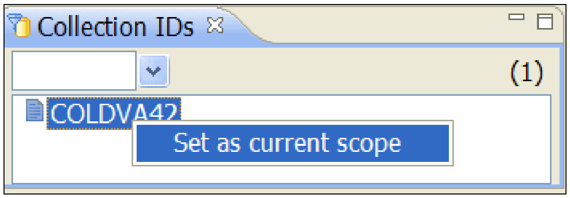


Figure 9-25 Setting the SCOPE

After the SCOPE is set, it is displayed in the toolbar view (Figure 9-26).

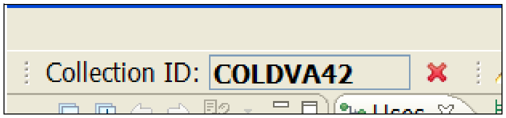


Figure 9-26 Scope set to COLDVA42.

We can now start to analyze the resources. First, we look at all resources used by the MAIL transaction.

9.4.4 Analyzing the MAIL transaction

To analyze the MAIL transaction:

1. In the **Transactions** view (Figure 9-27), limit the transactions to all transactions starting with the letter M by using the filter `M*`.

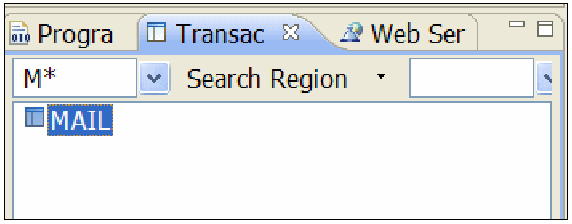


Figure 9-27 Viewing all transactions starting with M

2. To see all the resources used by the MAIL transaction, right-click the transaction, and select **Uses Resources** → **Specific Region** (Figure 9-28).

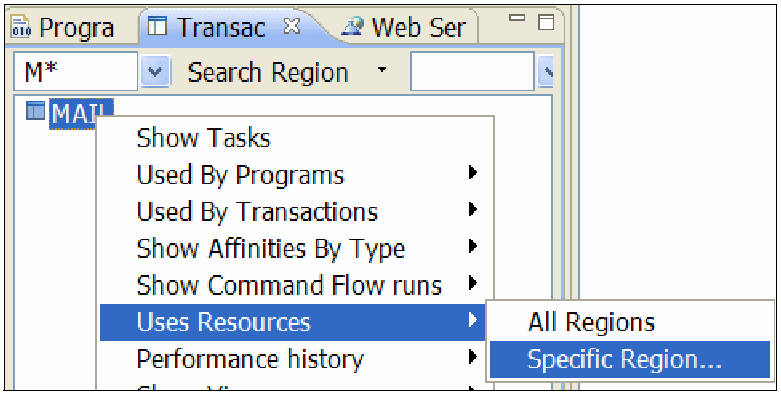


Figure 9-28 Viewing the resources used by the MAIL transaction

3. In the Resources used by window (Figure 9-29), select the region, which in this example has a CICS Applid of IYDZEJ07. Click **OK**.

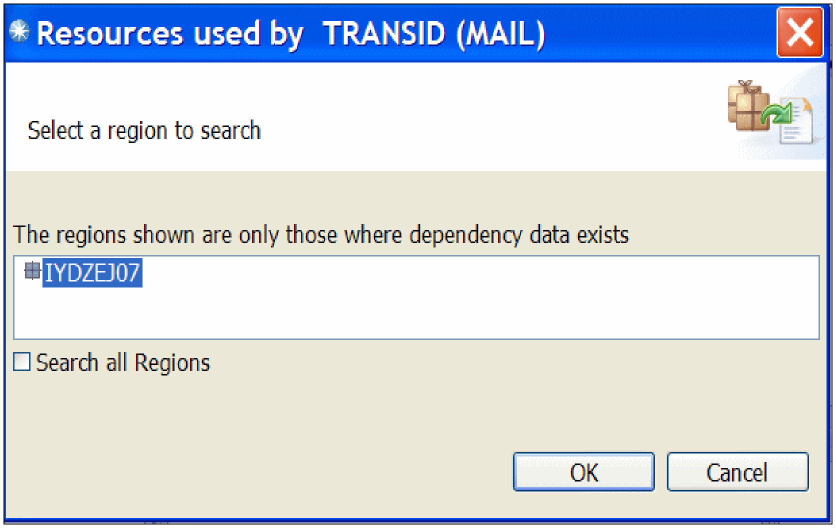


Figure 9-29 Selecting the IYDZEJ07 region

The **Uses** view (Figure 9-30) is now populated with all the resources used by the MAIL transaction. In this example, we see that the application is simple, consisting of only four programs. The initial program is TST4CVD1. Expanding the MAIL transaction on the right side shows which programs link to one another.

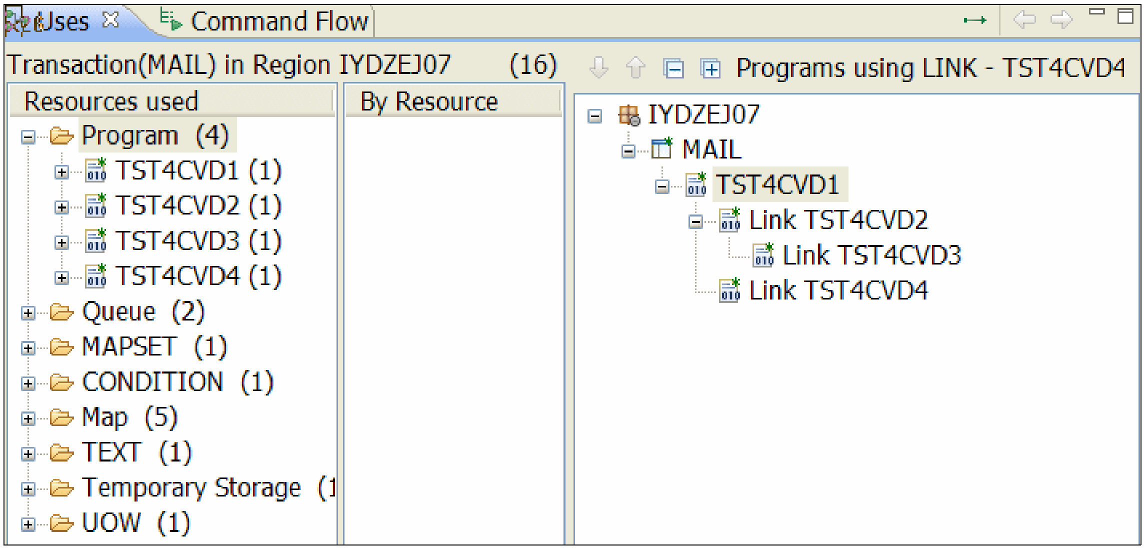


Figure 9-30 Resources used by the MAIL transaction.

Expanding Resources used on the left side shows the resource names and the commands issued against them. Figure 9-31 shows the two MQ queues that are used and the commands that are issued against the resource.

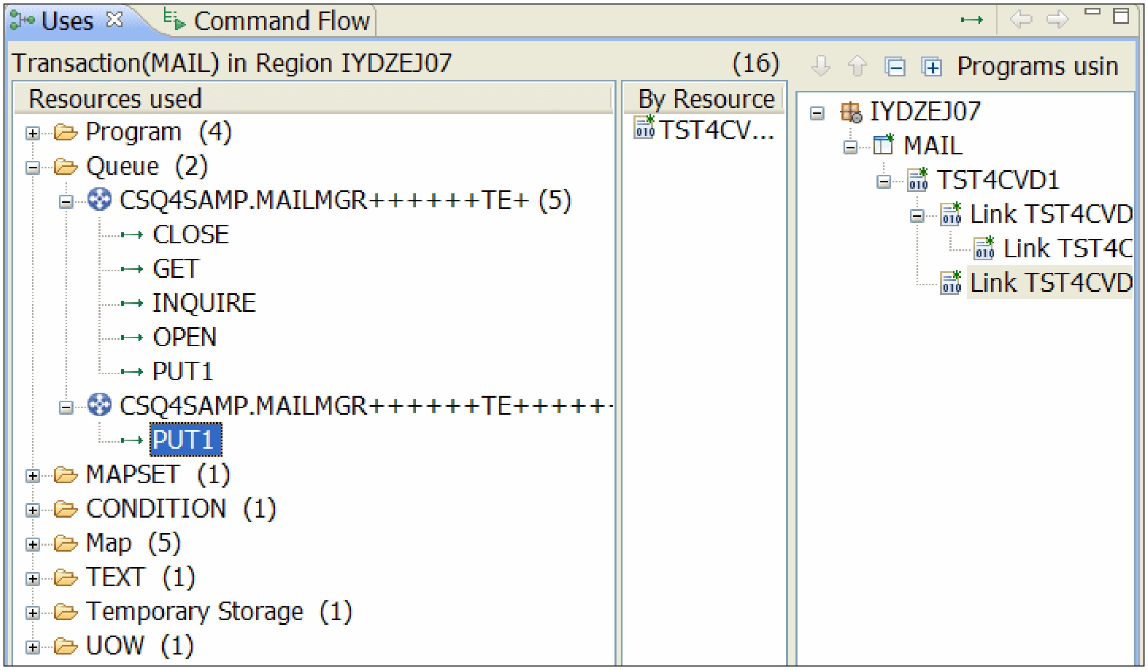


Figure 9-31 MQ resources used by MAIL transaction

Only a PUT1 is issued against the second queue. Selecting this command shows that it occurs in the TST4CVD4 program. Therefore, we can make the following assumptions:

- No commands are issued by the MAIL transaction that can cause data integrity issues. That is, we do not see any of the following commands that might cause a problem:
  - **ADDRESS CWA**
  - **GETMAIL SHARED**
  - **EXTRACT EXIT**
  - **LOAD HOLD**
- There are no dynamic COBOL calls. All programs calls are done by using the **EXEC CICS LINK** command.
- Commands are in here that do not cause a TCB swap. Therefore, it might be beneficial if we define the programs to have a concurrency of REQUIRED or THREADSAFE.

To further understand where these TCB swaps occur, we need to run the MAIL transaction through the CICS IA Command Flow feature.

9.4.5 Running the CICS IA Command Flow feature

By using the CICS IA Command Flow feature, an individual developer can capture all the commands that are issued by a transaction in chronological order. For information about the Command Flow feature, see “Command Flow feature” on page 349.

We use the Command Flow feature to capture more information about the MAIL transaction. In CICS IA V3.2, the Command Flow feature can be administered and operated from within the CICS IA Explorer plug-in.

To capture more information about the MAIL transaction:

1. In the CICS IA perspective, open the **IA Operations** view.
2. Right-click the developer user ID, which is **JAMESE** in this example, and select **Edit options** (Figure 9-32).

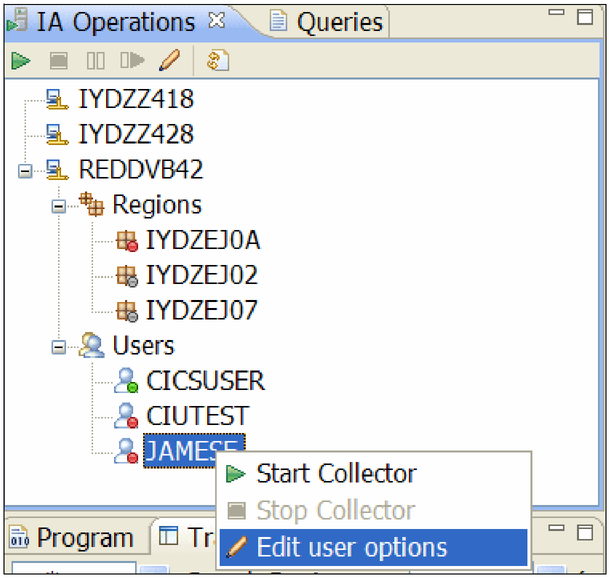


Figure 9-32 Edit Command Flow options

The Command Flow collection has an identifier of MAILQR. We capture information for the MAIL transaction only and in CICS region IYDZEJ07.

3. In the Edit Command flow collector options window, edit the command flow user options to reflect the following values (Figure 9-33):
   a. Under Command Flow Options, complete the following fields:
      i. For Command Flow ID, enter `MAILQR`.
      ii. Enter the date of last start and stop.
      iii. Enter a user name.
   b. Under Transaction List, for Transaction 1, enter `MAIL`.
   c. Under Region Applid List, enter a value for Applid 1, which is `IYDZEJ07` in this example.

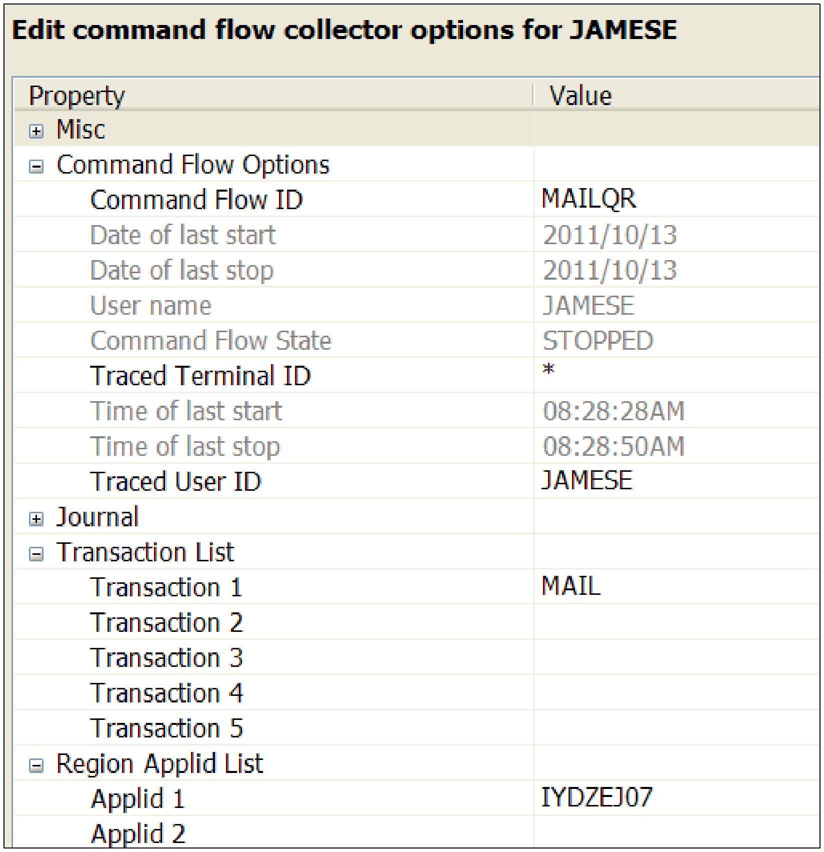


Figure 9-33 Editing the Command Flow Options

4. To start the command flow option, right-click the user, which is **JAMESE** in this example, and select **Start Collector** (Figure 9-34).

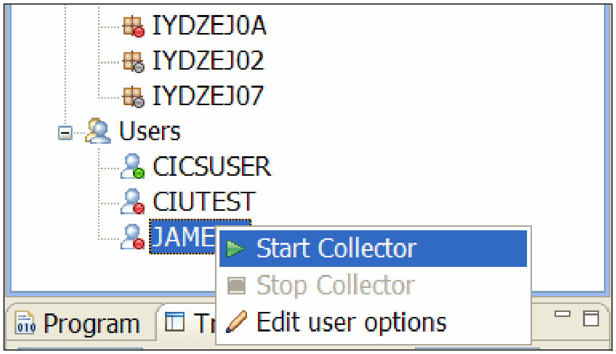


Figure 9-34 Starting the Command Flow

5. Run the MAIL transaction.
6. After capturing some data, stop the command flow run by using the CICS IA Explorer plug-in.
7. Load the collected data into the CICS IA DB2 database to use the CICS IA Explorer plug-in to analyze the data. This step requires running the following sample jobs from SCIUSAMP:

   CIUJLCPY Copies the command flow data from the log stream to the GDG data set.

   CIUUPDB5 Loads the command flow DB2 tables from the latest GDG.

   CIUJLDEL Clears out the log stream.

You can submit these jobs and check them in CICS Explorer by using the z/OS perspective as shown in Figure 9-35.

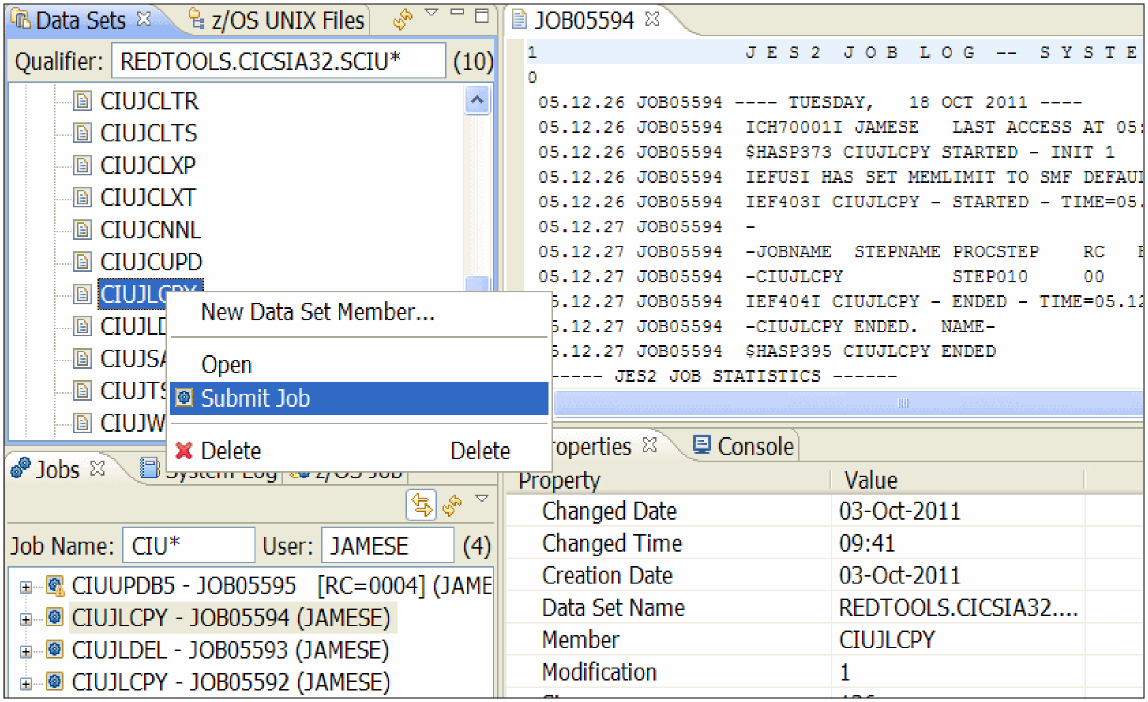


Figure 9-35 The z/OS perspective in CICS Explorer

9.4.6 Analyzing the CICS IA command flow data

We now use CICS IA Explorer to analyze the data that we captured for the MAIL transaction:

1. In the CICS IA Perspective, select the **User** view (Figure 9-36). This view lists all command flow collections by the user ID of the collectors.
2. Expand the tree to see some of the data for collection ID MAILQR.

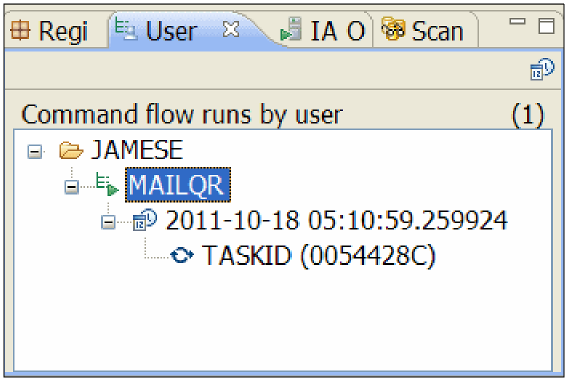


Figure 9-36 The User view

We see that we captured one instance of the MAIL transaction with a TASKID of 0054428C.

3. Right-click **TASKID (0054428C)**, and select **Show Execution** (Figure 9-37).

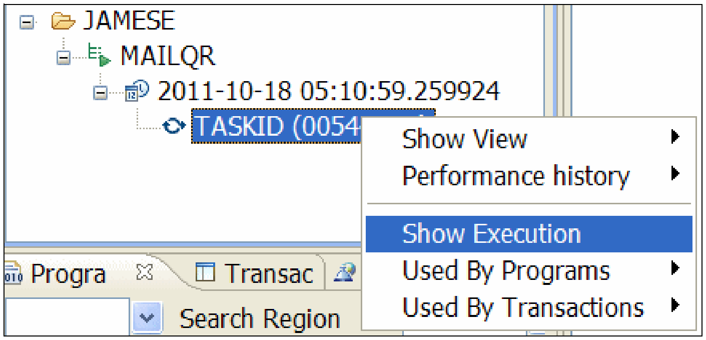


Figure 9-37 Selecting Show Execution

A new **Command Flow** view opens as shown in Figure 9-38.

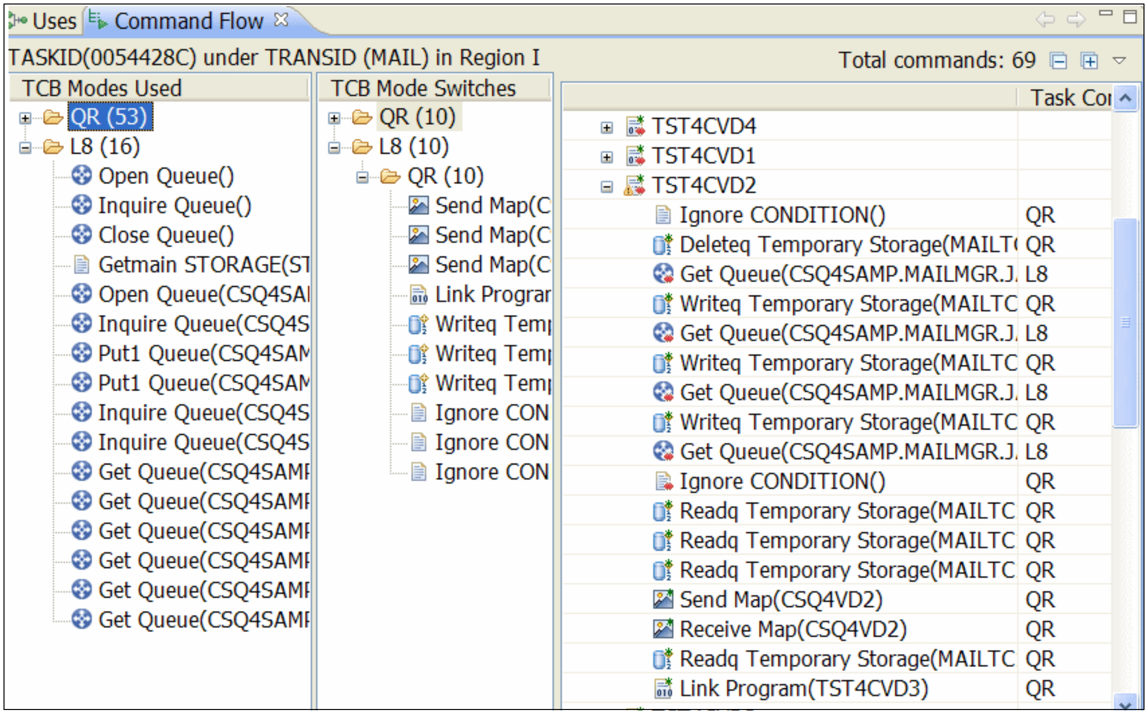


Figure 9-38 Command Flow view

The Command Flow view has three parts:

- TCB modes used
- TCB mode switches
- Execution Tree

TCB modes used

The **TCB modes used** view lists all the commands by the TCB that they used. This part can be useful when analyzing which commands run on the QR TCB.

TCB mode switches

The TCB mode switches section lists the commands that cause a TCB swap to occur. The previous example has two lists:

- Switches from L8 to QR
- Switches from QR to L8

Clicking one of the commands takes you to the execution tree on the right side. For example, clicking the **Writeq Temporary Strage(MAILTC11)** command causes a swap from L8 to QR. We see that this command is issued in the TST4CVD2 program as shown in Figure 9-39.

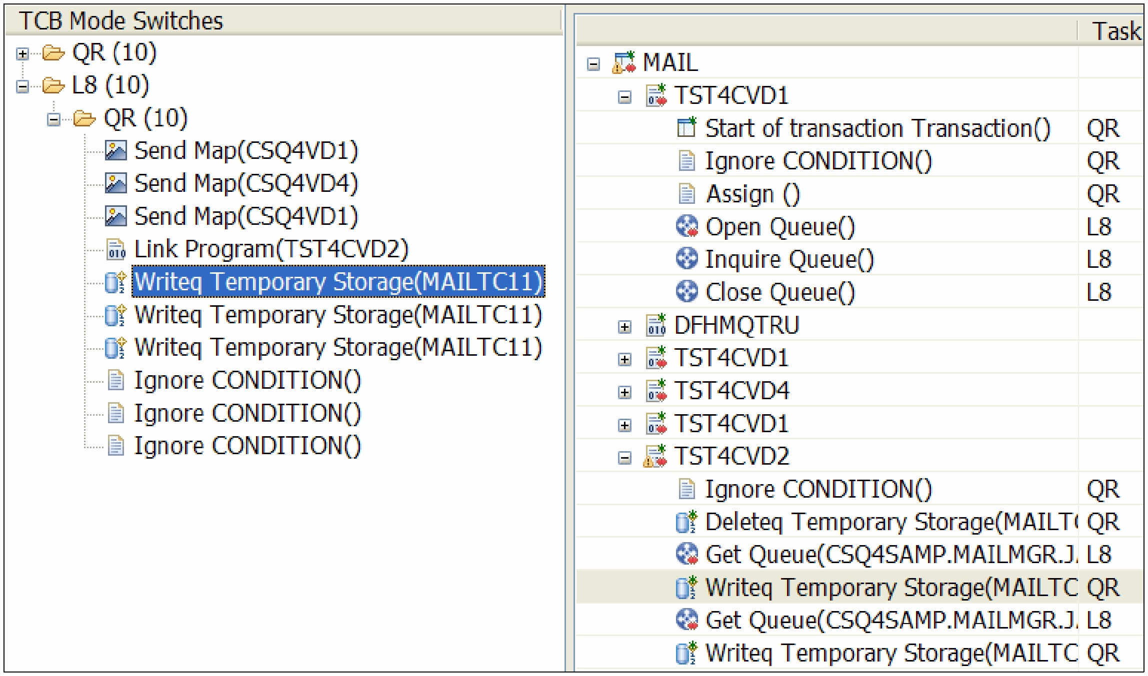


Figure 9-39 L8 to QR switch caused by WRITE TSQUEUE

Execution Tree

The main part of the **Command Flow** view is the execution tree. The execution tree lists all the commands issued in chronological order. The default view shows the initial transaction, the programs, and the commands issued by the programs. It also shows the TCB and the local time at which the command was issued. The command flow captures more information.

To add these columns to the view:

1. Click the **Customize Columns** option (Figure 9-40).

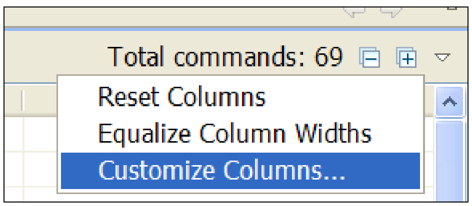


Figure 9-40 Customizing Command Flow columns

2. In the Customize Columns window (Figure 9-41), from the Available columns pane, select the columns you need, and click **Add** to move them to the Current columns pane. Then click **Close**.

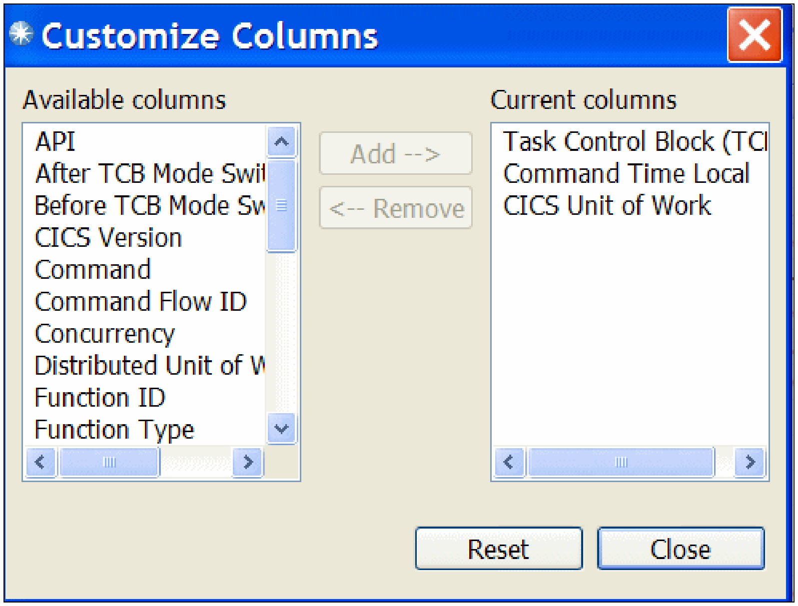


Figure 9-41 Selecting the columns to add to the ***Command Flow*** *view*

From the **Command Flow** view for the MAIL transaction, we see several places where we can benefit by defining the programs with concurrency THREADSAFE or REQUIRED. We also see that some commands, such as the SEND and RECIEVE maps, are not threadsafe and always cause a swap back to the QR TCB.

You can use CICS CM to manage these changes.

If you do not want to use the Explorer to assist with threadsafe analysis, you can use the Dynamic Threadsafe Analysis report.

9.4.7 The Dynamic Threadsafe Analysis report

Input into the Dynamic Threadsafe Analysis report comes from the CICS IA collector for detail interdependency APIs, which is then externalized to DB2. The following tables are used:

- CIU_CICS_DATA
- CIU_MQ_DATA

- CIU_DB2_DATA
- CIU_IMS_DATA
- CIU_REGION_INFO
- CIU_THREADSAFE_CMD
- CIU_PROGRAM_DETAIL
- CIU_FILE_DETAIL

This data was collected in "Collecting interdependency data" on page 268.

Before running the CIUJTSQ2 reporting job for the first time after database creation or migration, we run CIUTSLOD, which is in the SCIUSAMP data set, to establish the threadsafe table information with the appropriate CICS release levels.

We modify the CIUJTSQ2 job in the sample library to include a detail report for collection ID COLDVA42, region IYDZEJ02, and programs starting with TST* as shown in Figure 9-42.

```
//CIUOPTS  DD *
 COLLECTION_ID=COLDVA42
 REGIONNAME=IYDZEJ07
 PROGRAMNAME=TST*
 CICSLEVEL=
 REPORT=DETAIL
 LINESPERPAGE=60
/*
```

Figure 9-42 Threadsafe report options

The report in Figure 9-43 on page 287 indicates that we need to review the program because we have 10 threadsafe inhibitor calls. By reviewing the commands with * flags, we see that the inhibitor is ADDRESS CWA, which can cause an integrity issue if we make the program threadsafe without code review.

```
Program Dynamic Analysis - THREADSAFE DETAIL LISTING FOR CICS TS 4.2

COLLECTION_ID    APPLID   Program   Linkedit   Execution  Concurrency  APIST  Storage  CICS  LIB Dataset Name
                                      Date       Key                          Protect  Rel
---------------- -------- -------- ---------- ----------- ----------- ------- -------- ----- ----------------------------------

                  CMD  Function                Type            Resource                         Offset   Program  Use     Thread-
                  Type                                                                                   Length   Count   safe
                  ---- ----------------------- --------------- -------------------------------  -------- -------- --------------

COLDVA42         IYDZEJ07 TST4CVD1 0001-01-01 USER        QUASIRENT   CICSAPI INACTIVE 0670  REDTOOLS.TESTAPPL.LOADLIB
                  CICS IGNORE                  CONDITION                                           3B22     6618      3477    Y
                  CICS LINK                    PROGRAM         TST4CVD2                            4D08     6618      3408    I
                  CICS LINK                    PROGRAM         TST4CVD4                            4D66     6618      3436    I
                  CICS RECEIVE                 MAP             CSQ4VD0                             4132     6618      3473    N
                  CICS RECEIVE                 MAP             CSQ4VD1                             4E84     6618     10284    N
                  CICS RECEIVE                 MAPSET          CSQ4VDM                             4E84     6618     10284    N
                  CICS RECEIVE                 MAPSET          CSQ4VDM                             4132     6618      3473    N
                  CICS SEND                    MAP             CSQ4VD0                             40DC     6618      3477    N
                  CICS SEND                    MAP             CSQ4VD1                             4E2E     6618     10310    N
                  CICS SEND                    MAPSET          CSQ4VDM                             4E2E     6618     10310    N
                  CICS SEND                    MAPSET          CSQ4VDM                             40DC     6618      3477    N
                  CICS SEND                    TEXT            SEND TEXT                           402A     6618      3401    N
                  MQ   CLOSE                   QUEUE           CSQ4SAMP.MAILMGR++++++TE+           558A     6618      3401    Y
                  MQ   INQUIRE                 QUEUE           CSQ4SAMP.MAILMGR++++++TE+           57BA     6618     13737    Y
                  MQ   OPEN                    QUEUE           CSQ4SAMP.MAILMGR++++++TE+           54FE     6618      3473    Y
  Total CICS calls:        12  Threadsafe:            1  Non-Threadsafe:                     9  Indeterminate Threadsafe:        2
                               DB2 calls:             0  MQ calls:                           3  IMS calls:                       0
                               Dynamic Calls:         0  Threadsafe Inhibitor calls:         0
```

Figure 9-43 Threadsafe report for program TST4CVD1

9.4.8 Analyzing the TXM* transaction

We now repeat the CICS IA analysis that we performed for the MAIL transaction for the tone of the transactions in the TXM* application. We use the TXM1 transaction. First, we look at the resources used by the TXM1 transaction. Then we follow the steps in “Analyzing the MAIL transaction” on page 275.

Figure 9-44 shows that the WORKM program driven by the TXM1 transaction uses a DB2 table called DSN8810.EMP. This program performs an SQL **SELECT** command on this table. This command runs on the L8 TCB. Based on the CICS PA performance data we captured, this transaction performed many TCB swaps.

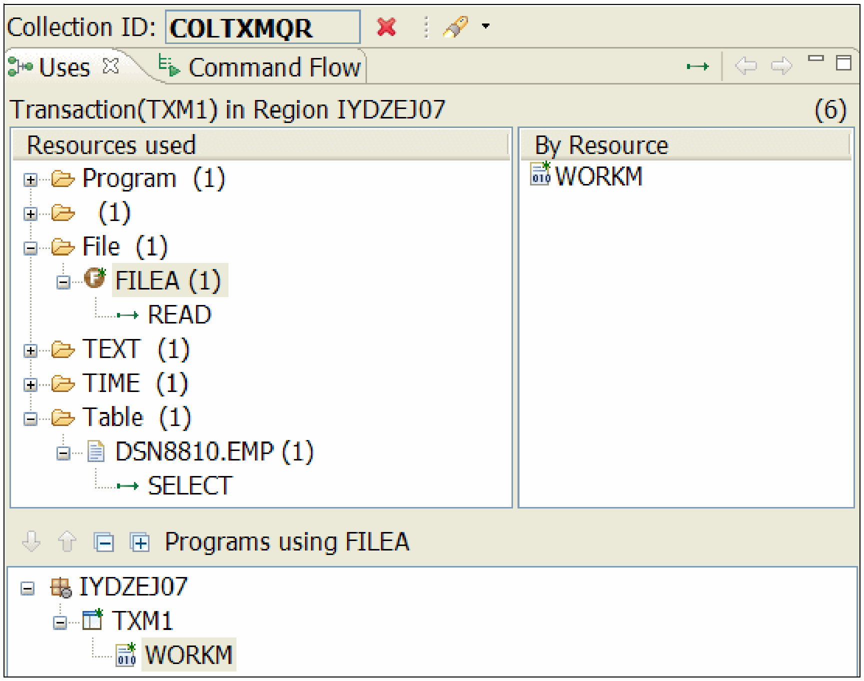


Figure 9-44 CICS IA - Resources used by TXM1

We now look at a CICS IA command flow execution for transaction TXM1. To capture command flow data for a transaction, follow the steps in “Running the CICS IA Command Flow feature” on page 278.

Figure 9-45 shows that the TCB swaps are caused by the SQL call as expected. The SQL calls are followed by the EXEC CICS READ FILE calls to FILEA.

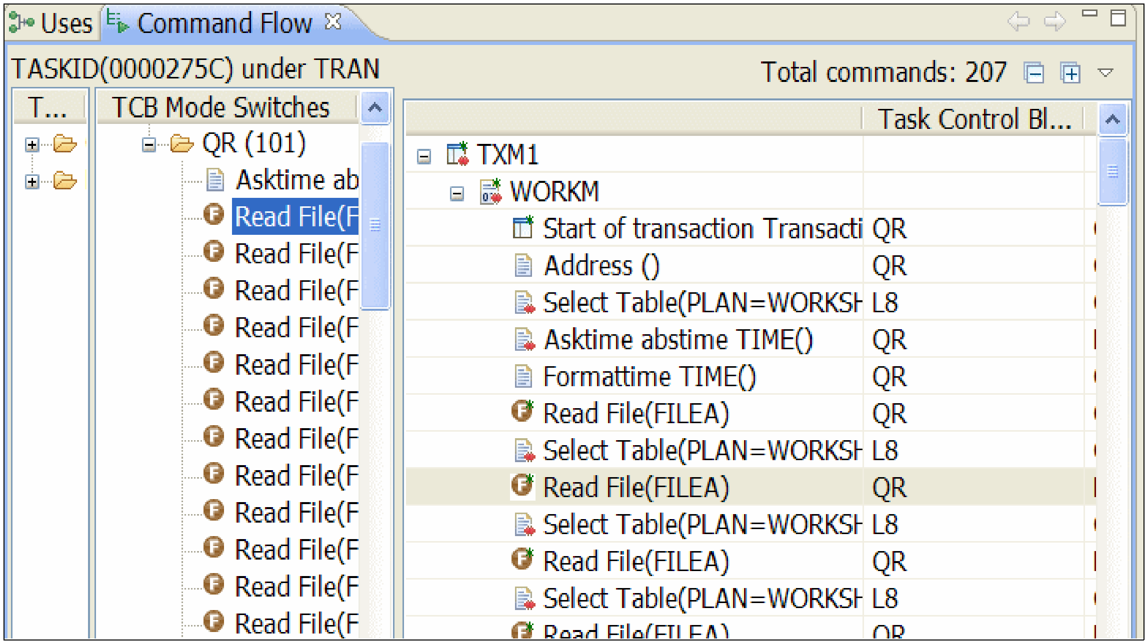


Figure 9-45 Command Flow view for transaction TXM1.

We can also see from Figure 9-44 and Figure 9-45 that the WORKM program issues an EXEC ADDRESS CWA. Before we can make this program threadsafe, we must investigate the source code further.

Example 9-1 shows the source for the WORKM program, which shows that a counter addressed by the CWA is updated.

Example 9-1 Source code for the WORKM program

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. WORKM.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 01 ws-abstime                        pic s9(15) comp-3.
 01 ws-filea                          pic x(08) value'FILEA'.
 01  wm-msg.
     05  Filler                      pic x(22)
      value 'Work program complete.'.
```

```
    05  Filler                      pic x(30)
     value SPACES.

* copy the filea record layout from sdfhsamp
 01 ws-filea-record.
    copy dfh0cfil.

 01 ws-ptr                           pointer.
 01 ws-counter2                      pic s9(8) comp.
 01 ws-counter3                      pic s9(8) comp.
 01 ws-count                         pic s9(8) comp.
 01 ws-queue                         pic x(08)
        VALUE 'OUTPUTQ'.
 01 WS-MSG.
    03 WS-TXN                        pic x(05).
    03 filler                        pic x(17)
        value "Counter value :- ".
    03 ws-counter                    pic 9(8).
    03 filler                        PIC x(13)
       value " Date/Time : ".
    03 ws-datestring                 pic x(64).

 01 ws-cwa-ptr                       usage is pointer.

    EXEC SQL
          DECLARE DSN8810.EMP TABLE (
          EMPNO                       CHAR(6),
          FIRSTNME                    CHAR(12),
          MIDINIT                     CHAR(1),
          LASTNAME                    CHAR(15),
          WORKDEPT                    CHAR(3),
          PHONENO                     CHAR(4),
          HIREDATE                    DATE,
          JOB                         CHAR(8),
          EDLEVEL                     SMALLINT,
          SEX                         CHAR(1),
          BIRTHDATE                   DATE,
          SALARY                      DECIMAL,
          SALARY                      DECIMAL,
          BONUS                       DECIMAL,
          COMM                        DECIMAL )
    END-EXEC.

    EXEC SQL INCLUDE SQLCA END-EXEC.
```

```
 LINKAGE SECTION.
 01  SHARED-AREA.
     03  SHARED-COUNTER                  PIC S9(8) COMP.

 PROCEDURE DIVISION.
     MOVE EIBTRNID TO WS-TXN.

*    Access our shared storage area - this time the CWA
     EXEC CICS ADDRESS CWA(WS-PTR) END-EXEC.

*    map our linkage section to the address of the shared area
     Set address of shared-area to ws-ptr.

*
*    Make DB2 Call which will transfer to the L8
*
     EXEC SQL
         SELECT count(*)
         INTO :ws-count FROM DSN8810.EMP
         WHERE EMPNO = "000990"
     END-EXEC.

*    read the value in shared storage.
     move shared-counter to ws-counter.
*    ... and change its value
     Add 1 to ws-counter.

*    ** Do some important processing **
     move zero to ws-counter2.
     Perform 100000 Times
       add 2 to ws-counter2
       subtract 1 from ws-counter2
     End-Perform.
*    **********************************

*    ** get the time
     exec cics asktime abstime(ws-abstime) end-exec.

*    ... and format it **
     exec cics formattime
        abstime(ws-abstime) datestring(ws-datestring)
     end-exec.

     Perform 100 Times
*    ** Read file **
```

```
          move '000100' to NUMB
          exec cics read
             file(ws-filea) ridfld(NUMB) into(ws-filea-record)
             nohandle
          end-exec

*
*    Make DB2 Call which will transfer to the L8
*
          EXEC SQL
              SELECT count(*)
              INTO :ws-count FROM DSN8810.EMP
              WHERE EMPNO = "000990"
          END-EXEC

     End-Perform.
*    update the shared storage with our new value
     Move ws-counter to shared-counter.

*    output the results ......
*    exec cics
*      writeq ts main queue(ws-queue) from(ws-msg)
*    end-exec.

     EXEC CICS SEND TEXT FROM(WM-MSG) ERASE FREEKB END-EXEC.

     exec cics return end-exec.
```

If we make this program threadsafe, we can introduce data integrity issues because several instances of the program might be running on L8 TCBs at the same time.

Figure 9-46 shows what happens if we run this program with a concurrency of THREADSAFE. The WORKM program is started by several transactions, each of which can run on an L8 TCB. The counter addressed by the ADDRESS CWA becomes corrupted as shown in the CEBR listing of the TSQUEUE.

```
 CEBR  TSQ OUTPUTQ          SYSID EJ03 REC   289 OF   601    COL     1 OF   107
 ENTER COMMAND ===> _
00288 TXM4 Counter value :- 00000266 Date/Time : Mon, 30 Nov 2009 14:12:33 GMT
00289 TXM3 Counter value :- 00000267 Date/Time : Mon, 30 Nov 2009 14:12:33 GMT
00290 TXM4 Counter value :- 00000268 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00291 TXM3 Counter value :- 00000269 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00292 TXM1 Counter value :- 00000269 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00293 TXM5 Counter value :- 00000269 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00294 TXM5 Counter value :- 00000270 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00295 TXM4 Counter value :- 00000271 Date/Time : Mon, 30 Nov 2009 14:12:35 GMT
00296 TXM1 Counter value :- 00000272 Date/Time : Mon, 30 Nov 2009 14:12:36 GMT
00297 TXM5 Counter value :- 00000273 Date/Time : Mon, 30 Nov 2009 14:12:36 GMT
00298 TXM4 Counter value :- 00000274 Date/Time : Mon, 30 Nov 2009 14:12:36 GMT
00299 TXM1 Counter value :- 00000275 Date/Time : Mon, 30 Nov 2009 14:12:37 GMT
00300 TXM5 Counter value :- 00000276 Date/Time : Mon, 30 Nov 2009 14:12:37 GMT
00301 TXM4 Counter value :- 00000277 Date/Time : Mon, 30 Nov 2009 14:12:37 GMT
00302 TXM1 Counter value :- 00000278 Date/Time : Mon, 30 Nov 2009 14:12:38 GMT
00303 TXM5 Counter value :- 00000279 Date/Time : Mon, 30 Nov 2009 14:12:38 GMT
00304 TXM4 Counter value :- 00000280 Date/Time : Mon, 30 Nov 2009 14:12:38 GMT
```

Figure 9-46 CEBR view of the WORKM TSQUEUE

Therefore, before proceeding with making the WORKM program threadsafe, we must serialize the update to the counter by using EXEC CICS ENQ or EXEC CICS DEQ technique described in 3.2, “Serialization techniques” on page 56.

Running the **CICS READ** command on the L8 TCB might greatly reduce the number of TCB swaps. Potentially the WORKM program is a good candidate to be defined as concurrency THREADSAFE or REQUIRED.

In CICS TS V3.2 and later, EXEC CICS calls to LOCAL files became threadsafe. In CICS TS V4.2, EXEC CICS calls to REMOTE files using IPIC connections became threadsafe.

Before changing the concurrency attribute for the WORKM program, we can use CICS Configuration Manager to help understand the nature of the calls to FILEA. Our CICS system is on CICS TS V4.2. Therefore, we must determine whether FILEA is LOCAL or REMOTE to region REDDVA42. If FILEA is REMOTE, we must also determine the type of connection that is used. By using the CICS CM search feature as described in “Viewing the program definition” on page 297, we can look at the resource definition for FILEA.

Figure 9-47 shows that FILEA is a REMOTE file with a remote system connection SYSID of EJ06. By using the base CICS Explorer, we must now determine whether the connection is an IPIC connection that supports threadsafe **EXEC CICS FILE** commands.

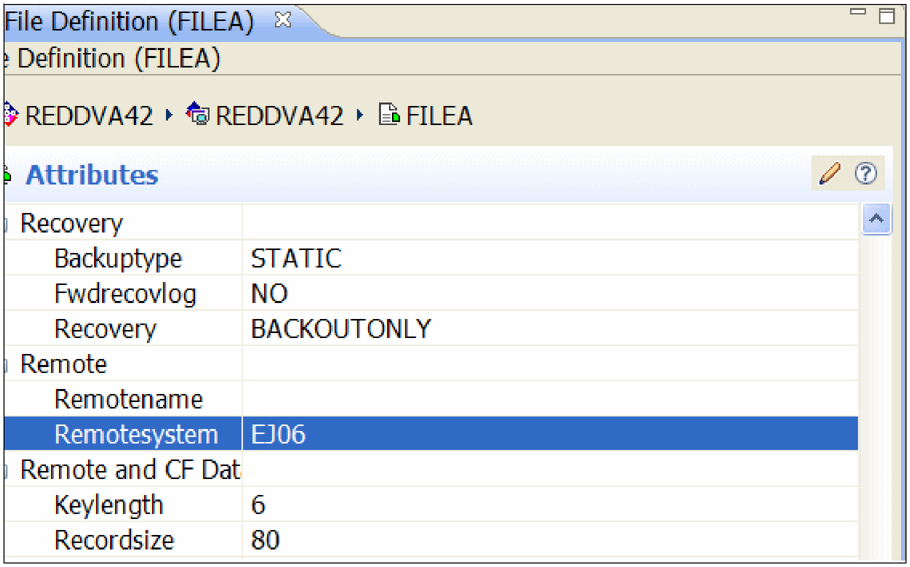


Figure 9-47 Resource definition for FILEA

To determine if the connection is an IPIC connection that supports threadsafe EXEC CICS FILE commands:

1. In the CICS Explorer, select the **CICS SM** perspective, and click the CICS region of interest, which is REDDVA42 in this case. The default **Regions** view is displayed (Figure 9-48).

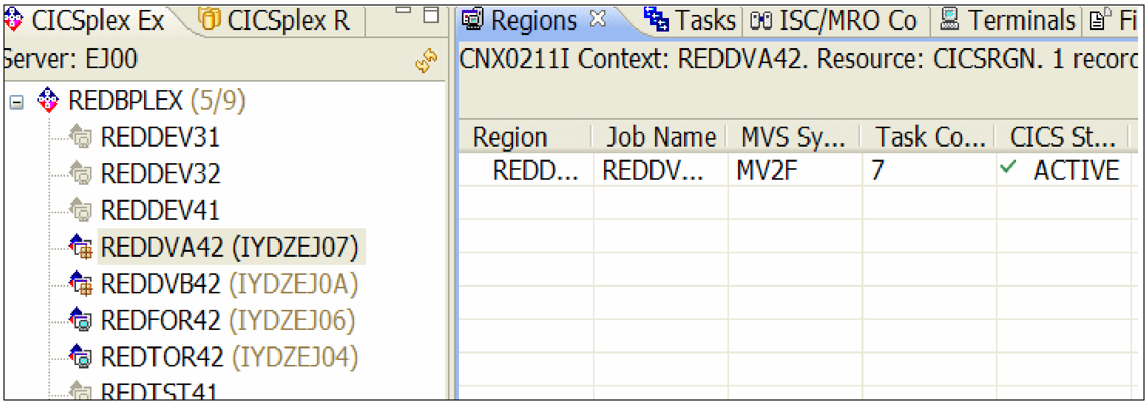


Figure 9-48 Region view for REDDVA42

2. To determine whether this region has any IPIC connections, select **Operations** → **IPIC Connections** (Figure 9-49).

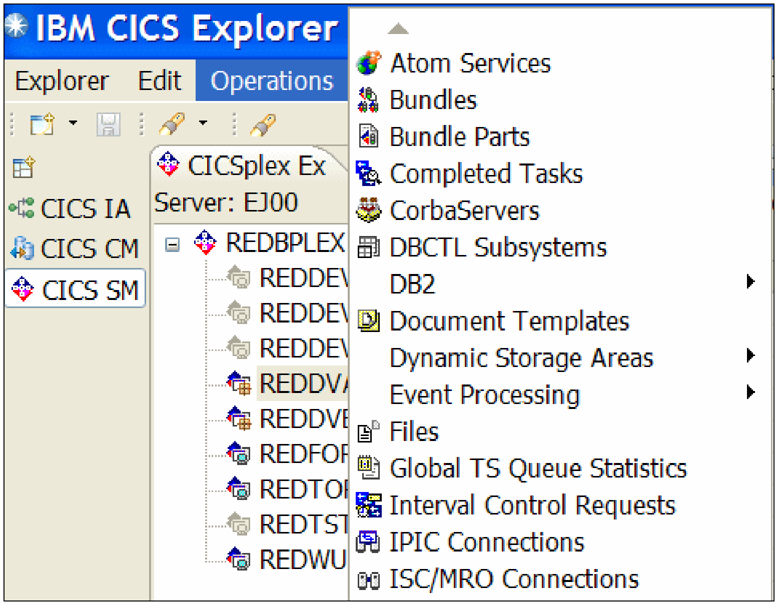


Figure 9-49 Selecting IPIC Connections

As shown in Figure 9-50, no IPIC connections are available.

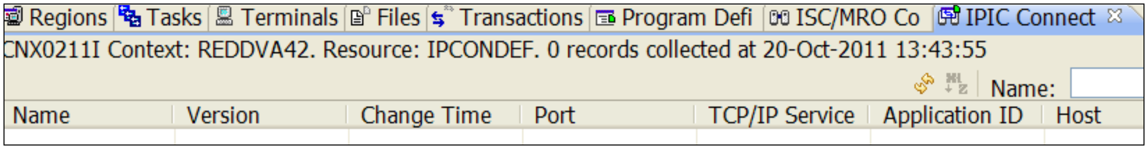
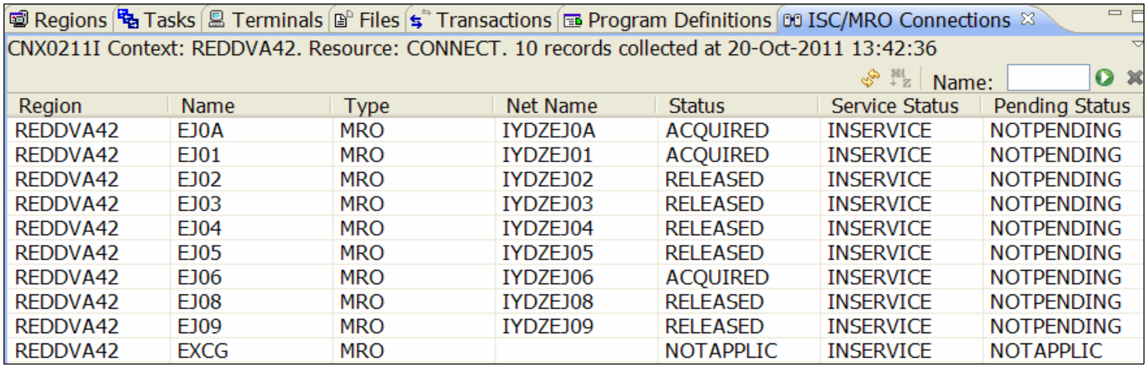


Figure 9-50 CICS Explorer - IPIC connections for REDDVA42

3. To confirm that connection EJ07 is an ISC/MRO connection, select the **ISC/MRO Connections** view for this region (Figure 9-51).



Region	Name	Type	Net Name	Status	Service Status	Pending Status
REDDVA42	EJ0A	MRO	IYDZEJ0A	ACQUIRED	INSERVICE	NOTPENDING
REDDVA42	EJ01	MRO	IYDZEJ01	ACQUIRED	INSERVICE	NOTPENDING
REDDVA42	EJ02	MRO	IYDZEJ02	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EJ03	MRO	IYDZEJ03	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EJ04	MRO	IYDZEJ04	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EJ05	MRO	IYDZEJ05	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EJ06	MRO	IYDZEJ06	ACQUIRED	INSERVICE	NOTPENDING
REDDVA42	EJ08	MRO	IYDZEJ08	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EJ09	MRO	IYDZEJ09	RELEASED	INSERVICE	NOTPENDING
REDDVA42	EXCG	MRO		NOTAPPLIC	INSERVICE	NOTAPPLIC

Figure 9-51 ISC/MRO Connections (EJ06 is ACQUIRED)

To gain any benefit from making program WORKM threadsafe, we create an IPIC connection between the regions and use this connection for the remote connection for FILEA.

To obtain the performance benefits of making the WORKM program threadsafe, we must complete the following tasks before proceeding:

- Modify the program to use EXEC CICS ENQ/DEQ to serialize updates to the CWA counter.
- Change the connection to the File Owning Region to use an IPIC connection.

We can then use CICS CM to change the concurrency attribute for the WORKM program definition to THREADSAFE as explained in the next section.

9.5 Changing the program definitions

We can simplify and provide controlled management of CICS threadsafe resource definition changes by using CICS Configuration Manager. We can accomplish this task by using the CICS CM ISPF interface or the CICS Explorer CM plug-in. For this exercise, we use the CICS Explorer CM plug-in.

9.5.1 Viewing the program definition

To view the program definition:

1. In the upper-right corner of CICS Explorer, click the **CICS CM Perspective** view.
2. Click the **Configurations** view to see a list of configurations, which were defined on this system for CSDs that apply to specify CICS regions.
3. Click the **REDDVA42** configuration, which populates the **Lists** view and the **Groups** view (Figure 9-52).

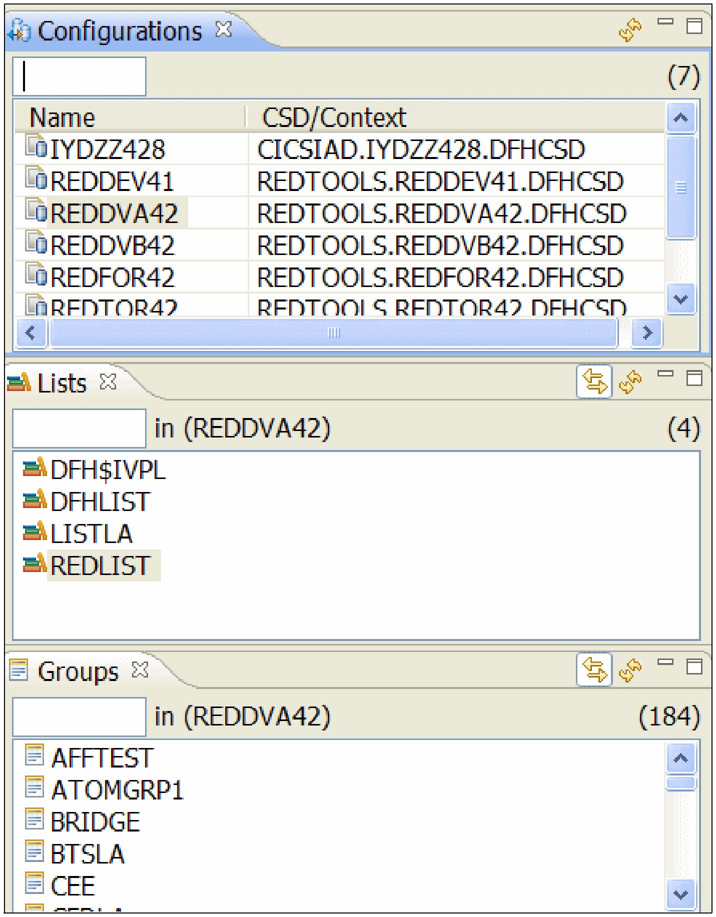


Figure 9-52 Lists and groups in the REDDVA42 configuration

4. Right-click **REDDVA42**, and select **Search** → **Search Programs** (Figure 9-53).

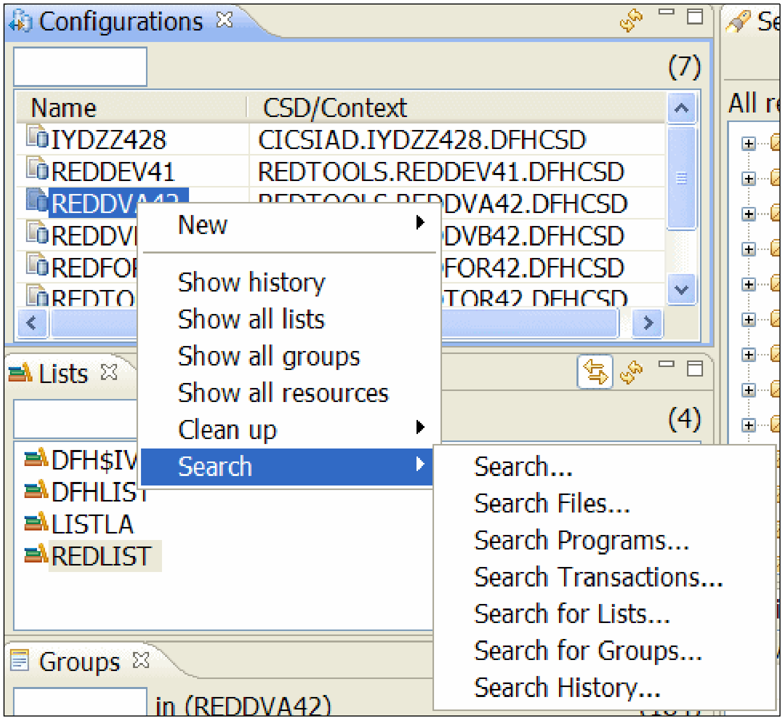


Figure 9-53 Searching for programs in REDDVA42

5. On the **Definition Search** tab of the Search window (Figure 9-54):

   a. In the Resource Name field, type `TST*`.

   b. For Configuration(s), select **REDDVA42** if it is not set automatically.

   c. For Group, leave as * for a generic search.

   d. Click **OK**.

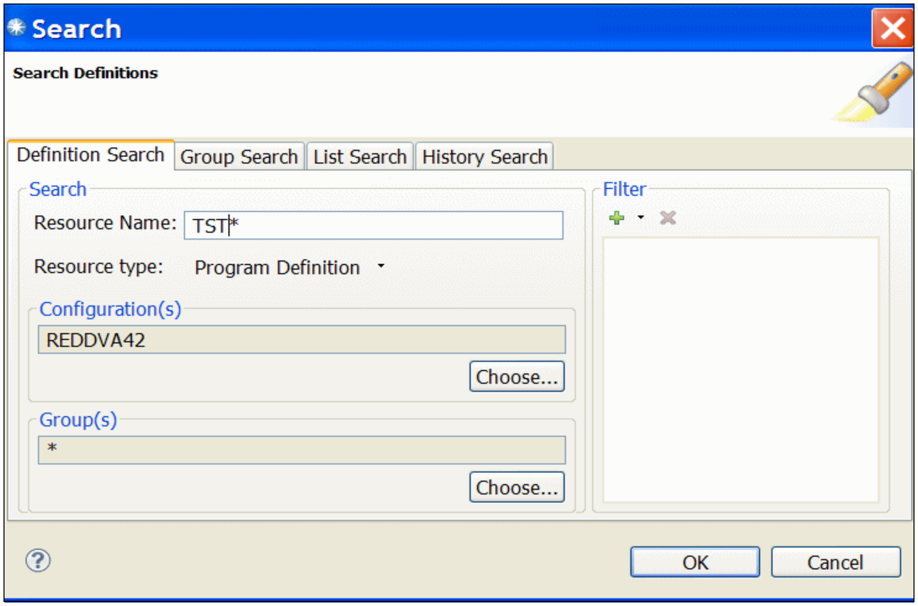


Figure 9-54 Selecting the search criteria

All programs starting with TST* are shown at the top of the perspective under the **Search Results** view (Figure 9-55).

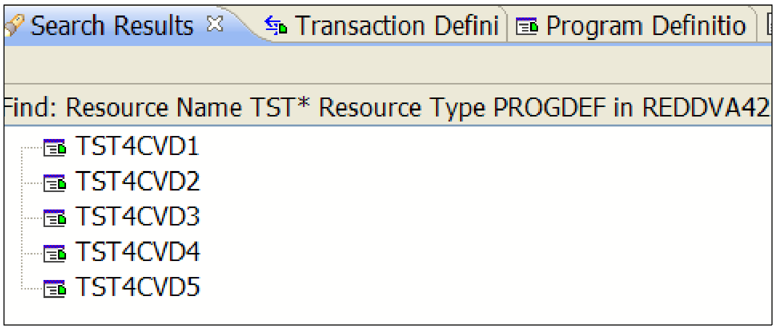


Figure 9-55 CICS CM Search Results

6. To view the definition for program, double-click **TST4CVD1**. Alternatively, right-click and select **Open**.

   You see a resource definition view, which is the **Program Definition** view in this example (Figure 9-56).

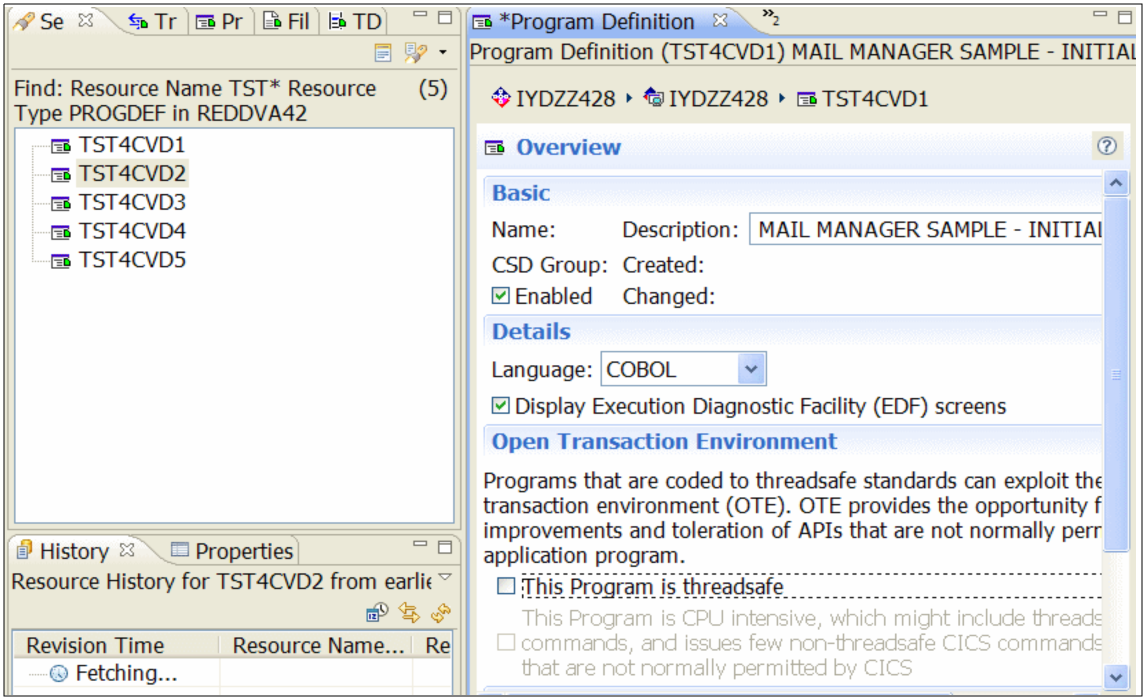


Figure 9-56 CICS CM Explorer: View of the TST4CVD1 program

The overview in Figure 9-56 contains common resource attributes that you might want to change or that are needed for a new definition. Notice that the Threadsafe box is not selected. You can also view the detailed attributes by clicking the **Attributes** tab at the lower-right corner.

9.5.2 Changing the program definition

We can change the definition to threadsafe by selecting the **Threadsafe** option:

1. Click the **Attributes** tab where you are presented the details of the resource.
2. Right-click the value for Concurrency, which is **QUASIRENT**. Figure 9-57 shows the detail view of the resource attributes.

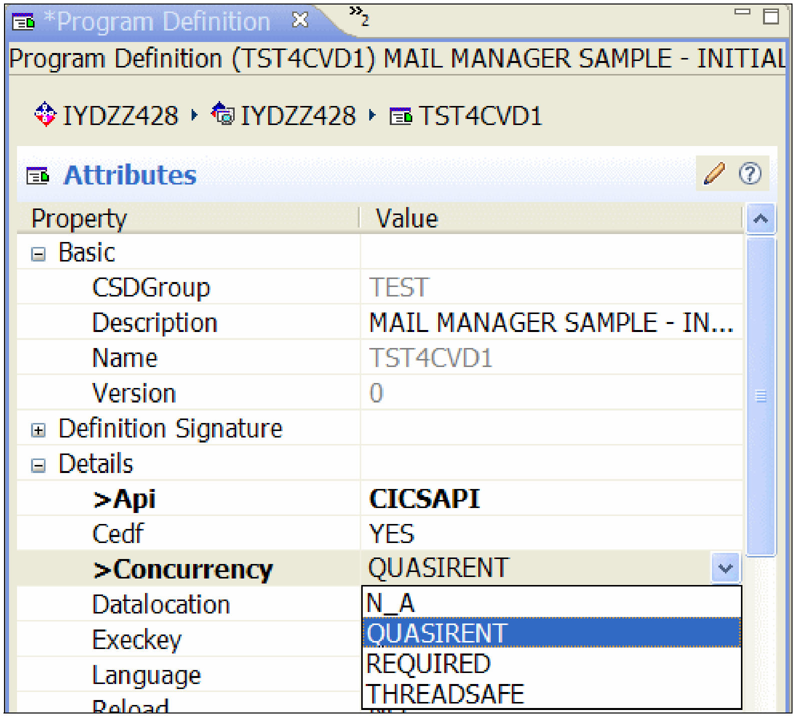


Figure 9-57 Editing the Attributes for the TST4CVD1 program

After analyzing the CICS IA data, we decide that all programs driven by the MAIL transaction can be made threadsafe. We change the concurrency to THREADSAFE. We can perform further analysis to see whether this candidate is a good one for being made concurrency REQUIRED.

3. Change the CONCURRENCY to threadsafe, and then press Ctrl+S to save the changes.
4. Repeat the change in the previous step for other programs that you want to make THREADSAFE.

If you are making many changes, use the CICS CM ISPF Package feature. By using this feature, you can package several changes together and then deploy and install them at the same time. This feature also helps you to back out the package if necessary.

9.5.3 Viewing Audit history

When you select and change the program resource definition in the CICS CM Explorer plug-in, the **History** view automatically refreshes to the resource you select. By selecting the TST4CVD1 program definition the **History** view, the resource is automatically displayed as shown in Figure 9-58. In this example, the concurrency attribute for this program definition changed from QUASIRENT to THREADSAFE. The **History** view also records the date, time, and user who made the change.

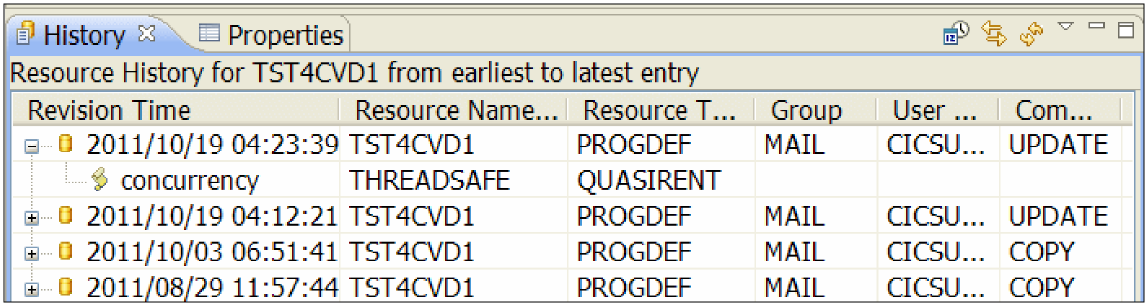


Figure 9-58 History log for the TST4CVD1 program definition

By using the CICS CM Explorer, you can view the history for the whole configuration or for an individual group. You can also perform history searches. For example, you can search for all programs starting with TST*, in configuration REDDVA42, that were changed before a date as shown in Figure 9-59 on page 303. You can also filter the search on several attributes including user ID.

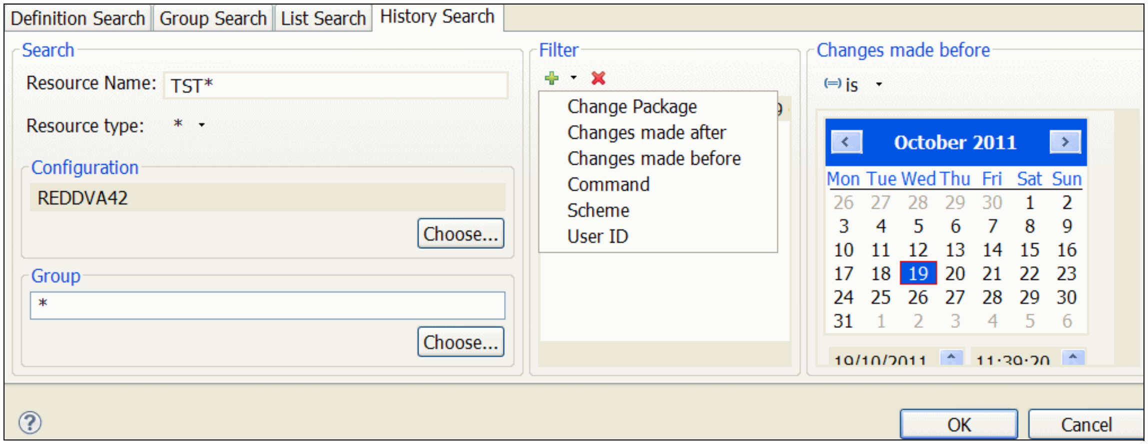


Figure 9-59 History Search criteria

9.5.4 Comparing the resource definitions

By using CICS CM, you can compare CICS resources. You can compare lists, groups, or resource definitions. You might want to compare a resource definition for a program in different configurations to see whether the attributes are different. In this example, we compare the TST4CVD1 program, which is defined in REDDEV41 and REDDVA42:

1. Search for the program in both configurations (Figure 9-60).

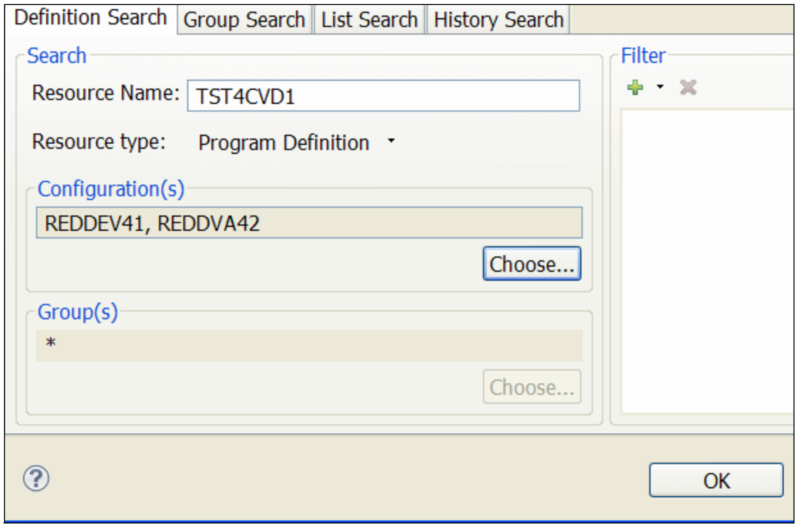


Figure 9-60 Searching for a program in two configurations

2. To select the TST4CVD1 program from both configurations, press the Ctrl key, right-click, and select **Compare with each other** (Figure 9-61).

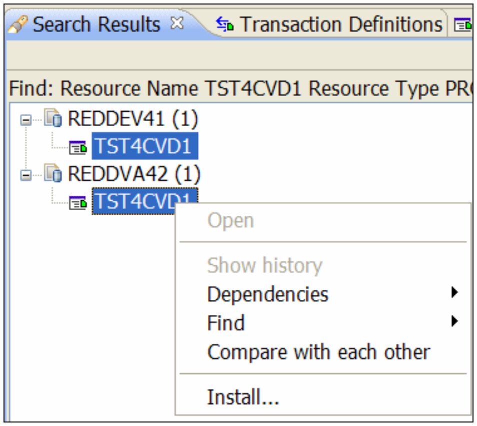


Figure 9-61 Comparing the program definitions

In the **Compare** view that opens (Figure 9-62), the attributes that are different are listed in the upper part of the view. Selecting one of these attributes highlights the difference in the lower part of the view. In this example, the concurrency attribute is different.

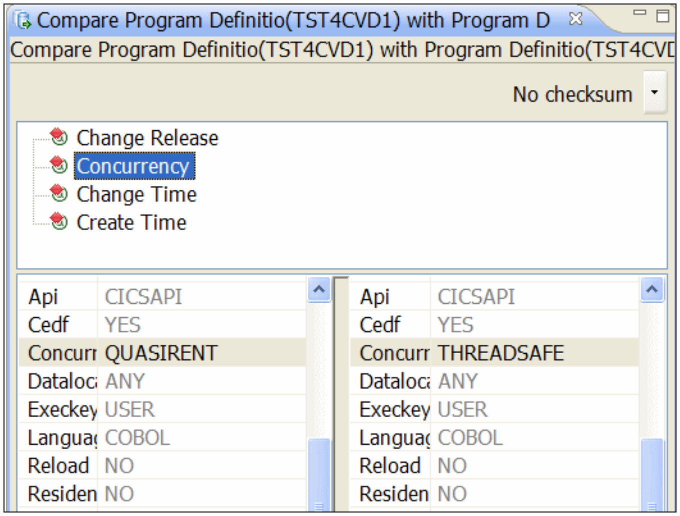


Figure 9-62 Results of the Compare program

9.5.5 Installing the program definition

To implement the program change, install the resource definition in the CICS Region. The **CICS CM Explorer** view still shows the program on the **Search Results** tab.

To install the program definition:

1. Right-click the **TST4CVD1** program, and then select **Install** (Figure 9-63).

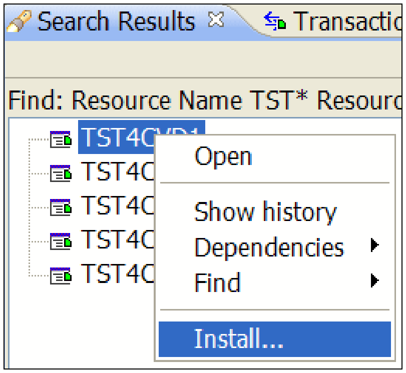


Figure 9-63 Installing the resource definition

2. In the **Perform Operation** view (Figure 9-64), select the CICSplex that you require, and then select the Target. Click **OK** to install the definition.

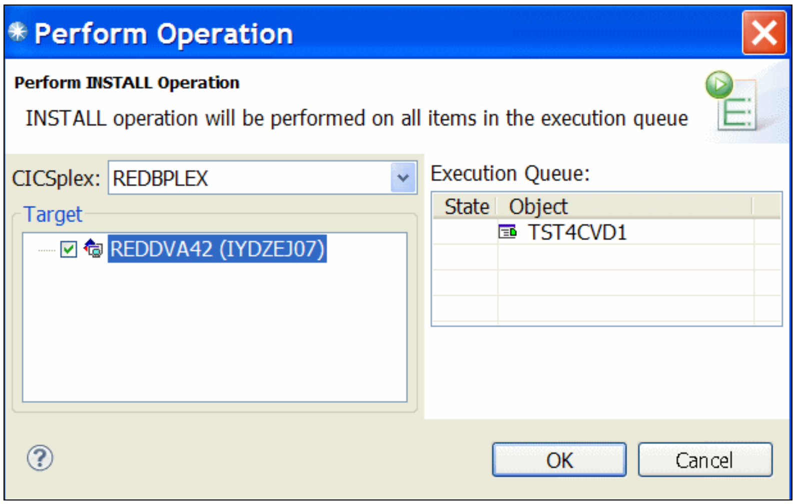


Figure 9-64 Selecting the target for the resource installation

We changed and installed the resource definitions using CICS Configuration Manager. Next, we test our changes and compare the performance results with the benchmarks that we captured earlier.

9.6 Testing and benchmarking the results

To test and benchmark the results, we capture SMF data for use by CICS PA. Then we run a CICS PA Transaction Profile report and rerun the CICS PA reports that we ran at the start.

9.6.1 Running the PA Transaction Profiling report

To run the PA Transaction Profiling report to verify the results:

1. In the CICS PA ISPF main menu (Figure 9-65), select option 8 for Profiling, and then select option 1 for SMF data against SMF Baseline.

```
 Transaction Profiling Menu
Command ===>

Select an option then press Enter.

1  1. SMF data against SMF Baseline
   2. SMF data against HDB Baseline
   3. HDB data against HDB Baseline

HDB Register . . . 'REDTOOLS.CICSPA.SHARED.REPOSTRY'              +

 F1=Help     F3=Exit     F4=Prompt   F6=Resize  F10=Actions F12=Cancel
```

Figure 9-65 Transaction Profiling Menu

2. In the Run Transaction Profiling menu, complete the fields as shown in Figure 9-66. The Report and Baseline Intervals are the same as noted for the application test script executions. Performance selection was made to include only transactions MAIL and DE1. We chose the default for the Report Form to select the Performance Report.

```
 Run Transaction Profiling
Command ===>

Specify Profiling data sources and options, then SUBmit to run.
                                                                   More:     +
Report System Selection:                sssss Report Interval sssssss
 APPLID . . IYDZEJ07  +                      YYYY/MM/DD  HH:MM:SS.TH
 Image  . . MV2F      +                 From 2011/11/27  12:27:00.00
 Group  . .           +                 To   2011/11/27  12:51:00.00

Baseline System Selection:              ssss Baseline Interval ssssss
 APPLID . . IYDZEJ07  +                      YYYY/MM/DD  HH:MM:SS.TH
 Image  . . MV2F      +                 From 2011/11/27  11:31:00.00
 Group  . .           +                 To   2011/11/27  12:26:00.00

Report Format:
 Report Form . . .           +          Baseline Form . .           +
 Title  . . Transaction Profile for MAIL and DE1


Summary Options:                       Reporting Options:
 Time Interval . . 00:01:00 (hh:mm:ss)  Lines . . . . /  Report  /  Baseline
 Totals Level  . . 8  (blank or 0-8)                  /  Delta   /  Change
                                        Threshold . .     % Above
Selection Criteria:                                       % Below Baseline
    Performance  *                      Exclude . . .    Within threshold
                                                      /  Blank lines
```

Figure 9-66 Transaction profiling options

3. On the command line, enter **SUB** to run the report. Figure 9-67 shows the Transaction Profiling report.

```
 V3R2M0                                                   CICS Performance Analyzer
                                                            Transaction Profiling
                                            __________________________________________________
PROF0001 Printed at 12:16:19  2/01/2012     Report   Data from 12:26:52 11/27/2011 to 12:50:59 11/27/2011
                                            Baseline Data from 11:31:47 11/27/2011 to 12:25:58 11/27/2011
Transaction Profile for MAIL

                                     Avg      Avg      Avg      Avg      Avg      Avg      Avg      Avg      Avg      Avg
Tran                        #Tasks Response Dispatch User CPU  Suspend DispWait  FC Wait   FCAMRq  IR Wait SC24UHWM SC31UHWM
                                      Time     Time     Time     Time     Time     Time    Count     Time    Count    Count
DE1      Report                620    .0571    .0031    .0010    .0541    .0063    .0025        5    .0000        0    98062
DE1      Baseline              465    .1936    .0046    .0010    .1890    .0402    .0060        5    .0000        0    98108
         Delta                +155   -.1364   -.0015   +.0001   -.1349   -.0340   -.0035       +0    .0000        0      -46
         Change%            +33.33   -70.49   -33.09    +6.11   -71.39   -84.46   -58.02     +.68      .00      .00     -.05

MAIL     Report               2681   4.5983    .0218    .0043   4.5764    .2423    .0000        0    .0000        0   165568
MAIL     Baseline             1962   4.6224    .0230    .0035   4.5993    .5486    .0000        0    .0000        0   165555
         Delta                +719   -.0241   -.0012   +.0008   -.0229   -.3063    .0000        0    .0000        0      +13
         Change%            +36.65     -.52    -5.22   +22.32     -.50   -55.83      .00      .00      .00      .00     +.01
```

Figure 9-67 Transaction profile results

9.6.2 Rerunning the PA threadsafe reports

An alternative to running the profiling reports is to rerun the CICS PA reports at the start of this exercise against the newly captured SMF data. Follow the instructions in "Identifying candidates and capturing a baseline" on page 254 to reproduce the four reports.

9.6.3 Analyzing the PA reports

Figure 9-68 shows the CPU report. This report shows that the average response time of the QR only application (such as transactions /FOR and DE1) improved. For example, the average response for transaction DE1 went from 0.1868 to 0.0589. The main saving is in the Suspend time and the Dispatch Wait time, which shows that we are getting more throughput on the OR TCB by moving work to the L8 TCBs. The Transaction Profile report in Figure 9-67 on page 308 shows the same results.

```
V3R2M0                                              CICS Performance Analyzer
                                                       Performance Summary
                                              ____________________________________________
SUMM0001 Printed at 12:55:17 11/27/2011     Data from 12:26:51 11/27/2011 to 12:51:54 11/27/2011
CICS TCB CPU Analysis - Summary
```

Tran	#Tasks	Avg Response Time	Max Response Time	Avg Dispatch Time	Avg User CPU Time	Avg Suspend Time	Avg DispWait Time	Avg QR CPU Time	Avg MS CPU Time	Avg KY8 CPU Time	Avg KY9 CPU Time	Avg RO CPU Time
/FOR	6164	.0189	.4954	.0013	.0003	.0176	.0001	.0003	.0000	.0000	.0000	.0000
DE1	733	.0589	2.9162	.0032	.0011	.0558	.0089	.0010	.0000	.0000	.0000	.0000
DE20	733	.0799	.6465	.0027	.0014	.0772	.0098	.0013	.0000	.0000	.0000	.0000
DLE1	6636	.2623	226.7467	.1634	.0027	.0989	.0457	.0008	.0000	.0019	.0000	.0000
DLE2	6634	.1589	226.3567	.0839	.0012	.0750	.0228	.0005	.0000	.0007	.0000	.0000
DLE3	6635	.2332	226.3793	.1532	.0019	.0800	.0302	.0006	.0000	.0013	.0000	.0000
DLE4	6636	.2538	226.5894	.1719	.0016	.0819	.0333	.0006	.0000	.0010	.0000	.0000
DLE5	6636	.2607	226.8339	.1706	.0019	.0901	.0415	.0007	.0000	.0012	.0000	.0000
HR1	247	.0570	.5612	.0030	.0009	.0540	.0073	.0008	.0001	.0000	.0000	.0000
HR2	246	.0376	.2444	.0023	.0007	.0353	.0034	.0007	.0000	.0000	.0000	.0000
IT1	230	.0176	.1986	.0022	.0006	.0154	.0006	.0006	.0000	.0000	.0000	.0000
IT2	535	.0455	.6360	.0036	.0019	.0419	.0112	.0019	.0000	.0000	.0000	.0000
IT8	732	.1376	2.9378	.0047	.0024	.1329	.0253	.0024	.0000	.0000	.0000	.0000
MAIL	2961	4.5344	4.7241	.0197	.0033	4.5147	.2142	.0015	.0000	.0019	.0000	.0000
OE1	426	.0236	.4802	.0023	.0007	.0213	.0015	.0007	.0000	.0000	.0000	.0000
OE2	423	.0751	.6734	.0028	.0011	.0724	.0079	.0011	.0000	.0000	.0000	.0000
OE4	418	.0258	.4962	.0033	.0014	.0225	.0018	.0013	.0000	.0000	.0000	.0000
OE5	413	.3019	2.7909	.0072	.0040	.2947	.0446	.0040	.0000	.0000	.0000	.0000
PA2	698	.0177	.2538	.0012	.0003	.0164	.0001	.0003	.0000	.0000	.0000	.0000
PS2	899	.0229	.4998	.0028	.0014	.0201	.0018	.0014	.0000	.0000	.0000	.0000
PS3	834	.0264	1.1468	.0040	.0021	.0224	.0026	.0021	.0001	.0000	.0000	.0000
SC2	255	.2319	.8985	.0060	.0027	.2259	.0368	.0026	.0001	.0000	.0000	.0000
SC6	910	.0745	.6151	.0026	.0010	.0720	.0089	.0010	.0000	.0000	.0000	.0000
TS1	358	.0191	.4930	.0013	.0003	.0178	.0002	.0003	.0000	.0000	.0000	.0000
TXM0	1	.0177	.0177	.0177	.0026	.0001	.0000	.0006	.0020	.0000	.0000	.0020
TXM1	**505**	**2.0735**	**11.6065**	**.0310**	**.0260**	**2.0426**	**.0159**	**.0014**	**.0055**	**.0191**	**.0000**	**.0000**
TXM2	**501**	**2.1226**	**11.5111**	**.0310**	**.0255**	**2.0917**	**.0154**	**.0009**	**.0055**	**.0191**	**.0000**	**.0000**
TXM3	**498**	**2.1420**	**11.4446**	**.0306**	**.0263**	**2.1114**	**.0158**	**.0017**	**.0055**	**.0190**	**.0000**	**.0000**
TXM4	**493**	**2.0994**	**11.2503**	**.0291**	**.0253**	**2.0703**	**.0137**	**.0008**	**.0055**	**.0190**	**.0000**	**.0000**
TXM5	**490**	**2.0727**	**11.6526**	**.0291**	**.0250**	**2.0435**	**.0144**	**.0005**	**.0055**	**.0190**	**.0000**	**.0000**
Total	53880	.5039	226.8339	.0947	.0028	.4092	.0354	.0008	.0003	.0017	.0000	.0000

Figure 9-68 TCB CPU Summary report (after)

This report also shows that the response for the MAIL transaction improved slightly. However, the response time for the TXM* transactions (our DB2 and File I/O application) doubled. This result shows that making programs THREADSAFE is not a straightforward task and needs tools, such as CICS PA, to assist at all stages.

We must investigate the reasons behind this result by using CICS PA reports. The CICS PA TCB summary report (Figure 9-69) shows that the number of TCB swaps doubled for TXM* transactions, which is the opposite of what we expected to see, which is a reduction.

```
V3R2M0                                                   CICS Performance Analyzer
                                                             Performance Summary
                                          ________________________________________________
SUMM0001 Printed at 13:01:03 11/27/2011     Data from 12:26:51 11/27/2011 to 12:51:54 11/27/2011
CICS TCB Usage and Delays
```

Tran	#Tasks	Avg TCBAtach Count	Avg DSTCBHWM Count	Max DSTCBHWM Count	Avg DSCHMDLY Count	Max DSCHMDLY Count	Avg MaxJTDly Count	Avg MaxOTDly Count	Avg MAXSTDLY Count	Avg MAXTTDLY Count	Avg MAXXTDLY Count	Avg KY8 Disp Count
/FOR	6164	0	0	0	0	6	0	0	0	0	0	0
DE1	733	0	0	0	0	42	0	0	0	0	0	0
DE20	733	0	0	0	0	42	0	0	0	0	0	0
DLE1	6636	0	1	1	37	40	0	0	0	0	0	18
DLE2	6634	0	1	1	17	20	0	0	0	0	0	9
DLE3	6635	0	1	1	30	32	0	0	0	0	0	15
DLE4	6636	0	1	1	26	28	0	0	0	0	0	13
DLE5	6636	0	1	1	38	46	0	0	0	0	0	19
HR1	247	0	0	0	0	26	0	0	0	0	0	0
HR2	246	0	0	0	0	6	0	0	0	0	0	0
IT1	230	0	0	0	0	6	0	0	0	0	0	0
IT2	535	0	0	0	0	24	0	0	0	0	0	0
IT8	732	0	0	0	0	46	0	0	0	0	0	0
MAIL	2961	0	1	1	18	28	0	0	0	0	0	11
OE1	426	0	0	0	0	26	0	0	0	0	0	0
OE2	423	0	0	0	0	6	0	0	0	0	0	0
OE4	418	0	0	0	0	26	0	0	0	0	0	0
OE5	413	0	0	0	0	6	0	0	0	0	0	0
PA2	698	0	0	0	0	6	0	0	0	0	0	0
PS2	899	0	0	0	0	26	0	0	0	0	0	0
PS3	834	0	0	0	0	64	0	0	0	0	0	0
SC2	255	0	0	0	0	24	0	0	0	0	0	0
SC6	910	0	0	0	0	26	0	0	0	0	0	0
TS1	358	0	0	0	0	6	0	0	0	0	0	0
TXM0	**1**	**0**	**0**	**0**	**2**	**2**	**0**	**0**	**0**	**0**	**0**	**0**
TXM1	**505**	**0**	**1**	**1**	**410**	**416**	**0**	**0**	**0**	**0**	**0**	**304**
TXM2	**501**	**0**	**1**	**1**	**409**	**410**	**0**	**0**	**0**	**0**	**0**	**304**
TXM3	**498**	**0**	**1**	**1**	**409**	**412**	**0**	**0**	**0**	**0**	**0**	**304**
TXM4	**493**	**0**	**1**	**1**	**409**	**410**	**0**	**0**	**0**	**0**	**0**	**304**
TXM5	**490**	**0**	**1**	**1**	**409**	**410**	**0**	**0**	**0**	**0**	**0**	**304**
Total	53880	0	0	1	38	416	0	0	0	0	0	23

Figure 9-69 TCB Summary report (after)

We can use additional CICS PA reports or the CICS PA Explorer to see the TCB switches that are happening. Figure 9-70 on page 311 shows that the TXM* transactions are now spending most of the time on the L8 TCB and a miscellaneous TCB. The miscellaneous TCB is the socket TCB that is used in CICS TS V4.2 to perform the SEND and RECEIVE of a remote EXEC CICS command over an IPCONN.

For each command, such as an **EXEC CICS READ FILE** command to a remote region using IPCONN, we see the following swaps:

- From L8 to SO for the send
- From SO to L8 for the send
- From L8 to SO for the receive
- From SO to L8 for the receive

However, a swap from L8 to SO TCB and back is not as expensive as a swap to QR. We are still freeing up the constraint on the QR TCB. Therefore, this swap cannot be the cause of the increase in response times for these transactions.

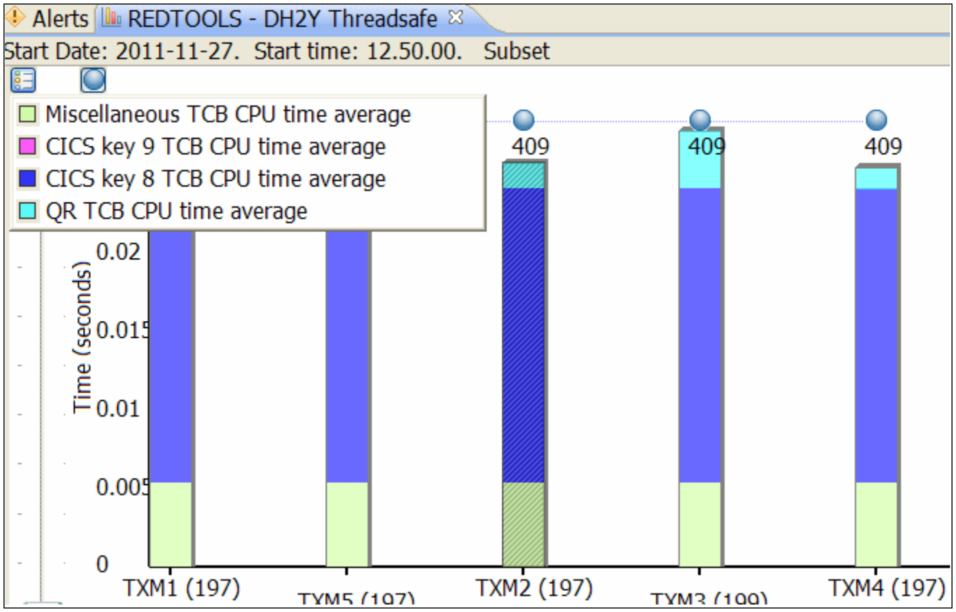


Figure 9-70 CICS PA Explorer: Threadsafe bar chart for TXM transactions*

Next we look at the Wait time Analysis report for the TXM1 transaction. This report provides more information about what is causing the increase in response time. As shown in Figure 9-71 on page 312, the increase is in the suspend time that is reported as “Other WAIT time”.

Without further investigation, which is beyond the scope of this book, we do not know whether our threadsafe changes or an external influence at the time of capturing the data caused the unknown WAIT time.

```
Tran=TXM1
  Summary Data                                             -------- Time ---------    ------ Count ------
                                                                 Total     Average       Total    Average
    # Tasks                                                                                505
    Response Time                                             1047.1259     2.0735
    Dispatch Time                                               15.6358     0.0310      259461      513.8
    CPU Time                                                    13.1345     0.0260      259461      513.8
    Suspend Wait Time                                         1031.4901     2.0426      259461      513.8
    Dispatch Wait Time                                           8.0210     0.0159      258956      512.8
      QR TCB Redispatch Wait Time                                1.4304     0.0028        3388        6.7
    Resource Manager Interface (RMI) elapsed time                3.2684     0.0065       53022      105.0
    Resource Manager Interface (RMI) suspend time                0.0000     0.0000           0        0.0

  Suspend Detail                                           ------------------- Suspend Time -------------
                                                                 Total     Average   %age  Graph
    N/A      Other Wait Time                                   953.2836     1.8877  92.4% |**************
    ISIOWTT  IPCONN link wait time                              69.3037     0.1372   6.7% |*
    DSCHMDLY Redispatch wait time caused by change-TCB mode      5.2426     0.0104   0.5% |
    DSPDELAY First dispatch wait time                            3.3951     0.0067   0.3% |
    GVUPWAIT Give up control wait time                           0.2360     0.0005   0.0% |
    LMDELAY  Lock Manager (LM) wait time                         0.0247     0.0000   0.0% |
    JCIOWTT  Journal I/O wait time                               0.0043     0.0000   0.0% |
```

Figure 9-71 Wait time analysis for TXM1

Finally, we revisit the “Worst Top 20 transactions causing TCB swaps” as shown in Figure 9-72.

```
 V3R2M0                                            CICS Performance Analyzer
                                                   Performance List Extended
                                             __________________________________________________
LSTX0001 Printed at 13:01:17 11/27/2011 Data from 12:26:51 11/27/2011 to 12:51:54 11/27/2011
Worst 20 transactions causing TCB swaps.

 Tran DSCHMDLY Userid     TaskNo Stop         Response Dispatch Dispatch User CPU  Suspend  Suspend DispWait
        Count                    Time           Time     Time     Count     Time     Time     Count    Time
 DLE1       40 CICSUSER      489 12:45:10.428    .0413    .0384       46    .0049    .0029       46    .0024
 DLE1       38 CICSUSER      702 12:45:11.977    .0164    .0087       43    .0025    .0077       43    .0071
 DLE2       20 CICSUSER      490 12:45:10.462    .0332    .0294       24    .0026    .0038       24    .0038
 DLE2       18 CICSUSER    52267 12:27:13.068  20.8679  19.3069       22    .1205   1.5610       22   1.4635
 DLE3       30 CICSUSER    52307 12:27:13.492  21.2029  19.2110       33    .1100   1.9919       33   1.9072
 DLE3       30 CICSUSER    52269 12:27:13.501  21.2930  19.2924       33    .1101   2.0006       33   1.9040
 DLE4       26 CICSUSER    52308 12:27:13.881  21.5790  19.3674       29    .1215   2.2116       29   2.1371
 DLE4       26 CICSUSER      499 12:45:10.597    .0039    .0022       27    .0011    .0017       27    .0017
 DLE5       46 CICSUSER      742 12:45:12.204    .0111    .0033       47    .0018    .0078       47    .0076
 DLE5       46 CICSUSER      928 12:45:13.244    .0315    .0029       47    .0019    .0286       47    .0117
 MAIL       28 CICSUSER    26230 12:47:15.899   4.5360    .0185       60    .0041   4.5175       60    .1954
 TXM1      410 CICSUSER      282 12:45:01.802    .1810    .0350      513    .0273    .1459      513    .0165
 TXM1      410 CICSUSER      286 12:45:02.310    .0893    .0296      512    .0252    .0597      512    .0086
```

Figure 9-72 Worst top 20 transactions causing TCB swaps

In this report, IMS transactions DLE1 through DLE5 are shown in the list as candidate items for being made threadsafe. We must now use CICS IA to investigate these transactions as explained in “Analyzing the program behavior” on page 268.

Part 3

Performance and general questions

This part explains performance indicators, presents the results from benchmark tests for a threadsafe conversion, and answers frequently asked questions about threadsafe.

This part includes the following chapters:

© Copyright IBM Corp. 2004, 2012. All rights reserved.

10

Performance case studies

This chapter documents the results of benchmark comparisons performed by IBM for applications running on CICS Transaction Server for z/OS (CICS TS) V3.2, and using DB2, WebSphere MQ (WMQ), and EXEC CICS file control calls. It compares the benefits obtained when redefining such applications from quasi-reentrant (QR) to threadsafe. In these examples, the applications are already analyzed and written to threadsafe standards. Therefore, you do not need to consider such issues as shared storage areas or serialization techniques.

This chapter includes the following sections:

- CICS DB2 and file control application
- CICS WMQ and file control application

© Copyright IBM Corp. 2004, 2012. All rights reserved.

10.1 CICS DB2 and file control application

The CICS DB2 and file control application test involved driving a transaction that linked an initial (quasi-reentrant) COBOL program EXEC CICS LINK to another COBOL application program. The second application then performed various DB2 and CICS commands in the following sequence:

- **`EXEC CICS READ`**
- **`EXEC SQL OPEN`**
- **`EXEC SQL FETCH`**
- **`EXEC CICS ASKTIME`**
- **`EXEC SQL UPDATE`**
- **`EXEC SQL CLOSE`**

> **Threadsafe commands:** The application logic involves EXEC CICS commands that are threadsafe in CICS TS V3.2, and therefore, can run under an open task control block (TCB) or QR TCB. In the same way, EXEC SQL calls to DB2 also run under open (L8) TCBs.

This application looped internally 100 times when linked to. Therefore, this series of commands was issued 100 times per task.

Testing involved defining the second program as **`CONCURRENCY(QUASIRENT) API(CICSAPI)`** and then redefining it as **`CONCURRENCY(THREADSAFE) API(OPENAPI)`**. In both cases, performance and diagnostic data was gathered to provide metrics for comparative results, including CICS Performance Analyzer (CICS PA) reports, CICS statistics, RMF data and CICS auxiliary trace.

Because the CICS system was not using storage protection for these tests, **`STGPROT=NO`** was specified.

> **Purpose of this example:** This example quantifies benefits when redefining a program (that is or was made a good threadsafe candidate application) as threadsafe to CICS. This example does not have the following intents:
>
> - Demonstrate issues with serialization of shared data.
> - Demonstrate performance problems with TCB switching due to interleaved threadsafe and nonthreadsafe EXEC CICS commands.
> - Reveal performance issues when switching between L9 and L8 TCBs for OPENAPI programs that are defined with **`EXECKEY(USER)`** and that issue calls to DB2.

The application can be considered a good candidate for being redefined as threadsafe in CICS TS V3.2 for the following reasons:

- It includes EXEC SQL calls to DB2, which require an L8 TCB.
- It has EXEC CICS commands that are threadsafe, and therefore, that have no affinity to a TCB environment.
- It does not interleave threadsafe and nonthreadsafe commands.

This type of application can be a good model for one that was prepared for threadsafe use before migrating to CICS TS V3.2. Before that release, **`EXEC CICS READ`** commands were nonthreadsafe and **`EXEC CICS ASKTIME`** commands were threadsafe. Therefore, the application is already structured to separate its nonthreadsafe and threadsafe work and can avoid TCB switching where possible. It has good construction in regard to threadsafety.

10.1.1 Environment

Performance testing was performed on a dedicated IBM test system to provide comparable results. The environment had the following hardware and software:

- IBM eServer™ zSeries® 990 (z990) 2084-303 with three dedicated central processors (CPs)
- z/OS Version 1.7
- CICS TS V3.2
- CICS PA V2.1
- DB2 Version 7.1

10.1.2 Results

Two sets of results were obtained:

- When the application was defined as **`CONCURRENCY(QUASIRENT) API(CICSAPI)`**
- After it was redefined as **`CONCURRENCY(THREADSAFE) API(OPENAPI)`**

CICS PA helped to investigate the CPU usage and response times for the application and to compare the number of invocations of the transaction (CICS tasks) that ran for the tests. Resource Measurement Facility (RMF) workload activity was used to review the total CPU usage, transaction rates, and internal response time for the comparison tests. In addition, a review of the CICS auxiliary trace taken during the tests was used (optional) to verify the TCB switching activity taking place when the transactions were running.

Figure 10-1 shows the results from CICS PA when comparing the characteristics of the transaction for a quasirent definition and a threadsafe definition of the main application. As shown in the figure, the average user CPU time and average dispatch time were reduced after the program was redefined because of the reduction in TCB switches that occurred when the program was redefined as threadsafe.

```
CICS Performance Analyzer
Performance Summary

                Avg      Avg               Avg     Avg      Avg      Avg
           Tran User CPU Dispatch  #Tasks  L8 CPU  L9 CPU   Response DSCHMDLY
                Time     Time              Time    Time     Time     Count
Results when
quasirent: FCDB .010710  .020827   2313    .007678 .000000  .031925  9050757

Results when
threadsafe: FCDB .008057 .013630   3452    .007944 .000000  .014356  1806209
```

Figure 10-1 Comparison of transaction performance between a quasirent and threadsafe definition

Previously, each EXEC SQL command required a switch from QR to L8 for the duration of the call to DB2, followed by a switch from L8 back to QR upon return to CICS. With 100 iterations of the loop within the application, 800 switches resulted for the EXEC SQL calls and 2 switches resulted for the end-of-task sync point flows to DB2. When the program was redefined as **CONCURRENCY(THREADSAFE) API(OPENAPI)**, two switches resulted for the **EXEC CICS LINK** command to the second program, and two switches resulted for the syncpoint flows to DB2. The switch from QR to L8 on the link to the second program occurred because it was defined as **API(OPENAPI)**, and therefore, ran under an open TCB. Likewise, the switch back from L8 to QR on the return from the link occurred because the top-level linking program was still defined as quasirent.

The results also showed that the comparison is more favorable when the program was redefined as threadsafe, because more than 1000 additional tasks were able to be run within the test time frame.

The average L8 CPU time increased when the application was redefined as threadsafe. However, this result was more than countered by the reduction in QR TCB CPU usage, as reflected in the total value shown by the average user CPU time. L9 TCB CPU usage was 0 because storage protection was not active, and therefore, the execution key of the application was not pertinent. An L8 TCB can be used instead.

The average response time for using the threadsafe application was less than half of the quasirent version.

Finally, the DSCHMDLY value (redispatch wait time caused by a change mode to switch TCBs) was reduced by 80%, which is a direct reflection that far fewer TCB switches occurred.

Figure 10-2 shows the results from CICS PA when comparing the DB2 performance characteristics of the transaction for a quasirent definition and a threadsafe definition of the main application. Similar to the results shown in Figure 10-1 on page 318, the response time reduced when redefining the application as threadsafe. The same result as in Figure 10-1 on page 318 is true for the average user CPU time.

```
CICS Performance Analyzer
DB2 - Long Summary

                                          Avg       Max       Avg       Max       Avg      Max
         Tran/  Program/  #Tasks/   DB2Rqst   DB2Rqst   UserCPU  UserCPU  Response Response
         SSID   Planname  #Threads    Count     Count      Time      Time      Time     Time

Results when quasirent:
         FCDB  FCDB2001   2313        400.0      400   .010710   .012918     .0319    .1539

         DF2A  DB9A       2313       Thread Utilization      Entry=    0    Pool=   2313
                                     Class1: Thread Time Avg: Elapsed=  .0312   CPU= .008933
                                                         Max: Elapsed=  .1533   CPU= .011068

Results when threadsafe:
         FCDB  FCDB2001   3452        400.0      400   .008056   .018088     .0144    .0546

         DF2A  DB9A         3452      Thread Utilization     Entry=    0    Pool=   3452
                                      Class1: Thread Time Avg: Elapsed= .0126  CPU= .007693
                                                          Max: Elapsed= .0462  CPU= .017703
```

Figure 10-2 Comparison of DB2 performance activity for a quasirent definition and a threadsafe definition

Figure 10-3 shows the results from RMF workload activity when comparing the CPU and throughput characteristics of the transaction for a quasirent definition and a threadsafe definition of the main application. The transaction rate increased from 49.40 per second up to 76.99 per second. This result occurred because using L8 TCBs allowed parallel processing to use multiple CPs in the hardware and increased the transaction throughput as a result. Likewise, the transaction time reduced from 49 to 13 seconds. The CPU time is reduced, reflecting the reduction in TCB switches when redefining the program as threadsafe.

```
W O R K L O A D   A C T I V I T Y

z/OS V1R7     SYSPLEX PLEX3      DATE 07/10/2007      INTERVAL 00.45.546      MODE = GOAL
RPT VERSION V1R7 RMF             TIME 11.51.14

Results when quasirent:

REPORT BY: POLICY=POLICY

   TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT    ---SERVICE----  SERVICE TIMES  ---APPL %---
   AVG      1.00  ACTUAL            36.024    IOC       160   CPU     30.9   CP     68.19
   MPL      1.00  EXECUTION         36.024    CPU     6209K   SRB      0.1   AAPCP   0.00
   ENDED       2  QUEUED                 0    MSO     2663M   RCT      0.0   IIPCP   0.00
   END/S    0.04  R/S AFFIN              0    SRB     26214   IIT      0.0

   TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT
   AVG      0.00  ACTUAL                49
   ENDED    2250
   END/S   49.40



Results when threadsafe:

REPORT BY: POLICY=POLICY

   TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT    ---SERVICE----  SERVICE TIMES  ---APPL %---
   AVG      1.00  ACTUAL            38.026    IOC         0   CPU     28.4   CP     62.73
   MPL      1.00  EXECUTION         38.026    CPU     5696K   SRB      0.2   AAPCP   0.00
   ENDED       2  QUEUED                 0    MSO     2444M   RCT      0.0   IIPCP   0.00
   END/S    0.04  R/S AFFIN              0    SRB     38026   IIT      0.0


   TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT
   AVG      0.00  ACTUAL                13
   ENDED    3506
   END/S   76.99
```

Figure 10-3 Comparison of RMF workload activity for a quasirent definition and a threadsafe definition

Figure 10-4 shows the output from the DFHSTUP CICS statistics utility program, comparing TCB activity when the application was defined first as quasi-reentrant and then as threadsafe.

```
CICS TCB Mode Statistics
________________________


TCB    < TCBs Attached >    TCB     Attach       MVS       Accumulated          Accumulated          Accumulated
Mode   Current       Peak  Attaches  Failures     Waits    Time in MVS wait      Time Dispatched        Time / TCB
______________________________________________________________________________________________________________________

Results when quasirent:

 QR          1          1         0          0    538040      00:00:35.590807      00:00:10.394348      00:00:10.552730
 L8         81         81        71          0    916768      00:14:08.888409      00:01:19.872928      00:00:20.877639


Results when threadsafe:

 QR          1          1         0          0      7474      00:00:44.388717      00:00:01.596551      00:00:00.637270
 L8         10         10         0          0      7333      00:06:58.286155      00:00:45.715037      00:00:28.074048



TRANSACTION MANAGER STATISTICS
______________________________


Results when quasirent:

Peak number of active user transactions        :          82
Total number of active user transactions       :        2287


Results when threadsafe:

Peak number of active user transactions        :          12
Total number of active user transactions       :        3547
```

Figure 10-4 Comparison of CICS statistics data for a quasirent definition and a threadsafe definition

In the quasi-reentrant case, both the QR and L8 TCBs entered more MVS waits than in the threadsafe case. For the L8 TCBs, the accumulated time spent in MVS waits was over twice as long as for the threadsafe case. The quasirent workload required a peak of 81 L8 TCBs to accommodate the transactions, where the threadsafe workload peaked at 10 L8 TCBs. This result occurred because, in the quasi-reentrant case, work built up in the CICS system as tasks were attached and competed for subdispatch processing under the QR TCB. This build up led to a higher peak of user transactions in the system (82 compared to 12).

Because L8 TCBs can be reused only after their owning task has completed, more L8 TCBs needed to be attached to accommodate the additional concurrently attached tasks as they issued their interleaving EXEC SQL calls to DB2. The higher number of L8 TCBs, coupled with the greater number of TCB switches between them and the QR TCB in the quasi-reentrant case, led to the L8 TCBs experiencing more MVS waits than in the threadsafe case. Because there were more occasions when no further work had to be performed, control was relinquished back to the operating system.

The total accumulated time for the TCBs was lower in the threadsafe case, which reflects the fewer TCB switches that were required.

Because fewer peak L8 TCBs were required in the threadsafe case, the need for below-the-line storage was reduced as a result, assisting with virtual storage constraint relief for this workload.

10.2 CICS WMQ and file control application

The CICS WMQ and file control application test involved driving a transaction that linked an initial (quasi-reentrant) COBOL program **EXEC CICS LINK** to another COBOL application program. The second application then performed various WebSphere MQ and CICS commands in the following sequence:

- **EXEC CICS READ**
- **WMQ PUT**
- **WMQ GET**

This application looped internally 100 times when linked to. Therefore, this series of commands was issued 100 times per task. In addition, an MQOPEN was issued before the loop, and an MQCLOSE was issued after the loop completed.

Testing involved defining this second program first as **CONCURRENCY(QUASIRENT) API(CICSAPI)** and then redefining it as **CONCURRENCY(THREADSAFE) API(OPENAPI)**. In both cases, performance and diagnostic data was gathered to provide metrics for the comparative results, including CICS PA reports, CICS statistics, RMF data and CICS auxiliary trace.

Because the CICS system was not using storage protection for these tests, **STGPROT=NO** was specified.

The purpose of this test was not to demonstrate serialization issues.

10.2.1 Environment

Performance testing was performed on a dedicated IBM test system to provide comparable results. The environment used the following hardware and software:

- z990 2084-303 with three dedicated CPs
- z/OS Version 1.7
- CICS TS V3.2
- CICS PA V2.1
- WebSphere MQ Version 6.1

10.2.2 Results

Two sets of results were obtained:

- When the application was defined as **CONCURRENCY(QUASIRENT) API(CICSAPI)**
- After it was redefined as **CONCURRENCY(THREADSAFE) API(OPENAPI)**

CICS PA helped to investigate the CPU usage and response times for the application and to compare the number of invocations of the transaction (CICS tasks) that ran for the tests. RMF workload activity was used to review the total CPU usage, transaction rates, and internal response time for the comparison tests. In addition, a review of the CICS auxiliary trace taken during the tests was used (optional) to verify the TCB switching activity taking place when the transactions were running.

Figure 10-5 shows the results from CICS Performance Analyzer when comparing the characteristics of the transaction for both a quasirent and a threadsafe definition of the main application.

```
CICS Performance Analyzer
Performance Summary

                Avg       Avg                 Avg      Avg      Avg       Avg      Avg
          Tran  User CPU  Dispatch   #Tasks   L8 CPU   L9 CPU   Response  DSCHMDLY DSCHMDLY
                Time      Time                Time     Time     Time      Time     Count
Results when
quasirent:  FCMQ  .011992   .014209    1500   .009574  .000000  .019020  .004250   7141728

Results when
threadsafe: FCMQ  .011003   .013148    1500   .010866  .000000  .015339  .000076    312592
```

Figure 10-5 Comparison of transaction performance between quasirent and threadsafe

As shown in Figure 10-5, the average user CPU time and average dispatch time were reduced after the program was redefined because of the reduction in TCB switches that occurred when the program was redefined as threadsafe.

Previously, each WMQ call required a switch from QR to L8 for the duration of the call to WebSphere MQ, followed by a switch from L8 back to QR upon return to CICS. With 100 iterations of the loop within the application, 400 switches resulted for the WMQ calls and 2 switches resulted for the end-of-task sync point flows to WebSphere MQ. When the program was redefined as **CONCURRENCY(THREADSAFE) API(OPENAPI)**, two switches occurred for the **EXEC CICS LINK** command to the second program, and two switches occurred for the syncpoint flows to WebSphere MQ. The switch from QR to L8 on the link to the second program occurred because it was defined as **API(OPENAPI)** and, therefore, ran under an open TCB. Likewise, the switch back from L8 to QR on the return from the link occurred because the top-level linking program was still defined as quasirent.

The reduction in the average user CPU time and average dispatch time was less marked than in the case of the file control or DB2 application. The WebSphere MQ application only issued two WMQ calls (**WMQ PUT** and **WMQ GET**) within the scope of its loop. Four EXEC SQL calls were in the file control or DB2 example program. Therefore, the CPU benefits of remaining on an L8 TCB and the reduction in TCB switching are less marked in the WebSphere MQ example than in the DB2 example. This result is another indication of the scalability of benefits that threadsafe exploitation brings: The more an application must drive an open transaction environment (OTE)-enabled task-related user exit (TRUE), such as for DB2 or WMQ calls, the more the savings occur if that application is suitable for redefining as a threadsafe program.

The average L8 CPU time increase when the application was redefined as threadsafe. As with the DB2 tests in 10.1, “CICS DB2 and file control application” on page 316, L9 TCB CPU usage was 0 because storage protection was not active, and therefore, the execution key of the task-related user exit was not pertinent. Instead, an L8 TCB can be used.

The average response time for using the threadsafe application was reduced compared with the response time of the quasi-reentrant version.

Finally, the DSCHMDLY value (redispatch wait time caused by a change mode to switch TCBs) and the DSCHMDLY count (number of TCB switches) were reduced by orders of magnitude. This result is a direct reflection of the fact that fewer TCB switches occurred after the application was redefined as **CONCURRENCY(THREADSAFE) API(OPENAPI)**.

Figure 10-6 shows the results from CICS PA when comparing the WebSphere MQ performance characteristics of the transaction for a quasirent definition and a threadsafe definition of the main application. CPU usage was reduced when redefining the application as threadsafe.

```
CICS Performance Analyzer
WebSphere MQ Class 1 Summary
                                                     ----- Average -----
                      SSID  APPLID    TRAN    Count    CPU         Calls

Results when
quasirent:            VICC  IYCUZC19  FCMQ    15282    0.007768    200.0

Results when
threadsafe:           VICC  IYCUZC19  FCMQ     6088    0.007634    200.0
```

Figure 10-6   Comparison of WMQ performance activity between quasirent and threadsafe

Figure 10-7 shows the output from the **DFHSTUP** CICS statistics utility program for comparing TCB activity when the application was defined first as quasi-reentrant and then as threadsafe.

```
CICS TCB Mode Statistics
________________________

TCB    < TCBs Attached >    TCB    Attach       MVS     Accumulated         Accumulated         Accumulated
Mode   Current       Peak Attaches Failures     Waits   Time in MVS wait    Time Dispatched       Time / TCB
______________________________________________________________________________________________________________

Results when quasirent:

 QR         1          1        0         0    264838     00:00:40.827822     00:00:05.157484     00:00:05.228039
 L8        81         81       71         0    312696     00:01:13.668300     00:00:18.511194     00:00:15.920050


Results when threadsafe:

 QR         1          1        0         0      5305     00:00:45.505023     00:00:00.480204     00:00:00.340059
 L8        10         10        0         0      4651     00:01:12.057051     00:00:19.912177     00:00:16.667971


TRANSACTION MANAGER STATISTICS
______________________________

Results when quasirent:

Peak number of active user transactions       :           6
Total number of active user transactions      :        1535


Results when threadsafe:

Peak number of active user transactions       :           8
Total number of active user transactions      :        1535
```

Figure 10-7   Comparison of CICS statistics data between a quasirent definition and a threadsafe definition

In the quasi-reentrant case, the QR and L8 TCBs entered more MVS waits than in the threadsafe case. The total accumulated time for the TCBs was lower in the threadsafe case, which reflects the fewer TCB switches that were required.

The quasi-reentrant workload required a peak of 81 L8 TCBs to accommodate the transactions, where the threadsafe workload peaked at 10 L8 TCBs. As in the DB2 tests in 10.1, “CICS DB2 and file control application” on page 316, these peaks resulted because (in the quasi-reentrant case) work built up in the CICS system as tasks were attached and competing for subdispatch processing under the QR TCB.

Because L8 TCBs can be reused only one time, with their owning task completed, more L8 TCBs needed to attach. In this case, the attachment occurred to accommodate the additional concurrently attached tasks as they issued their interleaving WMQ calls. The higher number of L8 TCBs, coupled with the greater number of TCB switches between them and the QR TCB in the quasi-reentrant case, led to the L8 TCBs experiencing more MVS waits than in the threadsafe case. Because there were more occasions when no further work had to be performed, control was relinquished back to the operating system.

The total accumulated time for the TCBs was lower in the threadsafe case, which reflects the fewer TCB switches that were required.

As with the DB2 example in 10.1, “CICS DB2 and file control application” on page 316, fewer peak L8 TCBs were required in the threadsafe case. Therefore, the need for below-the-line storage was reduced when the application was redefined as threadsafe.

11

Common threadsafe questions

This chapter answers several of the most frequently asked questions about threadsafe. The questions are divided into the following categories:

- CICS exits
- Defining applications as threadsafe
- Fields and parameters
- Load module scanner
- Nonthreadsafe CICS commands
- Performance
- TCB switching
- Threadsafe file control
- Using L8 or L9 TCBs

© Copyright IBM Corp. 2004, 2012. All rights reserved.

11.1 CICS exits

- **Question:** How do I determine which exits to use and whether they are defined as threadsafe?

  Answer: Use the sample `DFHOSTAT` utility supplied by CICS to examine user exits. This tool reports which exit programs are active and the concurrency setting of the exit program. The report includes any exits that are supplied by third-party vendors in support of their products.

- **Question:** If my exits are for a vendor product, can I define them as threadsafe and improve my performance?

  Answer: No, you must contact the vendor and have them tell you whether it is safe to change the concurrency attribute for the program definition of the exit.

11.2 Defining applications as threadsafe

- **Question:** Can I define all of my applications as threadsafe?

  Answer: No, you must perform a full analysis of each application before making the definition change. Otherwise, you can compromise the shared data of application. You might also see a performance degradation due to excessive task control block (TCB) switches caused by nonthreadsafe CICS commands and nonthreadsafe user exits. Changing only the definition of a program is not enough.

- **Question:** If my application is reentrant, can I define it as threadsafe?

  Answer: No, reentrancy is one aspect of being threadsafe. You must check whether the application accesses any shared resources. If it does, you must check whether it has the necessary serialization logic in place. An application can be reentrant, can be link-edited with RENT, and can be in a CICS read-only DSA. However, if it incorrectly accesses shared data without serialization logic, then it is nonthreadsafe.

- **Question:** What is the difference between a threadsafe program and an OPENAPI program in CICS TS V3?

  Answer: A *threadsafe program* is defined as `CONCURRENCY(THREADSAFE)` and `API(CICSAPI)`. It can run on QR TCB or an open TCB. Part of it might run on QR TCB, and then after a DB2 or WMQ request, part of it can run on an open TCB. A threadsafe program has no TCB affinity and no affinity to the key of the TCB. Use of APIs that are not supplied by CICS is not allowed, because they can run on QR TCB and, therefore, damage the CICS environment.

An *OPENAPI program* is defined as **CONCURRENCY(THREADSAFE)** and **API(OPENAPI)**. It always runs on an open TCB. It starts on an open TCB, and all application code runs on an open TCB. If CICS must switch to QR TCB to run a nonthreadsafe CICS command, CICS switches back to the open TCB when it returns control to the program. An OPENAPI program runs on an open TCB whose key matches the execution key of the program, which is an L8 TCB for EXECKEY(CICS) or an L9 TCB for EXECKEY(USER). Use of APIs that are not supplied by CICS is allowed at your own risk, because they do not run on QR TCB and do not block main CICS processing.

- **Question:** Are any tools available that I can run to detect if my application code is threadsafe or to convert my application automatically?

  Answer: No, an automatic way to make your programs threadsafe is not available. CICS provides the load module scanner that you can use as a starting point in analyzing your application. The load module scanner helps you to identify commands that can cause your application code to be nonthreadsafe and to identify CICS commands that are nonthreadsafe and will cause a switchback to the QR TCB. CICS Tools are available to assist you in your threadsafe project (see Appendix A, "Overview of CICS Tools" on page 337).

- **Question:** What happens if I define an application program as threadsafe to CICS when it is not threadsafe?

  Answer: CICS cannot protect you from such consequences, and the results are unpredictable. You risk the integrity of the shared data because multiple instances of the program, each running on its own TCB, can access the data at the same time. No protection is provided by using quasi-reentrancy because the application is not running on the QR TCB. The loss of data integrity might not be instantly detected, but can become apparent later. This scenario is similar to someone who discovers a storage overwrite long after it occurred.

11.3 Fields and parameters

- **Question:** Can I still address the task control area (TCA) of a task by using the CSACDTA field?

  Answer: No, with the introduction of OTE, you can no longer assume that the TCA address held within CSACDTA is the TCA of the task that is accessing the CSA. CSACDTA contains the address of the task that is currently dispatched *under the QR TCB*. The task that is examining the value in CSACDTA might be running under an open TCB, possibly leading to the wrong TCA address being used by the program, with unpredictable results.

Use the CICS system programming interface (SPI) whenever possible for programs that need to access state information about a task.

CSACDTA was renamed CSAQRTCA in CICS Transaction Server for z/OS (CICS TS) V3.1 to further discourage using the CSA to address the TCA of the running task. In CICS TS V3.2, IBM withdrew the ability to reference a TCA using this field, by loading CSAQRTCA with the address of an area of fetch-protected storage. The result is an abend ASRD with message `DFHSR0618` if it is referenced.

- **Question:** If SUBTSKS is specified so that CICS uses the CO TCB for concurrent VSAM calls on busy systems, is this parameter still honored for file control requests that are issued under an open TCB?

  Answer: No, if a CICS TS V3.2 application is running under an open TCB and issued an EXEC CICS file control command, do not switch to another TCB to process the request. The **`SUBTSKS`** SIT parameter is honored by CICS only if the application is running on the QR TCB when the file control command is issued.

11.4 Load module scanner

- **Question:** Are the commands listed in DFHEIDTH table nonthreadsafe commands?

  Answer: Many of the commands in the DFHEIDTH table are nonthreadsafe, but that is not the purpose of the table. The commands listed in the DFHEIDTH table give the application programmer access to shared storage. A potential exists for the application program code to be nonthreadsafe, unless serialization logic is implemented around updates to the shared storage. Therefore, the purpose of the DFHEIDTH table is to report programs that might contain nonthreadsafe code.

- **Question:** I ran the load module scanner **`DFHEISUP`** with DFHEIDTH table against my programs and found that they were using EXEC CICS ADDRESS common work area (CWA). However, when searching the code, I cannot find any reference to the CWA. Are these programs threadsafe?

  Answer: If the report from the **`DFHEISUP`** utility flags usage of a command that gives you access to shared storage, and you never reference the storage in question, there is no issue of threadsafety. Perhaps someone changed the code years ago but never removed the reference to the shared storage. If this issue is the only potential shared storage issue reported, your program is threadsafe.

- **Question:** Is there a table that lists all nonthreadsafe CICS commands?

  Answer: Yes, the DFHEIDNT table lists all nonthreadsafe CICS commands. Use of this table indicates whether you can experience excessive TCB switching due returning to QR TCB to run the nonthreadsafe CICS command. The table does not indicate anything about the application code and whether it is threadsafe. How much TCB switching will occur depends on the number of CICS commands and their relative position in the code to DB2 calls, WMQ calls, or both.

11.5 Nonthreadsafe CICS commands

- **Question:** What is a nonthreadsafe CICS command, and do such commands have a data integrity exposure?

  Answer: A *nonthreadsafe CICS command* is a CICS command that insists on running on QR TCB. The CICS code that implements the command relies on quasi-reentrancy, that is, serialization provided by running on the QR TCB. No, nonthreadsafe CICS commands do not have data integrity exposure, because serialization of shared resources is provided by QR TCB. However, a threadsafe CICS command is one in which the CICS code does not rely on running on QR TCB and can run safely on open TCBs concurrently.

- **Question:** Can I define a program as threadsafe if it contains nonthreadsafe CICS commands?

  Answer: Yes, by defining a program as threadsafe, you are informing CICS that the application code (for example, the COBOL source code) is threadsafe. You are not specifying to CICS which API commands the program uses. (CICS manages the threadsafe issues of its own code.) Nonthreadsafe EXEC CICS commands cause a switch back to the QR TCB, which affects the performance of the application, but not the integrity of your data.

  For a program defined as `CONCURRENCY(THREADSAFE) API(CICSAPI)`, after a nonthreadsafe CICS command is run, the program remains on the QR TCB until the next request to an OPENAPI task-related user exit (TRUE), such as a DB2 or WebSphere MQ request). For a program defined as `CONCURRENCY(THREADSAFE) API(OPENAPI)`, after a nonthreadsafe EXEC CICS command is run, the program receives control back on the open TCB (either L8 or L9).

11.6 Performance

- **Question:** I am planning to migrate to CICS TS V3 and am concerned about the potential performance impacts. Can I do the migration and then set FORCEQR to FORCE so that the system runs like my current CICS system?

  Answer: No, you must review and set your exits to threadsafe before you perform the migration to be safe. You cannot turn off the use of L8 open TCBs.

- **Question:** What is the cost of a TCB switch?

  Answer: The path length of a single TCB switch (such as from QR to L8) is approximately 2,000 instructions. Therefore, a nonthreadsafe application issuing an EXEC SQL call to DB2 incurs 4,000 additional instructions. Half of them occurs when switching from the QR TCB onto an L8 TCB to call DB2, and the other half occurs when switching back to the QR TCB upon return to CICS.

  You can see the benefits of being threadsafe when additional path length is scaled up by the number of calls to OPENAPI TRUEs such as DB2 and WebSphere MQ from within busy quasi-reentrant applications. In addition to the time required to start the TCB switches is the corresponding CPU cost, with the increased contention of using the QR TCB for nonthreadsafe application work.

11.7 TCB switching

- **Question:** Before CICS TS V2.2, did TCB switching occur for DB2 requests?

  Answer: Yes, for each DB2 request, two TCB switches occurred: one switch from QR TCB to a DB2 thread TCB before calling DB2 and one switch back to the QR TCB after the DB2 call completed. Activity on the DB2 thread TCB was not visible in a CICS trace.

- **Question:** Before CICS TS V3.2, did TCB switching occur for WebSphere MQ requests?

  Answer: Yes, for each WebSphere MQ request, two TCB switches occurred: one switch to a WMQ thread TCB before calling WebSphere MQ and one switch back to the original TCB after the WMQ call completed. Activity on the WMQ thread TCB was not visible in a CICS trace.

11.8 Threadsafe file control

- **Question:** What differences are observed when tasks are running in CICS TS V3.2 and file control commands are issued?

  Answer: CICS TS V3.2 supports threadsafe file control. Therefore, applications that are running on an open TCB can call VSAM under the open TCB as part of a file control request.

  In CICS TS V3.2 and earlier versions, file control is a nonthreadsafe EXEC CICS API. Therefore, all file control commands are processed under the QR TCB. If VSAM must suspend a task during its execution of a request to an LSR file, it drives the supplied UPAD exit in CICS, and the task is suspended by the CICS dispatcher. If the request is NSR, CICS issues the request to VSAM asynchronously and then suspends the task if needed. For example, tasks are suspended on FCIOWAITs or FCXCWAITs. The reason a task is suspended can be analyzed, for example, by using CEMT online or by investigating a CICS system dump offline.

  The IPCS system dump formatter is run against a system dump and returns, for example, the task environment (using the KE VERBEXIT) or the dispatcher environment (using the DS VERBEXIT). Because nonthreadsafe commands must run under the serialized QR TCB, only one task is seen as running on this TCB at any one time, and the KE VERBEXIT data identifies the running task at the time of the dump.

  In CICS TS V3.2, VSAM requests can run under an open TCB. Any suspends due to VSAM do not require calling the UPAD exit because blocking the TCB does not affect other tasks within CICS (unlike the effect that a blocking operation on the QR TCB has). Such requests do not result in the CICS dispatcher being started to suspend the task. Tasks still seem to be running when investigated by using such techniques as CEMT or IPCS, which can affect the analysis of task activity when using performance monitors or equivalent software.

11.9 Using L8 or L9 TCBs

- **Question:** Does a program defined with `CONCURRENCY(QUASIRENT)` calling DB2 6 or later use L8 TCBs?

  Answer: Yes, CICS always uses L8 TCBs with DB2 6 and later regardless of whether the application is threadsafe. For every DB2 call, the DB2 work is done on the L8 TCB, and after completion, CICS switches back to the QR TCB before returning to the nonthreadsafe application.

- **Question:** Does an application running on CICS TS V3.2 that calls WebSphere MQ use L8 TCBs?

  Answer: Yes, CICS TS V3.2 uses an open transaction environment (OTE)-enabled TRUE to handle CICS WMQ calls. L8 TCBs are used for the requests to WebSphere MQ. If the application is defined as threadsafe, control remains on the open TCB upon return from WebSphere MQ. If the application is defined as quasi-reentrant, control switches back to the QR TCB upon return from WebSphere MQ.

- **Question:** Can I stop using L8 TCBs by specifying **FORCEQR=YES** in the system initialization table (SIT) parameter?

  Answer: No, DB2 calls for DB2 V6 and later always switch to an L8 TCB. **FORCEQR=YES** overrides the **CONCURRENCY(THREADSAFE) API(CICSAPI)** setting for any program defined as such, forcing a switchback to the QR TCB after the DB2 call. **FORCEQR=YES** has no affect on a program defined as **CONCURRENCY(THREADSAFE) API(OPENAPI)** that must run on an open TCB.

- **Question:** Can I stop using L8 TCBs by specifying **FCQRONLY=YES** in the SIT?

  Answer: It depends. By using CICS TS V3.2, EXEC CICS file control requests can be processed under an open TCB. If your application is running under an L8 or L9 TCB when it issues a file control command, CICS runs this threadsafe API request under the open TCB. This same method is used for other threadsafe EXEC CICS commands.

  Implementation of file control threadsafety also provides the option to disable threadsafe support for EXEC CICS file control API commands by using the **FCQRONLY** SIT parameter. If this parameter is set to YES, file control commands are processed under the QR TCB within CICS, as in earlier releases. However, this setting does not remove support for and use of L8 TCBs, such as for WMQ or DB2 calls. Nor does this setting remove support nor use of L8 TCBS for programs defined with **CONCURRENCY(THREADSAFE) API(OPENAPI)**. Such programs must run their application logic under an L8 or L9 open TCB. **FCQRONLY** is specific to the execution path within CICS for EXEC CICS file control commands only.

- **Question:** If **STGPROT=NO** is specified, does CICS still need to use L9 TCBs for EXECKEY(USER) programs?

  Answer: No, if CICS is not using storage protection, open TCBs that match user key storage and execution are not necessary. L9 TCBs do not have to be used for **CONCURRENCY(THREADSAFE) API(OPENAPI) EXECKEY(USER)** programs. Instead, use L8 TCBs.

Part 4

Appendixes and related publications

This part includes additional information that supports the contents of this book. It includes the following information:

© Copyright IBM Corp. 2004, 2012. All rights reserved.

A

Overview of CICS Tools

Several CICS Tools can assist in threadsafe enablement, including the following tools:

- CICS Performance Analyzer for z/OS
- CICS Interdependency Analyzer for z/OS
- CICS Configuration Manager
- CICS VSAM Transparency performance on CICS Transaction Server V3.2, V4.1, and V4.2

© Copyright IBM Corp. 2004, 2012. All rights reserved.

CICS Performance Analyzer for z/OS

This section highlights the CICS Performance Analyzer (CICS PA) tool. It includes an overview of CICS PA, its components, and its purpose.

For more information about CICS PA, see the IBM Redbooks publication *CICS Performance Analyzer*, SG24-6063.

Overview of CICS PA

CICS PA provides comprehensive performance reporting and analysis for CICS Transaction Server for z/OS (CICS TS) and related subsystems, including DB2, WebSphere MQ (WMQ), IMS (Database Control (DBCTL)), and the z/OS System Logger. It provides information about the performance of your CICS systems and applications and helps to tune, manage, and plan your CICS systems effectively. CICS PA also provides a historical database facility to help you manage CICS statistics and performance data for your CICS transactions.

CICS PA produces reports and extracts by using data that is normally collected by your system in MVS System Management Facility (SMF) data sets:

- CICS Monitoring Facility (CMF) performance class, exception class, and transaction resource class data in SMF 110 records
- CICS statistics and server statistics data in SMF 110 records
- CICS Transaction Gateway (CICS TG) statistics data in SMF 111 records
- DB2 accounting data in SMF 101 records
- WebSphere MQ accounting data in SMF 116 records
- System Logger data in SMF 88 records
- IBM Tivoli OMEGAMON® XE for CICS on z/OS (OMEGAMON XE for CICS) data in SMF 112 records, containing transaction data for Adabas, CA Datacom, CA IDMS, and Supra database management systems

Benefits of CICS PA

CICS PA offers the following benefits:

- Analyzes CICS application performance
- Improves CICS resource usage
- Evaluates the effects of CICS system tuning efforts
- Improves transaction response time
- Provides ongoing system management and measurement reports
- Increases availability of resources

- Increases the productivity of system and application programmers
- Provides awareness of usage trends by assisting with future growth estimates

Components for CICS PA

Figure A-1 provides a view of the components for CICS PA that are explained in the following sections.

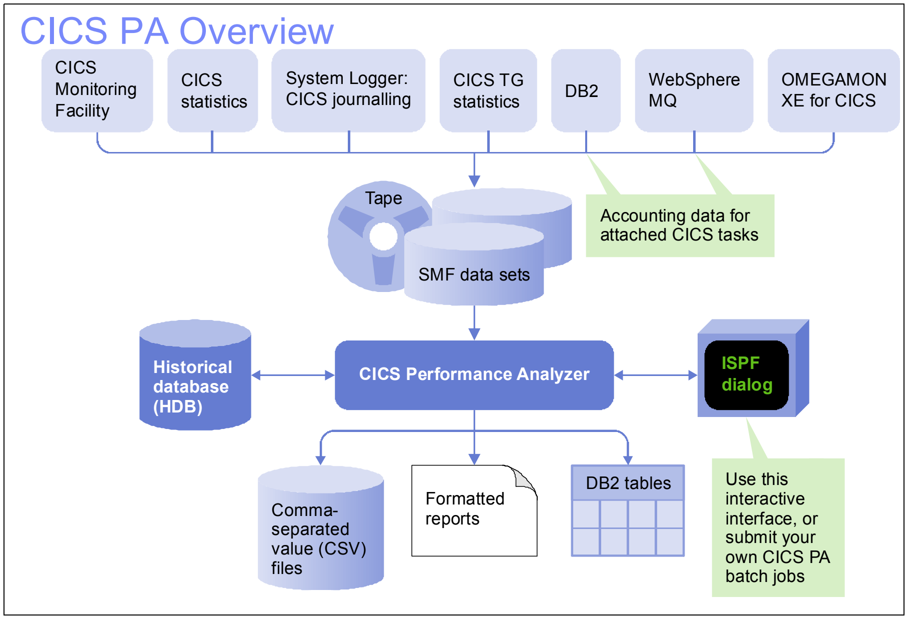


Figure A-1 Overview of CICS PA components

ISPF interface

Use the CICS PA Interactive System Productivity Facility (ISPF) dialog to generate reports and extract requests. The CICS PA dialog helps to build the reports and extracts requests that are specific to your requirements. This way, you avoid having to understand the complexity of the CMF data, CICS statistics, CICS server statistics data, CICS TG statistics, CICS System Logger data, DB2 accounting, and WebSphere MQ accounting data. The ISPF dialog has extensive online help facilities and a powerful command language that is used to select, sort, and customize the report formats and data extracts.

Reports and extracts

CICS PA provides a comprehensive suite of reports and data extracts for use in the following ways:

- System programmers can track overall CICS system performance and evaluate the effects of CICS system tuning efforts.
- Applications programmers can analyze the performance of their applications and the resources they use.
- Database administrators (DBAs) can analyze the use and performance of CICS Resource Managers, such as WebSphere MQ, and database systems, such as DB2 and IMS (DBCTL).
- MQ administrators can analyze the use and performance of their WebSphere MQ messaging systems.
- Managers can ensure that transactions are meeting their required service levels and to measure trends to help plan future requirements and strategies.

The Historical Database (HDB) facility provides a flexible way to manage and report historical performance and statistics data for your CICS systems. First you define an HDB template with customized data for historical reporting, and then you load it into the HDB. Next, CICS PA extracts it to a comma-separated values (CSV) file or exports it to DB2 for reporting with the CICS Explorer in spreadsheet view, charts, and graphs.

With report sets, you specify, save, and run your report requests. A report set contains a set of report and extract requests to be submitted and run as a single job. You can define any number of report sets, and any number of reports and extracts can be included in a single report set.

The following reports are most commonly used:

- The *performance list, list extended, and summary reports* provide detailed analysis of CICS transaction activity and performance.
- The *performance wait analysis report* provides a detailed analysis of transaction activity by wait time. This report summarizes, by transaction ID, the resources that cause a transaction to be suspended and highlights the CICS system resource bottlenecks that can cause bad response times.
- The *cross-system work report* combines CICS CMF performance class records from connected CICS (through multiregion operation (MRO) and intersystem communication (ISC)) systems for a consolidated network unit-of-work (UOW) report.
- The *DB2 reports* combine CICS CMF (SMF 110) performance class records and DB2 accounting (SMF 101) records to produce detailed or summary reports of the DB2 usage by your CICS systems. The DB2 list report shows

the DB2 activity of each transaction, and the DB2 summary report (short or long) summarizes the DB2 activity by transaction and program within an APPLID.

- The *transaction profiling report* benchmarks before and after results with detailed reporting. It can show differences between the report data and the baseline data as a delta (report data values minus their equivalent baseline data values) and as the percentage of change.

The extracts produce data sets that are intended for use by software applications, including CICS PA. The extract export facility creates a delimited text file that can be used in spreadsheet analysis or as input into the CICS Explorer.

CICS PA plug-in for CICS Explorer

The CICS PA plug-in for CICS Explorer (PA plug-in) is an Eclipse plug-in that operates on top of the IBM CICS Explorer to help you analyze CICS performance data.

By using the PA plug-in, you can perform the following tasks:

- View and sort the CSV or database data in a spreadsheet viewer.
- Select single or multiple transaction for analysis.
- Perform CPU time analysis.
- Perform file analysis.
- Perform response time analysis.
- Perform storage analysis.
- Perform threadsafe analysis.

For more information about the IBM CICS Explorer, see the CICS Explorer page at:

http://www.ibm.com/cics/explorer

New in CICS PA V3.2

The latest version of CICS PA for z/OS V3.2 includes the following new features:

- You can now export CICS TS and CICS TG statistics from the HDB into DB2 tables, making them available to the PA plug-in.

  Figure A-2, shows the statistic records for Event Binding.

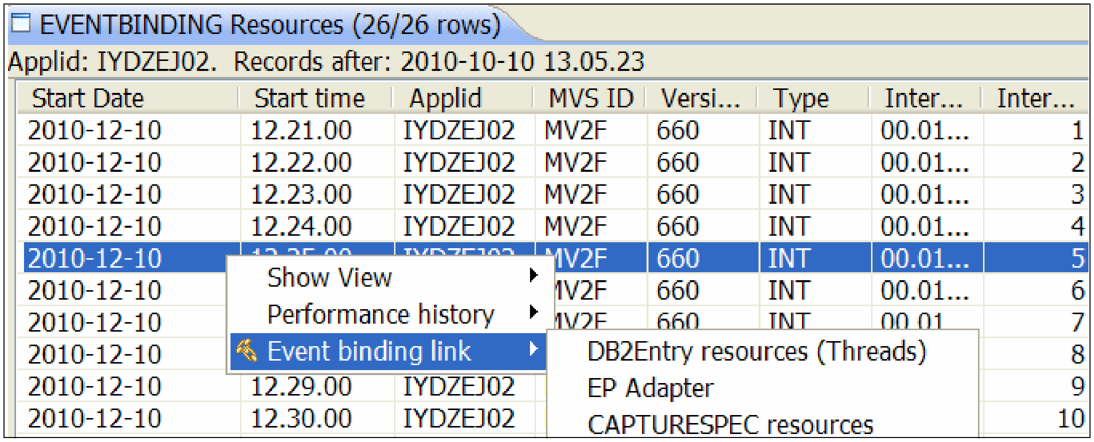


Figure A-2 Event Binding statistic records

- You can now load statistics alerts into the HDB and then a DB2 table, making the data available to the PA plug-in.

  Figure A-3 shows the CICS PA alert records.

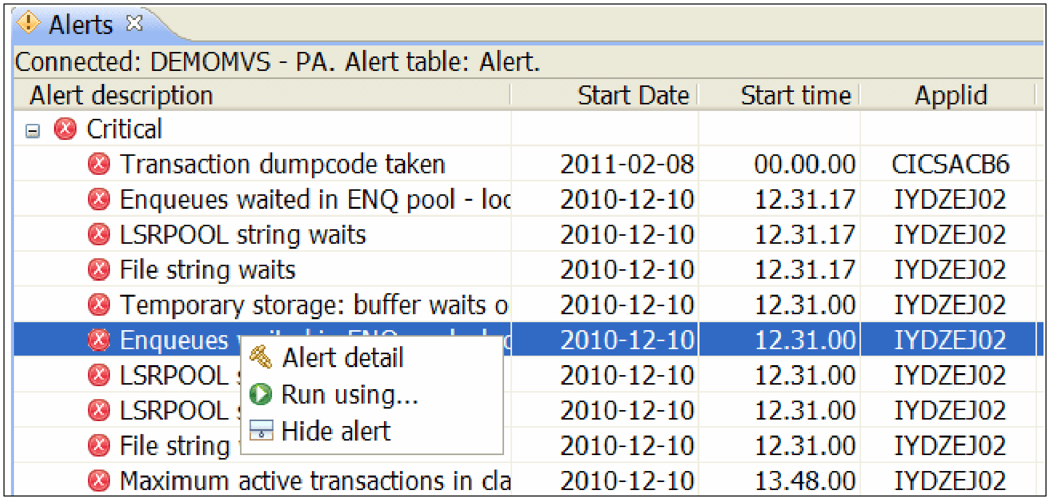


Figure A-3 CICS PA alert records in the Explorer

- You can format output from batch reports in Portable Document Format (PDF) files.
- Support is now available for CICS TG V8.
- The *IBM CICS Performance Analyzer for z/OS Getting Started Guide*, SC34-7155-01, is available at:

  http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.pa.doc/pdf/cpags-pdf.pdf

CICS Interdependency Analyzer for z/OS

This section highlights the CICS Interdependency Analyzer for z/OS (CICS IA) tool. It includes an overview of CICS IA, its purpose, components, architecture, and steps for configuring and using it for threadsafe analysis.

Overview of CICS IA

The CICS IA is a run-time and batch system for use with CICS TS for z/OS. It is used for the following purposes:

- Identifies the CICS application resources and their interdependencies so that you can understand the makeup of your application set, such as the following examples:
  - Which transactions use which programs
  - Which programs use which resources (such as files, maps, and queues)
  - Which resources are no longer used
  - Which applications a CICS region contains
  - Which commands within programs provide integrity exposures for threadsafe
  - Which commands cause a task control block (TCB) mode switch
- Analyzes transaction affinities. Affinities require particular groups of transactions to run in the same CICS region or in a particular region.

  Affinity information is useful in a dynamic routing environment. You must know about any restrictions that *prevent* transactions from being routed to particular application-owning regions (AORs) or that *require* particular transactions to be routed to particular AORs.

Benefits of CICS IA

Many large organizations have used CICS since the early 1970s, and their systems have grown and evolved with the business. During this time, many techniques for implementing applications have been used as a result of new functions, changing corporate standards, technical requirements, and business pressures.

Frequently, this growth has not been as structured as it might have been, with the result that many applications and services share common resources. Also, changes in one area typically affect many other areas, which can reach such a level that the system can no longer develop in a controlled manner without a full understanding of these interrelationships. CICS IA can help you achieve this understanding. For example, if you need to change the content or structure of a file, you must know which programs use this file because they must also be changed. CICS IA can provide this information and identify the transactions that drive the programs. CICS IA records the interdependencies between resources (such as files, programs, and transactions) by monitoring programming commands that operate on resources.

The application that issues such a command has a dependency on the resource that is named in the command. For example, if an application program issues the **`EXEC CICS WRITE FILE(myfile)`** command, it has a dependency on the `myfile` file. It might have similar dependencies on, for example, transient data queues, temporary storage queues, transactions, and other programs.

The commands that are monitored are typically CICS application programming interface (API) and system programming interface (SPI) commands that operate on CICS resources. However, you can also instruct CICS IA to monitor some types of commands (not commands provided by CICS) that operate on resources that are not CICS resources, such as the following examples:

- WMQ calls
- DLI calls to IMS Database resources
- DB2 calls
- Dynamic COBOL calls to other programs

CICS IA has the following features and capabilities:

- An Eclipse-based GUI to analyze collected data:
  - Sample queries with toolbar searches for common resources
  - Custom queries to interrogate dependency and affinity database objects
  - Integrated with the CICS Explorer, providing participation in cross-tooling capability from performance to resource definitions

- Timer-based collector control

  With this control, you can start the collector for a specific time of day to enable targeted data collection. For example, you can set the tool to schedule collection in different regions throughout the data collection process.

  It helps you to work around high volume time periods and to target collection for when an application is active.

- Enhanced single point of control capabilities:
  - You can turn data collection for multiple CICS regions on and off with a single `CINT` command to speed selection.
  - You can select default options for all your CICS regions with a single setting, or you can specify collection options to be region-specific.
- A selective program and transaction Exclude list that eliminates extraneous data and reduces overhead during data capture
- Provision of CICS system definition data set (CSD) name and group-list information
- Automation of tracking of runtime impact on application change by providing program version information, enabling removal of old data by version and comparison of data by program version
- Command Flow Feature for enhanced threadsafe analysis and tracking

Components for CICS IA

This section highlights the components of CICS IA, which include collector components and reporting components.

Figure A-4 illustrates the collector components for CICS IA.

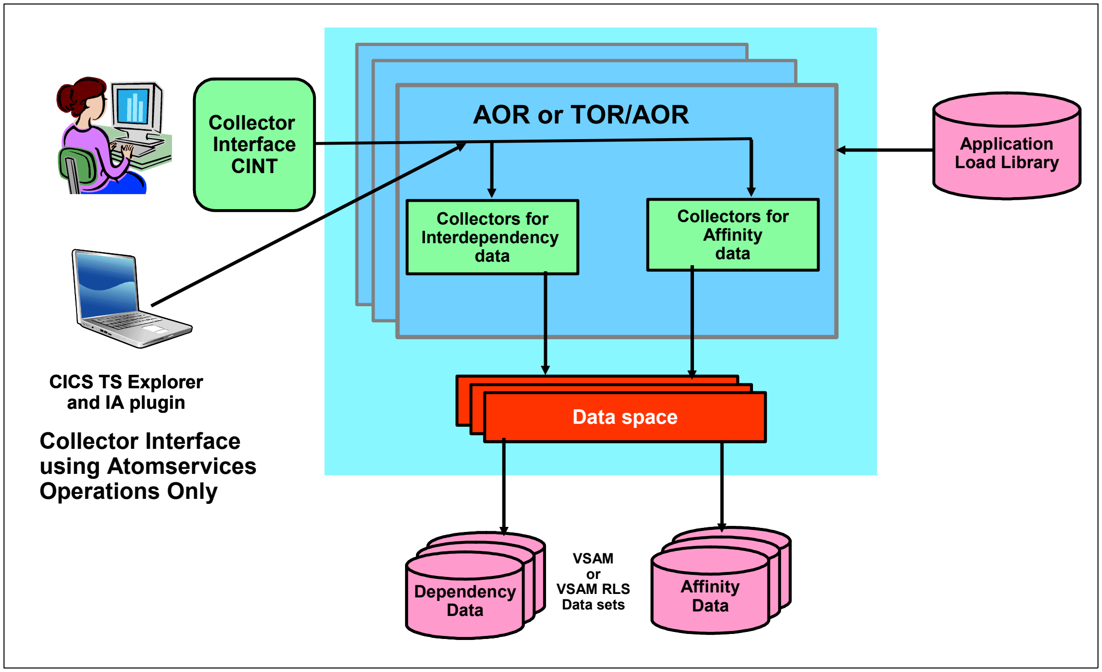


Figure A-4 CICS IA collector components (structure)

Figure A-5 illustrates the reporting components for CICS IA.

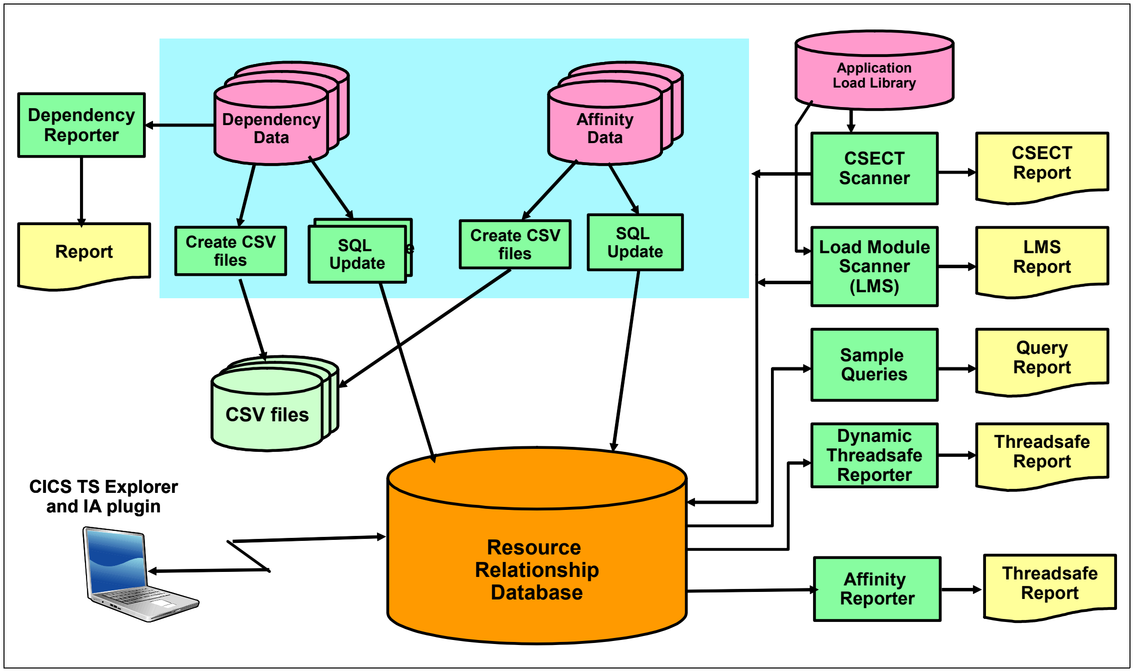


Figure A-5 CICS IA reporting components (reporting structure)

The design of CICS IA centers around the concept of examining the EXEC CICS commands that applications and systems programmers use. Each command and its parameters indicate the resources that the program will use. An analysis of these calls provides a view of resource interdependencies.

The scanner component

With the *scanner component* of CICS IA, you can write a program to examine the program load modules and report on the EXEC CICS commands and their parameters. This component produces a report that indicates, for each program, the commands issued, the programming language used, and the resources that are involved. The scanner also indicates whether the command is a possible affinity, a possible dependency, or both.

The collector component

The problem with using only the scanner is that it does not show the execution-time path through the code and which commands are run. The *collector component* intercepts the commands as they are run and captures the name of the program and its context (for example, which program called it and which transaction initiated it). This component also stores the data in an MVS data space.

The collector function can be activated across multiple CICS regions from a single point of control. The data can be collected across these regions and written to a Virtual Storage Access Method (VSAM) file that is shared between these regions using a file owning region (FOR) or using record-level sharing (RLS). The collector can collect dependency or affinity information. However, it cannot collect both types of information at one time. At specified intervals or on operator command, the data space is written to VSAM files.

From the interactive interface of CICS IA, you can control collectors that are running on multiple regions.

> **Tips:** To ensure that you monitor as many potential dependencies or affinities as possible, use CICS IA with all parts of your workload, including rarely used transactions and abnormal situations. You can store the collected information from several CICS regions into the same database. You can then review the collected dependencies and affinities by using the CICS IA query interface or produce your own SQL queries based on samples provided.

After the data is collected, CICS IA provides a set of utilities to enable this data to be loaded into a DB2 database. Having the data in DB2 provides many opportunities for detailed analysis by using standard SQL queries or using the online CICS basic mapping support (BMS) interface that CICS IA provides. This analysis can help you in the following ways:

- Use CICS resources more efficiently.
- Balance application workload for continuous availability.
- Improve the speed and reduce the cost of application maintenance.
- Minimize the impact of routine application maintenance for the user.
- Plan reuse of existing applications as business applications and build new applications more efficiently.

The reporter component

The *reporter component* is a set of batch programs that can produce reports from these batch files. You can run a summary report or a detailed report.

This component includes the following programs:

Dependency Reporter
Consists of a batch job that converts the dependency data collected by the collector component into reports that present the data in a readable format.

Affinities Reporter
Consists of a batch job that converts the affinity data collected by the Collector into reports presenting the data

in a readable format. It can also be used to create a file of affinity-transaction-group definitions in a syntax approximating the batch API of CICSPlex SM. This file is used as input to the builder component.

Threadsafe Reporter Consists of a batch job that produces reports displaying the threadsafe status of each command in the requested programs.

CICS IA also provides sample SQL queries for use with SPUFI or other DB2 query tools from IBM or other ISVs.

The builder component

The *builder component* is a batch utility that takes, as input, a file of basic affinity-transaction-group definitions created by the Reporter. It produces a file of combined affinity-transaction-group definitions that is suitable for input to CICSPlex System Manager (SM).

Command Flow feature

With the Command Flow feature, you can capture all EXEC CICS, SQL, MQ, and IMS calls in chronological order. In addition, it performs the following tasks:

- Traces command flows for up to five transactions.
- Writes to a CICS journal that uses the MVS logger.
- Provides a graphic format view in the CICS IA Explorer.
- Highlights, for example, TCB mode switches, non-zero return codes, and getmain freemain addresses.

Figure A-6 shows the Command Flow components.

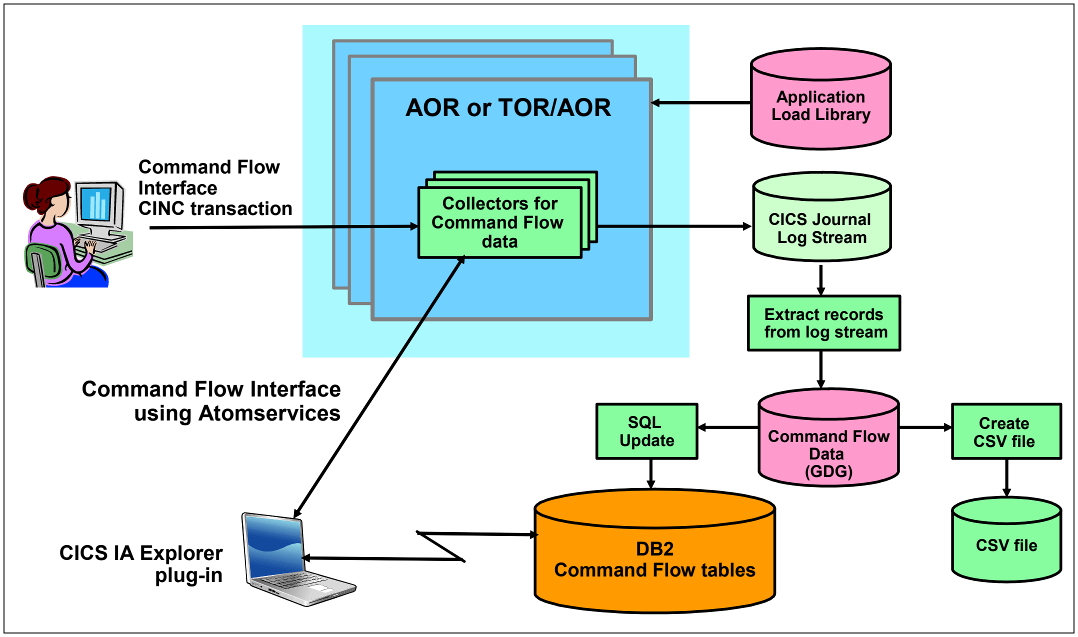


Figure A-6 CICS IA Command Flow components

CICS IA Explorer plug-in

The CICS IA Explorer plug-in provides an Eclipse-based infrastructure to identify CICS resources and their interdependencies and to analyze transaction affinities.

You use the CICS IA plug-in to analyze and explore the collected data about CICS interactions and CICS, Affinity, IMS, DB2, and MQ resources. The CICS IA plug-in requires the location of the DB2 database in which CICS IA stored data. Use the Connection preferences window to enter this information. When you make a successful connection between the CICS IA plug-in and the DB2 collector tables, you can search to find resources and analyze their usage and their dependencies.

New in CICS IA V3.2

CICS IA V3.2 has the following new features:

- You can now operate CICS IA from the Explorer plug-in. You can start, stop, pause, or continue the collector in any connected region (Figure A-7).

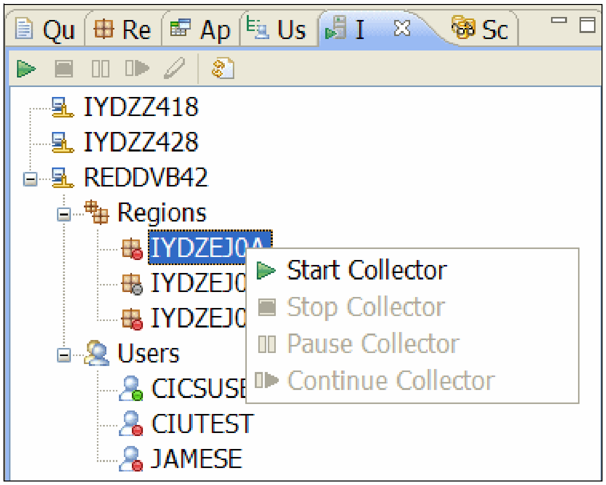


Figure A-7 CICS IA - Explorer Operations View

- The CICS Command Flow feature (Figure A-8) is separated into its own transaction and can be used simultaneously by many developers or system programmers. The command flow data is captured and viewed by user ID. It can be operated and administered from the Explorer plug-in.

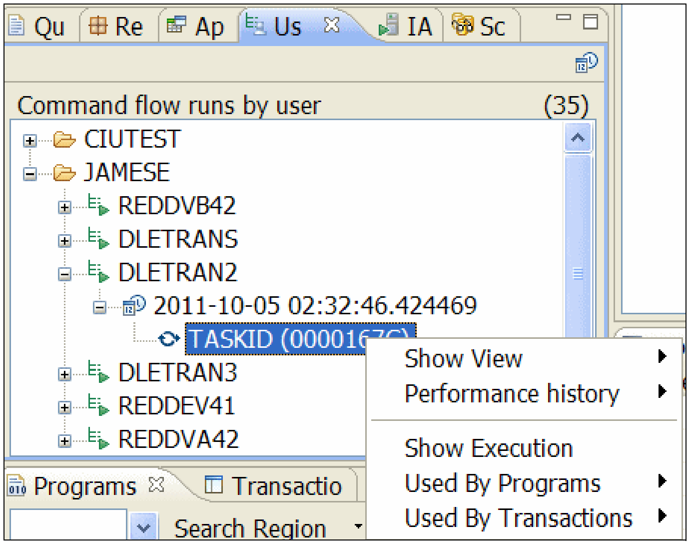


Figure A-8 CICS IA view to show command flows by user ID

- CICS IA now collects Event information with CICS Business Event support. You can view the Events in the Explorer (Figure A-9).

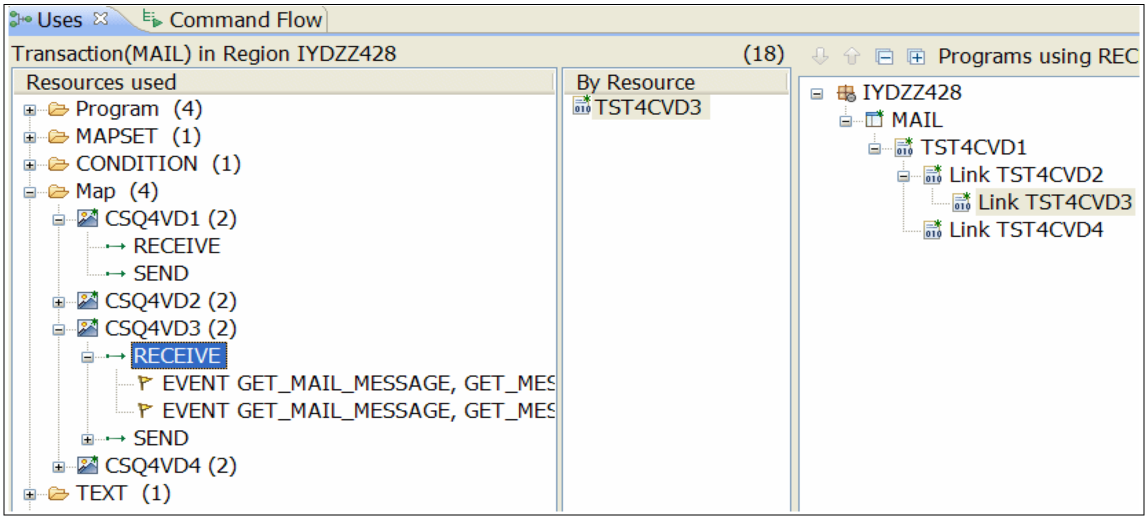


Figure A-9 CICS IA Events

- By using the Explorer plug-in, you can now generate events by calling the Events Wizard (Figure A-10) from both the **Resources** view or the **Command Flow** view. A new option is available for resource types on which you can generate an event, such as a program. The selected resource type, program name, and transaction name are passed to the event specification.

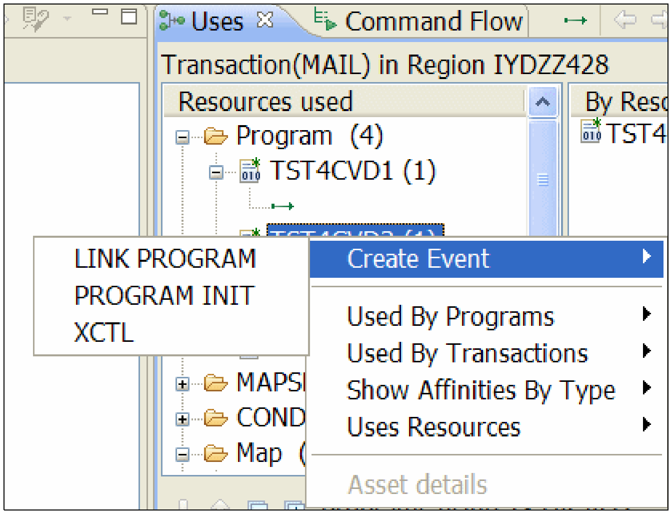


Figure A-10 CICS IA Event Generation

- By using the Explorer plug-in, you can analyze predefined business applications (Figure A-11).

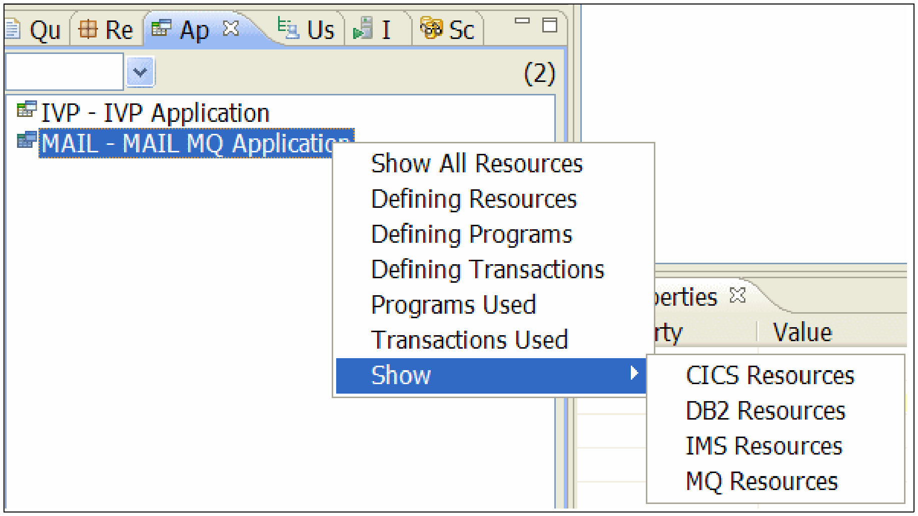


Figure A-11 CICS IA Application Analysis

- By using the Explorer plug-in, you can now enable Affinity analysis by region, program, or transaction (Figure A-12).

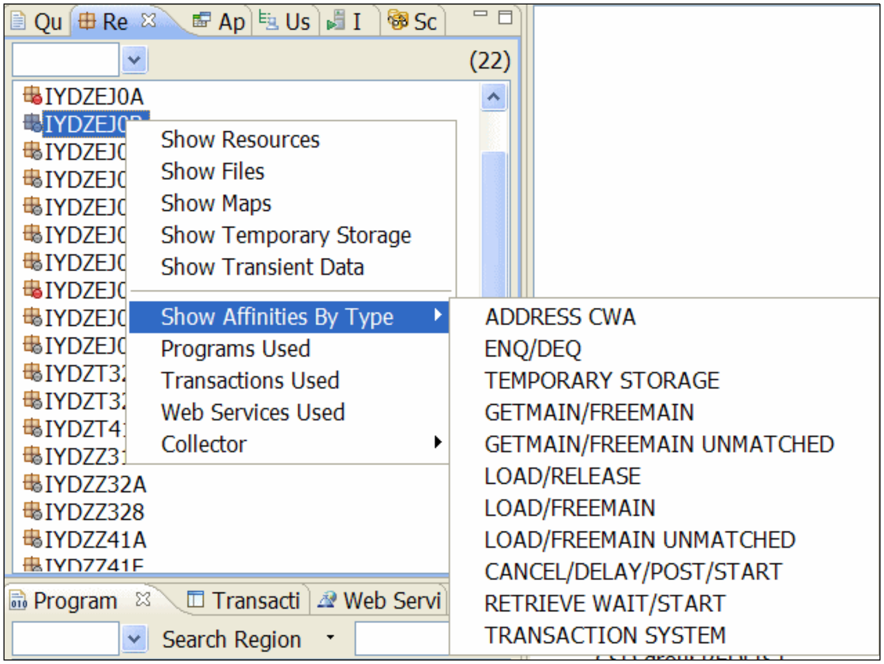


Figure A-12 CICS IA Threadsafe Analysis

- With the new **Scanner** view, you can see load module data sets that are scanned by the CICS command scanner and the CSECT scanner.
  - By expanding this view you can see all the CSECTs for a program if it is available. By selecting a CSECT, the **Properties** view shows the details for that CSECT (Figure A-13).

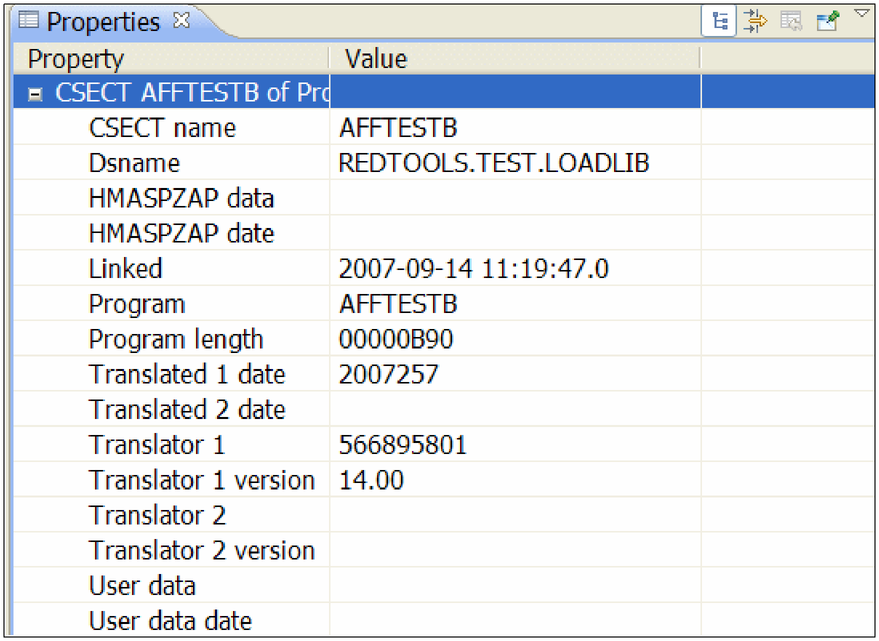


Figure A-13 CICS IA CSECT scanner in the Properties view

  - When you right-click a program, you can see the possible CICS commands that are captured by the resource scanner (Figure A-14).

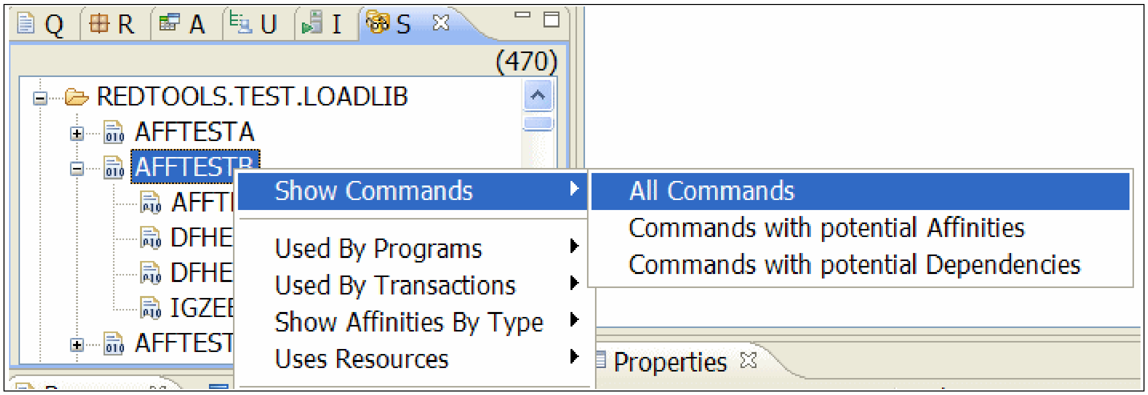


Figure A-14 CICS IA possible CICS commands from scanner tables

CICS Configuration Manager

This section highlights the CICS Configuration Manager (CICS CM) tool. It includes an overview of CICS CM, its purpose, components, architecture, and features.

Overview of CICS CM

CICS CM provides a single point of control for editing, reporting, and migrating CICS resource definitions across an enterprise. It provides change management capabilities to CICS resource definitions, change control package definitions, and audit history reporting for the lifecycle of CICS resource definitions.

Benefits of CICS CM

CICS CM includes the following benefits:

- Single point of control for all resource definitions (CSD and CICSPlex SM Business Application Services (CPSM BAS))
- Resource changes using migration schemes and transformation rules with end-to-end accountability and control
- Lower CICS system administration costs
- Lower risk of downtime due to user errors

Components of CICS CM

CICS Configuration Manager has the following main components:

- The server, which is a CICS application that can read from and write to CSD files and CICSPlex SM contexts
- The supplied clients, which include an interactive ISPF dialog interface and a batch command interface

  The clients communicate with the server by exchanging SOAP messages over an HTTP network.

As an alternative to using the supplied clients, you can use CICS Explorer with the CICS Configuration Manager plug-in, or you can develop your own clients.

Figure A-15 shows the CICS CM component architecture.

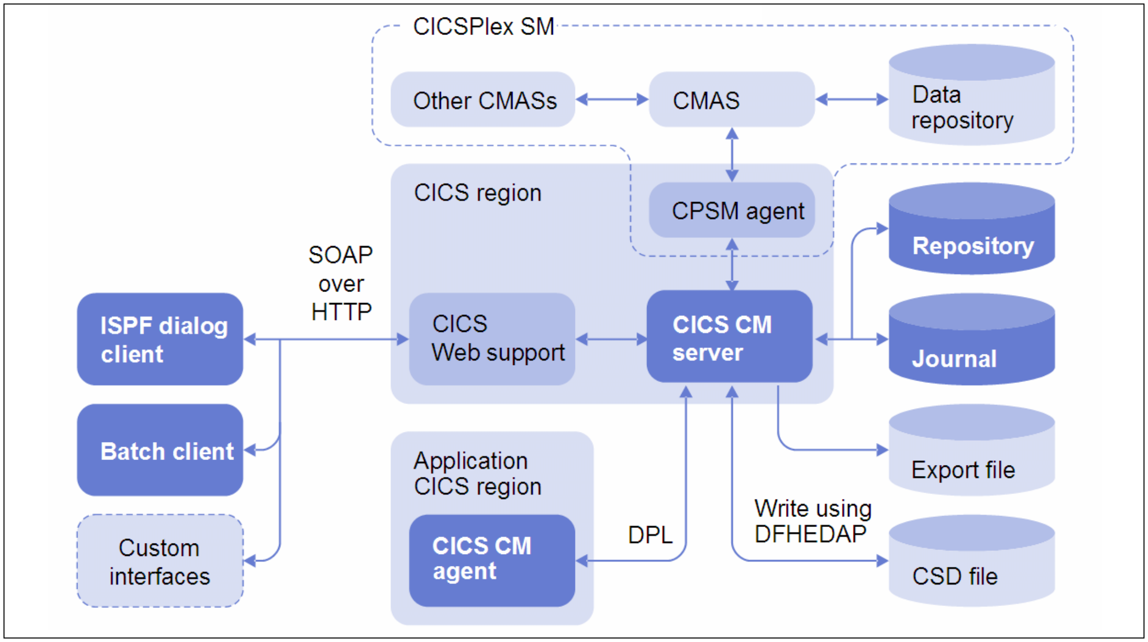


Figure A-15 CICS CM component architecture

Server

A *server* is a set of CICS programs that perform the actions requested by a client.

Client

A *client* is a user interface that allows you to send commands to, and receive responses from, the CICS CM server. The client and server communicate by exchanging SOAP messages using TCP/IP sockets.

CICS CM is supplied with two clients: an ISPF dialog and a batch command interface. As an alternative to using the supplied clients, you can use CICS Explorer with the CICS CM plug-in, or you can develop your own custom clients.

The CICS TS Explorer with the CICS CM plug-in provides an Eclipse-based GUI to many of the CICS CM functions that are available in the supplied ISPF user interface. CICS Explorer also provides an integrated interface to various CICS functions and other CICS tools. For more information about CICS Explorer and the CICS Configuration Manager plug-in, see the CICS Explorer page at:

http://www.ibm.com/cics/explorer/

For information about developing your own custom clients, see the API reference.

Repository

A *repository* is a VSAM key-sequenced data set (KSDS) that stores current CICS CM data.

Journal

A *journal* is a VSAM key-sequenced data set (KSDS) that records historical CICS CM data. A journal can also be a summary of processing events, such as updates to resource definitions and before and after copies of CICS resource definitions that CICS CM updated.

Agent

An *agent* is a CICS CM program, running in a target CICS region, that performs actions on that target CICS region on behalf of the CICS CM server. When a CICS CM client requests installation, newcopy, or discard actions for a target CICS region, the server uses a CICS distributed program link (DPL) to invoke the agent in that region. The agent then performs the action, which can be a CICS **`CEDA INSTALL`**, a CICS **`EXEC DISCARD`**, or a CICS **`EXEC SET PROGRAM`** (specifying either **`NEWCOPY`** or **`PHASEIN`**).

The agent is required only if you want to perform installation, newcopy, or discard actions on an active CICS region whose resource definitions are stored in a CSD file. You must make this program available within that CICS region in the same manner as any other application program. This agent is not used for CICS regions that are managed by CICSPlex SM. Instead, for those regions, CICS CM uses the CICSPlex SM API to perform these actions.

CICS CM ISPF interface

Menu panels show several options, from which you can select one. List panels show several items, with each item on a separate line, and you can enter a line action against one or more items. The available line actions depend on the type of item.

The CICS CM primary option menu presents the following options:

0 Settings

Customize the ISPF dialog for each user, and store the settings in each user's ISPF profile:

- Whether to show a prompt for confirmation of save and cancel commands
- Whether to automatically translate to uppercase some mixed-cased resource definition attributes
- Default job control information and stepped library for CICS CM batch jobs
- CICS CM server connection details, such as IP address and port number

These options are specific to each user, and are stored in each user's ISPF profile.

1 Administer

Set the system options that affect all users, and maintain the records for working with resource definitions:

- CICS configurations
- Migration schemes
- Approval profiles
- Transformation rules

2 CICS Resources

Work with resource definitions. Edit, compare, or package current resource definitions. View, compare, or restore historical versions.

3 Packages

Work with change packages.

4 Reports

Display sets of resource definitions that match various selection criteria, including historical versions of resource definitions.

New in CICS CM V2.1

The following features are new in CICS CM:

- Deployment Analysis reports, including cold-start analysis
- CICS Explorer plug-in
- Full function BAS definition support
- Change Package command stack
- Enhanced diagnostic tests

CICS CM Explorer plug-in

The CICS CM plug-in provides an Eclipse-based infrastructure to view and manage CICS CM resource definitions across an enterprise. It supports a subset of the function that is available in CICS CM.

By using the CICS CM plug-in, you can perform the following tasks:

- View all the CICS CM configurations (Figure A-16).

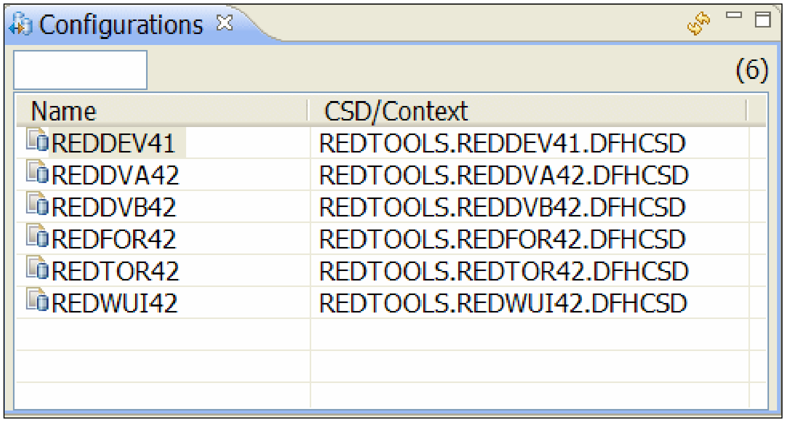


Figure A-16 Viewing all configurations (single point of control) in CICS CM

- View all lists in a configuration (Figure A-17).

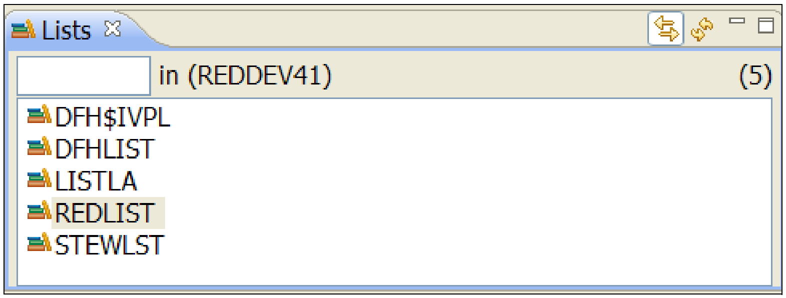


Figure A-17 CSD list in configuration REDDEV41 in CICS CM

- View all groups in a list (Figure A-18).

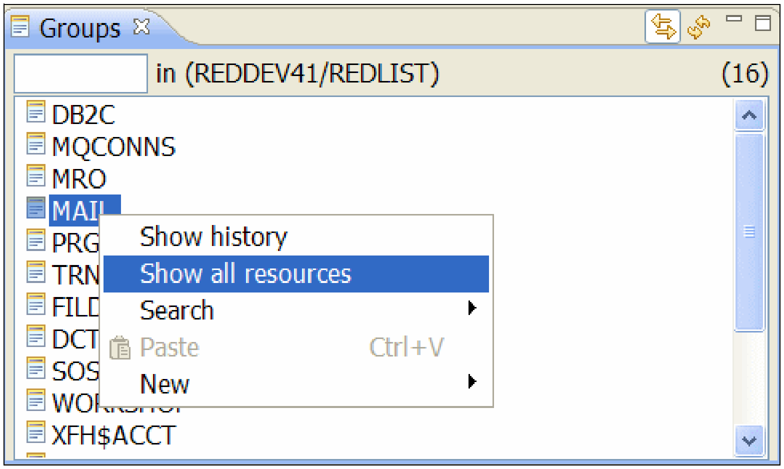


Figure A-18 Show groups in list REDLIST in CICS CM

- View resources by group (Figure A-19).

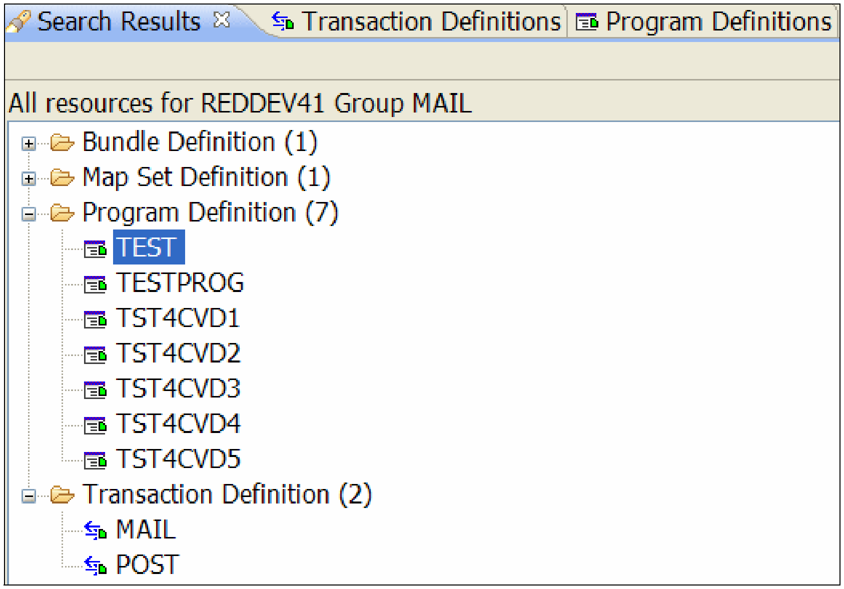


Figure A-19 Viewing resources for group MAIL in CICS CM

- View, edit, and delete resource definitions (Figure A-20).

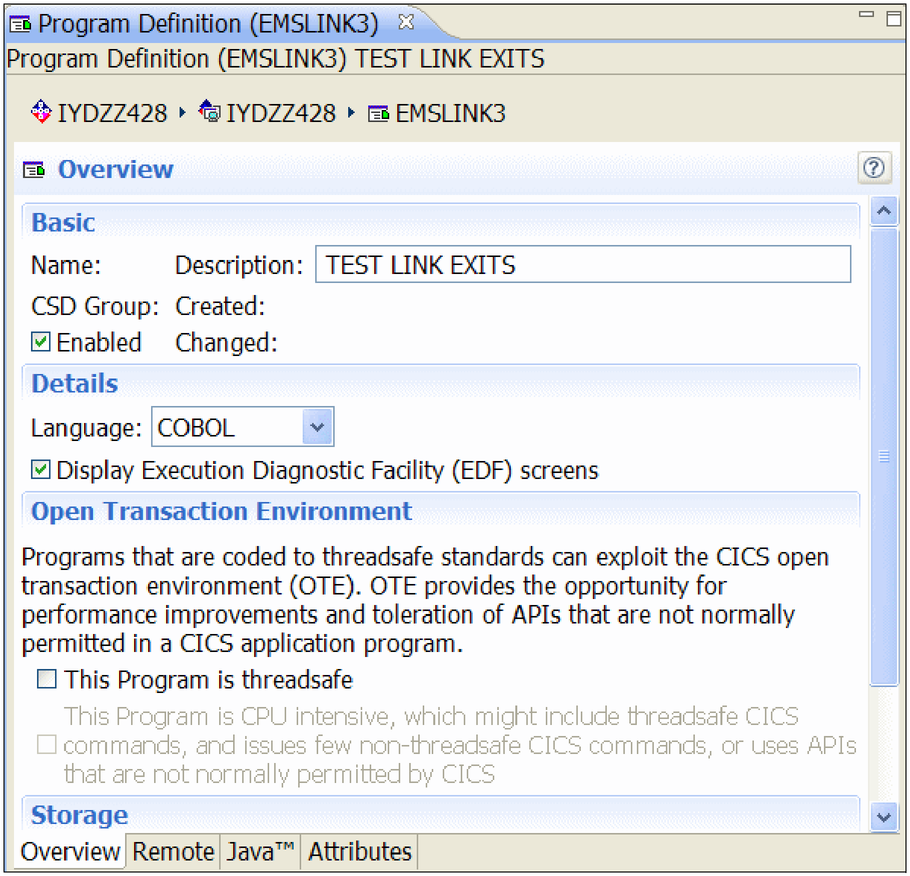


Figure A-20 Editing a resource (threadsafe explanation) in CICS CM

- View orphaned resources and groups in a CICSPlex SM configuration.

- View orphaned groups in a configuration (Figure A-21).

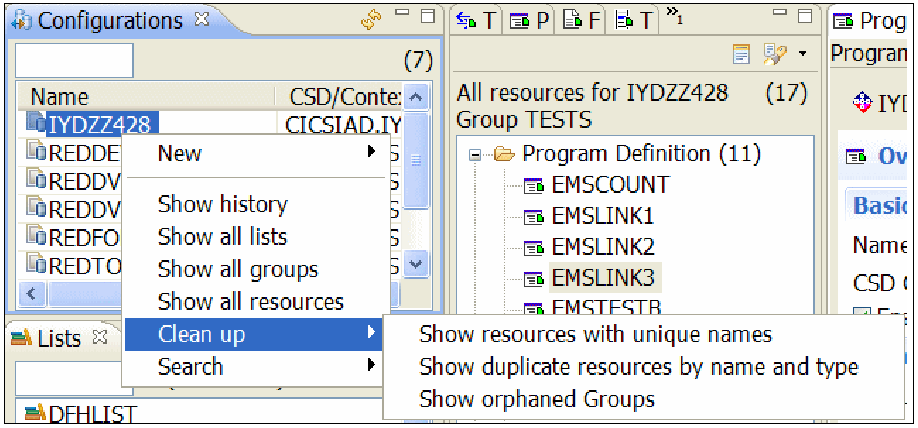


Figure A-21 Clean up feature in CICS CM

- View history for resource definitions, configurations, and groups, and restore changes made to a resource definition (Figure A-22).

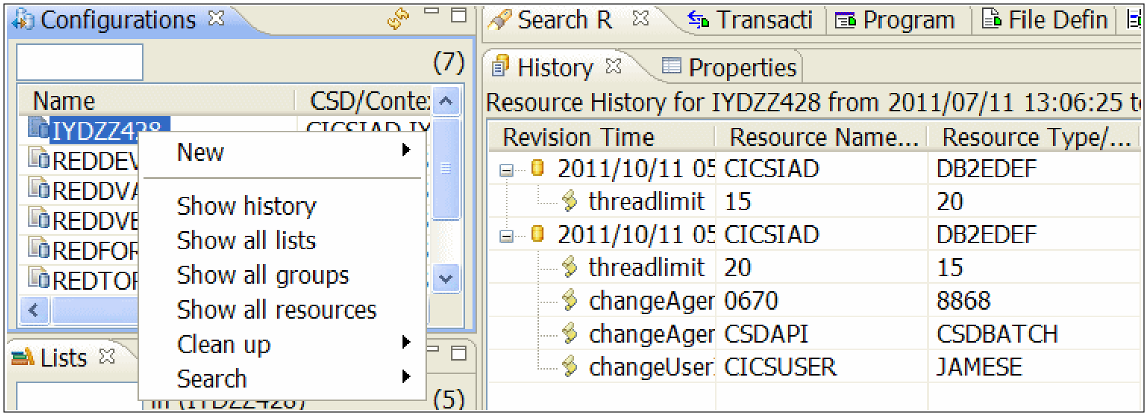


Figure A-22 History for configuration IYDZZ428 in CICS CM

- Search across one or more configurations, and search across one or more groups in a configuration (Figure A-23).

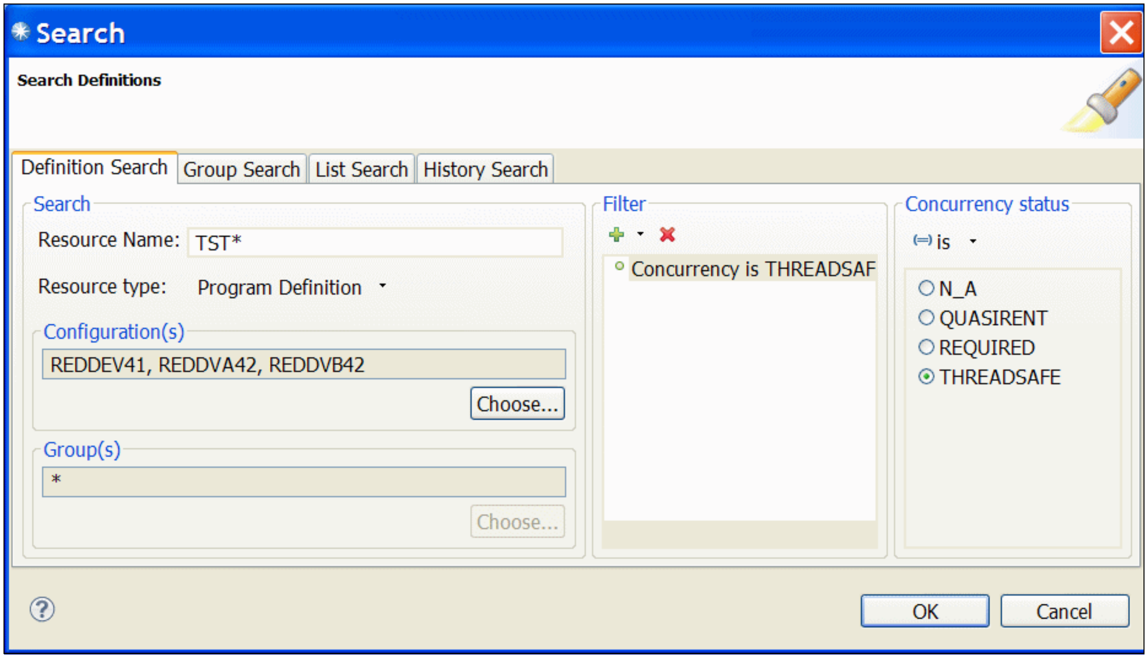


Figure A-23 Searching for programs with Concurrency Threadsafe in CICS CM

- Create new resources.
- Install resources from a Configuration in one or more active CICS systems.
- Search for groups.
- Search the history for a configuration or a group.
- Install a group if you have a CICSPlex SM connection.

- Compare two lists in the same configuration, two groups in the same or different configurations, or two of the same types of definition (Figure A-24).

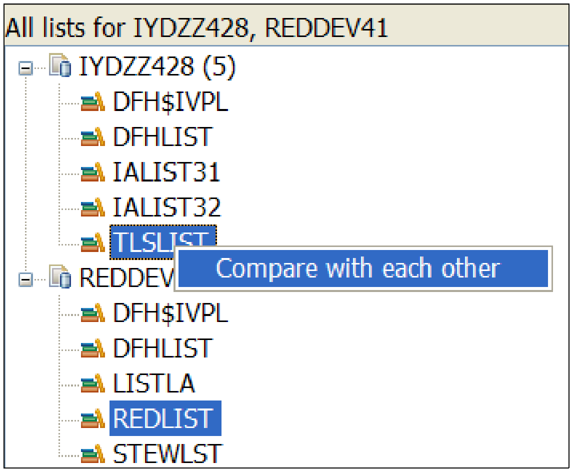


Figure A-24 CICS CM compare lists feature

CICS VSAM Transparency performance on CICS Transaction Server V3.2, V4.1, and V4.2

CICS VSAM Transparency (CICS VT) enables the migration of data from VSAM files to DB2 tables and ensures continued access to this data without modifying existing CICS and batch application programs. CICS VT supports CICS TS supported releases without any modification.

The threadsafe File Control API in CICS TS V3.2, V4.1, and V4.2 provides significant performance benefits for CICS VT.

CICS VT uses File Control GLUE programs to intercept File Control API calls and processes these requests as SQL calls to DB2. Although these GLUE programs have always been threadsafe, nonthreadsafe CICS File Control APIs in releases of CICS TS before CICS TS V3.2 resulted in a switch back to the quasi-reentrant (QR) TCB for every File Control API call.

B

Maintenance of CICS, DB2, and WebSphere MQ

This appendix provides a list of the maintenance APARs to apply to CICS, DB2, and WebSphere MQ. It includes the following sections:

© Copyright IBM Corp. 2004, 2012. All rights reserved.

APARs for CICS Transaction Server V3.1

CICS Transaction Server for z/OS (CICS TS) V3.1 has the following APARs:

- PQ05771

  Purge and forcepurge of tasks using OPENAPI True fails.
- PK05933

  Sqlcode -922 after a COBOL program precompiled in DB2 V8 new function mode.
- PK14003

  Task stuck in resumed early state.
- PK20040

  RMI 0C4 abend.
- PK21134

  Abend AD3K due to recovery backout failure after a task is purged.
- PK31859

  Abend AD3K and AEXZ on a task purge of a DB2 threadsafe transaction.

APARs for CICS Transaction Server V3.2

CICS TS V3.2 has APAR PK45354, which includes the following changes:

- File control threadsafe modifications.
- Change default for FCQRONLY parameter to YES.

APARs for CICS Transaction Server V4.1 and V4.2

CICS TS V4.1 and V4.2 has the following APARs:

- PM42455: Base code correct in 670; no route required
- PM52565: Routed to PM53485 at 670
- PM55984: Routed to PM59871 at 670
- PM59876: Routed to PM59329 at 670

APARs for WebSphere MQ 6.1

WebSphere MQ 6.1 has the following APARs:

- PK42616

  Checks the CICS release.

- PK38772

  Bridge code does not provide a reason code for sign-on failures after migrating to version 6.0.

APARs for the DFHEISUP utility

The **DFHEISUP** utility has the following APARs:

- PQ73890

  The **DFHEISUP** utility does not list the **EXEC CICS SEND MAP** command when the command contains the MAPONLY option.

- PQ76545

  Abend 0C4 in the DFHEISUP module for scanning application load libraries.

- PQ77185

  CEE3204S indicates that the system detected a protection exception (SYSTEM COMPLETION CODE=0C4).

- PQ78531

  DFHEISUP library problem. Runs short on storage.

- PQ82603

  Running the **DFHEISUP** utility returns an undocumented error message when certain commands are encountered.

C

Assembler routines

This appendix lists the assembler routines that were used in the migration as documented in this book. It includes the following routines:

- DB2MANY
- DB2PROG1
- DB2PROG4
- DB2PROG8
- PLANEXIT
- EXITENBL
- XXXEI exit
- XXXRMI exit
- XXXTS exit

© Copyright IBM Corp. 2004, 2012. All rights reserved.

DB2MANY

Example C-1 is a list of code for the DB2MANY program.

Example C-1 iDB2MANY code listing

```
***********************************************************************
 DFHEISTG DSECT
          EXEC SQL INCLUDE SQLCA
 *
 ***********************************************
 DFHEISTG DSECT
 ***********************************************
 VVEMP     DS   CL80
 EMPNO     DS   CL6
 FIRSTNME  DS   CL12
 MIDINIT   DS   CL1
 LASTNAME  DS   CL15
 WORKDEPT  DS   CL3
 PHONENO   DS   CL4
 HIREDATE  DS   CL10
 JOB       DS   CL8
 EDLEVEL   DS   HL2
 SEX       DS   CL1
 BIRTHDATE DS   CL10
 SALARY    DS   PL3
 BONUS     DS   PL3
 COMM      DS   PL3
 ***********************************************
 TERMNL         DC    F'0'
DATALEN        DS    F'0'
               DS    0D
               DC    C'EISTG   '
MESSAGES DS    CL80               TEMP STORE
KEYNUM   DS    CL9                TEMP STORE
COMLEN   DS    1H                 LENGTH OF C
         DS   0F
SQDWSTOR DS   (SQLDLEN)C     RESERVE STORAGE TO BE USED FOR SQLDSECT
SDARGDATA      DC    50F'0'
               DC    C'EISTG END'
SDARG          DSECT
SDREPEAT       DC    X'00000000'  NUMBER OF TIMES TO REPEAT DB2 CALL
SDTERMID       DS    CL4      TERMINAL ID
SDREPCNT       DC    F'0'     CURRENT NUMBER TO BE ATTACHED
SDPASSCT       DC    F'0'     NUMBER OF START TASK PASSES
SDTRAN         DS    F'0'
SDASKTIM       DS    CL4
SDEMPNO        DS    CL6      EMPLOYEE NUMBER TO USE
INPUT          DC    20F'0'   INPUT DATA
```

```
INMSGLEN       DS    OH        MESSAGE LENGTH
*
*********************************************************
SQDWSREG EQU   7
RETREG   EQU   2                   SET UP REGISTER USAGE
COUNTER  EQU   5
R06      EQU   6
R08      EQU   8
R9       EQU   9
COMPTR   EQU   4                   POINTER TO COMMAREA
SDPASSR  EQU   11                  PASS COUNT REG
*********************************************************
DB2MANY  CSECT
DB2MANY  AMODE 31
DB2MANY  RMODE ANY
*****************************************************************
* OBTAIN INPUT DATA
         LA    R08,SDARGDATA
         USING SDARG,R08
         MVC   SDREPEAT,REPEAT  SET TO THE NUMBER OF DB2 CALLS
*****************************************************************
*
* SQL WORKING STORAGE
         LA    SQDWSREG,SQDWSTOR   GET ADDRESS OF SQLDSECT
         USING SQLDSECT,SQDWSREG   AND TELL ASSEMBLER ABOUT IT
*
         EXEC SQL
               DECLARE DSN8710.EMP TABLE (
               EMPNO                       CHAR(6),
               FIRSTNME                    CHAR(12),
               MIDINIT                     CHAR(1),
LASTNAME                 CHAR(15),                  *
               WORKDEPT                    CHAR(3),                *
               PHONENO                     CHAR(4),                *
               HIREDATE                    DATE,                   *
               JOB                         CHAR(8),                *
               EDLEVEL                     SMALLINT,               *
               SEX                         CHAR(1),                *
               BIRTHDATE                   DATE,                   *
               SALARY                      DECIMAL,                *
               BONUS                       DECIMAL,                *
               COMM                        DECIMAL )
*
*
RESET    L     COUNTER,COUNT
*
READLOOP DS    OH
         EXEC  CICS ASKTIME
*
```

```
 EXEC SQL SELECT * INTO :VVEMP FROM DSN8710.EMP WHERE EMPNO='000140'
         LA    COUNTER,1(COUNTER)
         C     COUNTER,MAXREAD
         BNH   READLOOP
*
*****************************************************************
**  NOW START THE NEXT TASK                                   ****
**                                                            ****
         L     SDPASSR,SDPASSCT     LOAD THE WORK REG
         C     SDPASSR,NUMPASS
         BE    NOSTART
STARTLP  DS    0H
         LA    R08,SDARGDATA
         USING SDARG,R08
         MVC   SDTERMID,EIBTRMID
         MVC   SDREPEAT,REPEAT SET THE NUMBER OF DB2 CALLS PER TRAN
         MVC   SDREPCNT,NUMTRAN  PASS THE NUMBER OF TIMES TO RESTART
         MVC   SDTRAN,=CL4'DB21'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
********************************************************************
         MVC   SDTRAN,=CL4'DB22'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB23'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB24'
MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB25'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB26'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB27'
         MVC   TERMNL,EIBTRMID
```

```
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB28'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB29'
         MVC   TERMNL,EIBTRMID
EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)                   *
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         MVC   SDTRAN,=CL4'DB2A'
         MVC   TERMNL,EIBTRMID
         EXEC CICS START TRANSID(SDTRAN) INTERVAL(0)                   *
                   FROM(SDARG) LENGTH(SDLENG)
*****************************************************************
         LA    SDPASSR,1(SDPASSR)  INCREMENT THE COUNTER
         C     SDPASSR,NUMPASS
         BNL   NOSTART
         B     STARTLP
NOSTART  DS 0H
         EXEC CICS SEND TEXT FROM(AREA) FREEKB
         EXEC CICS RETURN
         DS    0F
AREA     DC    CL40'TRANSACTION COMPLETE'
***************************************************************
          DC F'0'
REPEAT    DC    X'000000C8'     TEST NUMBER OF TIMES TO REPEAT
COUNT     DC    X'00000000'
MAXREAD   DC    X'000000C8'    NUMBER OF DB2 CALLS FOR DB2M XACTION
SDEND     DC    X'00000001'    LAST ONE
NUMTRAN   DC    F'01000000'  NUMBER OF TIMES A TASK IS TO RESTART
NUMPASS   DC    F'00000001'  THE NUMBER OF PASSES AT STARTING TASKS
***************************************************************
SDLENG   DC    X'0030'    LENGTH OF TS RECORD
 ***************************************************************
          END
```

DB2PROG1

Example C-2 shows a listing of the DB2PROG1 code. The DB2PROG1 program is the same as the DB2PROG2 and DB2PROG3 programs and includes the following actions:

- **EXEC CICS RETRIEVE**
- **EXEC CICS POST**
- **EXEC CICS WAITCICS**
- **EXEC CICS START**

Example C-2 DB2PROG1 example (same as DB2PROG2 and DB2PROG3)

```
DFHEISTG DSECT
          EXEC SQL INCLUDE SQLCA
 *
 **********************************************
 DFHEISTG DSECT
 **********************************************
 VVEMP     DS   CL80
 EMPNO     DS   CL6
 FIRSTNME  DS   CL12
 MIDINIT   DS   CL1
 LASTNAME  DS   CL15
 WORKDEPT  DS   CL3
 PHONENO   DS   CL4
 HIREDATE  DS   CL10
 JOB       DS   CL8
 EDLEVEL   DS   HL2
 SEX       DS   CL1
 BIRTHDATE DS   CL10
 SALARY    DS   PL3
 BONUS     DS   PL3
 COMM      DS   PL3
 **********************************************
 TERMNL         DC     F'0'
DATALEN        DS    F'0'
               DS    0D
ECB1           DS    1F
**********************************************
* THE FORMAT OF THE TS QUEUE RECORD PASSED TO
**********************************************
               DC    C'EISTG   '
MESSAGES DS    CL80               TEMP STORE
KEYNUM   DS    CL9                TEMP STORE
COMLEN   DS    1H                 LENGTH OF C
         DS   0F
SQDWSTOR DS   (SQLDLEN)C     RESERVE STORAGE TO BE USED FOR SQLDSECT
SDARGDATA      DC    20F'0'
```

```
              DC    C'EISTG END'
SDARG         DSECT
SDREPEAT      DC    X'00000000'  NUMBER OF TIMES TO MAKE THE DB2 CALL
SDTERMID      DS    CL4       TERMINAL ID
SDREPCNT      DC    F'0'      CURRENT NUMBER TO BE ATTACHED
SDTRAN        DS    F'0'
SDASKTIM      DS    CL4      YES ISSUE ASKTIME,NO SKIP ASKTIMES
SDEMPNO       DC    2F'0'     EMPLOYEE NUMBER TO USE
*
*********************************************************
R1       EQU   1
SQDWSREG EQU   7
RETREG   EQU   2                   SET UP REGISTER USAGE
R06      EQU   6
R08      EQU   8
R9       EQU   9
SDREPCTR EQU   5
COMPTR   EQU   4                   POINTER TO COMMAREA
*********************************************************
DB2PROG1 CSECT
DB2PROG1 AMODE 31
DB2PROG1 RMODE ANY
*****************************************************************
         MVC   TERMNL,EIBTRMID
* OBTAIN START DATA
         EXEC CICS RETRIEVE SET(R08) LENGTH(DATALEN)
         USING SDARG,R08
*****************************************************************
*
* SQL WORKING STORAGE
         LA    SQDWSREG,SQDWSTOR   GET ADDRESS OF SQLDSECT
         USING SQLDSECT,SQDWSREG   AND TELL ASSEMBLER ABOUT IT
*
         EXEC SQL                                                     *
               DECLARE DSN8710.EMP TABLE (                            *
               EMPNO                        CHAR(6),                  *
               FIRSTNME                     CHAR(12),                 *
               MIDINIT                      CHAR(1),                  *
               LASTNAME                     CHAR(15),                 *
WORKDEPT                    CHAR(3),
               PHONENO                      CHAR(4),
               HIREDATE                     DATE,
               JOB                          CHAR(8),
               EDLEVEL                      SMALLINT,
               SEX                          CHAR(1),
               BIRTHDATE                    DATE,
               SALARY                       DECIMAL,
               BONUS                        DECIMAL,
               COMM                         DECIMAL )
```

```
*
*****************************************************************
         L     6,SDREPEAT
         EXEC  CICS POST SET(R9)
         ST    R9,ECB1              POST EVENT & STORE ADDRESS
*
AGAIN    DS    OH
         LA    R9,ECB1              WAIT UNTIL ECB POSTED
         EXEC  CICS WAITCICS
                    ECBLIST(R9)
                    NUMEVENTS(=F'1')
                    NAME(=C'APPLWAIT')
                    PURGEABLE
*
 EXEC SQL SELECT EMPNO INTO :EMPNO FROM DSN8710.EMP
                  WHERE EMPNO='000070'
BCT  6,AGAIN
*
*****************************************************************
**  NOW START THE NEXT TASK                                  ****
**                                                           ****
         L     SDREPCTR,SDREPCNT     LOAD THE WORK REG
         LTR   SDREPCTR,SDREPCTR
         BZ    NOSTART
         S     SDREPCTR,SDEND        DECREMENT THE COUNTER
         ST    SDREPCTR,SDREPCNT     SAVE IT BACK FOR NEXT START
         EXEC CICS START TRANSID('DB21') INTERVAL(0)
                   FROM(SDARG) LENGTH(DATALEN)
*****************************************************************
*        EXEC CICS PERFORM STATISTICS RECORD DISPATCHER
NOSTART  DS OH
*        EXEC CICS SEND TEXT FROM(AREA) FREEKB
         EXEC CICS RETURN
         DS    OF
BIG_NUMBER DC  X'00000500'     XXX,XXX  1280 TIMES
AREA     DC    CL30'TRANSACTION COMPLETE'
***************************************************************
          DC F'0'
REPEAT    DC    X'00007500'         NUMBER OF TIMES TO REPEAT
MAXREAD   DC    X'00000600'         MAX READ COUNT
MAXREAD2  DC    X'00000005'         MAX READ COUNT
SDEND     DC    X'00000001'    LAST ONE
***************************************************************
SDLENG   DC    X'0030'    LENGTH OF TS RECORD
***************************************************************
         END
```

DB2PROG4

Example C-3 shows a listing of the DB2PROG4 code. The DB2PROG4 program is the same as the DB2PROG5, DB2PROG6, and DB2PROG7 programs and includes the following actions:

- **EXEC CICS RETRIEVE**
- **EXEC CICS START**

Example C-3 DB2PROG4 example (same as DB2PROG5, 6, and 7)

```
DFHEISTG DSECT
         EXEC SQL INCLUDE SQLCA
*
**********************************************
DFHEISTG DSECT
**********************************************
VVEMP     DS   CL80
EMPNO     DS   CL6
FIRSTNME  DS   CL12
MIDINIT   DS   CL1
LASTNAME  DS   CL15
WORKDEPT  DS   CL3
PHONENO   DS   CL4
HIREDATE  DS   CL10
JOB       DS   CL8
EDLEVEL   DS   HL2
SEX       DS   CL1
BIRTHDATE DS   CL10
SALARY    DS   PL3
BONUS     DS   PL3
COMM      DS   PL3
**********************************************
TERMNL         DC    F'0'
DATALEN        DS    F'0'
**********************************************
* THE FORMAT OF THE TS QUEUE RECORD PASSED TO
**********************************************
               DC    C'EISTG   '
MESSAGES DS    CL80              TEMP STORE
KEYNUM   DS    CL9               TEMP STORE
COMLEN   DS    1H                LENGTH OF C
         DS   0F
SQDWSTOR DS   (SQLDLEN)C     RESERVE STORAGE TO BE USED FOR SQLDSECT
SDARGDATA      DC    20F'0'
               DC    C'EISTG END'
SDARG          DSECT
SDREPEAT       DC    X'00000000'  NUMBER OF TIMES TO MAKE THE DB2 CALL
SDTERMID       DS    CL4       TERMINAL ID
```

```
SDREPCNT       DC    F'0'      CURRENT NUMBER TO BE ATTACHED
SDTRAN         DS    F'0'
SDASKTIM       DS    CL4      YES ISSUE ASKTIME,NO SKIP ASKTIMES
SDEMPNO        DC    2F'0'     EMPLOYEE NUMBER TO USE
*********************************************************
CWASTG         DSECT
CWACOUNT       DS    F             COUNTER TO UPDATE
*********************************************************
SQDWSREG EQU  7
RETREG   EQU   2                   SET UP REGISTER USAGE
R08      EQU   8
R9       EQU   9
R10      EQU   10
COUNT2   EQU   9
SDREPCTR EQU   5
COMPTR   EQU   4                   POINTER TO COMMAREA
*********************************************************
DB2PROG4 CSECT
DB2PROG4 AMODE 31
DB2PROG4 RMODE ANY
*****************************************************************
         MVC   TERMNL,EIBTRMID
* OBTAIN START DATA
         EXEC CICS RETRIEVE SET(R08) LENGTH(DATALEN)
         USING SDARG,R08
*****************************************************************
*        EXEC CICS PERFORM STATISTICS RECORD DISPATCHER
*
* SQL WORKING STORAGE
         LA    SQDWSREG,SQDWSTOR   GET ADDRESS OF SQLDSECT
         USING SQLDSECT,SQDWSREG   AND TELL ASSEMBLER ABOUT IT
*
         EXEC SQL                                                      *
               DECLARE DSN8710.EMP TABLE (                             *
               EMPNO                        CHAR(6),                   *
               FIRSTNME                     CHAR(12),                  *
               MIDINIT                      CHAR(1),                   *
               LASTNAME                     CHAR(15),                  *
               WORKDEPT                     CHAR(3),                   *
               PHONENO                      CHAR(4),                   *
HIREDATE                    DATE,
               JOB                          CHAR(8),
               EDLEVEL                      SMALLINT,
               SEX                          CHAR(1),
               BIRTHDATE                    DATE,
               SALARY                       DECIMAL,
               BONUS                        DECIMAL,
               COMM                         DECIMAL )
*
```

```
*******************************************************************
         L    6,SDREPEAT
AGAIN    DS   OH
         EXEC CICS ASKTIME
NOASKT   DS   OH
****************************************************************
 EXEC SQL SELECT EMPNO INTO :EMPNO FROM DSN8710.EMP
                  WHERE EMPNO='000100'
         BCT  6,AGAIN
*****************************************************************
*                                    INCREMENT COUNTER IN CWA
         EXEC CICS ADDRESS CWA(R10)
         USING CWASTG,R10
         L     R9,CWACOUNT
         LA    R9,1(R9)
         ST    R9,CWACOUNT
*****************************************************************
**  NOW START THE NEXT TASK                                  ****
**                                                           ****
         L     SDREPCTR,SDREPCNT     LOAD THE WORK REG
         LTR   SDREPCTR,SDREPCTR
         BZ    NOSTART
         S     SDREPCTR,SDEND        DECREMENT THE COUNTER
         ST    SDREPCTR,SDREPCNT     SAVE IT BACK FOR NEXT START
         EXEC CICS START TRANSID('DB24') INTERVAL(0)
                   FROM(SDARG) LENGTH(DATALEN)
*****************************************************************
*        EXEC CICS PERFORM STATISTICS RECORD DISPATCHER
NOSTART  DS OH
*        EXEC CICS SEND TEXT FROM(AREA) FREEKB
         EXEC CICS RETURN
         DS    OF
BIG_NUMBER DC  X'00000500'     XXX,XXX  1280 TIMES
AREA     DC    CL30'TRANSACTION COMPLETE'
****************************************************************
          DC F'0'
REPEAT    DC    X'00007500'         NUMBER OF TIMES TO REPEAT
MAXREAD   DC    X'00000600'         MAX READ COUNT
MAXREAD2  DC    X'00000005'         MAX READ COUNT
SDEND     DC    X'00000001'    LAST ONE
****************************************************************
SDLENG   DC    X'0030'    LENGTH OF TS RECORD
****************************************************************
END
```

DB2PROG8

Example C-4 shows a listing of the DB2PROG8 code. The DB2PROG8 program is the same as the DB2PROG9 and DB2PROGA programs and includes the following actions:

- **EXEC CICS RETRIEVE**
- **EXEC CICS START**
- **EXEC CICS WRITEQ TD**

Example C-4 (DB2PROG8 example (same as DB2PROG9 and DB2PROGA)

```
***********************************************************************
DFHEISTG DSECT
         EXEC SQL INCLUDE SQLCA
*
**********************************************
DFHEISTG DSECT
**********************************************
VVEMP     DS   CL80
EMPNO     DS   CL6
FIRSTNME  DS   CL12
MIDINIT   DS   CL1
LASTNAME  DS   CL15
WORKDEPT  DS   CL3
PHONENO   DS   CL4
HIREDATE  DS   CL10
JOB       DS   CL8
EDLEVEL   DS   HL2
SEX       DS   CL1
BIRTHDATE DS   CL10
SALARY    DS   PL3
BONUS     DS   PL3
COMM      DS   PL3
**********************************************
TERMNL         DC    F'0'
DATALEN        DS    F'0'
**********************************************
* THE FORMAT OF THE TS QUEUE RECORD PASSED TO
**********************************************
               DC    C'EISTG   '
MSG      DS    CL80
KEYNUM   DS    CL9                 TEMP STORE
COMLEN   DS    1H                  LENGTH OF C
QTEST    DS    CL8
         DS   0F
SQDWSTOR DS   (SQLDLEN)C     RESERVE STORAGE TO BE USED FOR SQLDSECT
SDARGDATA      DC    20F'0'
               DC    C'EISTG END'
```

```
SDARG          DSECT
SDREPEAT       DC    X'00000000'  NUMBER OF TIMES TO MAKE THE DB2 CALL
SDTERMID       DS    CL4       TERMINAL ID
SDREPCNT       DC    F'0'      CURRENT NUMBER TO BE ATTACHED
SDTRAN         DS    F'0'
SDASKTIM       DS    CL4      YES ISSUE ASKTIME,NO SKIP ASKTIMES
SDEMPNO        DC    2F'0'     EMPLOYEE NUMBER TO USE
*
*********************************************************
SQDWSREG EQU  7
RETREG   EQU   2                   SET UP REGISTER USAGE
R06      EQU   6
R08      EQU   8
RA       EQU   10
COUNT2   EQU   9
SDREPCTR EQU   5
COMPTR   EQU   4                   POINTER TO COMMAREA
*********************************************************
DB2PROG8 CSECT
DB2PROG8 AMODE 31
DB2PROG8 RMODE ANY
*****************************************************************
         MVC   TERMNL,EIBTRMID
* OBTAIN START DATA
         EXEC CICS RETRIEVE SET(R08) LENGTH(DATALEN)
         USING SDARG,R08
*****************************************************************
*        EXEC CICS PERFORM STATISTICS RECORD DISPATCHER
*
* SQL WORKING STORAGE
         LA    SQDWSREG,SQDWSTOR   GET ADDRESS OF SQLDSECT
         USING SQLDSECT,SQDWSREG   AND TELL ASSEMBLER ABOUT IT
*
         EXEC SQL                                                   *
               DECLARE DSN8710.EMP TABLE (                          *
               EMPNO                       CHAR(6),                 *
               FIRSTNME                    CHAR(12),                *
               MIDINIT                     CHAR(1),                 *
               LASTNAME                    CHAR(15),                *
               WORKDEPT                    CHAR(3),
               PHONENO                     CHAR(4),
               HIREDATE                    DATE,
               JOB                         CHAR(8),
               EDLEVEL                     SMALLINT,
               SEX                         CHAR(1),
               BIRTHDATE                   DATE,
               SALARY                      DECIMAL,
               BONUS                       DECIMAL,
               COMM                        DECIMAL )
```

```
*
******************************************************************
         L    6,SDREPEAT
AGAIN    DS   OH
*****************************************************************
         EXEC CICS READQ TS QUEUE(QTEST) SET(RA) LENGTH(COMLEN)
                   NOHANDLE
*****************************************************************
NOASKT   DS   OH
 EXEC SQL SELECT EMPNO INTO :EMPNO FROM DSN8710.EMP
                  WHERE EMPNO='000140'
*
         BCT  6,AGAIN
*
*****************************************************************
         MVC   MSG,=CL80'DB2PROG8 ENDED'
EXEC  CICS WRITEQ TD QUEUE(=C'THDS') FROM(MSG) NOHANDLE
*****************************************************************
**  NOW START THE NEXT TASK                                  ****
**                                                           ****
         L     SDREPCTR,SDREPCNT     LOAD THE WORK REG
         LTR   SDREPCTR,SDREPCTR
         BZ    NOSTART
         S     SDREPCTR,SDEND        DECREMENT THE COUNTER
         ST    SDREPCTR,SDREPCNT     SAVE IT BACK FOR NEXT START
         EXEC CICS START TRANSID('DB28') INTERVAL(0)                   *
                   FROM(SDARG) LENGTH(DATALEN)
*****************************************************************
NOSTART  DS OH
*****************************************************************
         EXEC  CICS RETURN
         DS    OF
BIG_NUMBER DC  X'00000500'     XXX,XXX  1280 TIMES
AREA     DC    CL30'TRANSACTION COMPLETE'
***************************************************************
          DC F'0'
REPEAT    DC    X'00007500'         NUMBER OF TIMES TO REPEAT
MAXREAD   DC    X'00000600'         MAX READ COUNT
MAXREAD2  DC    X'00000005'         MAX READ COUNT
SDEND     DC    X'00000001'    LAST ONE
***************************************************************
SDLENG   DC    X'0030'    LENGTH OF TS RECORD
***************************************************************
         END
```

PLANEXIT

Example C-5 shows the code used for PLANEXIT.

Example C-5 PLANEXIT code

```
TITLE 'PLANEXIT - DB2 CICS ATTACH, DYNAMIC PLAN ALLOCATION EXIT'
*
PLANEXIT AMODE 31                        CAN ADDR STORAGE ABOVE THE LINE
PLANEXIT RMODE ANY                       CAN RUN ABOVE THE LINE
PLANEXIT DFHEIENT CODEREG=(3),EIBREG=(11),DATAREG=(13)
*
A100     EQU   *                        ADDRESS COMMAREA
         USING CPRMPARM,R2
         L     R2,DFHEICAP
         EXEC  CICS ASSIGN USERID(USERID) NOHANDLE
*
RETURN   EQU   *                        RETURN TO CALLER
         EXEC  CICS RETURN
*
*
         LTORG
*
*                                       WORKING STORAGE
         DFHEISTG
USERID   DS    1CL8
         DFHEIEND
*
*
               DSNCPRMA                 COMMAREA
*
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
```

```
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
         END   PLANEXIT
```

EXITENBL

Example C-6 shows the example code used to enable all exits.

Example C-6 Program to enable all exits

```
TITLE 'ENABLE - ENABLE EXITS FOR SAMPLE APPLICATION'
*
EXITENBL AMODE 31                       CAN ADDR STORAGE ABOVE THE LINE
EXITENBL RMODE ANY                      CAN RUN ABOVE THE LINE
EXITENBL DFHEIENT CODEREG=(3),EIBREG=(11),DATAREG=(13)
*
A100     EQU   *
         EXEC  CICS ENABLE PROGRAM(=CL8'XXXEI')                        X
                           EXIT(=CL8'XEIIN')                           X
                           START
*
         EXEC  CICS ENABLE PROGRAM(=CL8'XXXEI')                        X
                           EXIT(=CL8'XEIOUT')                          X
                           START
*
         EXEC  CICS ENABLE PROGRAM(=CL8'XXXRMI')                       X
                           EXIT(=CL8'XRMIIN')                          X
                           START
*
         EXEC  CICS ENABLE PROGRAM(=CL8'XXXRMI')                       X
                           EXIT(=CL8'XRMIOUT')                         X
                           START
*
         EXEC  CICS ENABLE PROGRAM(=CL8'XXXTS')                        X
                           EXIT(=CL8'XTSQRIN')                         X
                           GALENGTH(=H'64')                            X
                           START
*
RETURN   EQU   *                       RETURN TO CALLER
         EXEC  CICS RETURN
*
         LTORG
*                                      WORKING STORAGE
```

```
         DFHEISTG
         DFHEIEND
*
         END   EXITENBL
```

XXXEI exit

Example C-7 shows the source code for the XXXEI exit.

Example C-7 Source code for the XXXEI exit

```
DFHUEXIT TYPE=EP,ID=(XEIIN,XEIOUT)
          COPY  DFHTSUED             COMMAND LEVEL PLIST DEFINITIONS
 *
 DFHEISTG DSECT                      WORKING STORAGE
 RETCODE      DS XL4
 RESPONSE     DS F
 *
 XXXEI    DFHEIENT
 XXXEI    AMODE 31
 XXXEI    RMODE ANY
          LR    R2,R1                DFHUEPAR PLIST PROVIDED BY CALLER
          USING DFHUEPAR,R2          ADDRESS UEPAR PLIST
 *
          LA    R15,UERCNORM         SET OK RESPONSE
          ST    R15,RETCODE            IN WORKING STORAGE
 *
 RETURN   EQU   *
          L     R15,RETCODE          FETCH RETURN CODE
          DFHEIRET RCREG=15          RETURN TO CICS
 *
 R0       EQU   0
 R1       EQU   1
 R2       EQU   2
 R3       EQU   3
 R4       EQU   4
R5       EQU   5
 R6       EQU   6
 R7       EQU   7
 R8       EQU   8
 R9       EQU   9
 R10      EQU   10
 R11      EQU   11
 R12      EQU   12
 R13      EQU   13
```

```
R14      EQU   14
R15      EQU   15
         END   XXXEI
```

XXXRMI exit

Example C-8 shows the source code for the XXXRMI exit.

Example C-8 Source code for the XXXRMI exit

```
DFHUEXIT TYPE=EP,ID=(XRMIIN,XRMIOUT)
          COPY  DFHTSUED             COMMAND LEVEL PLIST DEFINITIONS
 *
 DFHEISTG DSECT                      WORKING STORAGE
 RETCODE      DS XL4
 RESPONSE     DS F
 *
 XXXRMI   DFHEIENT
 XXXRMI   AMODE 31
 XXXRMI   RMODE ANY
          LR    R2,R1                DFHUEPAR PLIST PROVIDED BY CALLER
          USING DFHUEPAR,R2          ADDRESS UEPAR PLIST
 *
          LA    R15,UERCNORM         SET OK RESPONSE
          ST    R15,RETCODE            IN WORKING STORAGE
 *
 RETURN   EQU   *
          L     R15,RETCODE          FETCH RETURN CODE
          DFHEIRET RCREG=15          RETURN TO CICS
 *
 R0       EQU   0
 R1       EQU   1
 R2       EQU   2
 R3       EQU   3
 R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
```

```
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END   XXXRMI
```

XXXTS exit

Example C-9 shows the source code used for the XXXTS exit.

Example C-9 Source code for the XXXTS exit

```
DFHUEXIT TYPE=EP,ID=(XTSQRIN)
*
GWA      DSECT                       GLOBAL WORK AREA
GWACOUNT DS    F
GWAL     EQU   *-GWA
*
XXXTS    CSECT
XXXTS    AMODE 31
XXXTS    RMODE ANY
         SAVE  (14,12)               SAVE REGS
         LR    R12,R15               SET-UP BASE REGISTER
         USING XXXTS,R12             ADDRESSABILITY
         LR    R2,R1                 DFHUEPAR PLIST PROVIDED BY CAL
         USING DFHUEPAR,R2           ADDRESS UEPAR PLIST
         L     R8,UEPGAA             GET GWA ADDRESS
         USING GWA,R8                ADDRESSABILITY
*
GWA_CHECK_LENGTH EQU *
         L     R10,UEPGAL            LOAD ADDRESS OF LENGTH OF GWA
         LH    R9,0(,R10)            LOAD LENGTH OF GWA
         LA    R10,GWAL              LOAD EXPECTED LENGTH OF GWA
         CLR   R9,R10                IS IT BIG ENOUGH?
         BNL   GWAUPDT               YES, CAN UPDATE DATA IN GWA
GWAERROR EQU   *
         B     RETURN                GWA NOT BIG ENOUGH, EXIT
*
GWAUPDT  EQU   *
         L     R6,GWACOUNT           GET THE COUNTER
         LA    R6,1(R6)              INCREMENT
         ST    R6,GWACOUNT             AND STORE
         B     RETURN                EXIT
*
RETURN   EQU   *
         L     R13,UEPEPSA           ADDRESS OF EXIT SAVE AREA
         RETURN (14,12),RC=UERCNORM RESTORE REGS AND RETURN
*
```

```
         LTORG
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END   XXXTS
```

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

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *IBM CICS Explorer*, SG24-7778
- *IBM CICS Interdependency Analyzer*, SG24-6458
- *CICS Performance Analyzer*, SG24-6063

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *IBM CICS Performance Analyzer for z/OS Getting Started Guide*, SC34-7155-01
- *IBM CICS Performance Analyzer for z/OS User's Guide*, SC34-7153-01
- Publications for CICS TS V2
  - *CICS Application Programming Guide*, SC34-6231
  - *CICS Application Programming Reference*, SC34-6232
  - *CICS Customization Guide*, SC34-6227
  - *CICS DB2 Guide*, SC34-6252
  - *CICS Performance Guide*, SC34-6247
  - *CICS System Programming Reference*, SC34-6233

© Copyright IBM Corp. 2004, 2012. All rights reserved.

- Publications for CICS TS V3
  - *CICS Transaction Server for z/OS CICS Application Programming Guide*, SC34-6433
  - *CICS Transaction Server for z/OS CICS System Programming Reference*, SC34-6435
  - *CICS Transaction Server for z/OS Performance Guide*, SC34-6452
  - *CICS Transaction Server for z/OS V3.1 CICS Application Programming Reference*, SC34-6434
  - *CICS Transaction Server for z/OS V3.1 CICS Operations and Utilities Guide*, SC34-6431
- *z/Architecture Principles of Operation*, SA22-7832
- *z/OS Communications Server IP CICS Sockets Guide Version 1 Release 7*, SC31-8807
- *z/OS MVS Initialization and Tuning Reference,* SA22-7592
- *CICS IA User's Guide and Reference, Version 3 Release 2*, SC34-7211

Online resources

These websites are also relevant as further information sources:

- CICS home page

  `http://www.ibm.com/cics`
- CICS Transaction Server Version 4.2 Information Center

  `http://publib.boulder.ibm.com/infocenter/cicsts/v4r2/index.jsp`

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM Redbooks

Threadsafe Considerations for CICS

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Threadsafe Considerations for CICS

New threadsafe options in CICS Transaction Server for z/OS V4.2

The value of threadsafe applications in a CICS environment

CICS Tools to help migrate applications to be threadsafe

Beginning with IBM CICS Version 2, applications can run on task control blocks (TCBs) apart from the quasi-reentrant (QR) TCB, which has positive implications for improving system throughput and for implementing new technologies inside of CICS. Examples of implementing new technologies include using the IBM MVS Java virtual machine (JVM) inside CICS and enabling listener tasks written for other platforms to be imported to run under CICS.

The newest release, CICS Transaction Server for z/OS (CICS TS) V4.2, includes scalability enhancements so that you can perform more work more quickly in a single CICS system. The advantage of this enhancement is that you can increase vertical scaling and decrease the need to scale horizontally, reducing the number of regions that are required to run the production business applications. The scalability enhancements in CICS TS V4.2 fall into two broad areas, which are increased usage of open transaction environment (OTE) and of 64-bit storage.

This IBM Redbooks publication is a comprehensive guide to threadsafe concepts and implementation for IBM CICS. This book explains how systems programmers, applications developers, and architects can implement threadsafe applications in an environment. It describes the real-world experiences of users, and our own experiences, of migrating applications to be threadsafe. This book also highlights the two most critical aspects of threadsafe applications: system performance and integrity.

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

SG24-6351-04 ISBN 0738436690