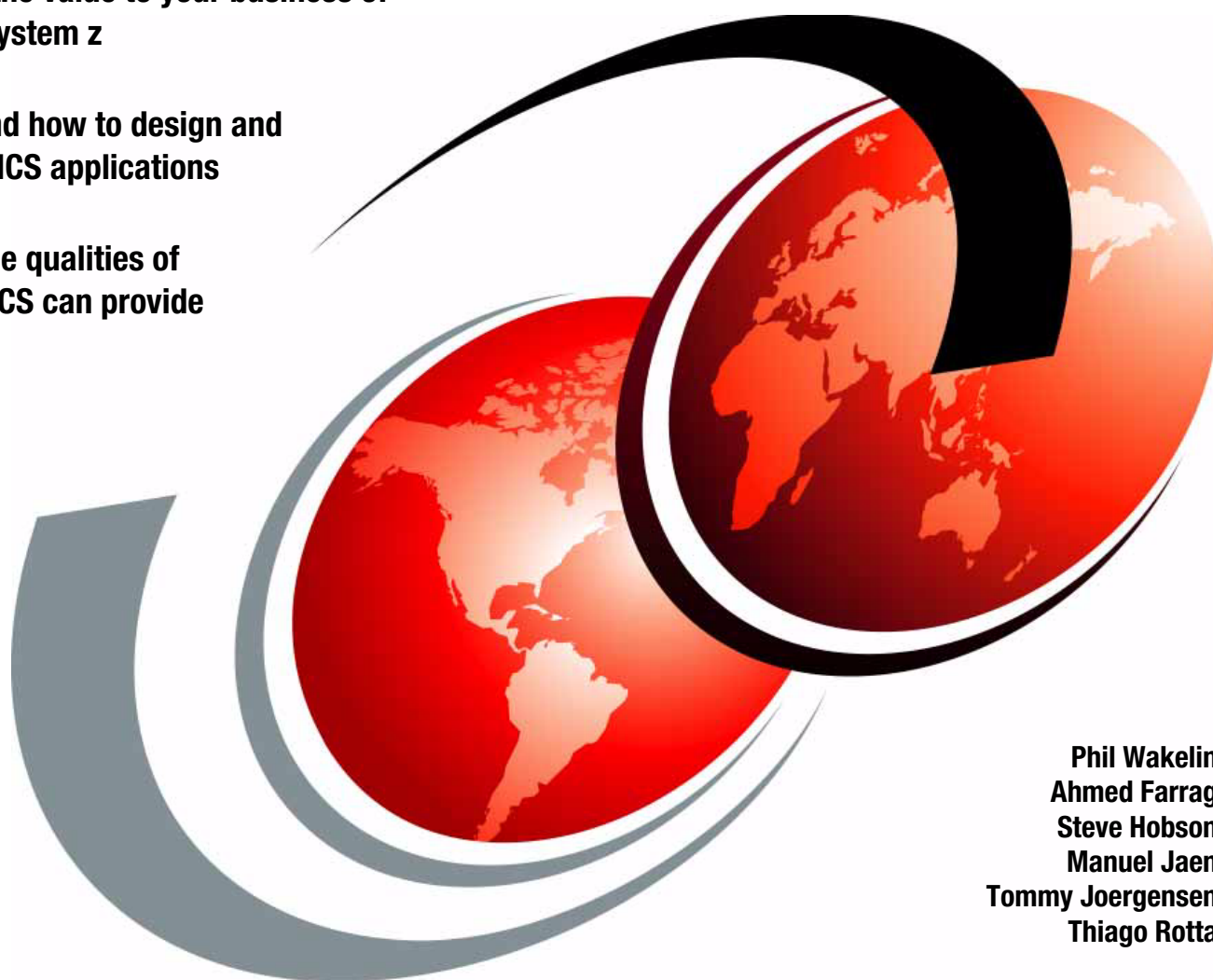


Architect's Guide to IBM CICS on System z

Discover the value to your business of
CICS on System z

Understand how to design and
develop CICS applications

Explore the qualities of
service CICS can provide



Phil Wakelin
Ahmed Farrag
Steve Hobson
Manuel Jaen
Tommy Joergensen
Thiago Rotta

Redbooks



International Technical Support Organization

Architect's Guide to IBM CICS on System z

November 2012

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

First Edition (November 2012)

This edition applies to Version 5, Release 1 of CICS Transaction Server for z/OS.

© Copyright International Business Machines Corporation 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

Notices	vii
Trademarks	viii
Preface	ix
The team who wrote this book	ix
Now you can become a published author, too!	xi
Comments welcome	xi
Stay connected to IBM Redbooks	xii
Part 1. Why CICS	1
Chapter 1. Business value of CICS	3
1.1 Today's enterprise computing	4
1.1.1 Reliability	4
1.1.2 Business agility	5
1.1.3 Flexibility	5
1.1.4 Cost effectiveness	5
1.2 CICS: The smart choice	6
1.2.1 Simplifying the environment	6
1.2.2 Application design	8
1.2.3 Reliability	12
1.2.4 Creating a market advantage with CICS TS	14
1.3 CICS market presence	15
1.4 Cost effectiveness	17
1.4.1 Understanding the cost of CICS	17
1.4.2 Cost benefits of CICS qualities of service	18
1.5 The future of CICS	20
Chapter 2. CICS capabilities	23
2.1 Agile development & deployment	24
2.1.1 CICS application programming interfaces	24
2.1.2 Programming languages	25
2.1.3 Java framework	26
2.1.4 Development and deployment lifecycle	26
2.1.5 CICS and cloud computing	27
2.2 Flexible integration options	27
2.3 Proven integrity	29
2.3.1 Transaction integrity	30
2.3.2 Data integrity	32
2.4 Secured environment	37
2.4.1 CICS security integration	38
2.4.2 End-to-end security	39
2.5 Highly available and scalable processing	40
2.5.1 High availability	40
2.5.2 Scalability	40
2.5.3 Techniques and facilitators	41
2.5.4 z/OS Workload Manager	42
2.6 Simplified administration with CICSplex System Manager	44

Part 2. Developing CICS applications	47
Chapter 3. CICS application architecture	49
3.1 Application architecture	50
3.1.1 CICS application architecture principles	50
3.1.2 Reusable modular model	52
3.2 Adapter layer	54
3.2.1 Message serialization adapters	55
3.2.2 Adapter technology	57
3.3 Presentation layer	58
3.3.1 The 3270 terminals	59
3.3.2 Web support	59
3.3.3 Liberty JSP support	59
3.3.4 State management	59
3.4 Integration layer	62
3.4.1 Service flows	62
3.4.2 Enterprise service bus	63
3.5 Business layer	64
3.5.1 Business rules	64
3.5.2 Event processing	66
3.5.3 Transaction integrity	67
3.5.4 Threadsafe programming	68
3.6 Data access layer	69
3.6.1 Data stores	70
3.6.2 Batch integration	70
Chapter 4. CICS application modernization	73
4.1 Introduction to application modernization	74
4.2 Business rules modernization overview	74
4.2.1 Analyzing existing CICS applications	76
4.2.2 Introduction to the business rules mining process	77
4.2.3 Preparation activities for rules mining process	79
4.2.4 Rule mining activities	80
4.3 Event processing in CICS	82
4.3.1 Why is events processing important to today's business?	83
4.3.2 Key concepts about CICS event processing	84
4.3.3 How CICS events work	85
4.3.4 Events consumers	86
4.3.5 Designing event driven solutions for CICS	88
4.3.6 Business scenarios for CICS events	90
4.4 Reusing business logic from CICS applications	91
4.4.1 Migration from COMMAREAs to channels and containers	91
4.4.2 CICS integration options	92
4.4.3 Java Connector Architecture through CICS Transaction Gateway	95
4.4.4 JCA through WebSphere z/OS Optimized Local Adapters (WOLA)	98
4.4.5 CICS web support	99
4.4.6 Asynchronous messaging	100
4.4.7 CICS sockets	101
4.4.8 Integration options comparison	102
4.5 Modernizing data layer using CICS VSAM Transparency	103
4.6 Modernizing 3270 presentation layer	104
4.6.1 Front End Programming Interface (FEPI)	105
4.6.2 Rational Host Access Transformation Services (HATS)	106

4.6.3 CICS Link3270 bridge	106
4.6.4 Service Flow Modeler (SFM).	107
Chapter 5. Application development lifecycle	111
5.1 CICS application development lifecycle	112
5.1.1 Eclipse-based environment.	113
5.1.2 CICS Explorer.	114
5.2 Procedural application development lifecycle	116
5.2.1 Application analysis and design	116
5.2.2 Development and build	117
5.2.3 Functional testing	118
5.2.4 Performance testing and tuning	119
5.2.5 Deployment to production.	119
5.2.6 Service delivery management.	120
5.3 Java application development and deployment lifecycle	122
5.3.1 Basic concepts of Java development in CICS	122
5.3.2 Java application development on CICS	126
5.3.3 Debugging and troubleshooting Java applications running on CICS	131
5.3.4 CICS diagnostics tools for Java	133
5.3.5 Profiling and tuning CICS Java applications	134
Chapter 6. Exploring the CICS development tools	137
6.1 Rational tools	138
6.1.1 Rational Asset Analyzer	138
6.1.2 Rational Developer for System z.	143
6.1.3 Rational Development and Test Environment for System z	148
6.1.4 Rational Team Concert.	150
6.2 CICS and z/OS tools	153
6.2.1 CICS Interdependency Analyzer (IA)	153
6.2.2 z/OS Problem Determination Tools.	156
6.2.3 CICS Performance Analyzer for z/OS.	164
6.2.4 CICS Application Performance Analyzer for z/OS	166
6.2.5 CICS Configuration Manager for z/OS	168
6.2.6 CICS Deployment Assistant for z/OS	170
6.2.7 IBM Session Manager for z/OS.	172
Part 3. CICS system design	175
Chapter 7. High availability and continuous operation	177
7.1 Availability requirements	178
7.1.1 Types of availability requirement.	178
7.1.2 Unplanned outages.	179
7.1.3 Planned outages	181
7.2 z/OS Parallel Sysplex and CICSplex	184
7.2.1 z/OS Parallel Sysplex	184
7.2.2 CICSplex.	186
7.3 Data availability	188
7.3.1 CICS data requiring high availability	188
7.3.2 Application data requiring high availability	188
7.3.3 WebSphere MQ	191
7.4 Application availability	195
7.4.1 Transaction deadlocks	195
7.4.2 Application errors: Damage limitation	195
7.4.3 Isolating applications in separate CICS regions	198

Chapter 8. Scalability and workload management.	201
8.1 Building a scalable enterprise system.	202
8.1.1 Scalable architecture	202
8.2 Single region considerations.	202
8.2.1 Virtual storage constraint relief	203
8.2.2 CPU contention.	203
8.2.3 Managing the workload in a single region.	204
8.3 Multiregion considerations	205
8.3.1 Client access.	206
8.3.2 Data access	206
8.3.3 Other multiregion considerations	208
8.3.4 Global CICS ENQ/DEQ.	210
8.3.5 Considerations for multi-LPAR CICS regions	211
8.3.6 Communication options	211
8.3.7 Workload management in a CICSplex	215
8.3.8 Workload separation.	217
8.3.9 Preparing for workload management	218
Chapter 9. Security and auditing	219
9.1 Security requirements	220
9.2 Authentication and identification	220
9.2.1 Considerations	220
9.2.2 Identification	221
9.2.3 CICS and identity propagation: The end-to-end security solution.	223
9.3 Authorization	225
9.3.1 Securing access to resources.	225
9.3.2 Securing access to DB2	228
9.3.3 Intercommunication security	229
9.3.4 Securing network access	230
9.4 Integrity	232
9.4.1 System z security and integrity capabilities.	232
9.4.2 CICS storage protection and transaction isolation	233
9.5 Confidentiality	233
9.6 Cryptography.	234
9.7 Auditing and non-repudiation	237
9.7.1 Auditing unauthorized access to CICS resources.	237
9.7.2 Auditing updates to CICS resource definitions	237
9.7.3 Auditing updates to CICS application programs	238
9.7.4 Non-repudiation	238
Part 4. Appendix and related publications	239
Appendix A. Supported standards	241
Networking standards	242
Web service standards.	242
Security standards	243
Other standards and frameworks.	243
Related publications	245
IBM Redbooks	245
Other publications	245
Online resources	246
Help from IBM	247

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 grant 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

COPYRIGHT LICENSE:

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

Trademarks

IBM, the IBM logo, and ibm.com 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:

AIX®	IMS™	Rational Team Concert™
CICS Explorer®	Informix®	Rational®
CICSPIlex®	Language Environment®	Redbooks®
CICS®	Lotus Notes®	Redbooks (logo)  ®
DataPower®	Lotus®	Symphony®
DB2®	MVS™	System Storage®
developerWorks®	NetView®	System z®
DS8000®	Notes®	Tivoli®
GDPS®	OMEGAMON®	VTAM®
Geographically Dispersed Parallel Sysplex™	Parallel Sysplex®	WebSphere®
IA®	QMFTM	z/OS®
IBM®	Query Management Facility™	zEnterprise®
	RACF®	zSeries®

The following terms are trademarks of other companies:

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

Microsoft, Windows, 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.

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

Preface

IBM® CICS® Transaction Server (CICS TS) has been available in various guises for over 40 years, and continues to be one of the most widely used pieces of commercial software. This IBM Redbooks® publication helps application architects discover the value of CICS Transaction Server to their business.

This book can help architects understand the value and capabilities of CICS Transaction Server and the CICS tools portfolio. The book also provides detailed guidance on the leading practices for designing and integrating CICS applications within an enterprise, and the patterns and techniques you can use to create CICS systems that provide the qualities of service that your business requires.

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, Hursley Center.



Phil Wakelin works for IBM UK in Hursley, and is a member of the CICS strategy and planning teams. He has worked with many CICS technologies for the last 20 years, and is responsible for new functionality in the areas of CICS interconnectivity, CICS integration and Java support. He is the author of many white papers, SupportPacs and IBM Redbooks in the area of CICS Transaction Processing.



Ahmed Farrag is an accredited IT Architect and has worked at IBM in Egypt since 2005. Ahmed holds a degree in electronics and communications from Cairo University. His areas of expertise include CICS application modernization, business process and rules management, and enterprise content management. During his work with IBM, Ahmed has led many services projects across various industries including government, banking, and insurance. He has participated, as a speaker, in several IBM and external conferences and technical briefings such as: IBM developerWorks® EGL technical briefing in Moscow and Zurich in February 2008, IBM Rational® Software Development Conference in Sharm El-Sheikh in June 2008, and SQLAdria conference on enterprise modernization in Croatia in November 2009.



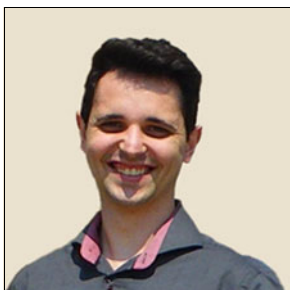
Steve Hobson works for IBM Hursley in the UK and is a member of the CICS strategy and planning teams and of the High Level Assembler (HLASM) development team. He has 45 years of experience in software development, in particular with TPF and ALCS, IBM WebSphere® MQ, WebSphere Application Server, and most recently CICS TS and HLASM. Steve is also an IBM Master Inventor.



Manuel Jaen is an IT Architect in Spain. He has 15 years of experience as an IT Consultant, IT Architect, Application Developer, and IBM System z® Technical Presales Specialist. He holds the degree of Computer Engineer from Universidad de Deusto and a Master's degree in IT Strategy Management from Universidad Politecnica Cataluña. His areas of expertise include HR Access package, System z application development, multi-platform integration, package integration and migration, business application transformation and modernization, SOA, and IBM zEnterprise®. Manuel has participated in projects across a broad range of industries, including the transportation, aeronautical, and chemical industries, and especially in the banking and financial services sector. Manuel was also an author for the Redbooks publication about zEnterprise business value: *Business Value Assessment Methodology for Business Application Deployment on zEnterprise*, ZG24-1781.



Tommy Joergensen is a Client Technical Specialist working for IBM Software Group in the Nordics, and has responsibilities in the System z software sales organization with a focus on CICS and related products. Tommy has participated in several Redbooks residencies, all with focus on CICS. He also spent three years working on CICS in IBM Hursley in the UK and two years on various customer activities in IBM Products and Solutions Support Centre in Montpellier, France. Tommy has more than 35 years of experience working in CICS.



Thiago Rotta has a Bachelor degree in Computer Engineering and has worked for IBM in Brazil since 2005. As an IBM Architect, specializing in application architectures, he has over eight years of experience. His responsibility spans the end-to-end lifecycle of IT projects, providing technical and architectural leadership to ensure overall integrity of the IBM USA post sales applications portfolio. Thiago has also merged his expertise as an Architect to the intellectual property arena and holds the title of IBM Master Inventor in Brazil. He has a solid background and experience in leading innovation, aiming to produce patent applications.

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

- ▶ Martin Keen, International Technical Support Organization, Raleigh Center, for coordinating the project
- ▶ John Knutson, IBM Hursley who was the inspiration behind the book
- ▶ Diane Sherman, Stephen Smith, Shawn Tooley, and Debbie Willmschen, International Technical Support Organization, for editing the drafts
- ▶ Linda Robinson, International Technical Support Organization, for providing graphics support
- ▶ Innes Read, IBM Somers, for explaining the business value of CICS
- ▶ Manuel Campos, Fernando Garcia de la Filia, Alberto Gonzalez, from IBM Spain, for providing helpful material and scenarios
- ▶ Marie-Therese Bouedo, IBM France for providing helpful material and scenarios
- ▶ Shane Babey, Chris Baker, Mark Cocker, Ian Mitchell, Catherine Moxey, Colin Penfold, Satish Tanna, John Tilling, Matthew Whitbourne, from IBM Hursley, for reviewing drafts
- ▶ Yann Kindelberger and Nigel Williams, IBM Montpellier, France, for reviewing drafts

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>



Part 1

Why CICS

For over 40 years, CICS Transaction Server (CICS TS) has been available in various forms, and continues to be one of the most widely used components of commercial software. Most people, however, are unaware of CICS, although they probably use it several times a week for almost every electronic transaction they undertake.

In this part, we look at the business value that CICS provides. We show how its technical capabilities can be rapidly used by a wide variety of industries, to more quickly and easily develop a modern and flexible online transaction processing solution, which integrates with a wide variety of applications.



Business value of CICS

Today, many enterprises are faced with the challenge of creating and extending applications quickly and effectively to meet demands of their clients. There is a need for rapid business agility adoption and also a pressure from business executives for cost reduction.

This chapter provides an introduction to the business values and benefits of using CICS Transaction Server (CICS TS) on IBM z/OS®. We describe how CICS can be used as a powerful transaction server that meets the transaction-processing needs of enterprises, seeking to redefine their business applications in each of these ways:

- ▶ Ensuring *reliability*
- ▶ Decreasing *development and operational lifecycles*
- ▶ Making the IT environment more *simple*
- ▶ Being more *cost effective*

1.1 Today's enterprise computing

A computing platform is defined as hardware and software that work together to provide an environment for application execution. And increasingly, businesses across all industries have heterogeneous computing platforms at the base of their operations. So the selection of new platforms for new applications has become a critical decision for most enterprises.

Platform selection has traditionally been based on an organization's technology preference. But now that there are common standards that enable applications to be delivered as services, this is no longer the case. Interoperable IT services can now be delivered regardless of the underlying technology implementation.

Business drivers are also a factor. Platform selection must support the need for the business to grow efficiently and respond quickly to new market demands. This support must be done without adding undue complexity to the IT environment, while reducing the overall cost of processing.

Frequently and increasingly, these considerations favor fully solid, virtualized platforms such as the IBM System z mainframe running modern and existing applications in IBM CICS Transaction Server for z/OS. In fact, CICS TS is now nearly ubiquitous; it is installed in almost every System z environment that is responsible for running and managing business transactions.

Based on business drivers, enterprises are asking important questions:

- ▶ How can we be more reliable?
- ▶ How can we be more agile in responding to global market needs?
- ▶ How can we be more flexible?
- ▶ How can we continue doing business and be more cost effective?

In this section, we provide a brief explanation of these key business drivers and how they are becoming such important decision factors for enterprise IT departments.

1.1.1 Reliability

The need for reliable application execution is putting increased strain on IT environments. The following topics are important when considering reliability.

Operational reliability

The IT environment must be available when the business needs it. Mission-critical application availability is directly related to business success.

Downtime, whether planned or unplanned, is not only unwelcome, it is costly. Downtime statistics are staggering and range from tens of thousands of dollars to multiple millions of dollars lost per hour when applications are not available. And, although the direct financial impact is significant, the damage can extend well beyond the financial realm into key areas of customer loyalty, market competitiveness, corporate reputation, and regulatory compliance.

Consistency of performance

Even when strong fluctuations in demand occur, the supporting IT systems must remain resilient. Infrastructures must be adequately provisioned and applications must be adequately developed and structured to handle unpredictable behavior, such as seasonal increases in transaction volume.

Integrity and security

Organizations are becoming more collaborative in the way they share information. Data integrity, therefore, is crucial in any environment where the information comes from multiple systems. This integrity is especially true with systems based on different technologies, or which are controlled by partners outside the direct control of the company.

Information must not be tampered with, and transactions that span multiple systems must be controlled. The applications and infrastructure must provide the level of security that is required for the business processes that are involved.

1.1.2 Business agility

Enterprises need to respond rapidly to client needs, and clients today know what they want, when and how they want it, and how much they are willing to pay for it. In the open and global market of today, clients can easily switch suppliers when the product and service no longer satisfies those clients.

Shifting client demands requires enterprises to respond quickly. Most new business solutions need to be up and running in a few weeks, not months or years. So the enterprise must know how to reuse its assets and implement solutions quickly to respond to market demands.

To build a solution from the start, without agile tools and without reusing assets, takes a long time. To succeed in a competitive market, businesses need to simplify their transaction processing architecture to improve their agility, reduce costs, and eliminate inefficiencies.

1.1.3 Flexibility

Today, enterprises must be more flexible than they used to be. Products and services must be capable of being customized quickly. And because mergers and acquisitions are common business occurrences, the ability to rapidly integrate different systems is increasingly important.

More than ever, businesses today need integrated processes. The organization must be able to analyze individual parts of specific business processes, and change or outsource them as necessary, without interfering with the overall processes.

1.1.4 Cost effectiveness

The IT industry as a whole is challenged with improving business flexibility and minimizing risk. At the same time, executives are expected to hold to constant or decreasing budgets. Across the industry there is a deep desire to eliminate expense where possible.

Over the past approximately 15 years, the cost dynamics of supporting corporate IT infrastructures have changed significantly. In particular, hardware costs have decreased significantly, while people and software costs have increased.

To control people and software costs, executives increasingly look to centralized transaction processing systems. Centralized infrastructures and processes enable shared services that, in turn, provide economies of scale. Optimizing processes and reusing resources also can improve business efficiency and flexibility. Standardization is another key to cutting costs, as is automation, where it makes sense.

1.2 CICS: The smart choice

CICS has been running in major enterprise environments around the world for more than 40 years. In that time, CICS has changed the field of transaction processing by providing innovative solutions to the most important new business drivers as they have emerged.

CICS is IBM transaction processing software that primarily runs on IBM mainframes under z/OS. It controls the interactions between applications and users (from a small number of users to thousands of them). CICS applications offer high availability and easy scalability. They have built-in redundancy, using various client interfaces ranging from terminals and web browsers to web services.

Transaction processors such as CICS are purpose-built to efficiently run business programs in secure, multi-user environments. Almost any computer system can be designed to process transactions, but only a purpose-built system can handle extremely large volumes of transactions efficiently.

Many organizations are running CICS applications that are 10 or 20 years old or older. Yet these applications still provide robust levels of service because CICS provides tools and capabilities to modernize even decades-old applications (see Chapter 4, “CICS application modernization” on page 73).

The following sections describe the key reasons that CICS TS has been and remains an ideal choice for enterprise transactional processing.

1.2.1 Simplifying the environment

Modern IT environments consist of multiple hardware architectures, a variety of software operating environments, complex internal and external networking, and applications that must interoperate to provide end-to-end service to clients and internal users.

The flexibility to meet market demands and seize new business opportunities is a must for businesses today. So, their IT environments must be capable of being changed on demand, as new needs arise. They must be able to easily communicate and integrate with other IT systems within the company, and also those of external partners, suppliers, and consumers.

Expanding the environment with new applications can raise concerns about existing applications and whether they will continue working properly. Adding a new application in a complex environment involves creating interfaces with current applications, databases, and networks. It also increases the complexity of the environment, adding network traffic and others factors that must be considered to ensure that deploying the new application will not degrade the overall performance of the environment in terms of efficiency, scalability, availability, security, and reliability.

CICS provides a transactional application server and complementary management tools that can execute large volumes of mixed-language applications with optimal performance and operational efficiency. CICS ensures that enterprise policies such as request priorities and security standards can be managed using a single perspective. CICS configurations can immediately scale to accommodate unpredictable peak workloads, while retaining a single point of control for managing resource utilization throughout the enterprise.

Figure 1-1 shows a complex, heterogeneous environment where adding a new application requires new interface connections.

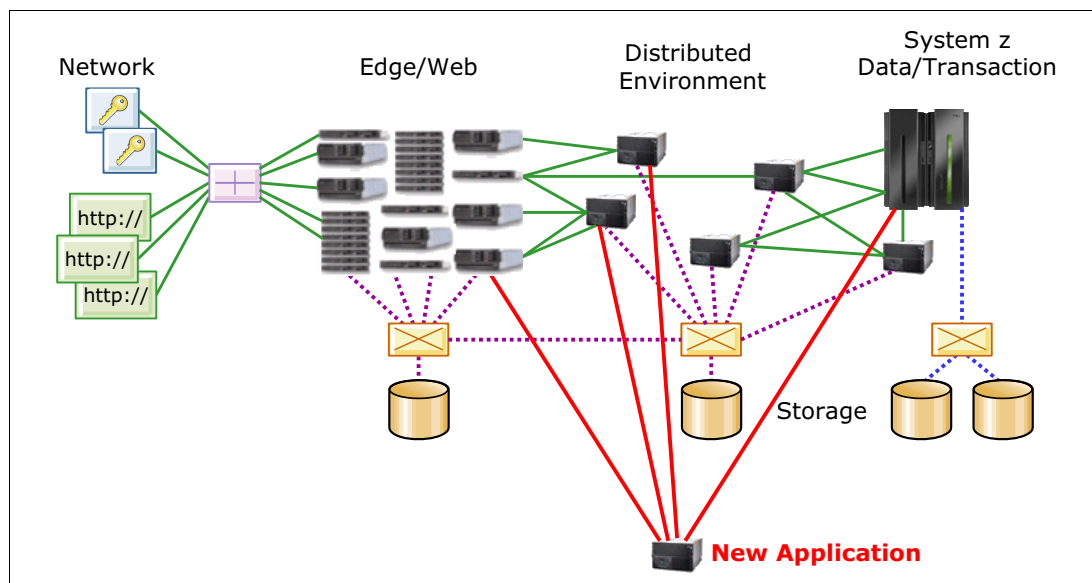


Figure 1-1 Complex environment

CICS TS for z/OS can help organizations move their distributed applications to the System z platform, simplifying the environment and making it more reliable, scalable, and cost effective. CICS TS can reliably run thousands of CICS applications in a single system, so even a complex environment with dozens or even hundreds of physically distributed servers can be consolidated into a single System z server.

Figure 1-2 shows how CICS TS can simplify an application environment to make changes or adjustments easier and faster.

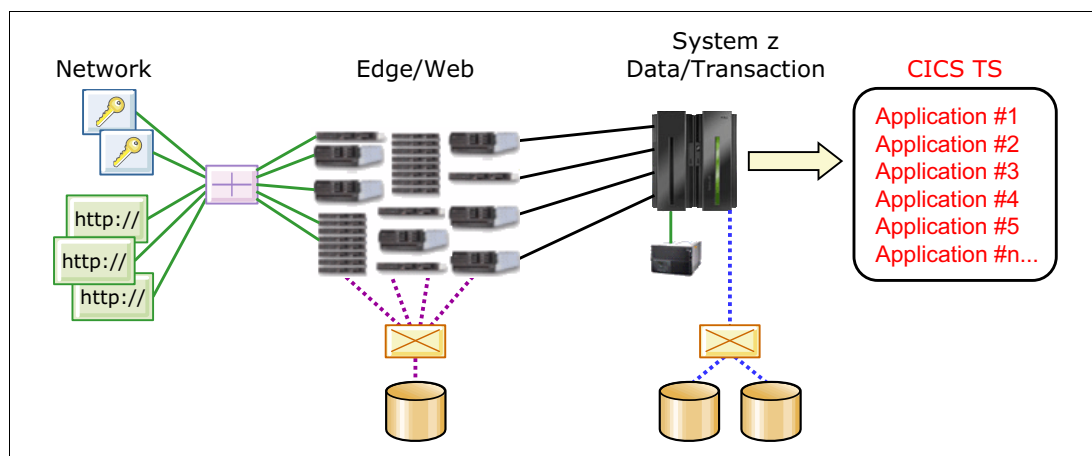


Figure 1-2 Simple environment running z/OS and CICS TS

Customers can run CICS TS on multiple LPARs for multiple production environments (such as when a bank runs sets of LPARs for core banking and other sets of LPARs for credit card processing), or they can run separate LPARs for development, testing, and quality assurance. All applications running under CICS TS can be integrated into a single disaster recovery environment that covers the entire System z environment, eliminating the need to purchase dozens of additional distributed servers for backup and disaster recovery purposes.

The business value of CICS TS, and its ability to consolidate applications and simplify IT environments, goes beyond the simple cost savings from eliminating server farms. System management is easier and higher qualities of service are possible, all while maintaining heterogeneous workload management and high resource utilization.

1.2.2 Application design

Business transactional systems have several common characteristics. Among them is concurrent access to shared information by many users, through multiple interfaces, with full preservation of data integrity.

Examples of transactions include a customer information inquiry, a financial transfer, an inventory update, a travel reservation, an insurance claim, a stock trade, and an order for a good or service. Each transaction represents a direct cost to the business which, if minimized, can help the business's bottom line.

Figure 1-3 illustrates the steps involved in a simple business transaction between a vendor and a customer.

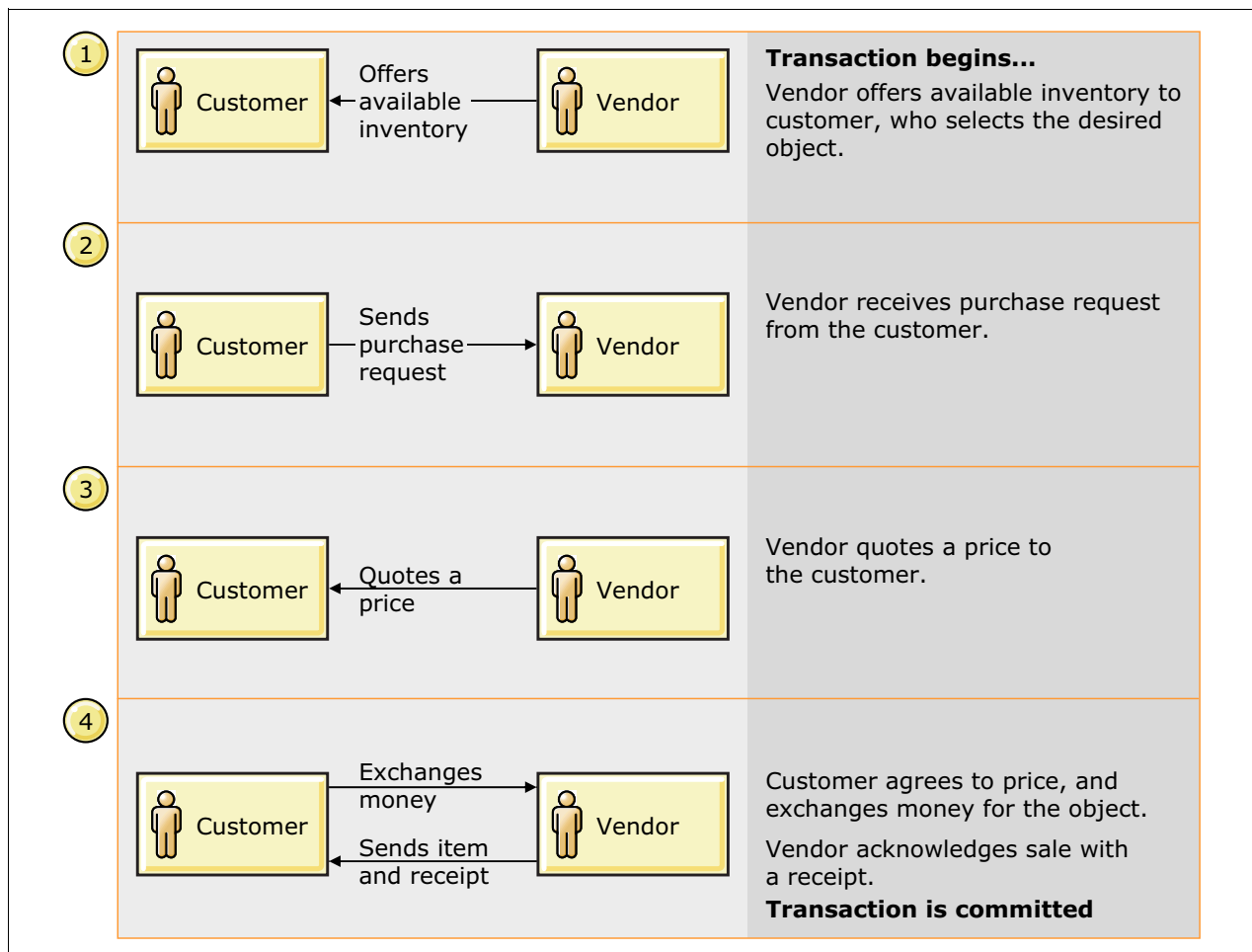


Figure 1-3 Simple business transaction

Standalone applications can be written to execute transactions in shared environments, but they will likely be more complex and less efficient than is necessary. Developers of such applications must consider authorizations and user rights, communication with the user

interface, integration with databases and other types of middleware, preventing concurrent changes in the same resource, and so on.

CICS TS is a managing environment for hosting high-volume, highly efficient, transactional applications. Instead of requiring developers to write system functions in their applications, CICS TS can perform these services automatically or upon request from an application.

By using the capabilities of CICS TS, developers can focus on solving business problems and coding business logic into their applications rather than worrying about system functions such as database integration and security control points.

This approach offers several advantages, including shorter development cycles, easier maintenance and easier incorporation of new functions. Applications do not have to be changed to work with new operating system releases or databases because CICS TS handles the interfaces with these systems.

Figure 1-4 presents an overview of CICS TS system functions.

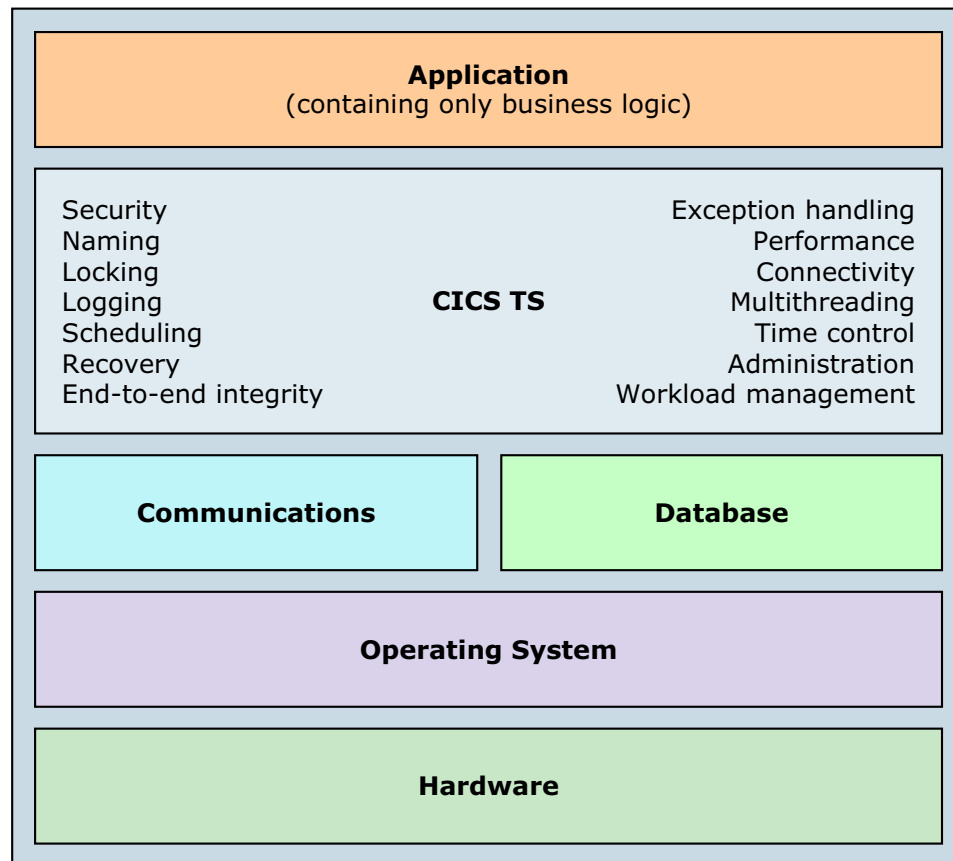


Figure 1-4 Architecture overview of CICS system functions

Because CICS TS was developed to exploit the full potential of the System z platform, applications that run in CICS TS inherit many of the System z qualities of service. This way makes the applications massively scalable, available, and high-performing.

Componentization

In general, a good practice is to split applications into a part containing the business code that is reusable and a part that is responsible for presentation to the client. This technique enables you to optimize the parts separately and to easily reuse your business logic when necessary (see 3.1.2, “Reusable modular model”).

To users accessing an application, such as through a web browser, the application can seem to be one service with many capabilities, or that each screen holds its own capabilities. However, from an IT application perspective, the view differs.

For reasons of performance and extensibility, large business solution areas such as online banking, which seem like a single service to the user, are actually made up of hundreds or even thousands of linked applications that each perform discrete functions. Such complex solutions are good examples of why application design must not inhibit scalability, because the solution must always be ready to respond to increases in demand.

Lines of business are no longer willing to accept long development cycles. For a business to remain competitive and retain clients, the way that applications are developed and deployed now typically follows these key design principles:

- ▶ Reuse existing assets to avoid the time and expense of redeveloping useful business logic.
- ▶ Simplify the development cycle and increase the speed of application delivery by minimizing the need to write new code.
- ▶ Support self-service development of situational applications that have a limited life and user base, to free IT staff to work on more strategic initiatives.

CICS TS provides more than just new principles for application design, however. It enables improved performance and efficiency by allowing you to optimize pieces of each application separately. You can reuse your business logic and rely on the individual presentation logic of each application, simplifying maintenance considerably.

CICS TS provides a component-based approach to application design, development and deployment. Elements of programs (for example, the presentation logic, the business logic and the data access logic) are deployed separately. These elements can be further divided into separate business-logic components for separate business rules. All program elements are linked together using CICS TS capabilities, either internally, between CICS applications within the CICS TS environment, or externally, to databases or to other IT systems through standards-based interfaces such as web services.

Figure 1-5 on page 11 shows how the different elements of an application can be separated into different components in CICS. By separating the elements, changes made in certain elements do not affect the good operation of the other elements.

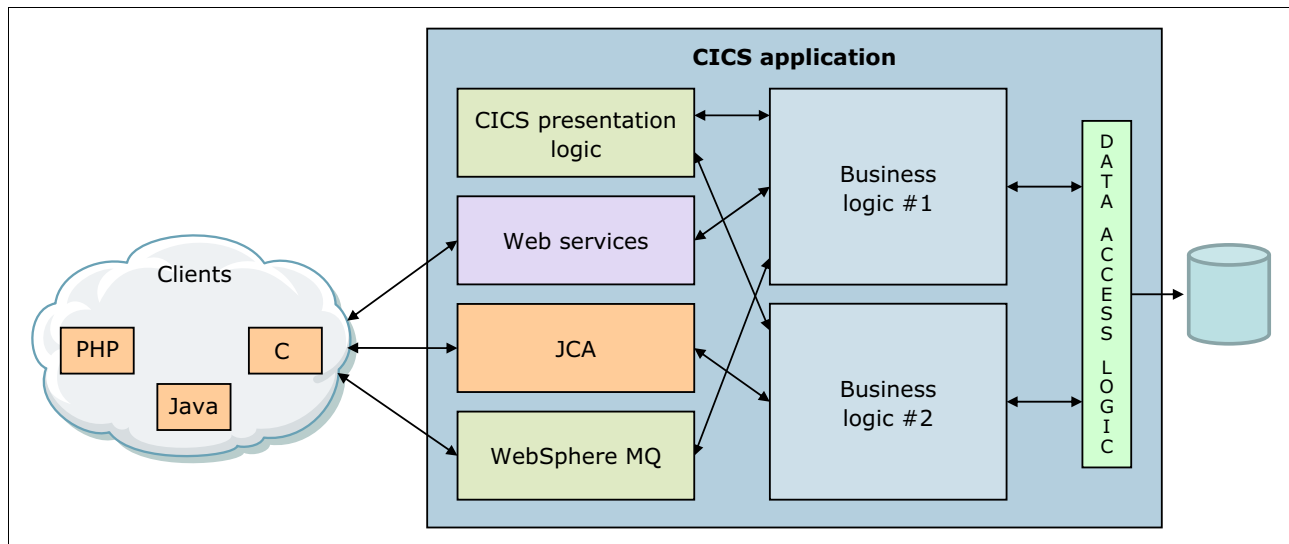


Figure 1-5 Elements of a CICS application as separate components

With this component-based approach, CICS TS simplifies the development of CICS applications and their integration with other systems. This simplification of development can lead to other benefits that will be described in the following chapters.

Performance and efficiency

Performance involves many areas, including the efficiency of the developed code, the efficiency of how the system interacts with applications, and the efficiency of deploying the code, which can be affected by the number of dependencies with other deployed applications and systems.

Another performance issue is speed. Every computer system waits at the same speed, so reducing the amount of time that a system spends waiting is a critical aspect. CICS TS promotes two important design rules to reduce system wait times:

- ▶ Tasks should exist in the system for the shortest time possible
- ▶ Resources should be locked for the shortest time possible

CICS TS reduces wait times by enabling a pseudo-conversational design model.

A *non-conversational transaction* processes one input, responds, and then ends. It never pauses to read a second input from the user or system involved, so there is no real, back-and-forth conversation. A *conversational transaction*, in contrast, occurs when a program interacts with a user for more than just a single input and response.

The *pseudo-conversational* approach used in CICS produces a series of non-conversational transactions that give the appearance of a single conversational transaction. Unnecessary resource locks are avoided because no transaction exists while the application waits for input; CICS reads the input when the user sends it. This way reduces the amount of time that an application requires exclusive access to a shared resource.

Reducing the duration of a resource lock can cut the time that other applications must wait for the resource to become available, increasing overall system throughput. It also reduces the potential for a deadlock situations, where two or more applications are waiting for each other to release locks to finish their jobs. In the relatively rare event that a deadlock occurs, CICS TS takes action to resolve it.

Reusability

Beyond the benefits of extensibility and performance, deconstructing CICS applications into components enhances their potential for reuse. New technology can be implemented as it becomes available, without having to change the underlying applications themselves. This way helps eliminate the costs and risks of rewriting applications for each new software release or technical advancement.

Figure 1-5 on page 11 presents a scenario where different applications can reuse the same business logic components without requiring special code and development effort. Even applications that run outside of the CICS environment can be integrated and incorporate the existing business logic.

A well-structured CICS application that was written over ten years ago can be reused in numerous technology evolutions, including client/server, enterprise messaging, enterprise Java, web services, and so forth. CICS TS capabilities make it possible for such an application to fully participate in even the most modern SOAs, helping the enterprise achieve notable savings in time and resources.

With componentized code, creating applications that add new features, or to updating programs with new functions becomes much easier. CICS TS provides options to introduce new or upgrade existing applications while the system is running. It is even possible to make these changes without making the system unavailable to users at any time, which helps avoid downtime and the potential for lost business.

In addition, critical CICS application modules can be assigned a relative priority to ensure that they continue to achieve predefined service levels, such as certain response times, even when the system is running at 100% utilization.

Chapter 3, “CICS application architecture” on page 49 covers CICS application architecture options and how they can help an enterprise address new business requirements and achieve greater cost effectiveness.

1.2.3 Reliability

Previously, we described the importance of being fast and flexible in today’s enterprise world. If the business of a given company is not available when its clients need it, the company is likely to lose those clients. A reliable and resilient infrastructure and integrated applications are critical to success.

Availability

A basic requirement for today’s IT environments is to provide continuous business operations even in the event of planned or unplanned disruptions. The availability of mission-critical applications, based on a highly available platform, directly correlates to successful business operations.

System z hardware, operating systems, and middleware elements are engineered to work together, providing an application environment with a high level of availability. The System z environment approaches application availability with an integrated and cohesive strategy that encompasses single-server, multi-server, and multi-site environments.

The System z hardware is a highly available enterprise server. All of the hardware elements are redundant. From the power supply to the spare components that enable hot failovers, each element can be switched out automatically in the event of a problem. Because of this redundancy, fixing or changing any element without stopping the machine is possible.

CICS TS delivers an environment that is designed for virtualization by providing the ability to run many virtualized CICS application runtime environments, known as CICS regions, within a single CICS TS installation.

A *CICS region* is a named instance of CICS TS that runs in its own z/OS address space and can be started and stopped independently of other CICS regions. Figure 1-6 shows possible interactions across CICS regions or z/OS systems.

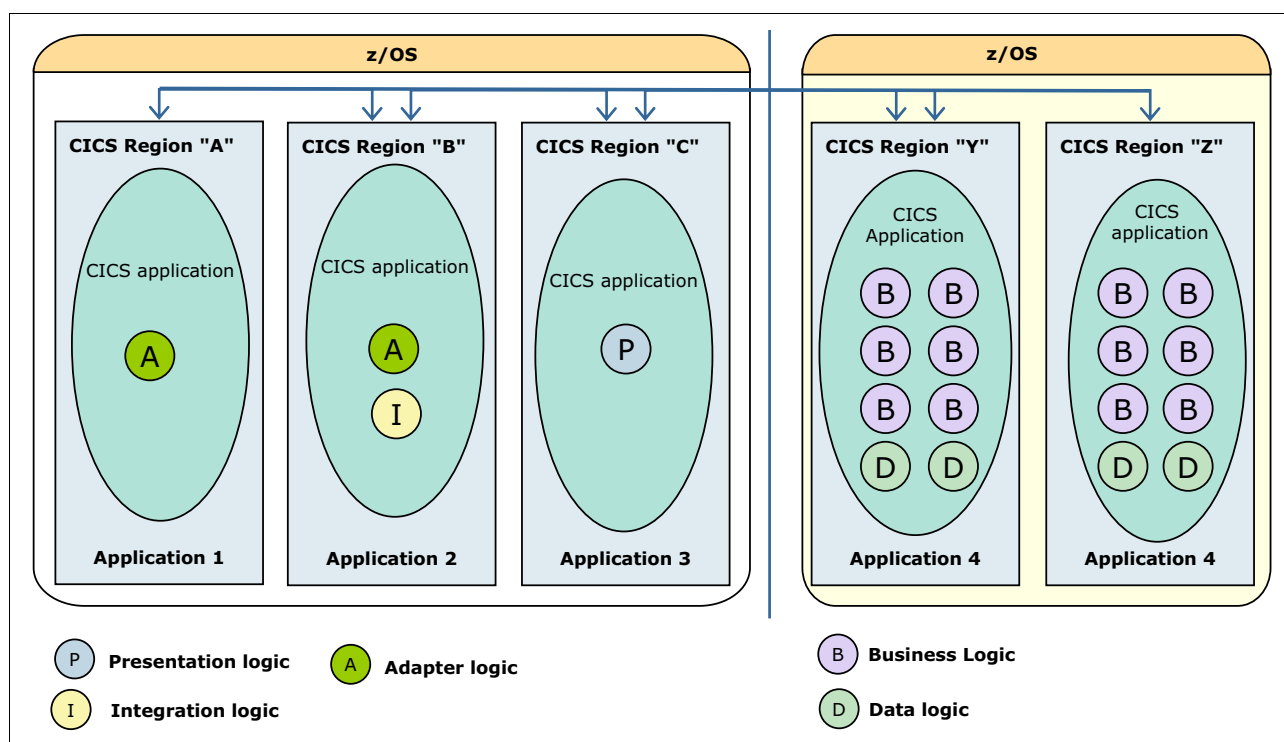


Figure 1-6 Applications can be distributed across multiple regions

Running CICS applications in separate CICS regions provides a separation of workloads that isolates applications from each other. Any application-based problem that occurs in one CICS region do not affect the availability of applications running in other CICS regions. In addition, separate components of the application, such as presentation or adapter logic and business and data logic, can be deployed into separate regions, providing for improved scalability and availability. For further details about the benefits of modular CICS applications, see 3.1.2, "Reusable modular model" on page 52.

Performance

When new workloads are added to a System z server, they are not simply added randomly. Some workloads, such as those associated with customer ordering and fulfillment, have a higher degree of importance than, for instance, applications that are only used internally. Making resources available to mission-critical applications when they need them is a priority, both for software designers and within System z.

System z servers running a z/OS image or images can take advantage of the z/OS Workload Manager (WLM). Using its advanced workload management capabilities, an administrator can establish policies and business priorities that ensure resources will be directed to an application when needed. These features allow a System z environment to operate effectively at average utilization levels exceeding 75%, and sustained peak utilization levels of 100%, without degradation.

In addition to what the System z platform provides, CICS has a rich set of functions that enable application developers and system administrators to explore performance and other statistical information. Using these tools, CICS analysts can focus on specific problem areas and resolve things quickly.

The tools that are used to manage CICS environments are described, in detail, in Chapter 6, “Exploring the CICS development tools” on page 137.

Integrity and security

When integrating separate systems that are based on separate technologies, and possibly controlled by partners that are outside the control of the company, information integrity and security grow in importance.

Information must not be tampered with before, during, or after a transactional exchange. The integrity of a business transaction that spans multiple systems must be controlled, which requires support for atomic transaction capabilities.

CICS TS uses the services provided by an external security manager, such as IBM RACF® program, to control access to application and system resources.

1.2.4 Creating a market advantage with CICS TS

Because the need to provide higher levels of service is rising, budgets are tending to move in the opposite direction. IT executives are constantly challenged to provide more and better services at the same or even lower cost. Hardware and software expenses, in particular, draw great attention.

The System z platform is known for advantages such as high availability, planned redundancy, rapid I/O capabilities, and more. Yet there is a common perception that the platform is too expensive. But is it really that expensive, considering what it offers?

Comparing one System z server with one distributed server shows that, in some ways, the System z environment is indeed more expensive. But is this the proper way to compare the two platform solutions?

Organizations that are researching alternatives typically compare a mainframe server with a distributed server. But what organizations might not consider is that most likely, more people will be needed to manage a distributed environment than a mainframe. Studies indicate that the cost of people has increased dramatically over the past decade, and the cost of the hardware has decreased. So the indirect cost of an IT environment is growing, just at the time IT organizations are looking for less costly alternatives.

Real experiences prove the effectiveness of CICS TS running on System z, and how this combination helps organizations create a market advantage. Consider a medium-sized company that is seeking to extend its operation internationally. Because operating in other countries will demand huge changes in the current IT environment, the company is looking at options and comparing the mainframe approach with a distributed server-based solution.

In a scenario involving, for example, 10,000 virtualized server images with an equal number of applications, software costs will clearly be higher with a mainframe than with a distributed system. However, a case could be made that 10,000 server images and applications in a distributed environment will consume more storage and network capacity and require more people to support the solution than would be the case with a mainframe. Using a mainframe running CICS TS to manage the applications can reduce the use of human and computational resources and therefore drop the costs dramatically.

From an application development perspective, an IT department can use many types of programming languages for different scenarios. But using different programming languages in complex applications can lead to long development cycles, greater complexity and higher maintenance costs, particularly when new functions need to be integrated into the solution. IT executives want to avoid such situations. They want rapid development cycles and low costs, but at the same time they expect to have applications that are well designed, stable, secure, and reliable.

CICS TS provides a fast, low cost, low risk way to evolve business applications. It provides significant opportunities for cost savings while improving productivity. Because applications can be adjusted using CICS capabilities, and because CICS can support multiple programming languages, the savings can be substantial.

1.3 CICS market presence

Despite frequent updates over the years, CICS TS still retains the unique architecture that made it such a popular purpose-built transaction processor. Developed primarily in the United Kingdom (with additional development at facilities around the world), CICS TS and CICS tools are used in more than 80 countries worldwide. In Figure 1-7, the black marks indicate countries where CICS products are used and the red marks (circles) show where they are developed.



Figure 1-7 The world of CICS

The creators of CICS built it with the most advanced technology available at that time. Their successors followed this example, evolving CICS to work with the many groundbreaking and innovative technologies that have emerged over the past 40 years.

With thousands of CICS mainframe licenses now active in 73 countries, CICS processes an estimated 30 billion transactions per day, essentially providing the foundation for electronic commerce worldwide. Figure 1-8 on page 16 illustrates the distribution of CICS in mature markets (such as those in North America, Europe, Australia and Japan), growth markets (such as Brazil, Russia, India, China and Turkey) and emerging markets, and also by region.

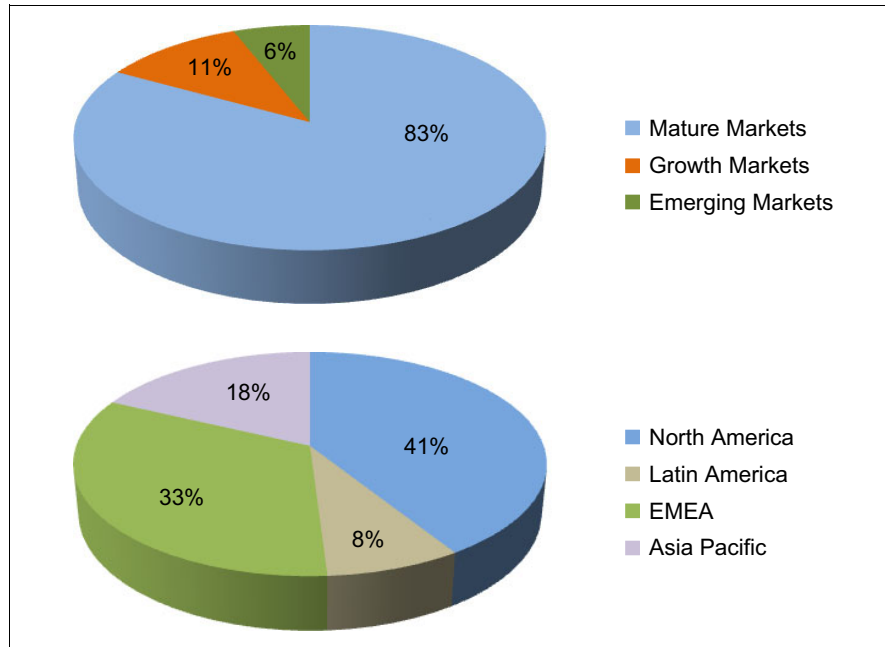


Figure 1-8 CICS usage in today's market

Enterprises in nearly every area of industry and commerce rely on CICS transaction processing. These enterprises include the industries depicted in Figure 1-9 on page 17 and listed here:

- ▶ Agriculture
- ▶ Architecture
- ▶ Automotive
- ▶ Banking
- ▶ Chemical
- ▶ Construction
- ▶ Education
- ▶ Financial services
- ▶ Government
- ▶ Insurance
- ▶ Manufacturing
- ▶ Media and broadcasting
- ▶ Medical and pharmaceutical
- ▶ Military
- ▶ Oil, gas, and mining
- ▶ Real estate
- ▶ Research and development
- ▶ Retail
- ▶ Shipping and transport
- ▶ Telecommunications
- ▶ Travel

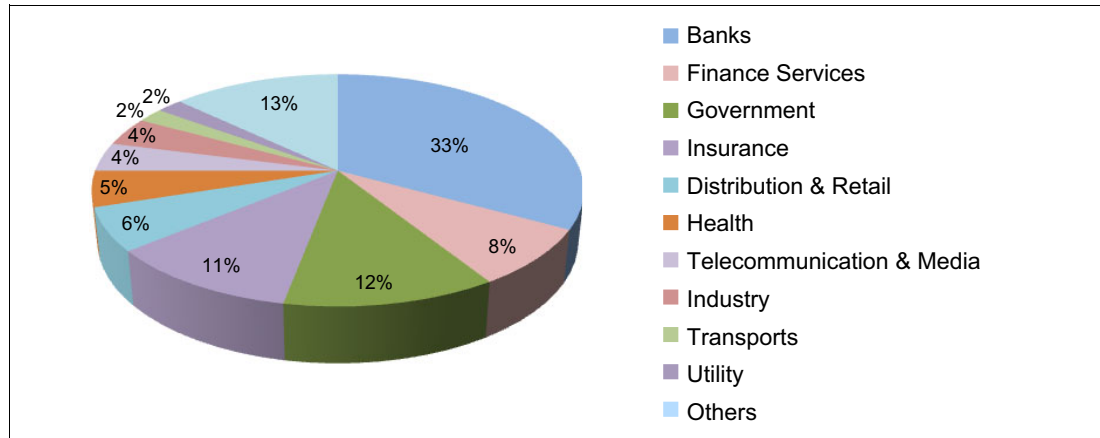


Figure 1-9 CICS usage by industry

The near-universal presence of CICS and its ability to securely manage so many transactions simultaneously has directly contributed to advancements in information communication, the growth of industry and commerce, and globalization. Yet, although it is at the heart of daily life today, CICS is still relatively anonymous.

Users might not realize that CICS is there when they order lunch online, schedule a visit by their utility provider, buy an e-book, or schedule an airline flight. It could be the most successful piece of software of all time, with millions of users unknowingly running CICS transactions every day of their lives.

1.4 Cost effectiveness

CICS provides a highly efficient environment for the development of business-critical applications. The fact remains, however, that scenarios for new business application deployment into CICS are often limited for cost reasons. In this section, we highlight several key factors that show why CICS can be considered a cost-effective solution for IT applications in any organization.

1.4.1 Understanding the cost of CICS

Quantifying the cost of a solution is crucial to understanding how profitable it can be. But evaluating cost effectiveness is not a simple matter. Multiple factors must be considered, from hardware and software to people and administration. This section shows how the technical virtues of CICS drive its cost effectiveness.

Figure 1-10 illustrates the cost per unit of CICS (as a general pattern, not in a specific customer scenario). The black line represents the cost per unit of workloads on a given system. The curve indicates that as more workload is added, the cost of adding it goes down. In this view, the cost effectiveness of a solution is determined by actual increment costs, not on average or linear costs.

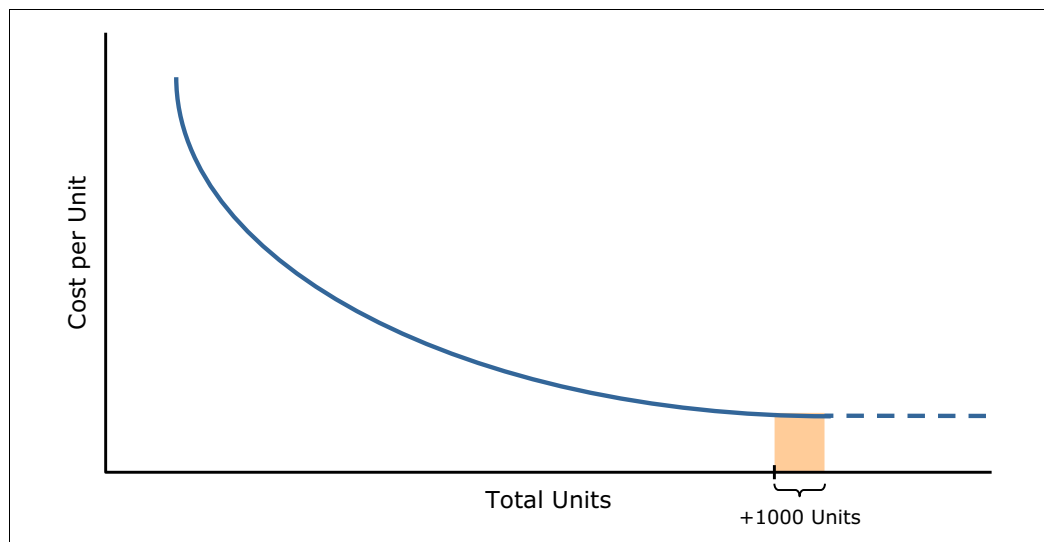


Figure 1-10 Understanding CICS cost

After understanding the calculation of actual increment costs, deploying new applications onto CICS clearly becomes a cost-effective solution, mostly because of the high efficiency of CICS and System z. The initial investment can be substantial, but the return over time is significant.

1.4.2 Cost benefits of CICS qualities of service

As shown previously, CICS provides a range of system functions that provide value in the development, deployment, and operation of applications. Here is a short list:

- ▶ Agile development: See Chapter 5, “Application development lifecycle” on page 111.
- ▶ Flexible access: See 2.2, “Flexible integration options” on page 27 and 2.6, “Simplified administration with CICSplex System Manager” on page 44.
- ▶ Robust integrity: See 2.3, “Proven integrity” on page 29.
- ▶ Highly available and scalable processing: See Chapter 7, “High availability and continuous operation” on page 177 and Chapter 8, “Scalability and workload management” on page 201.
- ▶ Secure environment: See Chapter 9, “Security and auditing” on page 219.

Together, these qualities of CICS provide synergistic value as the complexity and processing requirements of the system increase.

Figure 1-11 shows how, with increasing workload, the cost per unit of workload decreases. The reason is because of the economies of scale of CICS and System z and the inherent elastic scaling of CICS. A non-CICS solution does not have the same number or quality of these capabilities, so the cost per unit of workload in such solutions can grow considerably higher as system complexity increases.

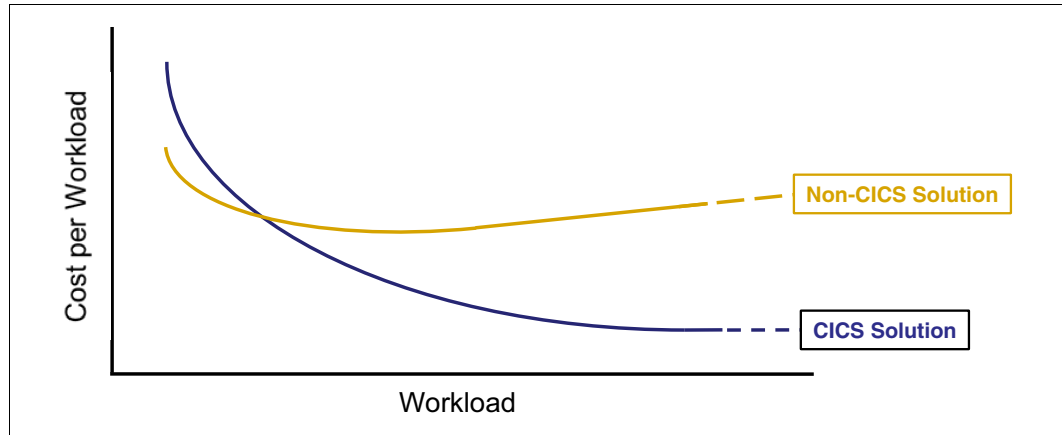


Figure 1-11 Cost behavior with increasing workload

1.5 The future of CICS

CICS was conceived in the late 1960s and has since evolved into one of the most widely used and functionally capable transaction processing monitors (Figure 1-12).

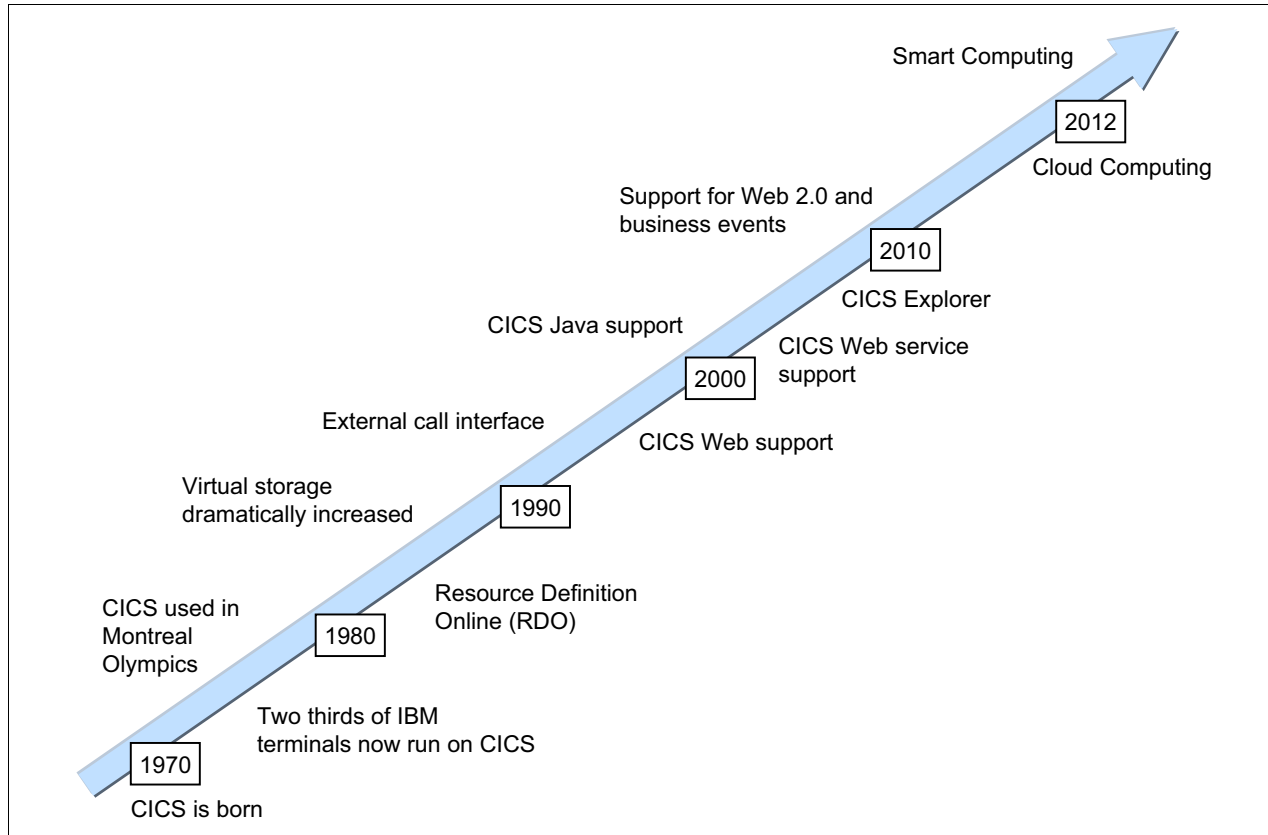


Figure 1-12 Evolution of CICS

The benefits of constant availability, robust security, and massive computing power stimulated increased sales for this type of computing in the early 1990s, which was approximately the time when critics said the mainframe would cease to exist. CICS grew into a family of application servers, spanning a variety of platforms and offering business solutions for all sizes of organizations. Although introducing entry-level systems at the low end of the marketplace, CICS has always kept pace with technological developments in the mainframe world.

CICS continues to deliver client value and give clients the most recent technology capabilities by making them available as a service, offering the best of what the *cloud computing model* has to provide in terms of agility, scalability, and resource management.

An application running in a cloud environment has five essential characteristics:

- ▶ On-demand self-service
- ▶ Broad network access
- ▶ Resource pooling
- ▶ Rapid elasticity
- ▶ Measured services

These characteristics are provided by System z and can be readily exploited when deploying applications as CICS services, such as resource measurement using CICS monitoring, broad network access using TCP/IP or SNA connectivity, rapid elasticity through the CICS open transaction environment, and System z capacity on demand.

However, even with such exceptional capabilities already present in CICS, the CICS development team continues to extend the core values of the product values to align them even more directly with the essential characteristics of a cloud, as shown in Figure 1-13.

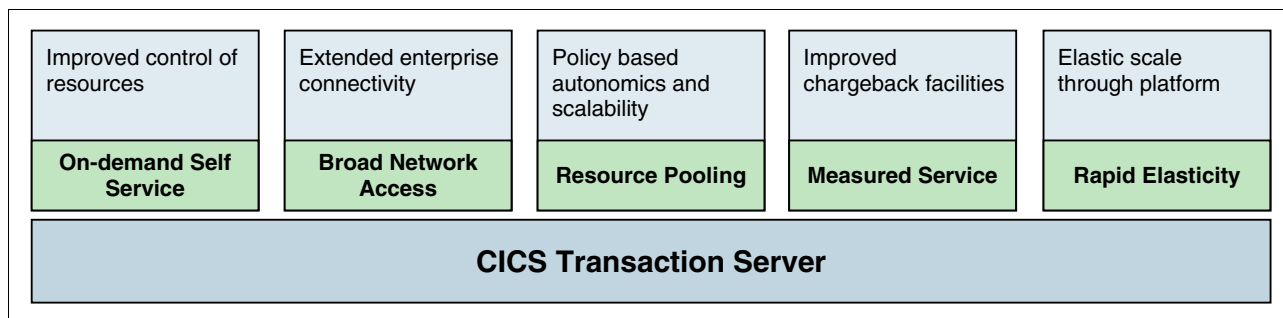


Figure 1-13 Extending CICS core values

The latest release, CICS TS V5.1, provides cloud-style CICS development, deployment, and operations. Foundational advancements in both capability and scalability allow CICS applications to do more things, more easily than ever. CICS now provides *operation efficiency* and *service agility* with cloud enablement:

- ▶ Driving operation efficiencies:
 - Greater capacity: Achieve cost savings through consolidation.
 - Managed operations: Control critical resource thresholds with policies.
 - Increased availability: Reduce the need for planned downtime.
 - Deeper insight: Extend performance and compliance information.
- ▶ Increasing service agility:
 - First-class applications: Create agile services from existing assets.
 - First-class platforms: Create agile service delivery platforms.
 - Modern interfaces: Build rich web experiences for critical applications.
 - Foundational enhancements: Extend core capabilities.

The combination of new application, platform, and policy definitions in CICS (see Figure 1-14) greatly simplifies the assembly, deployment, and lifecycle management of CICS applications, providing many of the characteristics found in private-cloud systems. CICS TS gives you the opportunity to explore the concept of a cloud-style CICS deployment in your own environment.

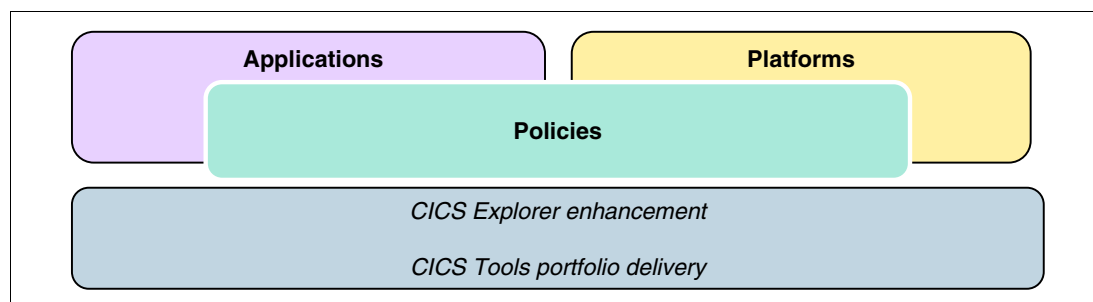


Figure 1-14 Cloud-style characteristics as part of continuous CICS enhancements

CICS evolved over multiple decades, providing support on multiple operating system platforms, using a variety of traditional programming languages and all of the common forms of data management. CICS supports systems network architecture (SNA) terminal networks, TCP/IP sockets, and web input and output. It continues to build on previously introduced capabilities, and expands to provide support for new technologies, such as the IP networking, web, Java, JSPs, and increased performance through its open transaction environment (OTE) facilities.

A number of technological trends are having a significant impact on transaction processing systems of today. CICS embraces these trends, which include the following trends:

- ▶ Evolving architectures based on service-orientation and web services
- ▶ Mobile solutions
- ▶ Web 2.0-based technologies
- ▶ New paradigms for application development
- ▶ Opportunities in business event processing and the encapsulation of business rules into flexible rules engines
- ▶ Business Process Management (BPM)
- ▶ Business analytics
- ▶ Hybrid systems

The fundamental nature of CICS will not change, however. It will continue to be *CICS for everyone*. No matter how many applications are added, an organization can still design, test, and run simple transactions on the largest or smallest systems. This is why CICS was created and why it will continue to be here in years to come.



CICS capabilities

IBM CICS TS is proven general-purpose transaction processing software for z/OS, with an established reputation in the marketplace. Its main purpose is to process requests to shared data from a large number of connected users or systems. It is designed to be scalable and secure, to operate with minimal errors and downtime, and to exploit the built-in qualities of the IBM System z platform.

This chapter describes the core capabilities that make CICS unique within the industry and capable of providing the business value needed for the smarter enterprise applications of today:

- ▶ Agile application development and deployment, including a rich set of APIs and support for multiple programming languages and the latest development tools and techniques
- ▶ A wide range of flexible interconnectivity options that support integration of CICS services within a service-oriented architecture (SOA)
- ▶ Proven integrity for coordinating access to recoverable data
- ▶ A secure processing environment integrated with the System z security controls
- ▶ A highly available and scalable processing environment able to meet the varying needs of today's businesses
- ▶ Simple administration

These capabilities are based on over 40 years of development and customer feedback. They are designed to exploit the intrinsic abilities of the System z hardware, and both industry standards and open standards.

2.1 Agile development & deployment

Businesses today exist in an ever-changing environment that requires them to be responsive to new opportunities as they emerge. There is constant demand to deliver new products and services within shorter time frames, while still maintaining quality standards. So, businesses need an agile way of developing and deploying new business applications.

CICS has a rich set of application interfaces that can be used to quickly build and deploy new application components, and it supports a wide variety of programming languages. But for maximum agility, development and deployment must be coordinated. Therefore, CICS includes the Open Services Gateway initiative (OSGi) framework, which can be exploited through a private cloud model using tools, such as the IBM CICS Explorer®.

2.1.1 CICS application programming interfaces

CICS provides a useful set of application programming interfaces (APIs) that enable simplified access to CICS-controlled resources and allow CICS container services to provide runtime qualities of service such as transactionality, security, and workload management.

The CICS APIs significantly simplify application development by providing a neutral, standard language with which to access resources such as files, queues, or other programs. And the APIs enable a modular development model by separating the business logic from the system management logic that supports it. You can read, write, and rewrite a file without using the syntax of the selected program language; you simply reference the alias of the resource and perform the CICS command through the appropriate API.

An additional benefit of the CICS APIs is that load-module compatibility is maintained across CICS releases. Even when APIs are deprecated and removed, modules can still be used in newer CICS releases if they are coded to expect the relevant errors.

All commands, with their options, are described in the CICS Application Programming Reference. The functions they replace are listed in the CICS Application Programming Guide. In addition, CICS provides customizing facilities that you can use to set up and run your environment. The following efficient facilities allow additional customization:

- ▶ Global user exits (GLUE): Entry points in a CICS module or domain where CICS can transfer control to a global user exit program and then resume control when the exit program finishes its work.
- ▶ Task-related user exits (TRUE): Customized programs to access a resource manager that is not otherwise available to the CICS system. CICS itself also uses these. For example, the task-related user exit DSNCSQL is used to access IBM DB2® when a CICS application program issues an SQL call.

In addition, you can customize the behavior of CICS programs by creating a modified version of the original.

2.1.2 Programming languages

Businesses use a wide variety of application development languages based on their requirements and existing skill sets. By supporting a wide range of language options, CICS allows applications to exist in a truly heterogeneous environment consisting of both procedural modules (such as COBOL) and object-oriented languages such as Java or C++.

The languages that are supported by CICS are summarized in Figure 2-1.

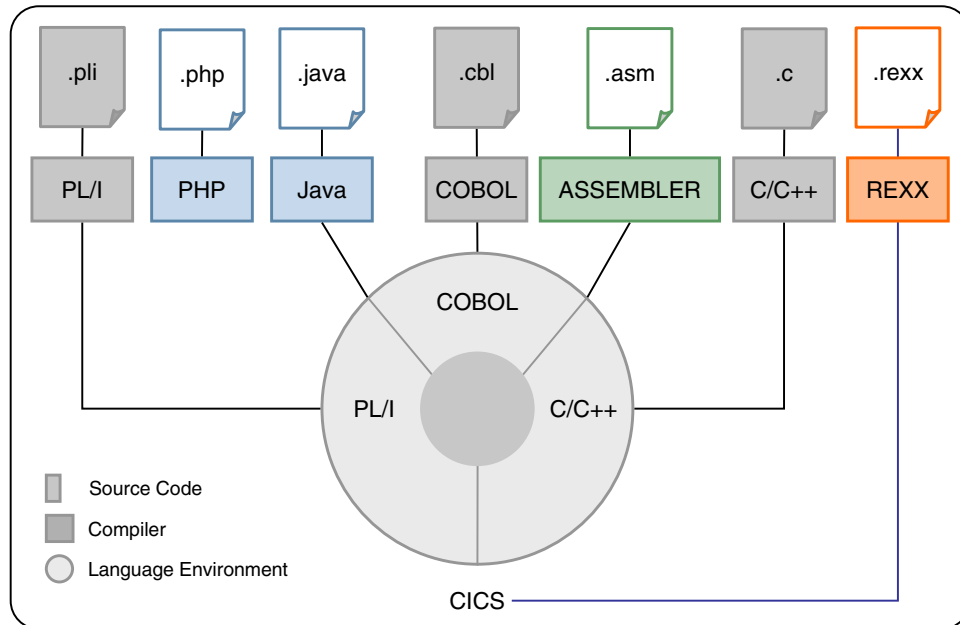


Figure 2-1 The CICS programming environment

Central to CICS language support is the z/OS IBM Language Environment®, which CICS uses to integrate the various programming language run times. Each programming language (for instance, PL/I, COBOL, and C/C++) has its own library within the language environment run time, which allows CICS to provide simple communication between separate programs at the language level. The assembler language does not require a runtime library to work in CICS, but it is suggested that the common execution libraries be installed in the CICS environment. REXX uses the CICS services directly and has no link with the language environment libraries.

The CICS multi-language run time includes the following benefits:

- **Inter-language communication:** This benefit provides flexibility and efficiency in the design and development processes, and allows developers to use a *fit for purpose* methodology to select the best programming language for each project.
- **Standard interfaces:** The CICS APIs provide a standardized set of interfaces for interacting with any CICS-controlled resources. CICS also provides an API for Java, which, although implemented in Java, behaves identically to a standard CICS API. This API provides simplicity of development, consistency between languages, and an easy way to exploit the CICS runtime qualities of service such as security, workload management, and transactionality.
- **Central diagnosis:** CICS uses a common layout for the information provided by the system when errors occur. This way reduces the time and effort needed to determine the source of problems, which can improve application quality.

2.1.3 Java framework

Java applications are widely used within CICS and run in a multithreaded Java virtual machine server (*JVM server*) environment, by using the facilities of a dedicated language environment enclave (Figure 2-2), which allows multiple tasks to be dispatched as parallel threads within a single JVM server. The JVMSERVER resource provides users with a single resource to manage a complete JVM and related hosted applications. Each task will run on a dedicated task control block (TCB) thread in the JVM, rather than requiring a whole JVM.

Concurrency is vastly increased, with up to 256 transactions running simultaneously in the same JVM server. In CICS TS V5.1, the pooled JVM environment support is removed.

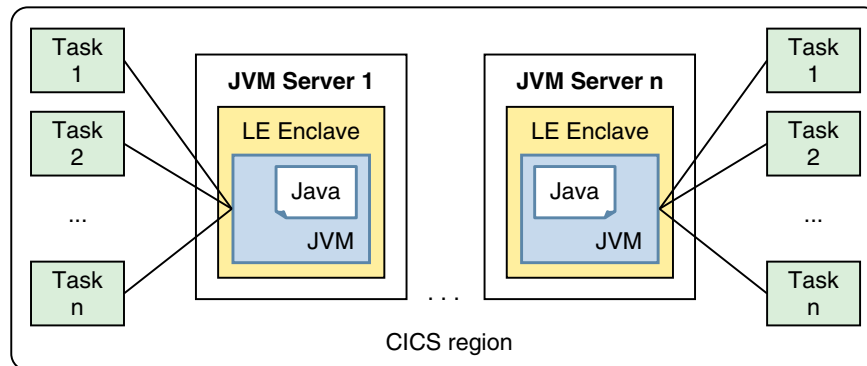


Figure 2-2 The CICS TS JVM server environment

Important: The PHP and Groovy programming languages run inside the CICS JVM server environment, using the CICS TS Feature Pack for Dynamic Scripting.

This new functionality allows CICS to run Java workloads on a JVM server under the OSGi framework. This way provides clear benefits for business applications that are developed and deployed using this model:

- ▶ The JVM server provides a more industry-standard server-side programming model. Applications can share an engine in addition to sharing data between tasks. This benefit greatly improves the potential for reuse of existing Java components.
- ▶ Support for the OSGi standard means that applications written for the JVM server (and migrated from pooled JVMs) will be packaged as OSGi bundles. This support allows dynamic deployment, replacement, and versioning of applications within the JVM server. Therefore, you can upgrade, enhance, or patch applications without restarting the JVM.
- ▶ Standard tools can be used for debugging, problem determination, and performance monitoring. This benefit allows for skills reuse and knowledge transfer.
- ▶ Rapid web service development is provided in Java through support for Java API for XML Web Services (JAX-WS) applications.

2.1.4 Development and deployment lifecycle

Existing and new CICS business applications can benefit not only from the virtues of CICS as a transaction manager at run time, but also from the tools and features within the CICS ecosystem that support the development and deployment phases of the system development lifecycle (SDLC).

Among these tools are the tools for the following purposes:

- ▶ Application analysis: CICS Interdependency Analyzer, Rational Assets Analyzer
- ▶ Developing, debugging, and managing source code: Rational Developer for System z, Debug Tool for z/OS, IBM Rational Team Concert™
- ▶ Functional and regression testing: Rational Development and Test Environment for System z, Fault Analyzer for z/OS
- ▶ Performance analysis: Application Performance Analyzer for z/OS, CICS Performance Analysis for z/OS
- ▶ Deployment to production: CICS Deployment Assistant, CICS Configuration Manager

Many of these tools are part of the Eclipse platform, which is used by CICS and other IBM systems. Using the Eclipse platform helps unify different products and provides integration between tools, utilities, add-ons, plug-ins, and extensions. It also allows for easier integration with applications and tools that comply with the Eclipse standards but are not from IBM.

2.1.5 CICS and cloud computing

Cloud computing is a model for enabling convenient, self-service, network-based access to a shared pool of configurable computing resources such as servers, images, operative systems, middleware, business applications and business processes. These resources can be provisioned and released with minimal management effort or service provider interaction.

The following benefits are inherent to this model:

- ▶ Economies of scale and efficient sharing of resources
- ▶ Agility in the processes related to development and deployment

The cloud model builds on the service enablement features of applications, and promotes additional availability and flexibility. Cloud applications have five essential characteristics:

- ▶ On-demand self-service
- ▶ Broad network access
- ▶ Resource pooling
- ▶ Rapid elasticity
- ▶ Measured services

CICS support for cloud computing is described in Chapter 1, “Business value of CICS” on page 3.

2.2 Flexible integration options

Businesses need to develop and deploy new functionality in a speedy way to meet the pace of the evolving marketplace. However, businesses also have existing IT assets that might be key to the success of the business; therefore, reusing these assets is often a better approach than redeveloping them and risking destabilizing the business.

Reusing existing assets to create new business processes or enhance existing ones is often the best way to achieve a fast time-to-market and maximize profits. Reuse also carries less operational risk, because the reused asset has already been tested, enhanced, and deployed in the real world.

CICS integration technologies (see Figure 2-3) support a comprehensive set of transports and architectures that can be used to integrate CICS applications with most common service-requester environments.

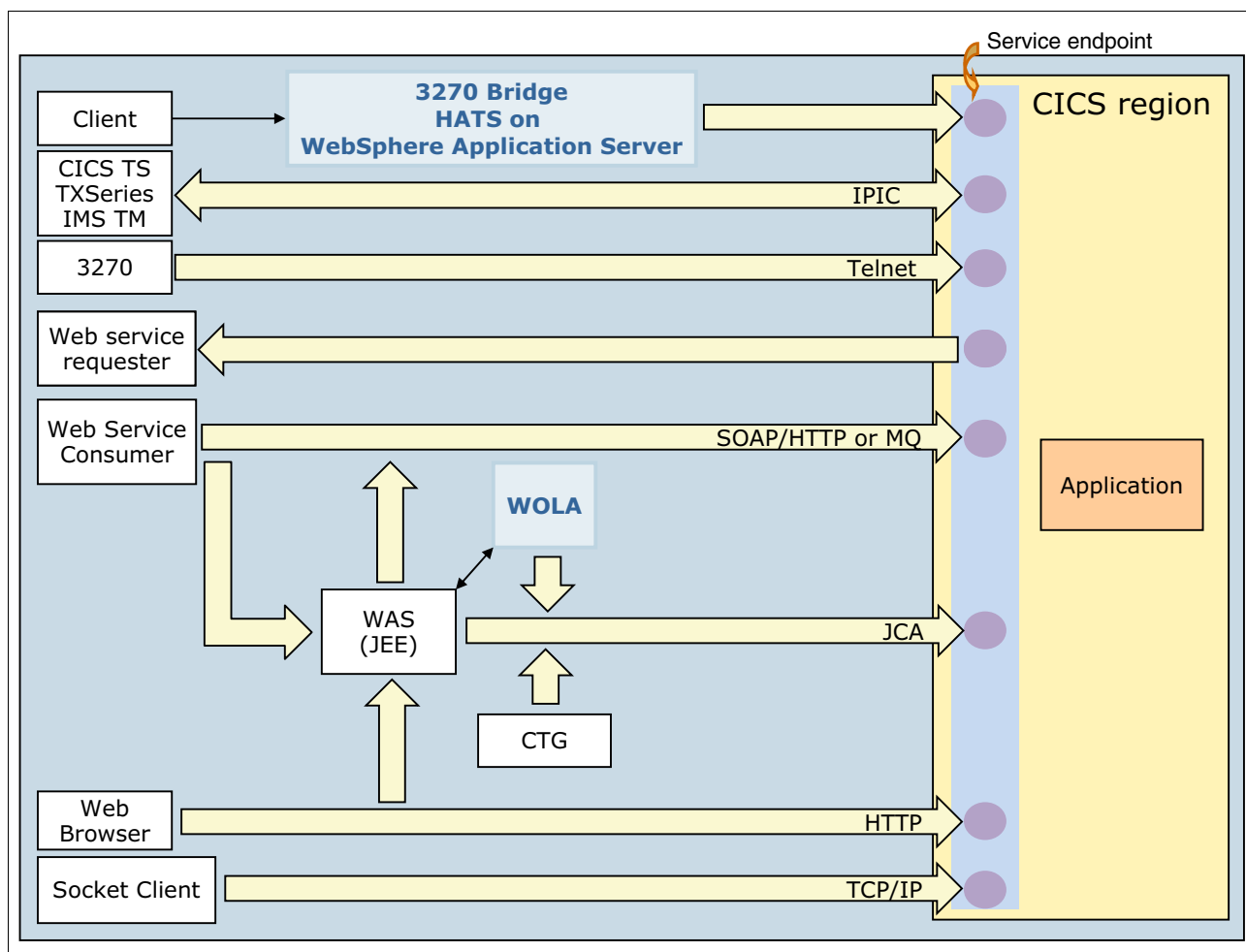


Figure 2-3 CICS access technologies

Each of these access technologies has separate components and qualities that are covered more thoroughly in 4.4.2, “CICS integration options” on page 92. A brief explanation of them is provided here:

- CICS web services

Web services are a proven technology within the IT industry. CICS TS has a rich set of functions to enable applications to be exposed as services and consumed by a requester. CICS applications themselves can also act as a service requester to other services hosted on internal or external providers. With this flexibility, CICS can integrate in a heterogeneous SOA environment across one or more organizations.

- CICS Transaction Gateway

The CICS Transaction Gateway (CICS TG) provides a simple connector package that consists of client and server software components, which allows external clients to invoke CICS services with no change to the CICS system. It provides a range of client APIs, supporting native-language applications in C/C++, COBOL, or the .NET framework, and also integration from Java-based client applications such as a JEE application server or Java applets or stand-alone applications.

- ▶ WebSphere Optimized Local Adapter

The WebSphere Optimized Local Adapter (WOLA) is a functional component of WebSphere Application Server for z/OS that provides an efficient cross-memory mechanism for inbound and outbound calls. And because it avoids the overhead of other communication mechanisms, it is capable of exchanging a high volume of messages.

- ▶ CICS web support

CICS has long provided support for direct access through HTTP. This function is known as CICS web support and is a simple way of using CICS as a web server or web client over HTTP. When CICS is an HTTP server, a web client can send an HTTP request to CICS and receive a response. The response can be a static response created by CICS from a document template or static file, or it can be an application-generated response that is created dynamically by a user application program.

- ▶ WebSphere MQ

WebSphere MQ is a family of messaging products that is available for all major operating system platforms. It provides an open, scalable, industrial-strength messaging and information infrastructure that helps enterprises integrate business processes. WebSphere MQ provides Java Message Service (JMS) APIs and native WebSphere MQ APIs, making it suitable for communication between a wide range of applications and ideal for integrating CICS with third-party software components.

- ▶ CICS sockets

The TCP/IP Socket Interface for CICS is also known as CICS sockets. It is an interface of IBM Communication Server for z/OS that provides a low-level socket API for use by CICS applications. This communications interface allows custom connectivity with any IP-enabled device within a TCP/IP network.

2.3 Proven integrity

CICS was conceived in the late 1960s to handle complex business transaction processing. It was engineered as an independent system to coordinate all transaction requests while maintaining the integrity of the data for ongoing processes. The success of CICS is based on the following pillars:

- ▶ By building upon System z and the z/OS operating system, CICS is able to communicate with hardware systems and subsystems to provide high performance response and reliability.
- ▶ By separating business logic from transaction control and management, CICS has a modularity that improves flexibility, reusability and cost effectiveness.
- ▶ By managing the workload with policies that are closer to the business process than to the actual hardware, CICS is able to predict certain system behavior when various operational conditions arise. This approach means that variations to the business process, planned or unplanned, can be handled by changing configuration settings, not re-engineering or repeating any previous work.

2.3.1 Transaction integrity

Two basic models exist to implementing transactional integrity within a business application:

- ▶ Transaction integrity is developed *within the logic* of the business application. Developing this level of integrity can be complex, especially if multiple components are involved, and it is likely to require complex compensating logic to be developed in case of failures.
- ▶ Transaction integrity is provided *by the system that supports* the business application. This simplifies the business logic that is required and makes it easier to develop transactional applications.

The second model is the method used for CICS applications. Transactional recoverability is automatically provided by the CICS run time when the CICS API is used to access recoverable resources.

Defining a transaction

In general, a transaction is a group of heterogeneous operations that succeed or fail as a *single unit*. In CICS, a transaction is a unit of activity that consists of multiple updates to recoverable resources under a single unit of work, such that either all or none of the updates are made permanent.

A key competency of CICS is the ability to process transactions quickly, efficiently, and reliably. Figure 2-4 on page 31 depicts a basic transaction environment that extends over more than one system. The environment includes the following components:

- ▶ **Application:** A program (or group of programs) that provides a known functionality to users or other systems or platforms.
- ▶ **Transaction Manager (TM):** Coordinates the overall transaction by communicating with transaction managers on other platforms (remote TMs) and with resource managers on the same platform to maintain task integrity.
- ▶ **Resource Manager (RM):** Can be any component that provides a recoverable data service. Resource managers can be data servers (such as DB2, IBM Informix®, or VSAM), queue managers (such as WebSphere MQ, WebSphere Message Broker, Microsoft Message Queuing, and so on), or application servers (such as WebSphere Application Server, Geronimo, JBoss, or Weblogic).

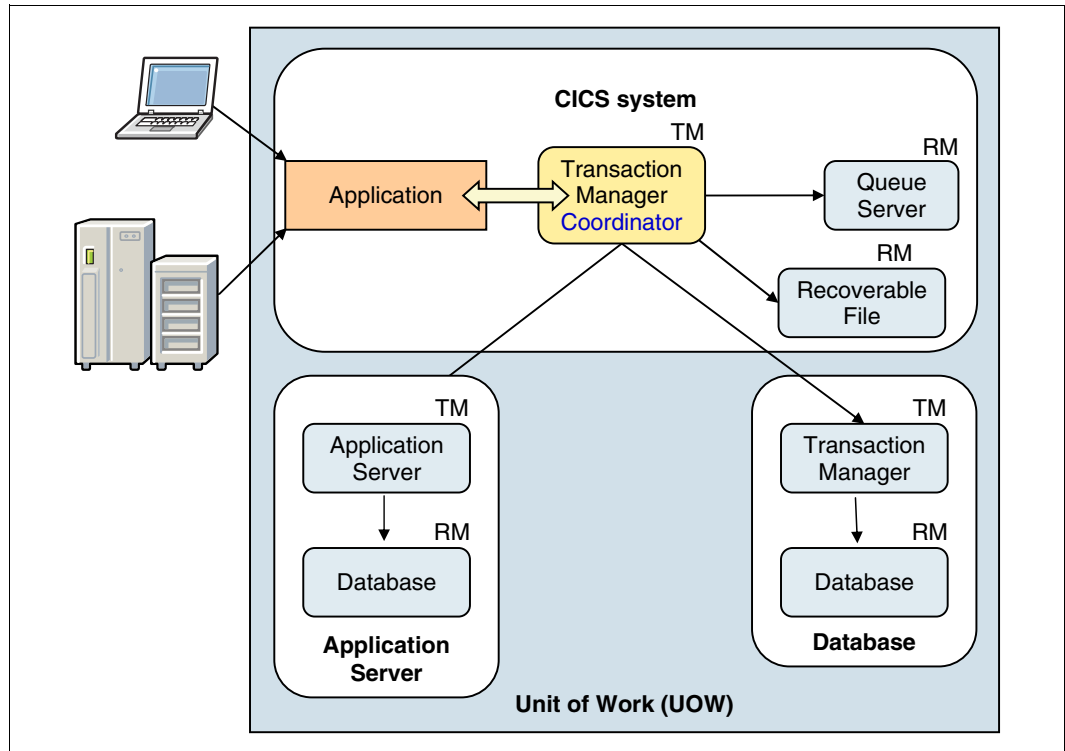


Figure 2-4 Transaction environment

ACID properties

From a CICS architectural perspective, integrity is obtained through adherence with the four properties: atomicity, consistency, isolation, and durability (ACID).

Atomicity

All operations are either performed completely or are not done at all; if some or all of the operations are not completed, then all of the operations must be reset to their starting state. For example, in a transaction that transfers funds from one account to another, the atomicity property ensures that, if a debit is made successfully from one account, the corresponding credit is made to the other account.

Consistency

The operations transform a system from one consistent state to another. Along the way, the system may pass through various internal states, but externally the changes are always consistent across the coordinated resources. For instance, in an application that transfers funds from one account to another, consistency ensures that the account is either viewed as having a complete credit or none at all.

Isolation

All transactional operations must be isolated from each other. Therefore, while operations are still in progress, the system prevents any transaction from observing or acting upon any inconsistent or intermediate state of the operation.

Durability

The results of consolidated operations must be persisted to stable storage, even in cases of catastrophic failure. For example, in an application that transfers funds from one account to another, the durability property ensures that the changes made to each account will not be reversed, even if the system fails.

Two-phase commit protocol

The two-phase commit (2PC) protocol is used in the coordination operations that span over several resource managers to ensure the ACID properties are achieved. Under the 2PC protocol, an actor must coordinate all operations.

For example, in the food industry there is a manufacturing transaction that receives customer requests for a certain product (for example, donuts) and orchestrates the food processing. The transaction has a Resource Manager (RM), which is a queue manager that sends a message to start the mixer machines to make the dough. If the coordinator needs to abort the transaction and the machines are already working, the dough could be totally or partially manufactured and so would have been subtracted from the pool of raw materials. However, the database of raw materials may now be inconsistent with the reality. This is where a transaction monitor with extended 2PC functionality is needed.

Additional information about transaction management is available in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

Stored procedures

A *stored procedure* is a user-written program that can be called by an application with an SQL CALL statement. It is a compiled program that is stored in a relational database (such as DB2), and can execute SQL statements, and also invoke other systems including CICS or IBM IMS™ systems by using the relevant batch interfaces.

DB2 stored procedures can be used to coordinate transactional updates in other z/OS systems, such as CICS, IMS, or WebSphere MQ, by using z/OS resource recovery services (RRS) and the DB2 RRS attachment facility (RRSAF). When used with the CICS external communication interface (EXCI), updates within CICS can be coordinated by the calling stored procedure, which then issues the relevant RRS commands to commit or back out the unit-of-work across all the connected systems

2.3.2 Data integrity

Many, if not all, enterprise business processes today rely heavily on the integrity of their data access facilities. A business process without guaranteed data integrity opens the door to possible legal issues and liabilities, can damage the company's reputation, and might even affect the company's general financial ledger.

To fully exploit the benefits of the various data management systems in the z/OS arena, CICS delegates much of the low-level data integrity to underlying systems such as DB2 or virtual storage access method (VSAM), which control the data access. This section summarizes the types of index files, databases, temporary storage facilities, and queued data.

File control

CICS uses the file control (FC) facility to access the storage devices, data sets, files, and data tables listed here. The FC allows access through the virtual storage access method (VSAM) and the basic direct access method (BDAM). The supported storage devices can be used in shared or non-shared mode.

VSAM data sets

Through VSAM, CICS supports access to various types of indexed data. The following data sets are the three basic types:

- ▶ Key-sequenced data set (KSDS): A file where each record is identified by a unique key. A KSDS is similar to an indexed table on a database.
- ▶ Entry-sequenced data set (ESDS): A file where each record is identified by a relative address.
- ▶ Relative record data set (RRDS): A file where each record is identified by a unique calculated number. An RRDS is similar to a hash table.

Data sets can be shared concurrently between different CICS regions using record-level sharing (RLS), or they can be accessed directly by a dedicated CICS region (known as a *file owning region*). CICS does not handle the sharing directly but instead uses the capabilities provided by the data facility storage management system (DFSMS), also known as the VSAM server. For more information about restrictions in this approach, see Chapter 3 of *CICS Transaction Server from Start to Finish*, SG24-7952.

Figure 2-5 shows the elements that are necessary to provide shared data sets within a CICS environment. A parallel sysplex with a coupling facility is required, each LPAR must have an active SMS VSAM server, and the requestor CICS has to be in the same sysplex. To have recoverable capabilities similar to those given by a DB2 datasharing group with the VSAM data sets, CICS VSAM Recovery for z/OS is required.

CICS VSAM Recovery for z/OS provides forward recovery log capabilities. The changes to the VSAM data set are stored on a system log. If there is a malfunction of a VSAM data set, such as a backup failure, a previous backup can be restored. The changes that are stored in the forward recovery log are also restored on the VSAM data set.

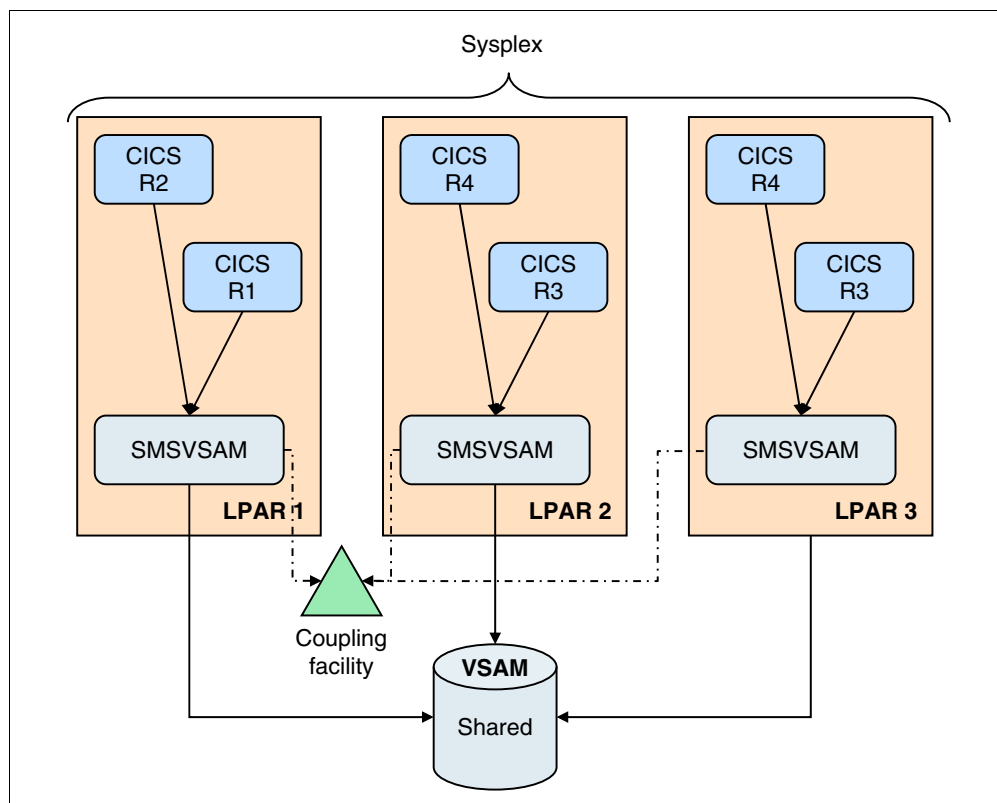


Figure 2-5 Data set sharing under CICS

Data tables

CICS manages data that is stored directly in memory by using a facility known as *data tables*. Data tables can be shared between CICS regions on the same LPAR by using the cross-memory services of z/OS. The data tables are backed by VSAM data sets and support only KSDS. This capability is similar to the concept of *files in memory* available in other platforms.

CICS supports the following data tables:

- ▶ CICS-maintained data table (CMT): Tightly coupled with the source data set. CICS changes the data table and its data set. This table cannot be shared.
- ▶ User-maintained data table (UMT): Uncoupled from the source data. CICS changes only the data table. The data set has to be managed by the user and so must be included in the logic of the application. This data table can be shared.

From the application perspective, a data table is another KSDS data set that is accessed through VSAM. CICS uses the definition of the data and accesses the memory directly. This way allows faster response time compared to a regular data set.

Data tables can be a way to optimize the response time of a CICS application without touching a single line of code in the application. The application handles a VSAM data set, which CICS uploaded into memory and later handles its access from the application. This way can improve the performance of the application.

Important: The CICS TS file control facility handles a special type of data table. The coupling facility data table (CFDT) is used to share data between CICS regions across a sysplex.

Databases

CICS has the ability to exchange information with both DB2 and IMS databases. The supported database subsystems are viewed as resource managers with which transactional updates need to be coordinated.

DB2

CICS provides the CICS DB2 attachment facility for accessing DB2 relational data. The connection between CICS and DB2 is multi-threaded, and the threads are individually handled by CICS depending on the type of configuration being implemented.

Figure 2-6 on page 35 shows how you can coordinate several CICS regions in your environment. When CICS works with DB2, the unit of work is inside the scope of CICS. If CICS needs to coordinate an additional RM, coordination can be done by using another TM such as CICS or WebSphere Application Server, because CICS can only coordinate a single DB2 RM. Between CICS instances on the same sysplex you can use a distributed program link (DPL) request by using CICS MRO connectivity; or, for a remote CICS system, DPL requests can be sent across intersystem SNA or IPIC connections, or by web services.

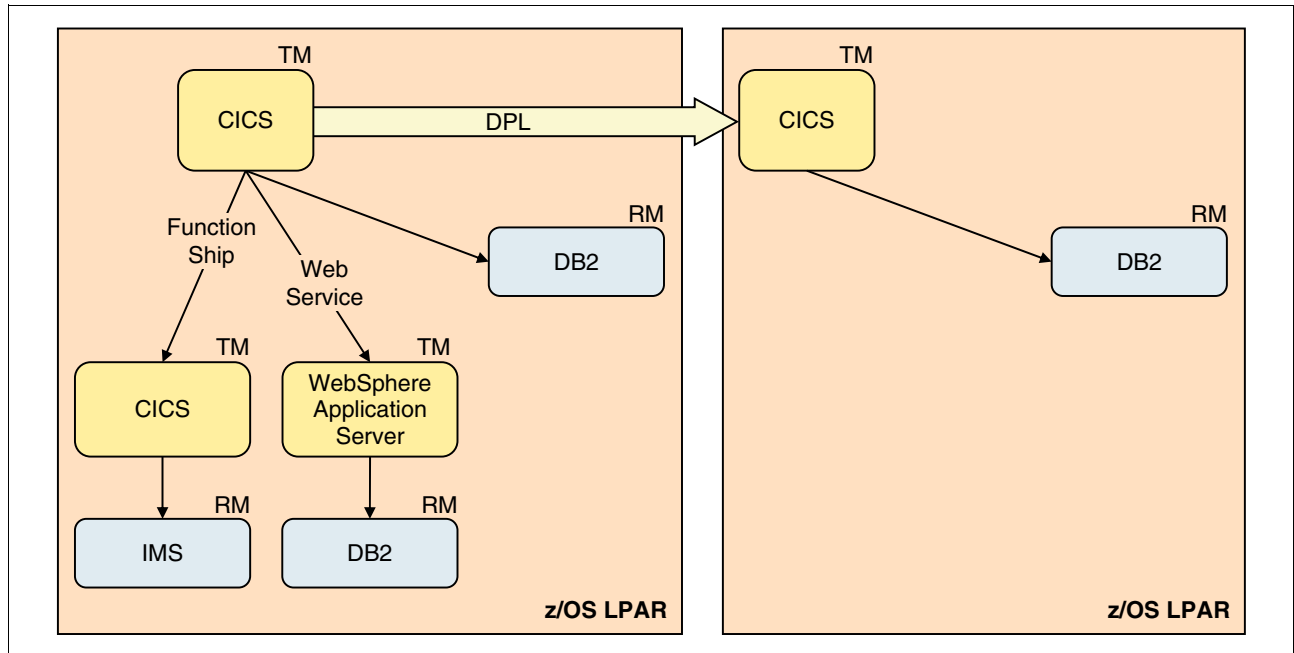


Figure 2-6 Transactional Environment CICS-DB2

There are several aspects of the CICS DB2 attachment facility that must be understood:

- ▶ There are no DB2 release dependencies within the attachment facility. CICS can connect to a DB2 subsystem running any supported level of DB2.
- ▶ A CICS address space can connect to only one DB2 system at a time.
- ▶ More than one CICS system can be connected to a DB2 system.
- ▶ CICS does not support SQL function shipping (where requests are shipped to a remote CICS region before being executed).

IMS

CICS has two ways of coordinating the resource manager from an IMS database. In Figure 2-7, CICS can work directly with the DBCTL interface by using the DL/I resource manager. It can also use the DBCTL interface to coordinate the IMS Database Manager or Transaction Manager if both transaction managers are on the same LPAR. In both cases shown, there is a single unit of work.

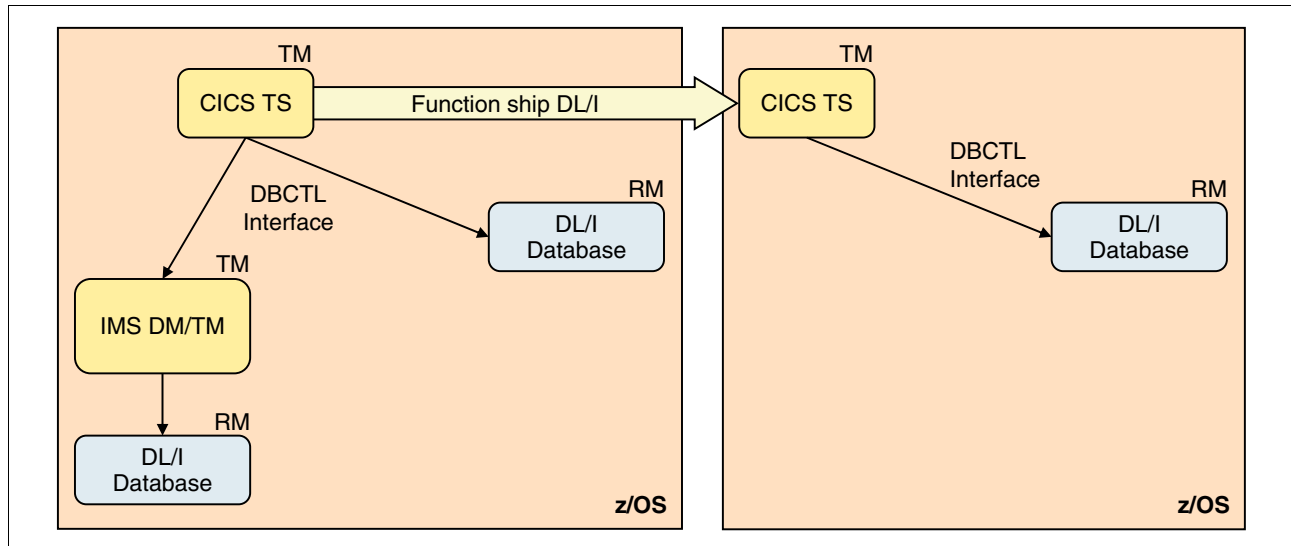


Figure 2-7 Transactional environment CICS-IMS

CICS TS can access the IBM DL/I databases in two ways:

- ▶ Using the CICS-DBCTL interface
- ▶ Using Remote DL/I with CICS function-shipping a DL/I request to another CICS system, where the DL/I support can be remote DL/I or DBCTL.

Temporary storage

CICS temporary storage is a scratchpad facility that is heavily used in many systems. Data in temporary storage tends to be short-lived, so emphasis is placed on ease of storage and retrieval.

You can set up temporary storage for a CICS region in three locations:

- ▶ Main storage in memory
This option is the fastest, but it is limited to the addressable memory (currently 64 bits). The main storage can be in the CICS region or in a remote queue-owning region (QOR). This location supports recoverability.
- ▶ Auxiliary storage on disk
This option is slower but has fewer size constraints. The storage can either be local to the CICS region or in a remote queue-owning region (QOR). This location allows the data to be defined as recoverable, thus maintaining transactional integrity.
- ▶ Shared temporary storage pools
These pools are shared by using the z/OS coupling facility, through the facilities of the CICS temporary storage server.

Timer and counter servers

CICS provides two additional servers that provide sysplex-wide services:

- ▶ **Timer service:** An interval timing and alarm clock service. It is usually used in a parallel sysplex environment to provide coherent timing services across the sysplex LPAR.
- ▶ **Counter service:** A facility for generating unique sequence numbers for use by application programs in a parallel sysplex environment. It is usually used to generate an understandable sequence of numbers across the sysplex LPAR, which can then be used by applications as unique-state tokens.

Transient data

CICS transient data provides generalized queuing capabilities that enable data to be stored for internal or offline processing, and also supports a triggering facility that can automate processing based on predetermined queue depths.

Queues are classified as either *intrapartition* or *extrapartition*. Intrapartition and extrapartition queues can be referenced through indirect destinations. Indirect queues provide some flexibility in program maintenance because data can be routed to one of several queues with only the transient data definition, and without the program itself having to be changed.

- ▶ **Intrapartition queues** are queues of data held in a CICS-managed direct-access data set. The data is only accessible by CICS transactions within the CICS address space. Once the queue is read, the information is purged from the queue.

Examples of data queued for intrapartition processing:

- Transactions that require processes to be performed serially rather than concurrently (for example, a process in which pending order numbers are to be assigned)
- Data to be used in a data set update (file update) and which can pass through the queue to allow the data to be applied in sequence

- ▶ **Extrapartition queues** are external data sets that reside on any sequential device such as DASD or tape.

Examples of the data queued for extrapartition processing:

- Logging data, statistics, and transaction error messages.
- Data that can be routed to an output device (such as to a printer)
- Data that CICS uses to send and receive messages to and from domains

2.4 Secured environment

Businesses depend on the data and information managed by their IT systems. Therefore, they need to manage and control the credentials of the users and applications that access those systems.

With the advent of pervasive web-based interfaces and new mobile channels, CICS services must cope with a potentially diverse range of users from a wide variety of systems. Without corresponding growth in data security practices, unauthorized access to critical data, or modification or even destruction of that data, could result.

As an online transaction processing system with up to thousands of users, CICS clearly needs the security to ensure that the application programs and data it manages are protected from unauthorized access.

2.4.1 CICS security integration

All requests or tasks that run in CICS do so under a specific credential or user ID. How this security context is established is key to limiting the activities of the different CICS users, and providing various levels of security for various types of requests (system-to-system requests, user requests, or administration activities).

CICS uses an external security manager (such as RACF, TopSecret, or ACF2) to manage the profiles that are used to authenticate and authorize access to its resources. When there is a request to attach a task or access a protected resource, the credentials of the requestor are validated using the information from the external security manager.

Figure 2-8 summarizes the main user identities that are used within CICS to manage security.

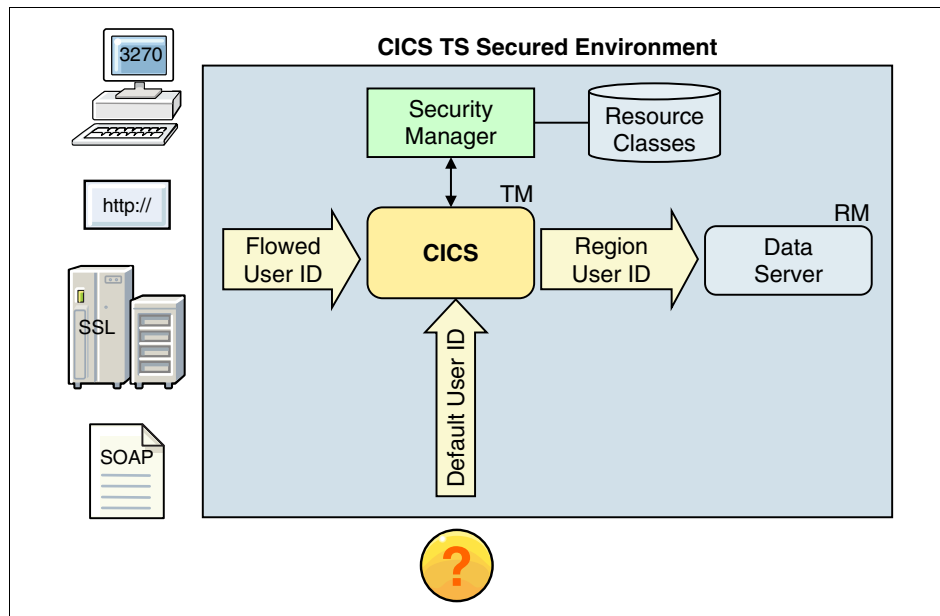


Figure 2-8 CICS security

The identities shown in Figure 2-8 are explained in the following list:

- ▶ Flowed user ID: An authentication token for an external user (human or machine). It is established when a request is initiated from one of the following routes:
 - A terminal session
 - The distinguished name from a digital certificate or WS security token
 - A flowed user ID from another CICS system
 - A connection from a third-party system such as the CICS TG, CICS sockets, WebSphere MQ, or WebSphere Optimized Local Adapters.
- ▶ Region user ID: Used for checking authorization when CICS requests access to system resources such as CICS data sets or other servers.
- ▶ Default user ID: Assigned to a request by CICS when no credentials have been established. These IDs typically will have limited authorization.

For further details about creating a secure CICS infrastructure, see Chapter 9, “Security and auditing” on page 219.

2.4.2 End-to-end security

Business applications must comply with various governmental regulations and laws, which can require, for example, the recording of which individuals accessed what data, and when. Typically, a means of *identity assertion* is used, whereby external identities are mapped to a set of internal identities known to the CICS security manager. However, the original distributed identity can be lost during the mapping process, so the true identity behind each request can be difficult to track.

To address this requirement, CICS, together with RACF, supports *identity propagation*, which allows the RACF security manager to map credentials from the distributed environment to known internal CICS credentials, and then to maintain this information with the new, mapped credential. From a distributed perspective, this CICS identity propagation is a clean solution that marries the LDAP capabilities with the RACF capabilities, without requiring extensive re-engineering of the security environment.

Figure 2-9 shows an example of identity propagation. The external credential (distributed identity) is based on LDAP as the remote security manager for the WebSphere Application Server. The distributed identity is sent to CICS with each request and mapped to a preset RACF user ID by using rules established in the RACF security registry. Authorization to access CICS resources is then controlled using the mapped RACF user ID, but the mapping information (such as the LDAP distinguished name and realm) is stored with the new mapped credential and is available in the SMF audit records for the resources that CICS uses.

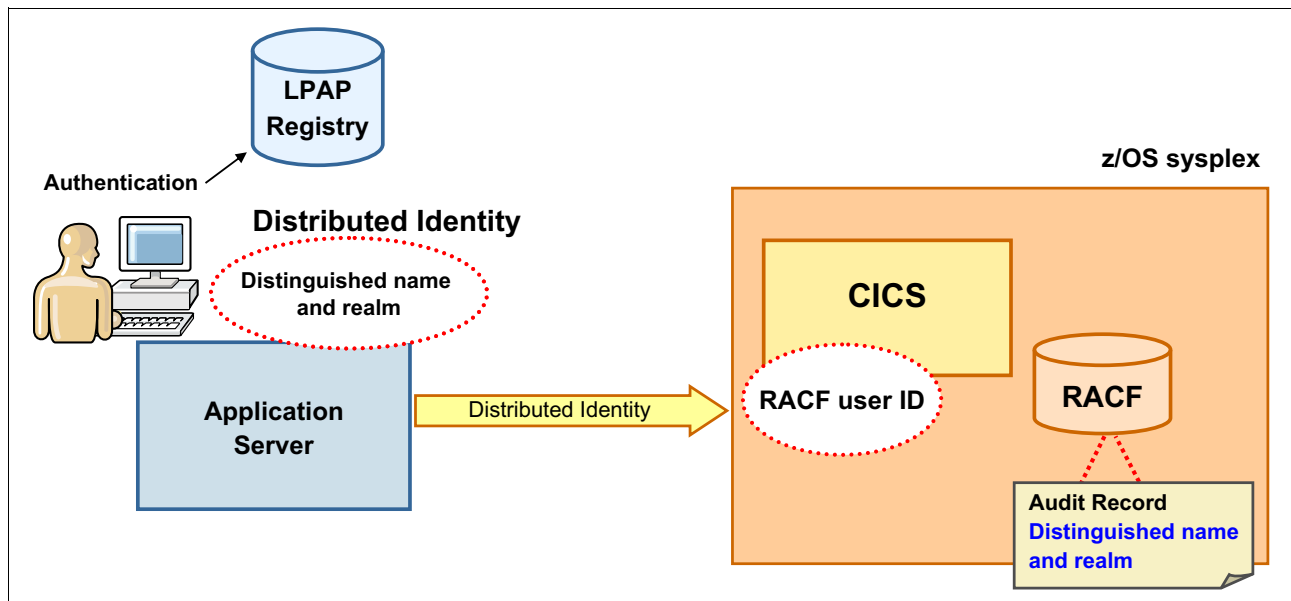


Figure 2-9 Identity propagation on CICS

This approach allows a common credential to be used across multiple systems, even if each uses inherently different security systems. Supported scenarios for identity propagation in CICS include the following items:

- ▶ Inbound to CICS from WebSphere Application Server through the CICS Transaction Gateway (CICS TG)
- ▶ Inbound to CICS as a web service request using the IBM DataPower® Appliance to map web services (WS) security headers to the correct format
- ▶ Propagated CICS inter-region requests between CICS regions using IPIC or MRO connections

2.5 Highly available and scalable processing

The cost of failing to provide a critical transactional service can be enormous. Beyond the monetary and market competitiveness aspects, a disruption of service can also affect other areas such as customer loyalty and regulatory compliance.

Just as businesses usually start small and grow over time, the systems that support them must be able to grow over time also.

Businesses need solutions that keep behaving as efficiently as the first time they were run, no matter what new demands are placed on them. Businesses also need solutions that can grow in capacity, throughput, and utilization as needs increase.

2.5.1 High availability

For business-critical applications, the underlying IT infrastructure must support continuous service availability and accommodate varying usage levels without failure. A well-designed, high-availability infrastructure can resolve these issues by building on the unique technology that is provided in IBM System z Parallel Sysplex®.

Benefits

The cost of downtime is so great that many enterprises today can no longer afford planned or unplanned outages. Even beyond the financial aspects, downtime can also affect key areas of customer loyalty, market competitiveness, and regulatory compliance.

A high-availability infrastructure can counter this situation by continuing to function even if individual components cease to work properly.

Barriers to implementation

Although high-availability configurations are highly desirable, implementation can be made challenging by a variety of technical factors, including the following most important factors:

- ▶ Management of state
- ▶ Persistent connections
- ▶ Transactional recovery implications
- ▶ Additional complexity
- ▶ Fail-back
- ▶ Uneven distribution

The z/OS solutions for balancing IP connections and dynamically routing CICS requests between CICS regions are the main techniques to attain high availability. Further details are available in Chapter 7, “High availability and continuous operation” on page 177.

2.5.2 Scalability

The graphs in Figure 2-10 on page 41 show how the ideal workload should behave. The graph on the left shows the expected increase in system throughput as an increasing number of requests come in (each request is assumed to initiate the same amount of work). The graph on the right shows system throughput increasing in direct proportion to the workload until an inherent system limit is reached, after which throughput remains constant but does not degrade.

Because the available resources are not infinite, there is always a point where a system is saturated. The questions are whether this limitation is acceptable and what are the limiting factors.

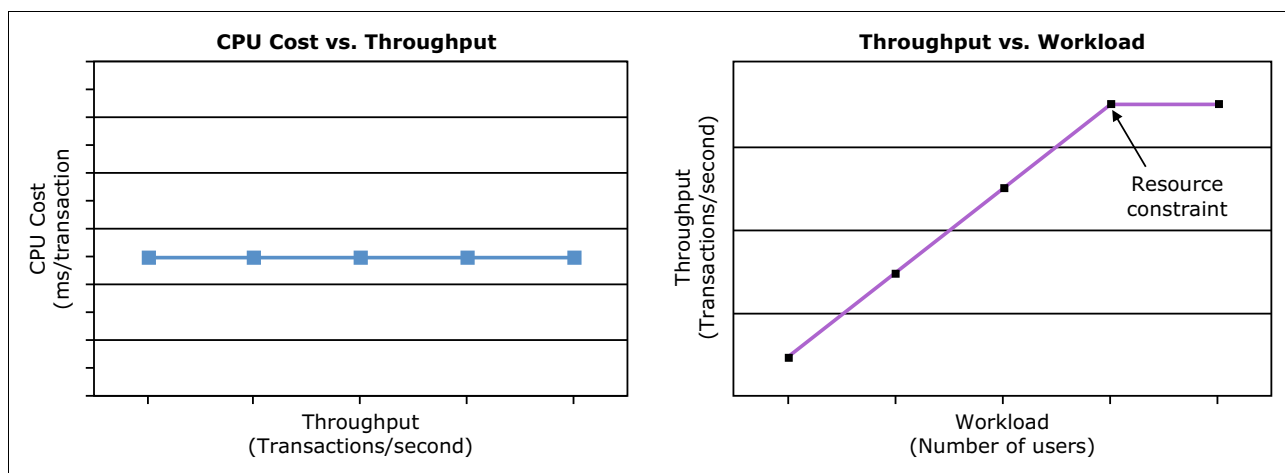


Figure 2-10 Positive linear scalability

Important: These graphs are for illustration purposes and are not intended as an accurate representation of real throughput levels.

For additional details about implementing scalable applications using CICS, see Chapter 8, “Scalability and workload management” on page 201. Further information is also in the following publications:

- ▶ *CICS Web Services Workload Management and Availability*, SG24-7144
- ▶ *Threadsafe Considerations for CICS*, SG24-6351

2.5.3 Techniques and facilitators

CICS is engineered to get as close as possible to the ideal of *positive linear scalability*, and to achieve as close as possible to 100% processor utilization without degrading critical application performance. CICS can scale in response to increases in workload by exploiting the load management facilities of IBM CICSplex® SM and z/OS workload manager.

CICS still has inherent system limits, as any other system. What makes CICS different is that it is specifically engineered to be capable of scaling to meet the demands of business transactions.

Multitasking

CICS supports multitasking. A single CICS region can run up to 2,000 tasks simultaneously. And a rich set of controls allows you to set priorities and limitations for specific transactions, including response times. As non-functional requirements change over time, CICS can respond to those changes without having to re-configure the business application.

CICS can use the following techniques to scale and meet new requirements for throughput:

- ▶ Horizontal scaling: The most common method is to *scale horizontally* by creating new CICS regions to handle the increased loads, and then distribute the work across the regions using a form of workload management. In this way, scaling is hidden from the business application, which does not to be stopped, recompiled, and restarted to benefit from the new resources.
- ▶ Vertical scaling: CICS also can use *vertical scaling* to support multitasking in the same region. To benefit from this technique, the business application must be built to exploit parallel processing on multiple threads (or TCBs) within a region. Such applications are marked as *threadsafe* and allow significant scalability within a single region. Because of these advantages, this is the preferred approach for building most new CICS applications.

Scaling is independent of the application code, which means scaling can be performed without either of the following typical IT roadblocks:

- ▶ Needing to change the business logic and deploy new code
- ▶ Disrupting the service for users

Optimizers

CICS provides the ability to use the special-purpose processors available on System z. Using these processors not only allows CICS to scale an application, but also enables more cost-effective processing, improving the price-to-performance ratio.

The following speciality processors are supported:

- ▶ System z Application Assist Processor (zAAP): A specialty engine to offload Java-based workloads from the general purpose engine.
- ▶ System z Integrated Information Processor (zIIP): A specialty engine to offload DB2 and XML-based workloads from the general purpose engine.
- ▶ Cryptographic coprocessors: Specialty engines to offload cipher processes from the general purpose engine.

Tip: Starting with z/OS V1.11, you can run zAAP-eligible workloads on zIIP engines. This way allows increased use of the zIIP engines and potential future savings.

Appliances

The WebSphere DataPower appliance is an SOA appliance built for delivering optimized and scalable SOA solutions. As specialized SOA hardware, it can be used to offload expensive operations from middleware solutions such as CICS, thus improving the overall scalability of a solution.

Typical operations using the DataPower appliance include processing large XML messages, validating XML digital signatures, and XML schema validation. These capabilities are also available in the DataPower XI50z, a blade-form factor that can be installed as a dedicated zEnterprise Blade Extension (zBX) into an zEnterprise ensemble.

2.5.4 z/OS Workload Manager

Organizations today process several types of work with various completion and resource requirements. Every organization wants to make the best use of its resources and maintain the highest possible throughput for optimum system responsiveness. Workload management makes this possible.

To understand the basic concept of z/OS Workload Manager, consider customers in a supermarket, as illustrated in Figure 2-11. When only a few people are paying for their groceries, they spend the expected time in the check-out line. But as more customers go to pay, the time required to check out grows. Therefore, the number of customers is the crucial factor in determining how long it will take for all of them to check out.

To avoid long waits for customers, more check-out lines can be opened. Some of these new lines can be fast-path lines for customers with few items. Optimization is achieved by identifying where contention might occur and responding as necessary.

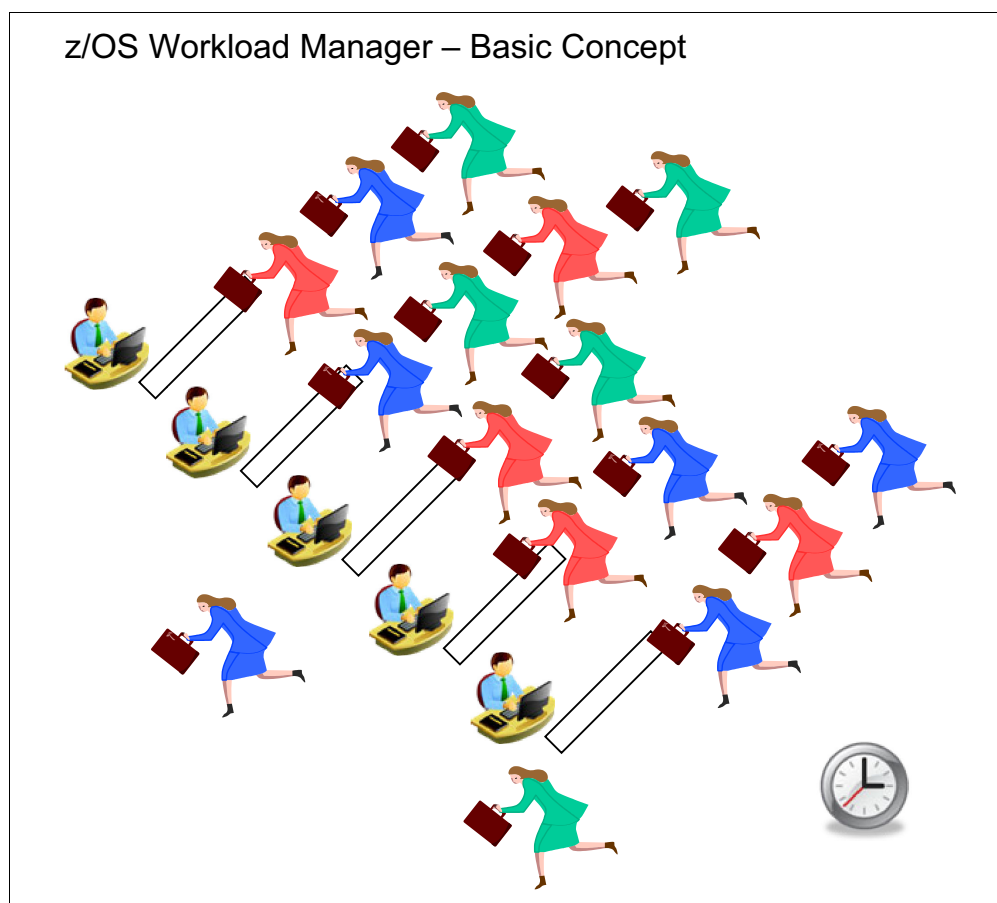


Figure 2-11 Workload management

Overview

One of the virtues of the IBM zSeries® platform and the z/OS operating system is the ability to run multiple workloads simultaneously within one z/OS image or across multiple images.

The function that makes this possible is *dynamic workload management*, which is implemented in the Workload Manager component of the z/OS operating system. The idea of z/OS Workload Manager is to establish a kind of contract between the installation (user) and the operating system.

The installation classifies the work running on the z/OS operating system in distinct service classes, and defines goals for each class in terms of the work that must be performed. Workload Manager uses these goal definitions to manage the work across all systems of a sysplex environment.

With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms and the system decides which resources, such as CPU and storage, will be allocated to meet those goals. Workload Manager constantly monitors the system and adapts processing as necessary.

Integration with CICS

From the CICS perspective, workload classification categorizes workloads with various priorities (for example, production or test). A service class can be thought of as a subset of a workload that has the same goal.

The following goal types are used:

- ▶ Response time
 - Percentile response time.
 - Average response time.
- ▶ Velocity
 - When the work “wants” to run, it can.
- ▶ Discretionary
 - The work is run when resources are available to do it.

Besides these goals, defining which work is most important is also necessary. In a case where several service classes do not achieve their target goals, the stated importance of the workloads helps Workload Manager decide which service class must get additional resources.

For further details about how CICSplex SM workload management functions in conjunction with the z/OS Workload Manager, see 8.3.7, “Workload management in a CICSplex” on page 215.

2.6 Simplified administration with CICSplex System Manager

An organization has to run day-to-day operational tasks on its runtime environment. These operations are commonly known as system administration or system operations. The goal can be summarized in a single phrase: *Keep it alive and running*.

Because it is based in System z architecture, CICS has a centralized administration style that saves time and money by encouraging IT staff to automate as many things as possible. CICS provides an extension, CICSplex System Manager, to simplify administration.

simplified administration has the following primary benefits:

- ▶ Less IT staff is required to support the infrastructure, freeing those resources for project and development initiatives.
- ▶ More efficient utilization of valuable technology assets occurs, protecting the previous investments of the organization
- ▶ Improved availability and performance of technology infrastructure and services allows users to work more efficiently and ensures that business is not lost because of unexpected downtime.
- ▶ Optimized IT service delivery makes IT part of the profit-generating side of the business rather than just the cost side.

The CICSplex System Manager (CICSplex SM) element of CICS TS is a system management infrastructure that enables you to manage multiple CICS systems across multiple images from a single control point. Enterprises in which CICSplex SM might be needed range from those running only a few CICS systems to those running hundreds of CICS regions or more. In the latest z/OS sysplex environments, having such large numbers of CICS systems to support a transaction-processing workload is becoming an increasing requirement.

All CICSplex SM components and resources, along with the system management requirements and the relationships between them, are held as objects in a VSAM-based data repository. These objects can be manipulated by using the CICS Explorer, an Eclipse-based management tool, or the web user interface, which is HTML-based. A batched repository-update facility is also provided for offline creation of CICS resource definitions.

Resource definitions are managed through a Business Application Services (BAS) component. Workload management, real time analysis, and monitoring services are used to manage the CICSplex SM configuration and CICSplex environment, and to gather statistical information.

From a system administration point of view, CICSplex SM has the following main characteristics:

- Single system image

CICSplex SM provides a real-time, single-system image of all CICS regions and resources that make up each CICSplex in the transaction processing environment. It provides a single point of control for the definition and deployment of resources in a CICSplex. CICSplex SM BAS is used to manage CICS resources that are stored in the data repository of the CICS CSD. However, the full power of BAS to deploy resources is realized when resources required by business applications are defined in the data repository.

- Monitoring

CICSplex SM provides facilities to monitor resource usage and performance in a CICSplex. CICSplex SM monitoring uses data collected by CICS monitoring and statistics, and retrieved by CICS INQUIRE commands to track the performance of individual resources. Although a full set of views are provided in the web user interface starter set to display CICSplex SM monitoring data, the primary purpose of monitoring allows performance exceptions to be detected by real-time analysis exception monitoring.

- Exception management

The RTA component of CICSplex SM monitors managed CICS systems for exceptional conditions that impact transaction throughput. The types of notifications that are provided when an exception is detected include issuing messages to the system console (of the creation of events for the CICS Explorer, or web user interface by an API program), sending a generic alert to IBM NetView®, and causing the system in which the exception occurred to be restarted by the z/OS automatic restart manager (ARM).

Developing CICS applications

Millions of CICS applications are running everywhere in today's world and there are more to come. In this part, we discuss the following aspects about CICS applications:

- ▶ How to design and develop CICS applications
- ▶ Ways to modernize existing CICS applications
- ▶ The development and operation lifecycle of CICS applications
- ▶ The necessary tools that can leverage the development and operation of CICS applications



CICS application architecture

This chapter is focused on the preferred practices for designing and developing CICS application programs. It discusses a range of techniques that are valid for both extending existing applications and developing new projects. It also shows how the CICS application run time provides a viable and modern option for a wide range of business applications.

This chapter includes the following topics:

- ▶ Application architecture
- ▶ Adapter layer
- ▶ Presentation layer
- ▶ Integration layer
- ▶ Business layer
- ▶ Data access layer

3.1 Application architecture

CICS Transaction Server has a wide range of capabilities that can readily be used by applications that are deployed into its run time, as illustrated in Figure 3-1. Application architects can rely on the runtime qualities of service that are provided by CICS, such as security, transactional integrity, and the dispatching of work for multiple processes or hardware clusters, without needing to design these qualities of service into the application code.

Because CICS provides an extensive range of system services, it is used by a wide range of industries and organizations around the world. A common factor for many of the usage scenarios for CICS applications is the provision of *high volume updates to shared data*, which is where the value of transactional integrity, security, and high-performance task-dispatching excel.

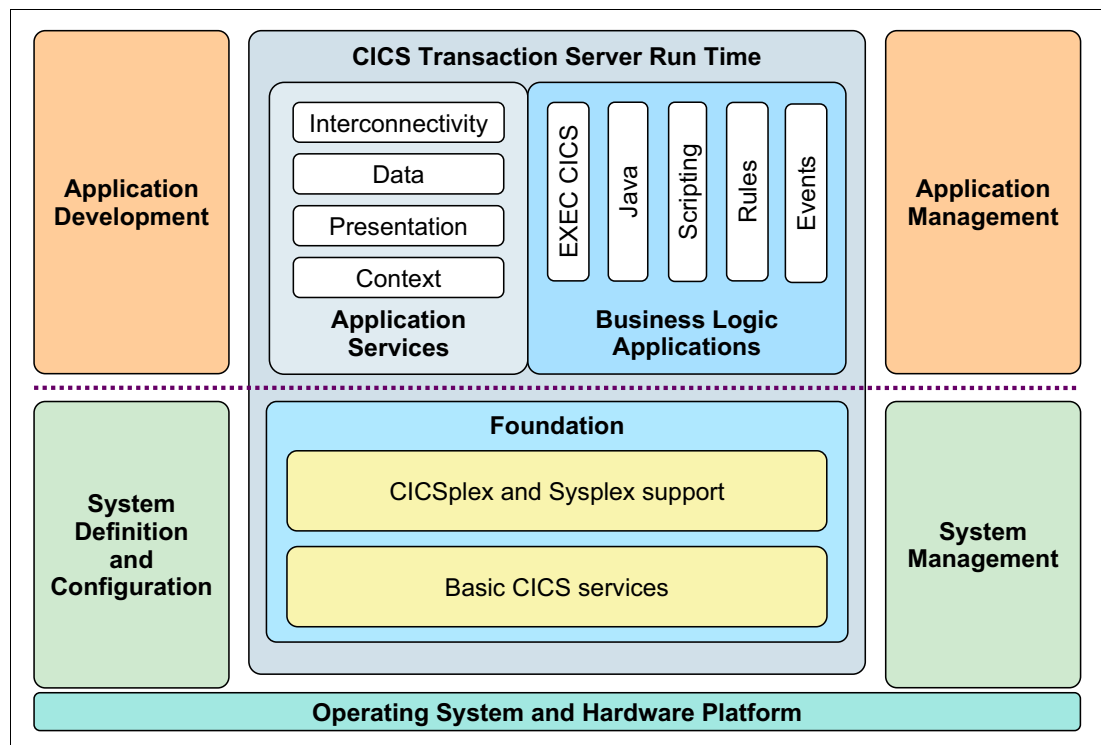


Figure 3-1 CICS services

3.1.1 CICS application architecture principles

When you define an application architecture, consider the following key design principles to obtain an application that meets the business requirement based on the qualities of service that are provided by CICS:

- ▶ Simplicity in the API model
- ▶ Agility for development languages
- ▶ Reliability in the design
- ▶ Scalability for performance and availability

Simplicity in the API model

CICS offers an extensive set of programming commands that an application program can use to request CICS services from the CICS run time. CICS also uses the full set of functions and procedures that are provided by the native language run time. The set of CICS commands is known as the *API*. The commands are macro statements that are supported in a wide range of languages and that are translated by the CICS translator before compilation. They provide the key to taking advantage of the qualities of service, such as transactionality and security from the CICS run time.

The commands that are provided through the API are categorized into the following areas:

- ▶ *Presentation services* are used for communication between the user, directly or with an intermediate system, and the transaction processing server. Presentation services work with the presentation management facilities of the system, such as a web browser or 3270 display device, which might be external to the CICS system.
- ▶ *Data services* retrieve and update data and provide access to files, temporary storage or transient data queues, and CICS journals.
- ▶ *Business services* manipulate data from the time that it is retrieved from storage to the time it is either presented or updated, and cover a wide range of functions.

With the CICS API, the application developer can maintain independence between the services of the business application and the transaction run time. This independence allows the creation of business applications that are focused on solving business processing.

Agility for development languages

Traditionally, CICS application development has focused on procedural languages, such as COBOL, PL/I, C, and assembly language. However, CICS Transaction Server has a range of capabilities that provide support for technologies such as Java, web services, PHP, and Web 2.0. As such, CICS provides a run time that allows you to continue to use previous investments while integrating modern components and third-party technologies. For further information about supported standards, see Appendix A, “Supported standards” on page 241.

Reliability in the design

Design the application to provide reliability, both in terms of providing transactional integrity when accessing data and the ability to respond in an acceptable and reliable fashion to incoming requests. The CICS API provides a rich set of response codes for each function, which provides a flexible means of handling application and system errors. These codes are integrated with the CICS transactional recovery routines, so applications can more easily provide error recovery while maintaining data consistency. For further details, see 2.3.1, “Transaction integrity” on page 30.

Scalability for performance and availability

CICS Transaction Server is designed together with System z to provide applications with a scalable application environment. In addition, a rich set of scheduling and administrative controls are available so you can control the runtime behavior of the application without modifying the business application. For further details about how to use these functions, see Chapter 8, “Scalability and workload management” on page 201.

3.1.2 Reusable modular model

The modular design model standardizes components that can be assembled together to build a bigger component. This model is often referred to as *component-based software engineering*. This kind of design decomposes the process into simpler elements that can be arranged in many patterns to benefit from synergies between the components. Modularity is the best way to meet many of the goals previously described. It can improve the potential to reuse existing assets in any design, either as-is or with limited composition.

In CICS applications, consider the following key functional layers, illustrated in Figure 3-2, for each application:

- ▶ Adapter
- ▶ Presentation
- ▶ Integration
- ▶ Business
- ▶ Data access

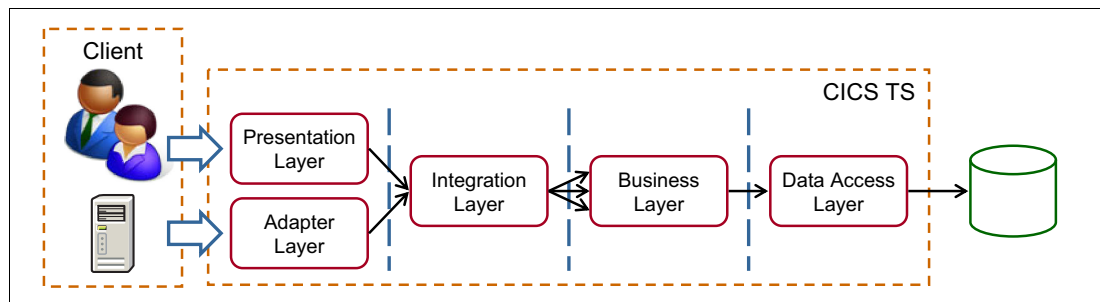


Figure 3-2 CICS modular application architecture layers

The *client*, or *service consumer*, initiates the request. Examples include a web service requester, web browser, WebSphere MQ client, TCP/IP socket client, 3270 device, z/OS batch program, or other CICS applications. Then, the request flows through the following architecture layers:

- ▶ The *adapter layer* processes the protocols and data with the client, establishes the transaction and security context in CICS, and works with the integration or business layer. For further details, see 3.2, “Adapter layer” on page 54.
- ▶ The *presentation layer* is a special case for web or 3270 applications, where the user interface is provided directly within the CICS application. For further details, see 3.3, “Presentation layer” on page 58.
- ▶ Optionally, when required, the *integration layer* implements a sequence of calls to business logic for situations where it is more efficient, offers better encapsulation, or simply makes the services easier to consume if the calls are done in CICS rather than the client making several calls directly to the business layer. For further details, see 3.4, “Integration layer” on page 62.
- ▶ The *business layer* implements the business functions of the service and is often the most extensive layer that has the most processing requirements. For further details, see 3.5, “Business layer” on page 64.
- ▶ The *data access layer* provides the logic to access the data store in use, such as DB2, IMS, VSAM, or other resources. Separating the data access from the business layer, facilitates reuse and allows changes to the data store type, without affecting other modules. For further details, see 3.6, “Data access layer” on page 69.

Designing the CICS application with these layers provides opportunity for reuse by many types of application. Layers provide callable interfaces that facilitate the opportunity for CICS workload management through the dynamic routing of **LINK** command requests between CICS regions. This approach provides benefits in terms of isolation, scalability, and high availability, as shown in Figure 3-3.

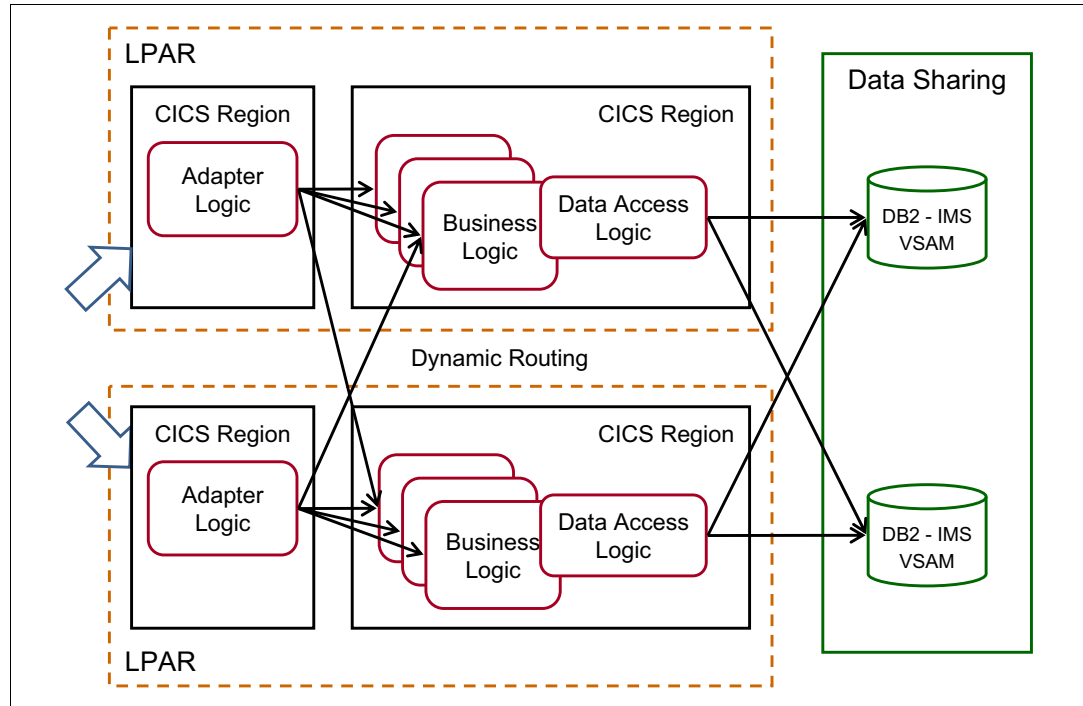


Figure 3-3 Dynamic routing of LINK commands

The **LINK** command provides the following options for passing data:

- Channels and containers

A *channel* is the interface that provides a set of one or more data areas, termed *containers*. Both the channel and the containers have user definable names and are scoped to the task automatically. A container can hold either character data, for which CICS manages the character encoding, or binary data (known as *BIT*), which is transferred with any conversion. The size of each container is limited only by the addressable storage in the CICS region. It can potentially be up to hundreds of megabytes.

Figure 3-4 shows an example channel and an analogy with an XML document.

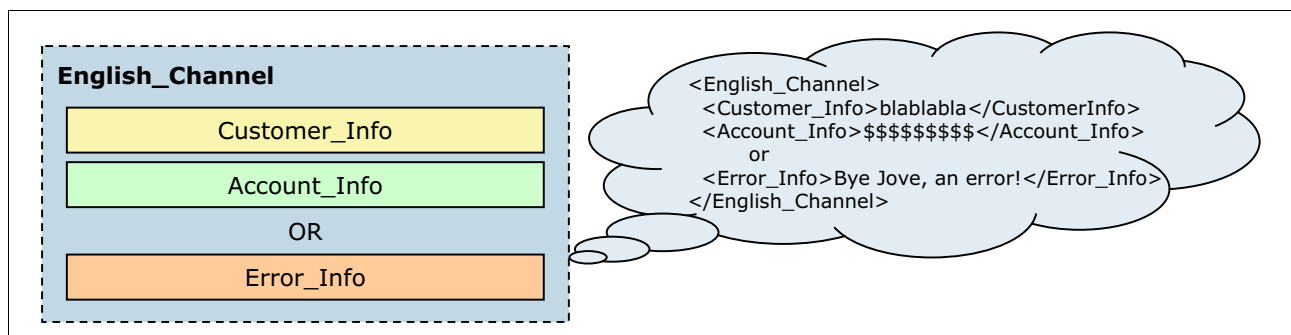


Figure 3-4 CICS channel message representation

- Communication area (COMMAREA)

The COMMAREA is a single predefined data area that was widely used by CICS applications before the channel interface was available. It can be passed between CICS applications and provides an automatically addressable data area that is limited in size up to 32 KB.

COBOL CALL statement: CICS also supports the COBOL CALL statement for one program to interface with another, which can be more efficient at run time than the CICS LINK command. However, the target program is required to run in the same CICS region. Thus, it is better suited when interfacing between programs within a layer, because it circumvents the ability to use CICS security or workload management functions.

3.2 Adapter layer

The *adapter layer* processes requests from the client system and adapts them to the interface that is provided by the business layer. This process likely involves managing the message formats and protocols that are supported by the client system. It also involves performing data conversion and marshalling between differing encodings (such as EBCDIC and Unicode) or big-endian and little-endian numeric formats. The transaction and security contexts for the application can also be established in the adapter layer.

An adapter typically addresses one or more of the following requirements:

- Business logic access technology transparency in two categories:
 - A *transport* adapter that handles the low-level transport, such as TCP/IP
 - A *protocol* adapter that identifies the operation and that supplies message envelope transparency
- Message serialization or deserialization transparency separate from the business logic

Such an adapter can also address EBCDIC data conversion. Typical examples are Java or XML to and from COBOL data types.
- CICS LINK transparency

When the existing interface is not using a CICS LINK command interface (for example, a COBOL call) and the technology that is used implies a LINK command interface (for example, a CICS web services provider application), a simple LINK command adapter is required that adapts a LINK command interface into a COBOL call.
- Business service exposure with no modification to the existing programming interface

This message mapping adapter maps an existing CICS program interface to a new business interface. For example, an adapter can expose an interface of 20 XML elements that are mapped from the selected 20 fields to an existing COMMAREA that contains many more fields.
- Business service granularity

A micro flow adapter can be used to increase the granularity of a service interface. It maps the course-grained business interface into multiple fine-grained program interfaces.

CICS application adapters can be implemented on the client side (a JCA connector, for example) or the server side, or both (web services or WOLA). The complexity of the implementation depends on the access technology that is used. The use of application tooling can significantly simplify the solution.

Depending on the access technology, CICS adapters can take various forms:

- ▶ CICS Transaction Server native support
- ▶ Generated code from IDE tools
- ▶ Connector client code
- ▶ Bespoke user code
- ▶ A mix of implementations

Adapters can also be deployed as part of an enterprise service bus (ESB) implementation so that the message adaptation is performed by the ESB, either partially (message simplification for example) or fully.

3.2.1 Message serialization adapters

A primary function of a message adapter is serialization or deserialization that handles the mapping of a Java object or XML representation to a byte array. Such an adapter can be hand-coded. However, this coding is a non-trivial task that involves the calculation of field lengths and placement and data-type conversions. The message serialization adapter implementation depends on the message representation technology, such as Java objects, XML, or JavaScript Object Notation (JSON).

Java and rational J2EE Connector architecture tooling

The J2EE Connector architecture (J2C) wizards that are available with IBM Rational Developer for System z and IBM Rational Application Developer provide tooling that generates code to handle message serialization or deserialization. This code produces getter and setter methods for each field in the message structure. The generated classes can then be used by Java methods to manipulate the COMMAREA or channel containers as required.

You can use the Rational J2C wizards to create a complete CCI proxy that generates the serialization or deserialization classes and generates a skeleton J2C bean that can access a CICS application. You can also use batch procedures for this process. The wizards also support optional EBCDIC character data and numeric conversion between little-endian and big-endian.

You can implement and test a typical CICS application interface access as follows:

1. Generate the serialization or deserialization data classes from the language structure.
2. Generate a skeleton J2C bean that can access the CICS application.
3. Create a J2C bean method that invokes the operation on the CICS server.
4. Create a web service to expose the functionality that is provided by the generated J2C bean.
5. Test the web service by using the web services explorer.

Java and JZOS batch toolkit

The JZOS batch toolkit for z/OS includes a set of tools for Java on z/OS. Within these tools, JZOS supplies a RecordClassGenerator utility for Assembler and COBOL language structures. This utility takes a sequential file as input and generates the serialization or deserialization Java bean class. The sequential file is generated from the ADATA compiler option using a two-step process:

1. Compile the CICS program with the ADATA compiler option to generate a binary file.
2. Run the JZOS Java utility to create the data classes from the binary file.

Pure XML or POX

With pure XML processing, the XML message is not handled by the CICS web service native support. XML message parsing and generation (deserialization/serialization) can be performed as follows:

- Using CICS XMLTRANSFORM API commands

You can use the **TRANSFORM XMLTODATA** command to convert XML to application data, and the **TRANSFORM DATATOXML** command to convert application data to XML. The **DFHSC2LS** and **DFHLS2SC** utilities generate the xsdbind file, which is analogous to the wsbind file that is created using the CICS web services tooling.

This method offers a balance between programming simplicity and infrastructure optimization.

- Using the COBOL XML PARSE verb

This method offers a CICS-neutral implementation. The programming style is more complex, as is the maintenance. The CPU consumption is higher than the CICS XMLTRANSFORM API, but the processing can be executed partly on a zAAP processor.

- Using Java facilities

This method offers a CICS-neutral implementation. The Java code is likely to be generated from standard Java tooling. The CPU consumption is likely to be higher than using CICS XMLTRANSFORM API, but the processing can be executed on a zAAP processor.

- Using the CICS DOCUMENT API commands for XML generation

With this method, you can use the DOCUMENT API commands to create XML documents from predefined skeletons. The variable parts of the skeleton (identified by the `&myVariable;` patterns) are substituted dynamically by the CICS **DOCUMENT** command support. This approach is a low-cost and convenient way to generate error or information messages.

Atom feeds in CICS

With emerging Web 2.0 technologies, messages often need to be serialized as XML documents in the Atom Syndication Format. CICS supplies a native message serialization adapter that exposes CICS resources, such as temporary storage queues and VSAM files, as Atom feeds with no need for user-written code. When you expose a CICS resource or application program as an Atom feed or collection, users can read and update the data by making HTTP requests from external client applications, such as feed readers or web mashup applications.

CICS Atom support allows you to quickly expose a variety of assets, such as VSAM files, a Transaction Server queue, or an application program as an Atom feed (Figure 3-5).

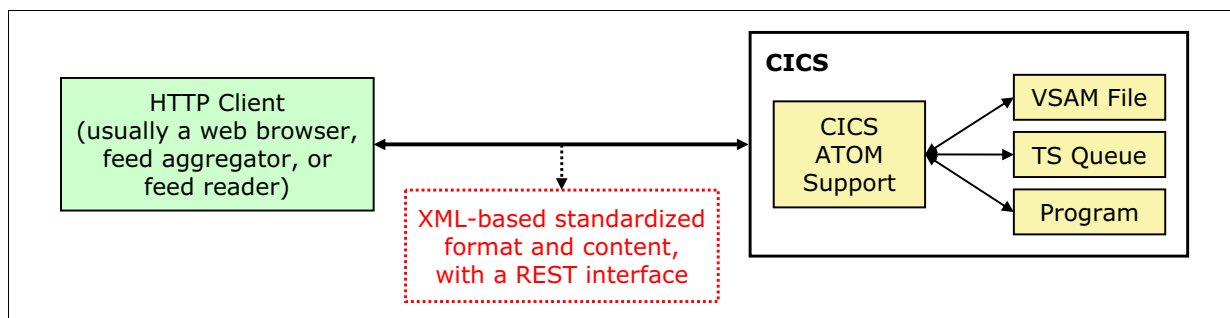


Figure 3-5 Atom support

An application program can also be exposed as a *feed*. In such a case, however, a user-written message adapter is required. Creating a message adapter can be quite a complex task. The CICS Dynamic Scripting feature offers a simpler adapter implementation using WebSphere sMash or Project Zero facilities. The adapter is written in Groovy or PHP from a few lines of code by using Project Zero renderers or APIs.

JSON

The JavaScript Object Notation (JSON) format is often used for serializing and transmitting structured data over a network connection. It is used primarily to transmit data between a server and web application in an AJAX web application programming model, serving as an alternative to XML. JSON message serialization can be implemented in CICS using the Dynamic Scripting feature. The message adapter is written in Groovy or PHP by using Project Zero renderers or APIs.

3.2.2 Adapter technology

The following adapter implementations are typically used in a CICS environment:

- ▶ Web service provider

The CICS native web-services support supplies adapters for the direct exposure of existing **LINK** command interfaces. Simple message optimizations can be performed, for example whitespace removal and specific field selection. The adapters are created by using the **DFHLS2WS** or **DFHWS2LS** utilities and are enabled by deploying the generated `wsbind` files. When required, a message adapter can be generated by using the Rational Developer for System z Enterprise Service Toolkit (EST), or they can be user-written.
- ▶ Web service requester

The CICS native web-services support supplies a full set of adapters for “simple” messages. The adapters are created by using the **DFHWS2LS** utility and are enabled by deploying the generated `wsbind` files. Web Services Description Language (WSDL) optimizations can be performed to simplify the messages with no impact on the service provider. An alternative adapter might be required for more complex messages. It can be generated by using the Rational Developer for System z EST wizards or implemented by user-written code.
- ▶ JCA or external call interface (ECI) with the CICS Transaction Gateway for z/OS

CICS Transaction Gateway and CICS Transaction Server supply a native transport adapter. The protocol and message serialization adapters are implemented on the client side. For JCA, the adapters are generated by using Rational J2C wizards. For Java on z/OS, the JZOS Toolkit can be used to generate the message adapter.
- ▶ JCA with WebSphere Optimized Local Adapters

The transport adapter is supplied by the WebSphere Optimized Local Adapters connector. The protocol and message serialization adapters are implemented on the client side. Rational J2C wizards or the JZOS Toolkit can be used to generate the message adapter Java beans.
- ▶ CICS WebSphere Optimized Local Adapters requester

The transport and protocol adapters are hand-coded from low-level WebSphere Optimized Local Adapters connector APIs. The message serialization adapter is typically implemented on the WebSphere Application Server z/OS server side. Rational J2C wizards or the JZOS toolkit generate the message adapter Java beans.

► CICS HTTP server

CICS Transaction Server supplies a native transport adapter. The protocol adapter is hand-coded by using high-level CICS web support APIs, typically a WEB RECEIVE/SEND sequence. The message format is optional, and a user-written message adapter is normally required.

► CICS HTTP client

CICS Transaction Server supplies a native transport adapter. The protocol adapter is hand-coded from high-level CICS web support APIs, typically a WEB OPEN/CONVERSE/CLOSE sequence. The message format is optional, and a user-written message adapter is normally required.

► WebSphere MQ

The CICS DPL bridge supplies native support for the transport and protocol adapters. The message adapter is likely to be a client implementation. When the DPL bridge is not used, a protocol adapter is required, typically a simple MQGET/MQPUT sequence. The message format is optional and a user-written message adapter is normally required.

3.3 Presentation layer

The *presentation layer* is the logic that defines the interface that is provided to a user. Historically, many CICS applications included logic to format data for display on SNA-based display devices, such as 3270 terminals. These devices are rarely used in most customer environments. Thus, ensure that the presentation logic is separate from other application components and can be modernized as business requirements adapt to new display technology, such as HTML, JSP, and Web 2.0 technology, all of which are now supported in CICS Transaction Server.

The sections that follow briefly describe the presentation functions that are available in CICS Transaction Server, which are summarized in Table 3-1.

Table 3-1 Comparison of presentation layer options

Option	When to use
3270 terminals	► Communication lines from the terminals have low bandwidth, and quick response time is needed. ► Experienced users can be highly productive. ► Can be used as an alternate presentation layer during a migration or modernization project.
Web support	► Integrates easily with existing web applications. ► Is a simple method for exposing CICS internal applications to web sites. ► Can increase the productivity of end-users with an integrated web application.
Liberty JSP support	► Is compliant with other JEE web containers, and HTML development tools. ► Is available only in Java.

3.3.1 The 3270 terminals

The 3270 terminals are a family of display and printer terminals, with supporting control units, that share common characteristics and that use the same encoded data format to communicate between terminal and host processor. Although this type of text-based interface is not as appealing for today's user, the simplicity of this interface is still used in many CICS applications.

The 3270 terminal supports the following basic functions:

- ▶ The screen sends and receives character information with predefined attributes, including placement, color, and visibility.
- ▶ An unformatted mode allows use of the screen as an input device. For example, you can use a laser scanner and send the stream of data to CICS.
- ▶ Support for special hardware and printers provides additional functions, such as light pen support.

Basic mapping support (BMS) is the CICS API interface between CICS programs and 3270 terminal devices and simplifies the building of display pages by using predefined maps.

3.3.2 Web support

CICS web support is a set of APIs and resources that are supplied with CICS Transaction Server for z/OS that enables a CICS region to act as an HTTP server and as an HTTP client. This support allows CICS applications to be invoked by and reply to HTTP requests.

When CICS is an HTTP server, a web client can send an HTTP request to CICS and receive a response. The response can be a static response created by CICS from a document template or static file, or an application-generated response that is created dynamically by a user application program.

3.3.3 Liberty JSP support

CICS Transaction Server V5.1 now offers a Java web container that provides developers with the features of Java Servlet and JavaServer Pages (JSP) specifications and local access to existing CICS applications and data. Built on the WebSphere Application Server Liberty profile technology, this web container runs in the CICS JVM server environment. You can use a wide range of Java development tools to develop web applications, such as WebSphere Application Server Developer Tools for Eclipse and Rational Developer for System z.

3.3.4 State management

In any interactive application, there is always a concept of state: information that allows each partner in the conversation to know how to proceed at the next step. Just as human memory stores short-term information during a conversation, two interconnected computer devices must also store short-term state to enable the conversation between the two devices to proceed, as illustrated in Figure 3-6 on page 60. Typically this state is managed in the client application that controls the presentation logic and in the interactions with the user.

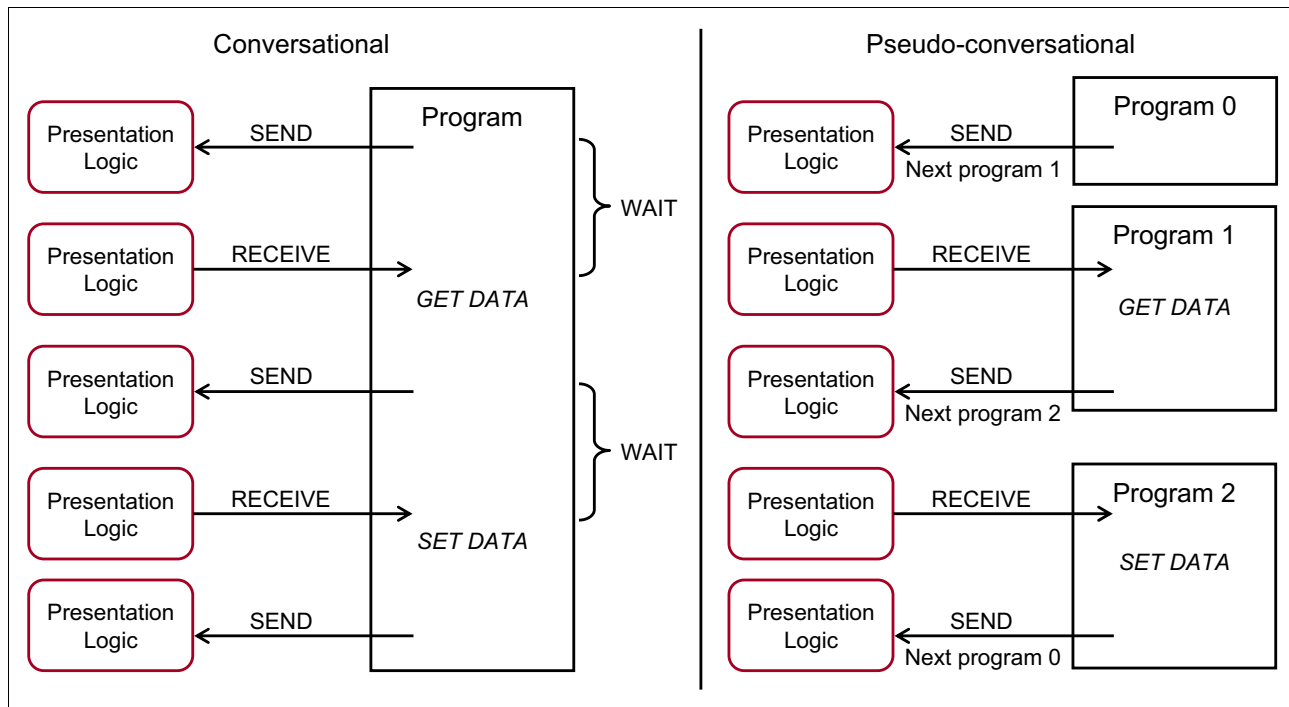


Figure 3-6 CICS pseudo-conversational processing

You can use the following principal models for developing interactive presentation logic in CICS:

- **Conversational programming**

Originally many CICS presentation modules were conversational in nature, so that the CICS transaction and the user enter into a long running conversational task. Although conversational tasks can be easier to write, they can also have serious disadvantages, both in performance (especially the need for virtual storage) and in their effect on the overall operability of the CICS systems containing them. Processors are now larger, with more storage and more power than in the past. With larger processors, conversational tasks are less painful in small amounts. However, if you use conversational applications, you can rapidly run into virtual storage constraints.

- **Pseudo-conversational programming**

The preferred practice in CICS presentation is to implement pseudo-conversational chains of tasks, with many separate tasks connected using a state data, and which appear to the user as one long conversation. The advantage of this practice is that the CICS transaction is not suspended while the user is thinking, and so fewer resources are consumed within CICS. Instead, the state of each side of the conversation is stored and managed within CICS, and it is this state that ties the sides of the conversation together into a meaningful dialog.

In CICS, the state of such a transactional conversation is usually maintained by the application, which for a 3270 based application would typically use a CICS data area such as the TCTUA or a COMMAREA passed between applications.

- **State in web applications**

Also, web-based applications must maintain state, to allow each series of web pages to be interlinked into a useful application. This state is somewhat more difficult to maintain in the web environment, because each HTTP request is stateless and might even involve the setting up and closing a TCP/IP socket for every HTTP request.

Consider, for example, an online share-trading application. In its simplest form, the user must provide an identity, select the company in which to trade, receive information about the current price, and perhaps other holdings, and then submit the order. At each stage, the application needs to know the information from the previous stage, and it needs to be able to prevent this information from being intermixed with any other transactions that are being processed at the same time. Because the web is based on the HTTP protocol, which is inherently stateless, the application must use its own mechanisms for passing this state information from one side of the conversation to the other.

Usually the presentation logic of a web application would run within the web server (or connected CICS system), and be driven by a Java servlet or JSP. At the end of each leg of the conversation, the presentation logic would need to pass to the web browser all the state information about the conversation so far (customer name, company chosen, and so on), and the web browser would need to flow this information back to the server with each subsequent request.

This situation then poses the problem that potentially large amounts of information would be flowed across the network, when it would much better be left on the web server. So instead of flowing such information with each and every request, it is possible to store the state data on the web server and flow only a unique identifier, termed a *state token*, with each request. This state token is used by the server to identify any given request, and to retrieve the correct state data from storage on the server, and so respond in the correct fashion. This management is illustrated in Figure 3-7.

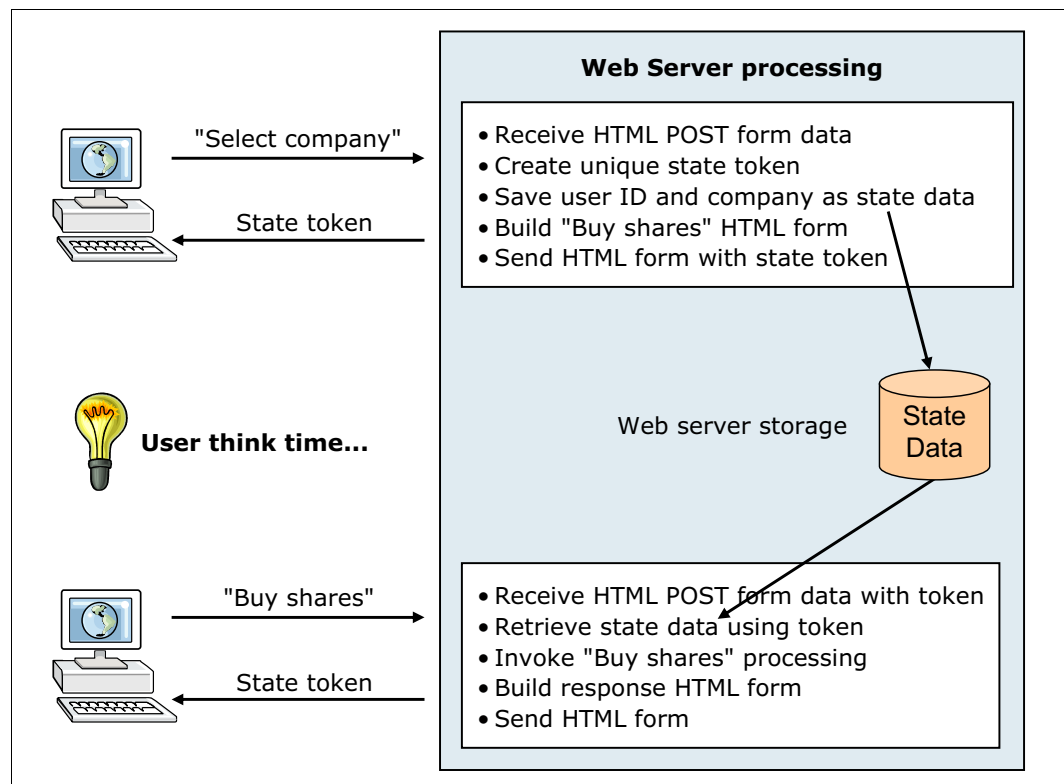


Figure 3-7 Web server state management

- Sharing of state information

An additional consideration when storing state data is that any locally cached information, such as state data, might need to be shared between CICS regions if a high availability solution is required. CICS provides extensive facilities to share data between CICS regions, which are described further in Chapter 8, “Scalability and workload management” on page 201.

3.4 Integration layer

The integration layer has a specific function in providing integration between the adapter and the business layers. When the adapter is implemented outside of CICS then the integration layer is also commonly implemented outside of CICS as it is often easier to handle the data formats in the adapter layer. However, if the principal function of the integration logic is in the grouping of multiple client requests, so that a single call can be made into the business layer, then consider implementing this function in CICS. This way provides obvious advantages in terms of reduced transmission of data and processing efficiency.

The integration layer can also be implemented in the business layer as part of the business logic in CICS. This behavior limits the reusability of the business logic, and so it can be beneficial to provide a separate module for this function.

3.4.1 Service flows

The integration layer can also be implemented within CICS by using a component referred to as the Service Flow Runtime (SFR). The Rational Developer for System z Service Flow Modeler (SFM) provides the tooling for generating flows that can be deployed into the SFR in CICS. SFR uses SFM-generated adapter services to provide the sequencing of 3270 screens in a CICS application. Thus, when a service request comes to CICS from a distributed application, SFR navigates the appropriate 3270 screen sequences, formulates a consolidated response, and sends a single service response to the requester.

Access terminal-oriented applications: SFM can generate flows that use FEPI, HATS, or the Link3270 bridge technology to access terminal-oriented applications that contain 3270 presentation logic.

SFM enables distributed applications to make business requests of existing CICS 3270 and COMMAREA applications as callable services. It enables customer-written applications to integrate seamlessly with business-critical 3270 and COMMAREA applications by generating service flows that contain a web service interface.

Figure 3-8 on page 63, shows the components that are used in request processing for a complex service flow.

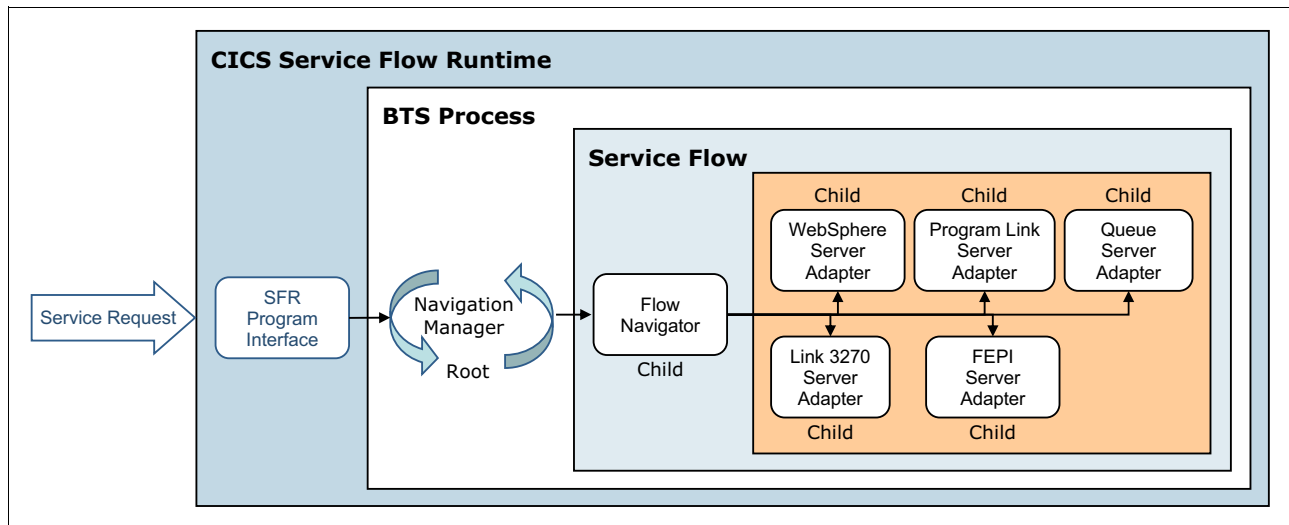


Figure 3-8 Service flow components

SFM provides modern graphical tooling based on the Eclipse platform. The tooling provides a means to simplify the reuse of existing 3270 and COMMAREA applications within business processes by assisting with tasks such as these:

- ▶ Interactions with terminal-based programs can be recorded in a simple manner by following a guided wizard.
- ▶ Application flow, conversion, and integration can be orchestrated within the tooling.
- ▶ The completed business process flow, including terminal interactions, can be generated as code for the Service Flow Runtime in CICS Transaction Server.
- ▶ The generated business process can be automatically deployed into CICS Transaction Server.
- ▶ Support is provided for exposing business processes as web services inside CICS Transaction Server.

For further information about the Service Flow Modeler, see the “Enterprise Service Tools for Web services and SOA” section of the IBM Rational Developer for System z information center:

<http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp>

3.4.2 Enterprise service bus

The enterprise service bus (ESB) is the component or set of components that provides connectivity between service consumers and service providers in a heterogeneous environment. CICS can collaborate with all of the IBM ESB solutions including WebSphere ESB, WebSphere Message Broker, and the WebSphere DataPower SOA appliances.

A main role of the ESB is to handle the following items:

- ▶ Connectivity and routing between your new and existing applications
- ▶ Data transformation and mapping
- ▶ Controlled deployment and versioning of services
- ▶ Policy enforcement, for example, enforcement of security standards
- ▶ End-to-end monitoring and measurement

WebSphere DataPower is frequently used in conjunction with CICS to help secure CICS services and to off load expensive operations by processing the complex part of XML messages (such as an XML signature). DataPower is also an ideal solution for other tasks such as auditing, XML validation, and identity propagation.

3.5 Business layer

The *business layer* implements the business functions of the service, and is usually the most extensive layer that will have the most processing requirements. Much of this layer depends on the complexity of the business applications and the service it provides; therefore, describing it is beyond the scope of this chapter. However, certain key principals should be understood; these principals can simplify the development and improve the efficiency of the business layers:

- ▶ Business rules
- ▶ Event processing
- ▶ Data integrity
- ▶ Threadsafe programming

3.5.1 Business rules

Business policies are usually embedded in a variety of documents and formats, such as technical requirement documentation, emails, minutes of meetings, or mentioned in conversation. You can also find rules hard-coded in applications as control logic in procedural code, data integrity constraints, or database triggers.

Business rules formalize a business policy into a series of if-then statements. For example, you can express a business policy to indicate that customers who spend a lot of money in a single transaction are upgraded, as shown in Figure 3-9.

```
if
    the customer category is Gold
    and the value of the customer shopping cart is more than 1500
then
    change the customer category to Platinum
```

Figure 3-9 Business rules example

All business rules have the same structure that includes a list of conditions that must be met before performing a list of actions.

As a business evolves and changes are required, hard-coded business rules and decisions inside application code complicate the necessary changes. By externalizing business rules and providing tools to manage these business rules, business experts can define and maintain the decisions that guide systems behavior. Tools can reduce the time and effort that is required to update production systems and can increase the organization's ability to respond to changes in the business environment. To help achieve these goals, a business rules management system can provide the following benefits:

- ▶ Storing rules in a central repository that can be accessible throughout the enterprise.
- ▶ Expressing policies and practices in natural language-based rules instead of computer code so that business analysts and experts can change decision logic themselves, without modifying application code, thus relieving overworked IT departments.

- ▶ Managing rule changes, including role-based management authority, testing and simulation, and configurable queries and reports, to ensure that rules are updated more quickly and accurately.
- ▶ Tools and environments that allow IT and operations departments to manage the deployment and monitoring of rules into the production environment.

Figure 3-10 shows the major components of a business rule management system (BRMS) system.

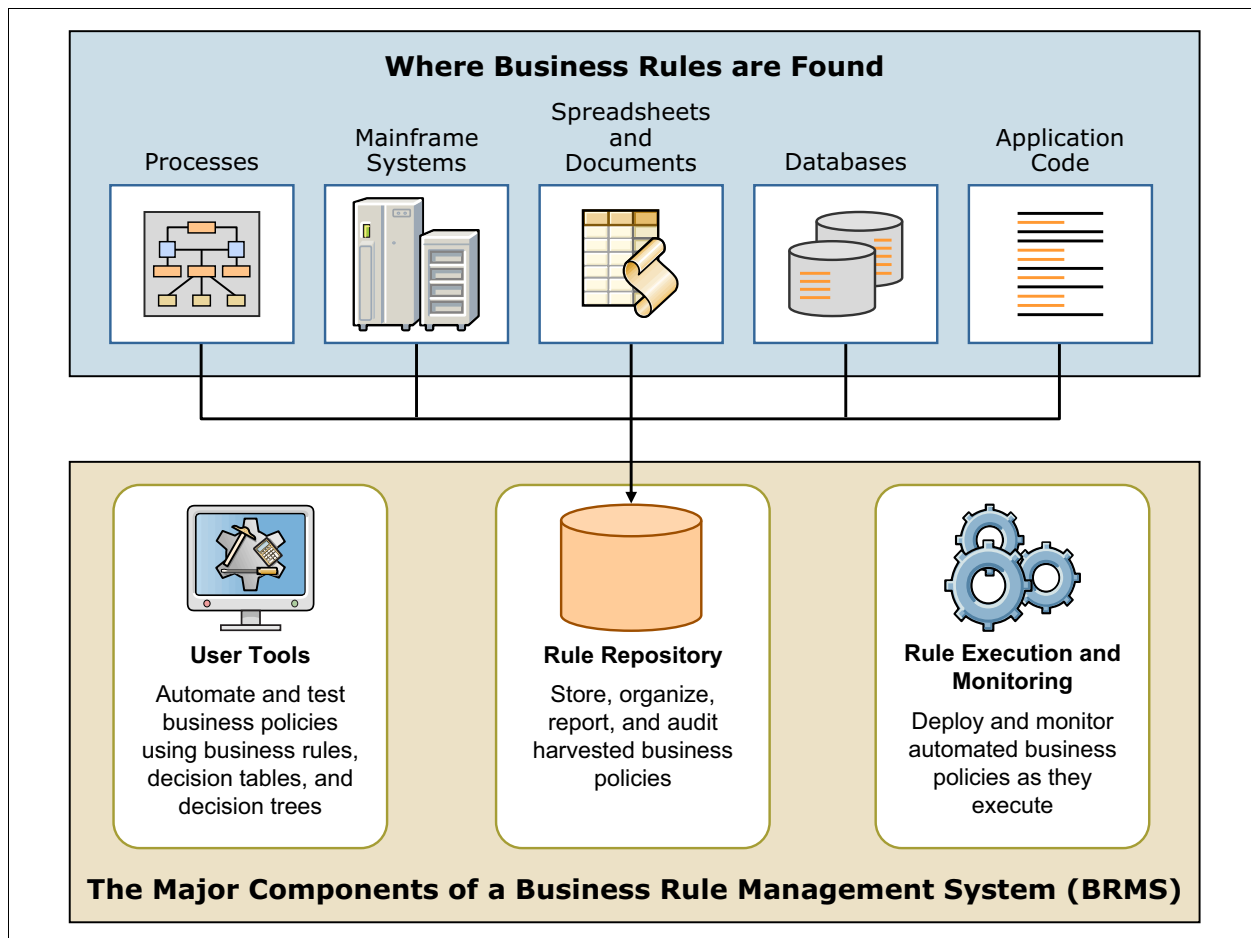


Figure 3-10 The major components of a BRMS system

Using WebSphere Operational Decision Manager for z/OS, you can create and manage business rules written in native COBOL data structures, deploy the rules in COBOL, and share these business rules across multiple platforms. You can also complete the following tasks:

- ▶ Document and manage business decisions executed in CICS applications.
- ▶ Author the rules for COBOL by using business terminology.
- ▶ Share rules and files between Java and CICS applications.
- ▶ Generate native COBOL code from rules and then execute inside CICS.

You do not need to re-engineer or rewrite an entire COBOL application to start managing the business rules for your CICS applications. You can base the scope of the rules on the following foundations:

- ▶ A set of rules for a specific region, territory, or type of customer
- ▶ A process or subprocess within a CICS application
- ▶ An entire COBOL program

You can then deploy the rules as native COBOL in the following applications:

- ▶ Batch
- ▶ CICS Transaction Server

For more information about business rules, see Chapter 4, “CICS application modernization” on page 73.

3.5.2 Event processing

A business *event* is an event that happens that is relevant to your business. An individual stock trade or the placement of an order are both examples of business events.

The CICS event processing run time detects instances of events that are enabled and captures the events and payload without the need to make application code changes. When CICS captures events, it carries out specified filtering, enriches the event with information about the application context in which it occurred, formats the event, and routes it to the appropriate event consumer. The event consumer can later, based on the detected pattern, take proper predefined actions.

The events consumer can be another CICS application, or the emitted events can be placed on a WebSphere MQ queue for the consumption by a complex event processing engine such as IBM WebSphere Operational Decision Management for z/OS.

CICS provides the following comprehensive support for simple business events:

- ▶ A CICS application can capture and emit business events with no change to the application itself, using non-invasive capture points, before and after selected EXEC CICS API calls, and when the program starts.
- ▶ For situations in which the noninvasive capture points are not sufficient to capture a specific business event, a new EXEC CICS API call, SIGNAL EVENT, allows events to be captured anywhere in a CICS application.
- ▶ The CICS Explorer includes the CICS event binding editor, a tool that helps you to define simple business events and create event bindings for your CICS applications. You deploy event bindings to a CICS system from the CICS Explorer.
- ▶ After CICS captures and processes a business event, the event is passed to an EP (Event Processing) adapter for formatting and routing. CICS provides EP adapters to allow business events to be emitted to the following products:
 - WebSphere MQ, either in XML format for consumption by WebSphere Operational Decision Management for z/OS, in Common Base Event XML format to the Common Event Infrastructure (CEI) for consumption by business monitors, or in a non-XML character format to a CICS transaction or to a CICS Transaction Server queue,
 - WebSphere Business Monitor with the Common Base Event REST format. The WebSphere Business Monitor Event Emitter REST interface will be used to receive events using HTTP.
- ▶ Custom EP adapters can be written by using the CICS API.

CICS event processing supports interoperability standards with business event consumers such as WebSphere Business Monitor and others, by emitting events in the Common Base Event specification V1.01 format. The common base event is an initial implementation by IBM of the Web Services Distributed Management (WSDM) Web Event Format (WEF) standard.

You can use event processing in many ways, such as to detect customer trends or to detect abnormalities in patterns of customer behavior to identify potential irregular or fraudulent situations or even to extend your current business logic by invoking other new CICS programs when a specific events occurs.

Important: When you plan to use CICS support for event processing, and in particular, when you design programs that consume CICS events, consider the following characteristics:

- ▶ The order in which events are emitted from a CICS program can vary from the order in which they were captured.
- ▶ The order in which events are emitted can vary from one run to another of an event enabled CICS program.
- ▶ In the unlikely situation of a CICS system failure after the capture of an event but before the emission of the event, the event cannot be emitted at all, whether or not the EP adapter in the event binding specifies that events are transactional.

More information about event processing is in 4.3, “Event processing in CICS” on page 82.

3.5.3 Transaction integrity

The business layer must always be designed with the integrity of the business data in mind. CICS provides comprehensive facilities to control the integrity of data that is written to recoverable resources (see 2.3, “Proven integrity” on page 29), and if resources are correctly defined as recoverable (or otherwise), then all access that uses the CICS or SQL APIs will ensure data integrity is maintained.

One of the key issues when you consider transactional integrity is the scope of the unit of work. In the CICS model, by default all services accessed within a CICS task are part of the same recoverable unit-of-work, which is implicitly started at task-attach, and completed at task-termination. But, to demarcate business logic into smaller recoverable units, use the CICS **SYNCPOINT** command to separate business processing into multiple units of work within the same business application request.

However, when considering a scenario with disparate application layers deployed into multiple CICS regions, the **SYNCPOINT** command can create a problem if the remote system where the adapter is running is designed to be the transaction coordinator. By design programs that are linked to, in CICS, cannot issue **SYNCPOINT** commands, because the CICS system is no longer the transaction manager. The CICS system is unable to coordinate the unit of work, which remains the role of the remote transaction manager, that is the client that invoked the **LINK** command. This model guarantees global transaction integrity but can limit reuse of business logic components by different adapters.

To address this issue, the client can issue a **LINK** command with the **SYNCONRETURN** option. This option ensures that the business logic runs within its own unit of work and is not coordinated by the client. It allows the linked program to issue **SYNCPOINT** or **ROLLBACK** commands to commit to, or to back out, specific units of work.

For further details about how this issue impacts access technologies, see *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

3.5.4 Threadsafe programming

A key benefit of CICS is that qualities of services, such as task dispatching, do not need to be developed within the application code. However, when considering task dispatching in CICS, you can use two separate models to provide parallel dispatching of applications within the CICS run time:

- ▶ Quasi-reentrancy
- ▶ Threadsafe

Quasi-reentrancy

Historically, most CICS application code ran under the main CICS TCB called the *quasi-reentrant TCB*. The CICS dispatcher sub-dispatched the use of the QR TCB between the various CICS tasks. Because only one CICS task can be active at any one time on a given TCB, each task voluntarily gave up control when it issued a CICS API command, which then caused a dispatcher to wait, allowing a different task to be dispatched to the TCB. However, the one and only QR TCB used only a single CPU. So, a single CICS region was limited to using one physical CPU, even if the LPAR contained multiple processors.

CICS applications that are defined as quasi-reentrant (using the CONCURRENCY attribute of the program resource) are invoked under the CICS QR TCB. Although this option can limit scalability and affect performance if accessing resource managers that run under OTE TCBs (such as WebSphere MQ or DB2), it has a distinct benefit that the development of quasi-reentrant programs are less stringent than if the program were to run concurrently on multiple TCBs. When running under the QR TCB, a program can be sure that no other quasi-reentrant program can run until it relinquishes control during a CICS command, at which point the user task is suspended, leaving the program still in use. The same program can then be invoked again for another task, which means the application program can be in use concurrently by more than one task, although only one task at a time can be running.

Quasi-reentrancy, therefore, allows programs to access globally shared resources, for example the CICS common work area (CWA), without the need to protect those resources from concurrent access by other programs. Such resources are effectively locked exclusively to the running program, until it issues its next CICS request. For example, an application can update a field in the CWA without using compare and swap instructions or locking (enqueueing) the resource.

Threadsafe

Modern CICS applications can now be developed as *threadsafe* and configured (using the CONCURRENCY attribute of the program definition) to run on multiple TCBs in parallel within a single CICS region. These TCBs are referred to as *open* TCBs and are assigned to a CICS task for its sole use. Several modes of open TCBs are used to support various functions, such as Java in CICS, DB2, open API programs, and C and C++ programs, which are compiled with the XPLink option. There is no sub-dispatching of other CICS tasks on an open TCB. Each new TCB can execute concurrently on one CPU, which gives the potential of increased throughput for a single CICS system in a multiprocessor system.

An application that is running on an open TCB cannot rely on quasi-reentrancy to protect shared resources from concurrent access by another program. Furthermore, quasi-reentrant programs might also be placed at risk if they access shared resources that can also be accessed by a user task running concurrently under an open TCB. The techniques used by threadsafe user programs to access shared resources must therefore take into account the

possibility of simultaneous access by other programs, and ensure appropriate locking techniques (such as compare and swap or enqueueing) are utilized when accessing shared storage areas. Typical examples of shared storage are the CICS CWA, the global work areas for global user exits, and storage that is acquired explicitly by the application.

Tip: CICS Interdependency Analyzer is a tool that can be used to analyze load modules and running CICS applications to help you identify CICS commands that give access to shared storage.

For most CICS resources, such as files, temporary storage queues, and DB2 tables, CICS processing automatically ensures access in a threadsafe manner. Most of the CICS API commands that operate on these resources are coded to use appropriate serialization techniques that allow them to run on open TCBs; that is, they are threadsafe commands. Where this is not the case (such as when using transient data queues) CICS ensures threadsafe processing by forcing a switch to the QR TCB, so that access to the resources is serialized regardless of the behavior of the command. For any other resources that are accessed directly by user programs, such as shared storage, it is the responsibility of the user program to ensure threadsafe processing.

Important: If you define a program as threadsafe, all routines that are statically or *dynamically* called from that program (for example, COBOL routines) must also be coded to threadsafe standards. When an **EXEC CICS LINK** command is used to link from one program to another, CICS can switch between threadsafe and non-threadsafe programs.

Performance benefits

For the reasons explained previously, you can design and implement new CICS applications as threadsafe to take advantage of the following benefits:

- ▶ Increased scalability allows CICS to take advantage of the multi processor capabilities of System z.
- ▶ Improved availability provides less queuing for the CICS TCB and allows for higher throughput per CICS region.
- ▶ Better performance reduces TCB switching within CICS when accessing resource managers, such as DB2 or WebSphere MQ, thus save unnecessary processing.

Tip: For improved performance, ensure that the program uses only threadsafe EXEC CICS commands. If you include a non-threadsafe **EXEC CICS** command in a program that is defined as threadsafe and running on an open TCB, CICS switches back from the open TCB to the QR TCB to ensure that the command is processed safely. The results of your application are not affected, but its performance might be affected.

3.6 Data access layer

The *data access layer* implements the data access for the business services. It is usually tightly coupled to the business layer (for reasons of efficiency) and is typically executed on the same CICS region as the business logic. Therefore, requiring a linkable interface to facilitate the dynamic routing of requests to the data access layer is not usual. However, it might still be desirable to have separate callable modules, which can help with reuse and facilitate development.

Many recommendations applied to the development of business logic, therefore, also apply to the development of the data access layer and in particular it is often highly desirable to ensure that applications accessing DB2 are threadsafe for the reasons discussed in 3.5.4, “Threadsafe programming” on page 68.

3.6.1 Data stores

CICS provides access to the following types of data:

- ▶ Record-based: Stored in VSAM files and accessed by using a key, a record number, or a relative byte address (RBA)
- ▶ Relational: Stored in a relational database, such as DB2 and accessed by using highly customizable SQL based queries
- ▶ Hierarchical: Stored in a hierarchical database, such as IMS DL/I

Each of these data access methods provide their own benefits and runtime characteristics. However, they all benefit from close integration with CICS and exploit the inherent security, transactional and run time.

3.6.2 Batch integration

On System z, batch processing where jobs are set up so that they can run to manual completion is central to many business process. Batch processing provides a simple and efficient way to perform validation and updates to business data during a *batch window*, when online access is not required. Batch processing can access data directly in DB2 or VSAM or can indirectly access data by using the CICS External Communication Interface (EXCI), which provides callable access to CICS applications from batch applications running anywhere in the sysplex.

However, as transaction volumes increase, the time window for batch access is under increasing pressure to be completed within the bounds of the typical overnight processing window. An application design needs to carefully consider the benefits of dedicated batch processing compared to online processing through CICS, as shown in Figure 3-11.

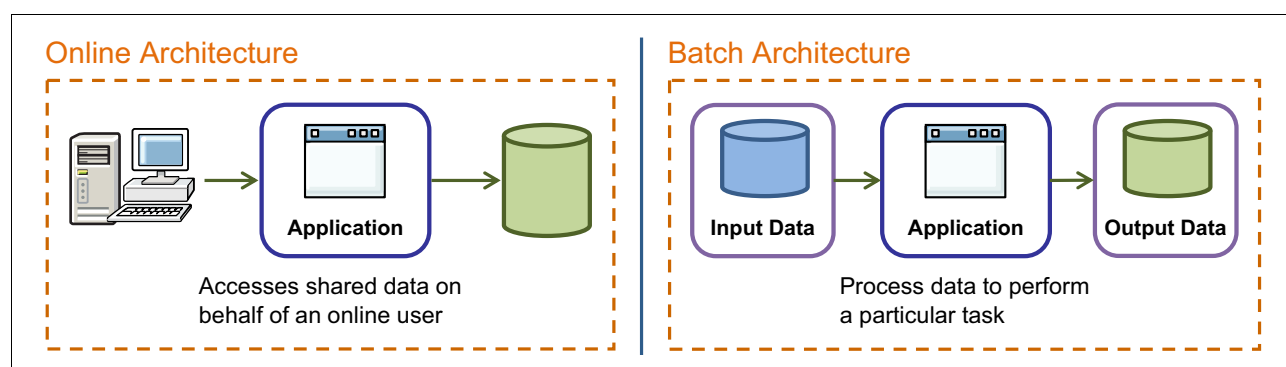


Figure 3-11 Online and batch

Integrating batch and online processing is often a key requirement in the provisioning of a complete solution architecture on z/OS. However, it is important to understand that dependencies exist between the technologies that cannot be separated easily, as summarized in Table 3-2 on page 71.

Table 3-2 Batch and online characteristics

Online	Batch
High volume updates to shared data	Processing of large amounts of consolidated data
Shared data	Exclusive access to data
Short lived transactional requests	Long lived transactions

Modular reuse

An efficient application design option is often to allow the reuse of modules in both the online CICS environment and the batch environment. This option requires careful design but can be facilitated by using one of the following techniques:

- ▶ Dynamic calls, such as the COBOL CALL statement, which allow modules to be called on demand
- ▶ Retranslation, where the source that is used in the online environment can be reused in the batch environments by retranslating and compiling source using the EXCI translator option

EXCI translator: The EXCI translator supports only the **LINK** command.

Running online and batch together

You can also run batch applications within the CICS run time as *hybrid batch*, where the CICS run time is used to provide a batch-like environment for processing. This option raises the following potential issues:

1. Batch jobs can take longer to run.

Traditional z/OS batch jobs are typically optimized to run as quickly as possible. When a batch job is moved into CICS, it might have to share resources with online applications. These resources can be CPU, files, or databases. Thus, the batch job might take longer to complete. At the same time, with the removal of the batch window, it might be that the batch job can start earlier or complete later.

For batch jobs with a hard time deadline, investigate whether the job can complete in the time required. These investigations can include Compute Grid capabilities such as the Parallel Job Manager, which enables a job to be partitioned into smaller segments and to run in parallel across Grid Execution Environments.

2. The impact to online application performance must be factored.

It is important that the batch processing does not negatively affect the performance of online applications. A user invoking a CICS transaction might expect a response time of tenths of a second. If the batch application locks too many resources at a time, it could affect the performance of the online applications. The checkpoint interval of the batch job can be adjusted to change the number of updates made within a checkpoint. The batch job that consumes too much CPU can also affect performance of the online application.

3. Data being updated by batch can be changed by the online applications.

Batch applications that are reliant on a set of records to remain consistent during the processing of the job might not work when run in parallel with online applications, because the online applications can update records that the batch job has read, or is about to read. This situation will need to be considered on a per application basis to determine if it is likely to be an issue.

WebSphere XD Compute Grid

WebSphere XD Compute Grid is a technology that can be used to decrease the batch window in a collaborative way. Compute Grid is a managed environment for batch applications that are scheduled, typically process large amounts of information, and can take hours to complete. Compute Grid provides checkpoint processing, repositioning of input and output data streams after a failure, and parallel job management, which are essential to efficiently manage, run, and possibly restart batch applications, in particular when batch application resources are shared.

Compute Grid has the following primary components:

- ▶ The Job Scheduler component determines when to dispatch a job, determines to where the job is dispatched, monitors what is happening with that job, and reports the status of the job to its caller.
- ▶ The Grid Execution Environment (GEE) is where a batch application actually runs. Jobs are dispatched into a GEE from the Job Scheduler. The job runs and on completion, its job log and return code is provided to the Job Scheduler.

The CICS support for the WebSphere XD Compute Grid enables a GEE to run inside a CICS resource called a JVM server. Compute Grid dispatches jobs to the GEE, and these jobs, running in CICS, have access to resources that are used by CICS and that CICS might have opened exclusively. Compute Grid jobs are Java-based, however CICS provides a Java API called JCICS, which enables a Java program running in CICS to access any other program defined to CICS, which includes programs written in COBOL or PL/I.

You can use the CICS Compute Grid support to write modern batch applications that include checkpoints and recovery. You can also reuse code written for online application programs and code written in procedural languages in these batch applications.



CICS application modernization

Application modernization helps you revitalize, modernize, and extend existing applications to realize greater return on technology investments and avoid costly rewrites. At the same time, it gives you the opportunity for the following accomplishments:

- ▶ Reduce IT costs and improve alignment with your market requirements
- ▶ Minimize project risk and achieve quick return on investment (ROI)
- ▶ Deliver new products and services faster with fewer resources

This chapter provides an overview of several techniques for rapid modernization in a way that use the investment and effort performed over years to build such applications.

This chapter includes the following topics:

- ▶ Introduction to application modernization
- ▶ Business rules modernization overview
- ▶ Event processing in CICS
- ▶ Reusing business logic from CICS applications
- ▶ Modernizing data layer using CICS VSAM Transparency
- ▶ Modernizing 3270 presentation layer

4.1 Introduction to application modernization

Many computer industry analysts agree that the IT investment by large enterprises over the last 30 years has been in building and maintaining legacy systems. Legacy systems not only include host-based COBOL business applications, but more generally systems that are not flexible enough to adapt to new business computing paradigms as they emerge

Current legacy systems that were built over the last 30 years used paradigms such as these examples:

- ▶ Batch programs
- ▶ Host terminals
- ▶ Client/server
- ▶ Distributed components and packaged applications

Typically, legacy systems, especially the older systems, have evolved over many years as business requirements have changed. These systems have become monolithic and unable to support evolutionary enhancements as new paradigms have emerged. Monolithic legacy systems also have layers of functionality and technologies (to manage data, process and presentation), which have hard-wired dependencies on each other. This situation is also true of some client/server systems, where a tight coupling exists between the presentation technology and the underlying business logic implementation.

Legacy applications also arise through a lack of foresight in skills and tools investment. Many existing systems are still being maintained using low-level APIs and tools such as Notepad. Replacing and maintaining such systems becomes difficult because the knowledge and skills are held by a few expert developers.

However, today's business is changing rapidly; to use the same set of paradigms for managing the existing infrastructure and applications will hinder all efforts for coping with market changes and the desire to expand. Therefore, finding new techniques becomes crucial for handling daily operations, maintaining existing applications, and developing new modernized applications that satisfy continuously changing business needs.

In this chapter, we cover various aspects of modernizing existing applications and infrastructure to better align with these demands. This modernizing includes business rules modernization, emitting and handling of events, exposing existing business logic to external consumers, enhancing the data access technology, and finally modernizing the presentation layer for existing legacy applications.

4.2 Business rules modernization overview

Business decisions are made every day in business transactions that are incorporated in web, online, and batch applications. However, business rules are often embedded redundantly throughout the application source, which can be subject to incremental application maintenance updates.

Over a period of time, because of this issue identifying and verifying the accuracy of these rules is difficult. With an effective process for identifying and validating the rules that govern business decisions, you can be more responsive to business needs and changes in the marketplace. By managing the rules centrally outside of the individual application source, you can be better positioned for lower application maintenance and IT integration costs.

Application modernization projects can benefit from the inclusion of a business rule mining process, which can help identify what decisions are made, by using which rules, and where in the source code those rules reside, and also by designing those coded rules to be executed and managed in a business rules management environment.

Business rule modernization, an integral part of application modernization, addresses the challenge of inflexible applications, helping corporations become more agile by quickly adapting the rules that govern their key business decisions to changes in the marketplace. Business rules management systems (BRMS) were designed to address these aspects.

BRMS are systems built for managing and storing business rules independently from the application source code. Using BRMS has the following advantages:

- ▶ Improves the agility (and therefore reduces the time, effort, and cost) that is required to develop and maintain applications by separating business decision logic from core application code
- ▶ Provides better management of rule-based decisions:
 - Rules are stored and managed in a centralized repository, with comprehensive capabilities for querying and reporting on rules and associated metadata.
 - Rules can be defined in a customizable business vocabulary using a variety of rule metaphors, including decision tables, decision trees, scorecards, and text-based rules.
 - Governance processes around business rules allow users to understand how and where rule changes affect business decisions.
 - Rules that are written to support existing applications can be shared with applications, providing a path for the application portfolio to be modernized over time.

IBM offers business rule modernization, by combining the application understanding and rule mining functionality of IBM Rational Asset Analyzer software with the capability of IBM WebSphere Operational Decision Management for z/OS, to forward design and manage these rules externally.

Using IBM WebSphere Operational Decision Management for z/OS, business rules are compiled and deployed into the existing COBOL applications as modules. The module interface is well defined, allowing it to be replaced at a faster rate than the existing applications, resulting in fewer architecture changes. Figure 4-1 on page 76 illustrates the deployment pattern for business rules that use IBM WebSphere Operational Decision Management for z/OS. Figure 4-1 on page 76 shows two Rule Execution Servers:

- ▶ A Rule Execution Server running in WebSphere Application Server on z/OS
- ▶ A zRule Execution Server that can run either in CICS TS in a JVM, or in a stand-alone JVM environment

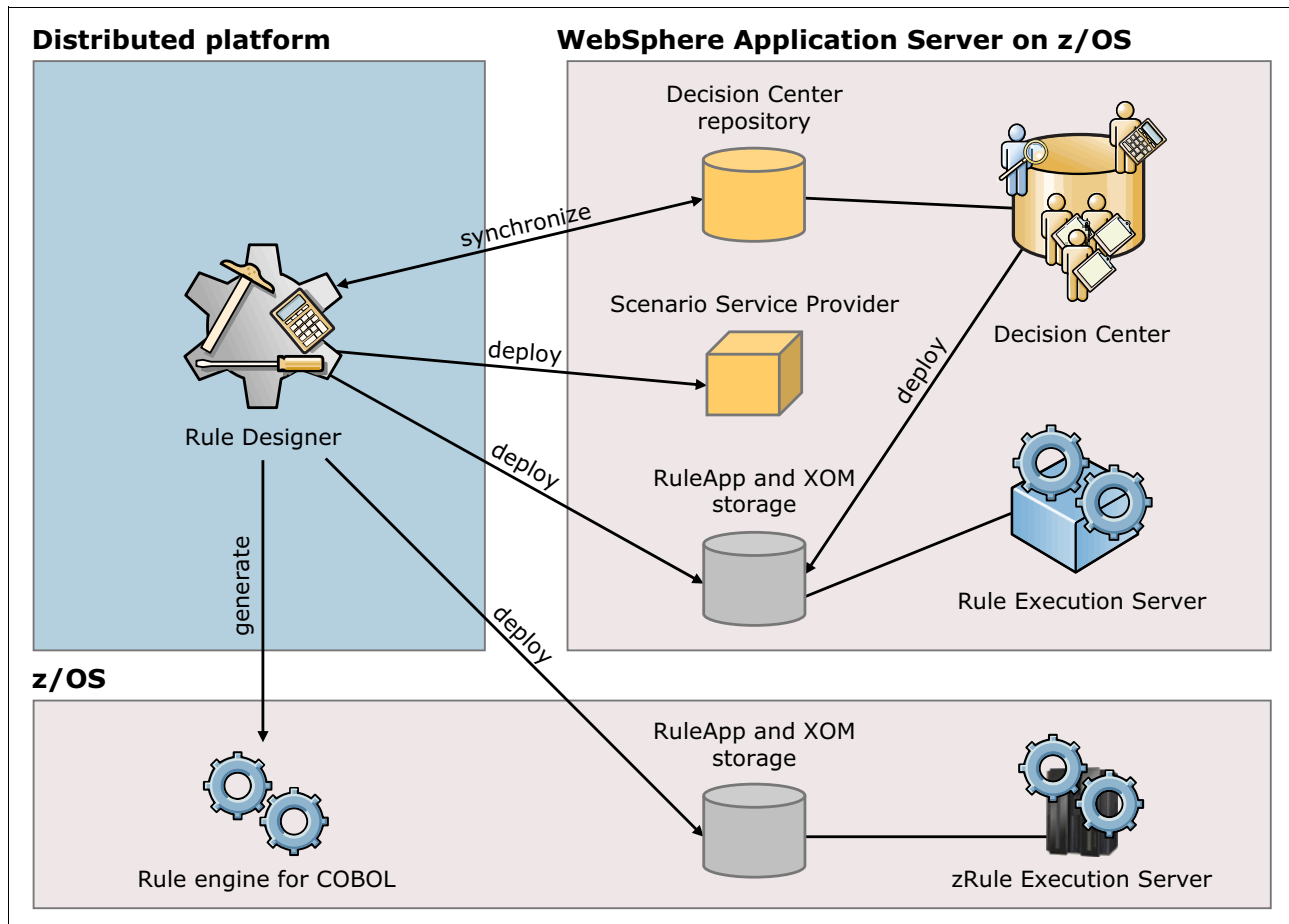


Figure 4-1 WebSphere Operational Decision Management for z/OS system components

4.2.1 Analyzing existing CICS applications

Extracting business rules from a complex application can be a daunting undertaking. With Rational Asset Analyzer, users can build the repository of business rules in stages by using the concept of a project. In this way, users define the scope of the effort to be performed in a given iteration.

Within the business rule mining (BRM) project, you can limit the scope of the assets that Rational Asset Analyzer searches for, such as a specific data element or concept, and build the business rules for that concept. This method allows you to incrementally build business rules one component at a time to meet the needs of your business community. Rational Asset Analyzer can also track the progress of BRM projects to give users a snapshot of how much is done (and how much remains) in extracting rules for each asset.

Using the business rule mining features of Rational Asset Analyzer jump-starts the process of identifying and automating business rules. The combination of Rational Asset Analyzer and WebSphere Operational Decision Management for z/OS results in an improved process for identifying and deploying business rules from start to finish.

So that Rational Asset Analyzer can derive a clear picture of existing business assets, it scans a wide variety of assets. Users can also specify vocabulary (specific nouns) to guide Rational Asset Analyzer in focusing on the candidate business rules. Vocabulary can also be imported from a rules repository.

After the business rule mining data is in the Rational Asset Analyzer repository, you can use it to develop the rules by using the business rule editors, which are incorporated into the Rational Asset Analyzer product. You can export the rules from Rational Asset Analyzer into WebSphere Operational Decision Management for z/OS, either as structured or unstructured rules. After the rules are in WebSphere Operational Decision Management for z/OS, they are positioned for ongoing maintenance throughout the rule lifecycle.

The synergy between Rational Asset Analyzer and WebSphere Operational Decision Management for z/OS is based on the common vocabulary between the products and the common rule mapping capabilities. The relationship between the two products is bidirectional and extends the value of both products.

4.2.2 Introduction to the business rules mining process

In this section, we describe how Rational Asset Analyzer can be used in combination with WebSphere Operational Decision Management for z/OS to perform business rule mining to allow identifying, capturing, and relating business rule assets to the existing set of COBOL applications in CICS environment. It suggests approaches and leading practices for using the rule mining tools available in Rational Asset Analyzer.

The rule mining process covers both preparation activities and rule mining activities, and also a general description of the necessary tasks to accomplish a rule mining project.

Preparation activities include the following steps:

1. Define rule mining project scope.
2. Gather supporting materials.
3. Take an inventory of application assets.
4. Import business vocabulary.

After preparation activities, the process continues with the rule mining activities:

1. Define a business process and related activities
2. Define business terms for decision inputs and outputs
3. Mine COBOL logic for candidate rules
4. Analyze candidate rules
5. Author structured business rules
6. Export business rules

It is advisable to follow an iterative approach to rule mining that allows you to refine the business vocabulary with its terms and properties to better fit the rules pattern that the process identifies. With each experience, your understanding of the rule patterns (that drive the business decisions encoded in the applications) is improved and therefore can guide the subsequent iterations.

Figure 4-2 on page 78 illustrates the rule mining process using Rational Asset Analyzer and WebSphere Operational Decision Management for z/OS.

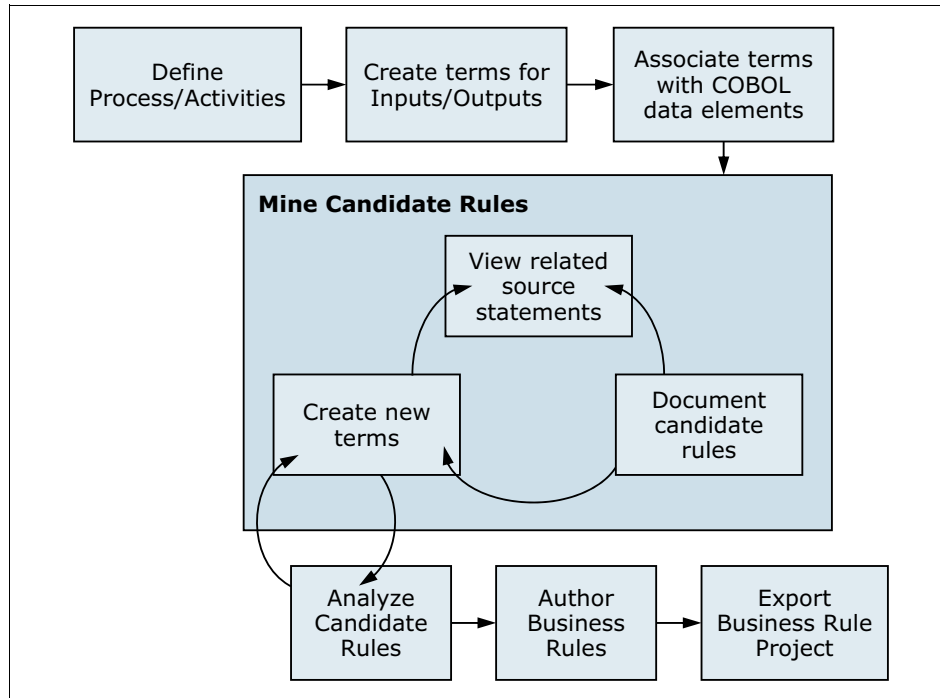


Figure 4-2 Rule mining activities

Rational Asset Analyzer uses the following asset types for rule mining:

- ▶ Business activity (*activity*): A named, structured process or task that produces a specific service or product for a particular customer or customers of a business. A business activity can be a collection of related business activities.
- ▶ Business category (*category*): An identifier assigned to business rules, business terms, and business term properties to let you filter which business terms and business term properties can be used when authoring a business rule.
- ▶ Business process model (*process model*): A named collection of business activities used to represent a core aspect of a business.
- ▶ Business rule (*rule*): A named statement, or set of statements, that defines or constrains an aspect of a business. Rational Asset Analyzer can capture business rules in either an unstructured form, a structured form, or both.
- ▶ Business rule mining (*BRM*) project: An optional, named collection of assets used to manage a given rule mining effort.
- ▶ Business term (*term*): A noun representing a concept that is used in the business.
- ▶ Business term property (*property*): A noun, of a specific type, representing an attribute or abstract quality that is associated with a business term. The relationships between business terms and business term properties are most often stated with the verb *has* (for example, car has driver) or the preposition *of* (for example, driver of car).

Rational Asset Analyzer uses the term *business element* to refer to a term, a property, or both. It uses the word *vocabulary* to represent a collection of related business elements. The vocabulary collection can be scoped by an application asset or a business process asset, or it can be unscoped, in which case, it contains all business elements. A business term dictionary represents an alphabetical rendering of a vocabulary.

4.2.3 Preparation activities for rules mining process

At this point, you and your team should have a good understanding of the objective of the effort, sufficient background knowledge about the application and access to the source, and tools needed to start the rules mining process. Thus, the first phase is to define the scope of the rule mining project.

Define rule mining project scope

Scope definition should be driven by business impact and level of effort or cost. Initial projects should define a realistically small scope to enable gaining rule mining experience. After gaining a greater level of experience and understanding, this knowledge and experience can be applied to projects with a larger scope.

The following steps are involved in defining the scope of a rule mining project:

1. Define the overall scope of the rule mining effort. Do this in business terms, such as rules related to loan eligibility or rules related to pricing of wireless family plans. It is not useful or productive to scope a rule mining project to a non-business-centric objective.
2. Identify the process that the new rules will affect. The process does not have to be defined or automated using a business process modeling or automation tool, but rather can be identified and simply sketched.
3. Develop high-level business process model. The process model needs only to be a high-level sketch defining the events, activities, and gateways in a process. The purpose of the process model is to identify decision points in the process.
4. Identify the decision points in the process. A decision point represents an activity in a business process where decisions are made such as the acceptance or rejection of a loan application to flag it for future processing downstream in the business process or even involve computing some values as part of a business transaction or process; for example, the calculation of discounts based on customer categories.
5. Prioritize the decisions.
6. Identify the inputs and outputs for a decision. Identify them in business terms and the applications associated with the inputs or outputs.

After establishing a project scope, rules can be identified and captured by the rule mining process which starts by gathering the supporting materials.

Gather supporting materials

After properly defining the rule mining project scope, gather the supporting materials. This activity helps to provide you with a better understanding of the existing application.

Supporting materials are any reference information that gives more insight into the application specifics such as documents and guides, glossaries or dictionaries, and data files or database tables that are referenced by the code to perform look-ups.

Inventory of application assets

The goal of this task is to produce an asset scan with no errors into the Rational Asset Analyzer inventory database. This task is not trivial, and often an inventory scan produces errors because of missing artifacts or “surprises” in the code, such as ADABAS calls. These issues might not have been known or communicated prior to the actual scan.

Establish the business vocabulary

An essential and often challenging part of a rule mining project is establishing the vocabulary that will form the backbone of the rule design. This vocabulary is a business-oriented abstraction of the vocabulary that the COBOL programmers and database administrator defined in COBOL copybooks, and the COBOL structures in the application. A key activity in rule mining is the mapping of the business vocabulary terms and their properties to the programming vocabulary, variables and data element names.

If a business vocabulary that is based on the business rules that are managed by WebSphere Operational Decision Management for z/OS is already available, it can be imported by using Rational Asset Analyzer. In the absence of an established business vocabulary, set up a “brainstorming” session between the application subject matter expert and the business rule designer to decide on business terms and their properties corresponding to the rules that are identified for the rule mining project. Be sure to discuss how to define these decisions in Rational Asset Analyzer, rather than importing the definitions.

An alternative approach is to generate business terms from the COBOL copybook. This way is useful if you know that all the data elements in a given level are referenced in statements that contain business rule logic. Although this approach is certainly convenient, the generated terms are not likely to be true business terms so you might need to adjust some of them as the mining process continues.

4.2.4 Rule mining activities

Consider the rule mining activities described here.

Define a business process and related activities

Rational Asset Analyzer supports defining a business process and related activities and associating other assets with them. Defining a process and its associated activities serves to document the scope of a rule mining effort. You can define processes and related activities and associate other assets to these assets. Although this is useful for the sake of documenting the business processes, impact analysis is not currently supported for rule mining asset types.

Define business terms for decision inputs and outputs

In this activity, you identify the inputs to and outputs from a decision, find the associated COBOL data elements by using Rational Asset Analyzer, and associate them to a business term. Rational Asset Analyzer discovers synonyms for the business term based on MOVE statements, and thus each COBOL data element can be associated with a business term.

Optionally, you can also assign the business term to one or more categories. Each term that is identified as an input or output to a decision activity will eventually need to be defined. It is important to define the business term or property name in business terminology so that the structured business rules are readable and understandable.

Mine COBOL logic for candidate rules

The process of mining COBOL logic for candidate rules, as illustrated in Figure 4-3, is an iterative process consisting of the following steps:

- View related source statements.

Starting from a business term, view the related source to see the lines of code that reference the business term.

- Document candidate rules.

Navigate the source view to look at source statements and associate candidate rules with statements that contain business logic that you need to track. Document the business logic for the candidate rule using defined business terms.

- Create new terms.

As you navigate the source and associate source statements with rules, you might find additional data elements of interest that need to be traced. Associate these data elements with business terms and continue the process of exploring the source for that term to find additional candidate business rules and terms.

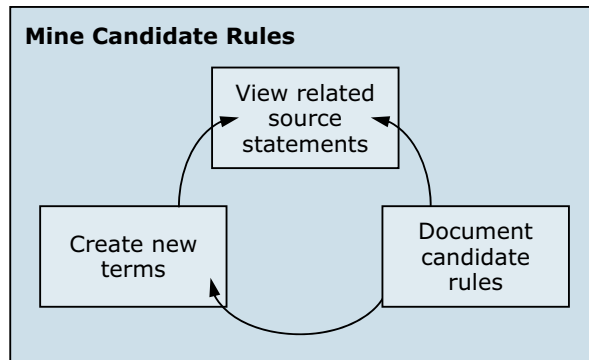


Figure 4-3 Mine candidate rules

Notes:

- A leading practice is to add new business terms to Rational Asset Analyzer as soon as you find them so that it can be used to document the candidate rules in natural language text. For this reason, the task of documenting candidate rules is likely to be interrupted by the task of creating new terms. The practice of using the business terms in the natural language representation of the rule makes it easier to analyze the rules and to write structured rules.
- Also, an advisable approach is to document the candidate rules by using consistent business terms that have been defined as terms or properties in Rational Asset Analyzer.

Analyze candidate rules

After you trace through all business terms, you have a final set of documented candidate rules. You can view the set of candidate rules on the “Business rules summary” page in Rational Asset Analyzer. If you use a naming convention for your candidate rules, you can filter on the name to see the candidate rules you defined.

The goal of rule analysis is to rework the candidate rule and business object model assets into a form that is closer to implementation, while keeping the meaning of the rules true to their business intent.

Author structured business rules

To support the authoring of structured rules, business terms are translated into business object model (BOM) classes and properties to members. Default verbalization is used to build the vocabulary using the guided editor in WebSphere Operational Decision Management for z/OS.

Export business rules

Rational Asset Analyzer provides the dialogs to create export packages that are used by WebSphere Operational Decision Management for z/OS. The export project contains a copybook, business object model, and business action language rules. The documentation field of exported rules, which holds the free-form business logic, includes any source statements copied during the capture process.

The rule project (and associated rules) is reworked in Rule Designer (for example to define packages, ruleflow, ruleset parameters, complete place-holders in rules and so forth). After the rules are exported and the project is modified in Rule Designer, the governance and management of the rules is taken over by WebSphere Operational Decision Management for z/OS. Traceability to the COBOL source is maintained in the documentation field inside Rational Asset Analyzer.

For more information about Rational Asset Analyzer and business rule mining, see the business rule mining process topic in the Rational Asset Analyzer information center:

http://publib.boulder.ibm.com/infocenter/rassan/v6r0/index.jsp?topic=%2Fcom.ibm.raa.analyze.doc%2Fcommon%2Fbrm_process.html

4.3 Event processing in CICS

An event is any electronic signal, or message, indicating a change in the state of an enterprise. For example, an event message can indicate the addition of a new customer, the sale of a product or receipt of a shipment.

Event processing is the capture, enrichment, formatting, and emission of events, the subsequent routing and any further processing of emitted events, and the consumption of the processed events. Event processing has evolved from addressing simple events and complex events to focusing on business events.

The event processing system can perform a variety of actions on events:

- ▶ Simple enriching of the event (for example, adding a time stamp to the event data)
- ▶ Adding information about the source of the event
- ▶ Processing multiple simple events, from multiple event sources, against event patterns to produce a new derived event. Processing of this kind is often referred to as *complex event processing*.

CICS event processing provides filtering, capturing, enrichment, formatting, and routing of business events, enabling CICS to act as a source of both system and also business events. The events resulting from processing are made available for consumption by other event consumers which reacts to the events.

4.3.1 Why is events processing important to today's business?

Business governance and compliance are increasingly important in many industries. These terms cover a crucial range of issues including financial transparency, information privacy and process control. The Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), or Basel II, and their associated information requirements are only some of the standards that are required for business compliance and governance.

Together, governance and compliance allow business and IT people to manage enterprise risks by reacting swiftly and effectively to capitalize on opportunities and to mitigate exceptions, providing flexibility to meet continuous changes in today's business requirements.

Events allow businesses to be more responsive and flexible, and to address governance and compliance concerns. Today's business solutions have to be highly responsive to continuous changes in business requirements and at the same time to comply with business controls, industry standards, and government legislation.

Historically, the event-processing capability was created and controlled at the IT level and required sophisticated IT skills to enable it. In contrast, business event processing allows the event patterns to be authored and managed directly by the business community by means of graphical, codeless authoring tools specifically designed to empower the business user. Additionally, the deployment of complex event processing was focused primarily on niche application domains (stock trades are a good example of this). In contrast, the processing of business events targets applications across industries and application domains, regardless of the type, time, and order of events.

Business event processing supports the need for increased responsiveness to change across industries and application domains. The following list has several examples of applications that benefit from the capabilities supported by this approach:

- ▶ Compliance
- ▶ Exception detection (alerts)
- ▶ Fraud detection
- ▶ Identity theft
- ▶ Application review and approval
- ▶ Case management
- ▶ Customer service
- ▶ Marketing and sales orchestration
- ▶ Supply-chain optimization

By using the event processing capability in CICS, you can extend existing applications by letting the emitted events invoke newly developed applications. This way can be done to invoke both new CICS programs or other Web applications. This approach can be considered as a way of modernizing existing CICS programs.

4.3.2 Key concepts about CICS event processing

Consider the following CICS event processing concepts:

- ▶ Capture specification

A typical business event is defined from one or more capture specifications and typically consists of a capture point, filter, and information source:

- Capture point

The capture point is a place in your CICS application where a particular event can be captured; for example, an EXEC CICS READ FILE command of a file where the record key has a particular value. The places in a CICS application that can be enabled as capture points consist of a number of the EXEC CICS API commands, and program initiation. There are many EXEC CICS commands that can be specified as a capture point, such as CICS commands that are related to files and queues access, link programs, send and receive maps, invoke services, and so on.

- Filter

The filter is a set of predicates connected by AND, used to determine whether an event is captured. If all predicates evaluate to TRUE, the event is captured. Predicates that evaluate to FALSE filter out events. A predicate is an expression used as part of a filter, consisting of a data item, an operator, and a value. A predicate is used with data values on the API call or context data, to restrict the occasions when an event is emitted to the occurrences of interest.

- Information source

If you want to supply a payload for an event, you can add business information items to the event specification in the order you want them emitted, and then define a source for each of these business items in the capture specifications for that event. The information source can be application context, data, or command options.

- ▶ CICS event specification

An event specification defines a business event to CICS. An event specification can be created by using the CICS event binding editor (one of the editors in the CICS Explorer) by business analysts or developers in response to a business requirement. An event specification describes an event and its processing in natural language. The event specification contains one or more capture specifications.

- ▶ Event binding

The event binding is the unit for installing, enabling, and disabling CICS events. An event binding consists of event specifications, capture specifications, and adapter and dispatcher information, which is needed to specify *normal* or *high* priority for control dispatching of events associated with the event binding. Event bindings are deployed to CICS in a bundle that can contain other resources. All of the resources in a bundle can be enabled and disabled together.

- ▶ Event processing adapter

The event processing (EP) adapter is responsible for emitting CICS events. EP adapters that are provided by CICS TS support the following transports and formats:

- WebSphere MQ queue EP adapter

This EP adapter emits events to a WebSphere MQ queue either format:

- In an XML format for consumption by products that use the common event infrastructure, such as WebSphere Business Monitor and WebSphere Operational Decision Management
- In a non-XML character format

- CICS transaction start ‘transport’

This EP adapter emits events in the CICS container-based event (CCE) format to a named CICS transaction that consumes the event and also the CICS system that will run the transaction.

- Temporary storage (TS) queue adapter

This EP adapter emits events in the CICS flattened event (CFE) format to a named CICS temporary storage queue and can be used to perform the following tasks:

- Validate that the correct events are being captured with the correct data.
- Emit events to any consumer that reads from a TS queue.

This EP adapter is a good choice when implementing events. You can use it as a temporary adapter because you can browse TS queues with the CICS CEBR transaction to check the payload. You can refresh the view to check for new events in the queue. However, TS queues are limited to 32,767 entries or less. You may delete only the entire queue rather than individual entries within, so you will need a strategy for clearing out events you processed.

- Custom (user written) EP adapter

This adapter emits events in any format that you require. A custom EP adapter is a CICS program that you write to provide a combination of formatting and routing of an event that is not supported by the CICS-provided EP adapters. It must not carry out any other processing, such as consumption of the event.

Specify the transaction ID for your user-written CICS application that formats, routes, and emits the event. You must specify a transaction ID. Write the data to be passed to the Custom EP adapter. Your custom EP adapter will receive this data, which may be used to configure the custom EP adapter.

4.3.3 How CICS events work

The CICS run time detects instances of events that are enabled, and captures the events and payload without the need to make application code changes. CICS event processing is a core component of the CICS run time, and provides all the qualities of service you expect of CICS. When CICS captures events, it carries out specified filtering, enriches the event with information about the application context in which it occurred, formats the event, and routes it to the appropriate event consumer.

It is possible to emit events in formats suitable for consumption by many event consumers including: WebSphere Operational Decision Management, WebSphere Business Monitor, and others. CICS Event Processing support is extensible, with options for customization.

Using CICS Explorer, you can do the following tasks in regard to handling events:

- ▶ Define events to be emitted and their payload data.
- ▶ Specify to the CICS run time how to detect when the events occur.
- ▶ Indicate how events are to be formatted and routed.
- ▶ Deploy the event definitions to zFS for installation into CICS.

Using event specifications defined to CICS, events can be captured from existing CICS application programs without altering the original code. Figure 4-4 on page 86 shows an overview on how CICS event processing works.

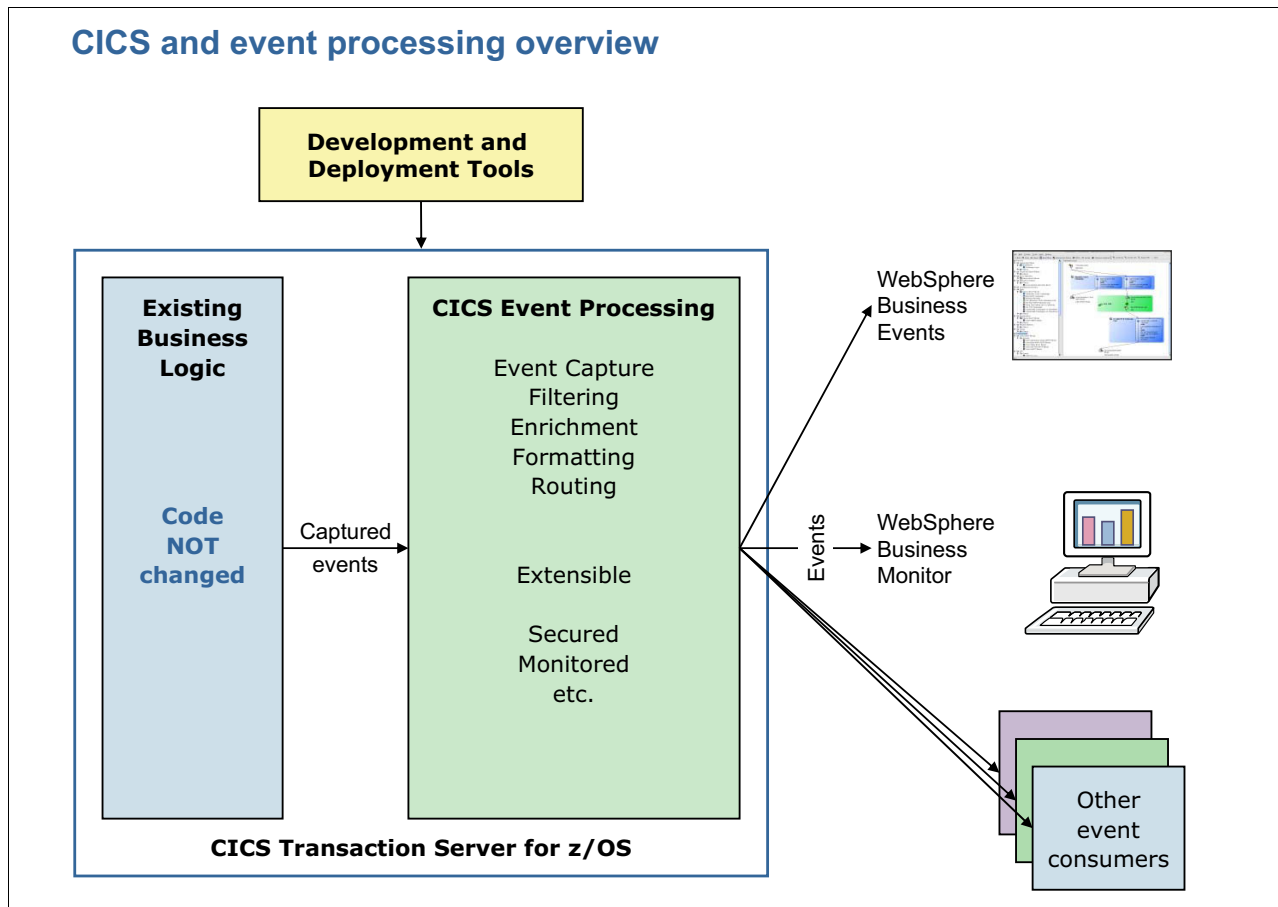


Figure 4-4 CICS event processing overview

An application analyst takes a defined business event and converts it to one or more capture specifications using the CICS event binding editor in CICS Explorer. During execution of the business application, many potential capture points occur. CICS checks each of the enabled capture specifications that match the capture point. Each matching capture specification contains optional filters to be compared against the application context, some command options on the API call, and data passed on the API call.

If the filters match, CICS collects the payload information from information sources described in the capture specification, enriches it with context data, then queues the event for dispatch so that the application can continue to execute quickly.

A separate process in CICS routes the event and any payload data through the event processing adapter described in the event binding. Events with a high dispatch priority are routed first. Events marked as transactional are only emitted after the transaction reaches a sync point and commits. The event processing (EP) adapter then emits the event to a consumer such as WebSphere Operational Decision Management, WebSphere Business Monitor or other event consumers.

4.3.4 Events consumers

Multiple products exist that can consume events that are emitted by CICS. We describe two of these products: WebSphere Operational Decision Management for z/OS and WebSphere Business Monitor.

IBM WebSphere Operational Decision Management for z/OS

WebSphere Operational Decision Management is an IBM software product that supports business event processing by meeting the high-volume demands and processing required across industries and application domains. WebSphere Operational Decision Management provides graphical, codeless user interfaces that simplify implementation and empower business users to develop and maintain event processing logic. WebSphere Operational Decision Management consists of the following basic constructs:

- ▶ Connectivity to business events
- ▶ Event processing engine for evaluating and detecting event patterns
- ▶ Initiation of business responses (actions)

Based on user definitions, the WebSphere Operational Decision Management processing engine detects and sifts through the mass of events occurring across the information infrastructure, identifying only those events and patterns of interest. Upon detecting a defined event or pattern (actionable situation), the engine initiates one or more business responses (actions).

Although discrete CICS business events are interesting in isolation, the real value of using WebSphere Operational Decision Management is in combining those events with other event sources within a business solution context; this combination becomes extremely informative. The combination of event capture and emission from key CICS business applications with the business event patterns and representation capabilities of WebSphere Operational Decision Management allows business managers to gain insight into a wealth of business behaviors and to accurately and immediately control critical business processes throughout the enterprise.

Proper actions and responses can range from sending alerts, to initiating the execution of follow-on processes. These actions are communicated directly to related systems (or over the communications backbone), indicating that an actionable event or pattern has been detected.

IBM WebSphere Business Monitor

WebSphere Business Monitor is comprehensive business activity monitoring (BAM) software that provides business users and managers with a real-time and end-to-end view of business processes, events and operations. WebSphere Business Monitor aggregates and correlates events into metrics that provide objective measurements on the status of business processes.

WebSphere Business Monitor provides customizable business dashboards that calculate and display key performance indicators (KPIs) and metrics that are derived from business processes, business activity data, and business events from a wide range of information sources, including CICS TS with events processing. Business users can view these KPIs, metrics, events, and alerts through various means, including lightweight Web interfaces, smartphones, corporate portals, and on desktops. These options give business users immediate actionable information and insight into their business operations to mitigate risk and take advantage of opportunities.

WebSphere Business Monitor offers business users real-time, business-relevant views into transactions. For example, WebSphere Business Monitor can provide information about how many payments originated from a particular customer, whether any payments delayed and might potentially incur a penalty, and similar data. Traditionally, IT provided this information, largely through manual mining of data logs and systems, leading to overloading of IT resources with routine data collection, and delays in getting time-sensitive information to business users. However, WebSphere Business Monitor enables business users to get this information in real time without any IT involvement, making business users more responsive to changing conditions and reducing the load on IT.

IBM has enabled a bidirectional flow of events between WebSphere Business Monitor and WebSphere Operational Decision Management. You can use one single dashboard to view the performance of the processes through KPIs, and view any alerts to business situations generated by WebSphere Operational Decision Management to get more complete real-time insight into what is happening within the organization and the ecosystem. Similarly, the bidirectional event flow enables customers to feed any alerts on processes or business generated by WebSphere Business Monitor to WebSphere Operational Decision Management for detecting patterns within those alerts which might otherwise go undetected and initiating follow-on processing. As a result, business users can receive better information to help them respond faster to their business needs.

4.3.5 Designing event driven solutions for CICS

To source a business event, we must have a point at which business activity and relevant information is known for constructing the event, such as during application execution. It is at this point that you have the greatest degree of business and contextual information about the business event that is occurring. While processing a new customer purchase, the application has all of the business context and associated information necessary to create and emit an event, indicating a new product order that is being added to the system. An example of this information is the exact time that the order was created.

In addition to business context information, other useful contextual information available might be relevant, such as the user ID of the operator who initiated the order. Detail about the transaction state is often available only at the time that the event is created and so might be useful to include in the event information.

Often there are many points in the application business logic that can be the source of a particular event. For example, adding a new order is an event that can be sourced from various points, as shown in Figure 4-5:

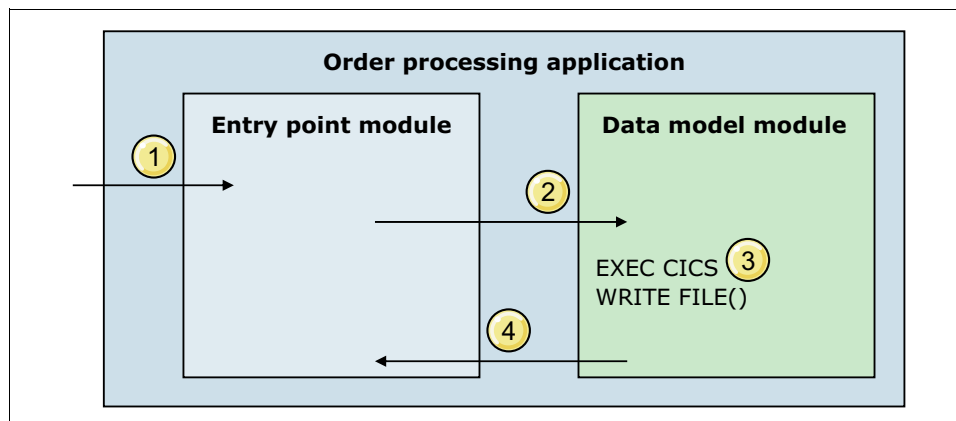


Figure 4-5 Potential points in CICS for event sources

The following event capture points are shown in Figure 4-5:

1. The initial entry point to the business application
“A new order request has been received”
2. The entry point to the module that hardens the new customer’s information
“A new order request is accepted and being processed”

3. The point in the new customer module where the data is actually hardened

“A new order record has been created”

4. The exit point from the module that hardens the new customer information

“A new order request has been processed”

Any of these points can create a valid business event for processing by the event-processing engine. Choosing which to use must largely be based on the need of the business for detection and response to actionable situations. This requirement helps to define the contextual information that is required to flow with the event. If the business requirement is simply to note the number of orders being placed through a particular channel, any of these event capture points can be used. However, if the requirement is to keep an audit of all the orders being written to the order file, event capture point 3 is the most logical, because that is where we can also capture additional data about the file being written.

After determining an appropriate point to specify a particular occurrence of the event, ensure that the required business information is also available at that point. In the preceding example, we might need to include the following additional contextual information:

- ▶ The customer number of the person placing the order
- ▶ The order reference number
- ▶ The value of the order being placed
- ▶ The user ID of the operator taking the order

The first three of these items are directly part of the application logic and are sourced from the application data within the application program. The final item, however, is not known by the application itself but by the system that is running the application. This item must be obtained from contextual information available to the application from another source, such as the CICS EXEC Interface Block (EIB).

After the event information is provisioned and a source capture point is created, an infrastructure must be put in place to create and process this event. Figure 4-6 shows a simplified event-processing infrastructure.

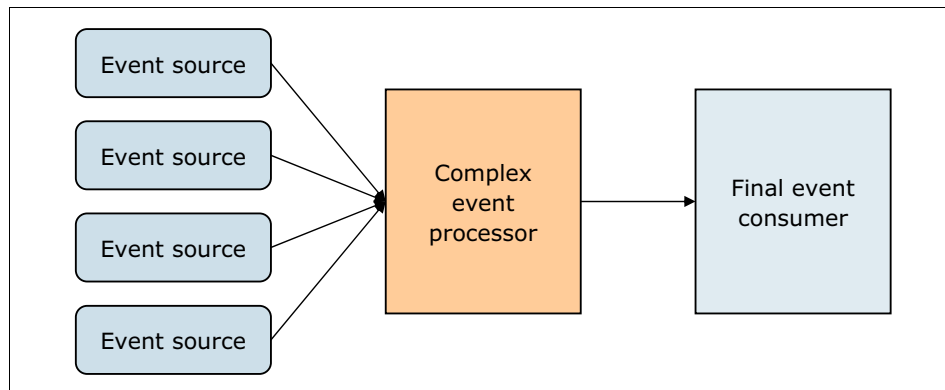


Figure 4-6 Simplified event processing infrastructure

Up to now, we have been discussing event sources, the initial locations to capture the data required to satisfy a business event. The other systems in this architecture must also contain an event consumer and optionally a complex event-processing engine. The event-processing engine is optional depending on the complexity of the original requirement for the event and how easily this can be mapped onto the existing business application.

Our example is a simple event-processing scenario. The requirement was to provide a business event when an order is placed in the system. Unless a level of message mediation is

required, this event can be consumed directly by the final event consumer (any other application). This event pattern is referred to as *simple event processing*. The pattern might need to be extended to help detect and respond to event patterns between like or related events, missing events, and aggregate events. As described previously, we would then need to use complex event processing.

For instance, in our simple example, we might need to address a requirement to recognize and provide incentives to customers whose cumulative order value for orders placed within a specified time period is greater than the promotion level. In this case, the complex event processor is the target for the simple event sources, and it applies specific business criteria to identify when the complex event pattern occurred. An additional event, a business action, is then emitted from the event-processing engine. This event can be directed to the ultimate event consumers (for example, the billing system) to inform them that the particular business situation occurred, or back to the business event-processing engine to be part of a larger interaction for detecting event processing patterns.

4.3.6 Business scenarios for CICS events

Using CICS events and events consumers such as WebSphere Operational Decision Management, you can build new systems or extend existing ones to be event-driven, which makes them flexible to the continuous changes in today's business requirements. The following examples for business scenarios can be realized by using this combination:

- ▶ Events from CICS can be used to ensure that regulations, which require banks and insurance companies to detect potentially fraudulent situations, are satisfied. Suspicious behavior relating to bank and credit card usage can be detected with the pattern-detection capabilities of WebSphere Operational Decision Management. Examples of these patterns include a new card ordered within a week of an address change request, several online purchases where none had been made before, or two or more cash withdrawals in quick succession when withdrawals are rare on this card, or normally for smaller amounts. A number of regulations require such situations to be detected in real time.
- ▶ Event patterns detected during order processing (order received, order dispatched, order cancelled and so on) can trigger action messages from WebSphere Operational Decision Management to WebSphere Business Monitor. These messages provide insight into order processing, such as studying KPIs of numbers of orders received per week, time to process and dispatch orders, and others.
- ▶ A student registration and course management system running in CICS can be extended to send email messages to students when classes are cancelled or relocated at short notice. This extension can be done by emitting events to WebSphere Operational Decision Management and using the embedded email action connector in WebSphere Operational Decision Management to notify students of the changes.

4.4 Reusing business logic from CICS applications

We describe how to reuse and extend CICS applications with the help of these techniques:

- ▶ Migration from COMMAREAs to channels and containers
- ▶ CICS integration options
- ▶ Java Connector Architecture through CICS Transaction Gateway
- ▶ JCA through WebSphere z/OS Optimized Local Adapters (WOLA)
- ▶ CICS web support
- ▶ Asynchronous messaging
- ▶ CICS sockets
- ▶ Integration options comparison

4.4.1 Migration from COMMAREAs to channels and containers

Historically, the way CICS programs exchanged data was typically by using a fixed block of addressable storage termed the communications area (COMMAREA). This COMMAREA is a multipurpose and fixed message interface. As such, it imposes redefinitions of the same physical area, the most trivial being the request and response layouts. To solve this problem, some implementations define two separate request and response areas in the same unique COMMAREA layout. Moreover, the CICS COMMAREA message has another limitation which is the maximum length cannot exceed the size of a signed 32-bit integer (32 KB). New programming styles often require the exchange of large amounts of data, rather than efficiently managed exchange of parameter data between CICS programs. As the interface evolves, this exchange introduces a level of complexity that might not be acceptable today.

To overcome the limitations in COMMAREA, CICS introduced a channel interface, which adds flexibility to the length and organization of the data passed on a LINK request.

Figure 4-7 shows an example channel and an analogy with an XML document. Programs can pass any number of containers between each other. Containers are blocks of data that are designed for exchanging messages between programs. Containers are grouped together in an interface referred to as *channels*.

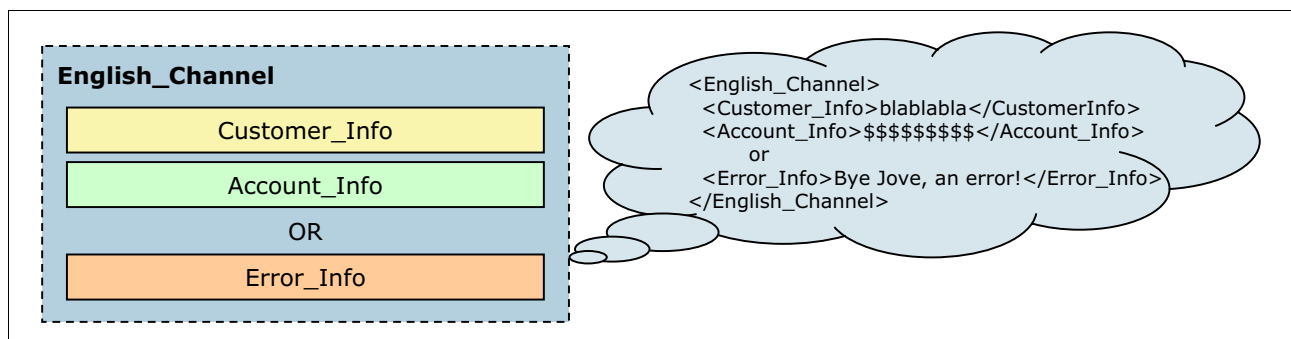


Figure 4-7 CICS channel message representation

The channel interface offers a rich EXEC CICS CONTAINER API that can be used to implement LCRUD (list, create, read, update, delete) services:

- ▶ List or discover the current set of containers in a channel.
- ▶ Create or PUT a container in a channel.
- ▶ Read or GET a container from a channel.
- ▶ Update or PUT replace a container in a channel.
- ▶ Delete a container from a channel.

The CICS infrastructure provides a set of services that simplifies the programming effort when dealing with channels:

- ▶ Channel lifecycle and memory allocation/de-allocation
- ▶ Monitoring and statistics data on channel usage
- ▶ Codepage conversion for character containers
- ▶ Message size optimization, for example, where read-only containers are not returned to the client application

The channel message enables a more structured interface so that different types of business data can be encapsulated within a container. It is common to have a separate set of containers for the request and response messages.

Using channels and containers offers the following benefits:

- ▶ More readable (understandable) code and interface
 - For easier maintenance
 - For better productivity
- ▶ An interoperable interface that is CICS implementation neutral
- ▶ Reuse of the container structure definitions within multiple channel interfaces

From a business perspective the channel message offers the best interface. From a runtime perspective, however, it has a higher cost of infrastructure than a COMMAREA interface. However taken into consideration the evolving of technologies nowadays and the growing complexity of program interfaces, it becomes a necessity to modernize existing program interface to use channel and containers rather than using a COMMAREA.

See the topic about migrating from COMMAREAs to channels in *CICS Application Programming Guide*, SC34-7158, for more information about migrating applications to use channels and containers:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.applicationprogramming.doc/dfhp3_pdf.pdf

4.4.2 CICS integration options

CICS provides a rich set of integration options for reusing and extending existing applications, including the following options:

- ▶ CICS web services
- ▶ CICS Transaction Gateway
- ▶ WebSphere Optimized Local Adapter (WOLA)
- ▶ CICS web support
- ▶ WebSphere MQ
- ▶ CICS sockets

Further details are in the following resources:

- ▶ *CICS and SOA: Architecture and Integration Choices*, SG24-5466
- ▶ *Options for Integrating CICS Applications in an SOA* white paper:

ftp://public.dhe.ibm.com/software/http/cics/pdf/Options_for_Integrating_CICS_Applications_in_and_SOA.pdf

CICS web services

Web services are the basic block of most applications of today. Web services take advantage of existing open-standard web technologies, such as XML, Uniform Resource Locator (URL), and Hypertext Transfer Protocol (HTTP), which are, themselves, a set of standards that facilitate open system-to-system communication.

Application programs running in CICS can participate in a heterogeneous web services environment as service requesters, service providers, or both. Figure 4-8 shows an outline of the web services support in CICS.

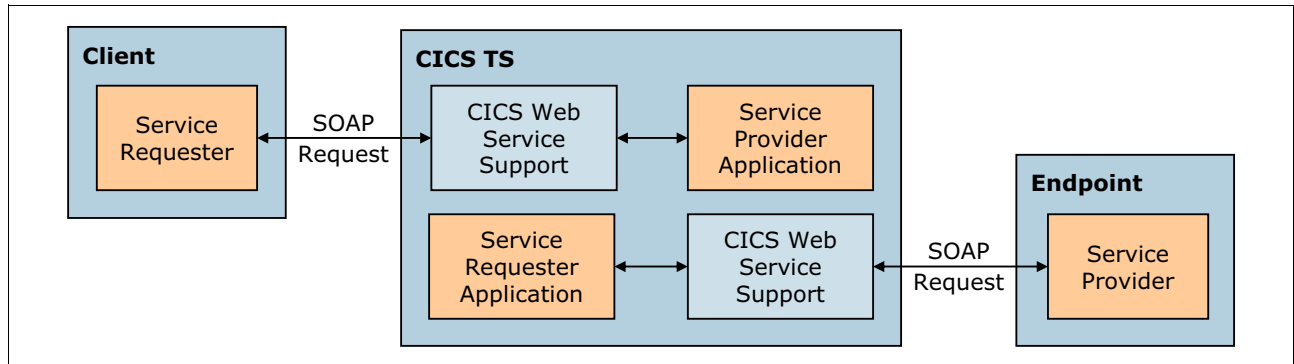


Figure 4-8 Web services Architecture inside CICS

CICS support for web services conforms to open standards, including these standards:

- ▶ SOAP 1.1 and 1.2
- ▶ HTTP 1.1
- ▶ WSDL 1.1 and 2.0
- ▶ Message Transmission Optimization Mechanism (MTOM)
- ▶ XML-binary Optimized Packaging (XOP)

CICS supports the most common type of communication between service requester and service provider: SOAP over HTTP, and also SOAP messages to WebSphere MQ.

Figure 4-9 shows an overview of CICS web services support. It makes a clear distinction between the tools and the runtime components of CICS web service support. The tooling includes the IDE, which is the Rational Developer for System z and the CICS web service assistant batch utilities. The run time includes pipeline, message handlers, and CICS resource definitions.

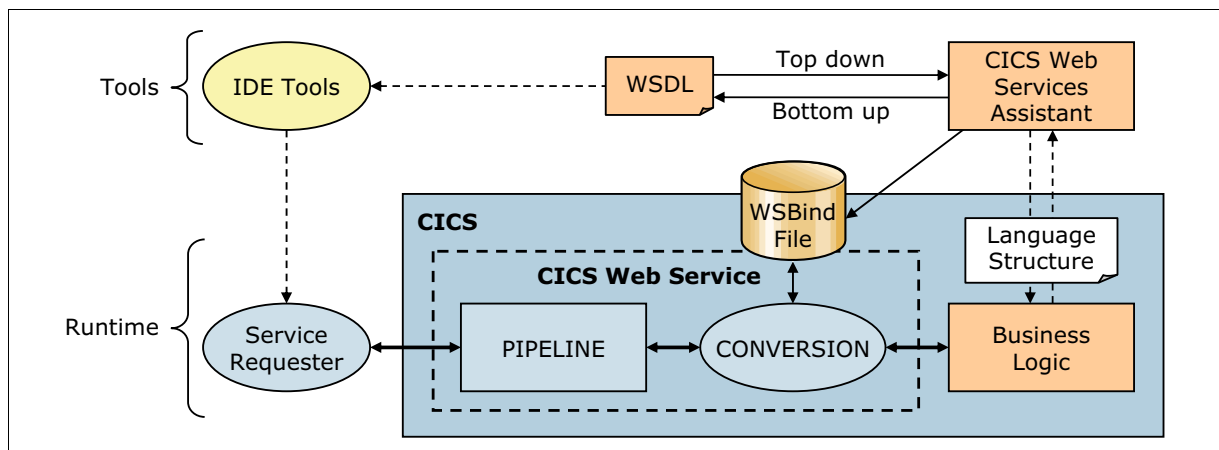


Figure 4-9 Overview of CICS web services support

Web services tooling

CICS web services provides the following tooling support to enable rapid service enablement of existing assets:

- ▶ CICS web service assistant

The CICS web services assistant is a set of batch utilities that can help you transform existing CICS applications into web services and enable CICS applications to use web services that are provided by external providers.

The assistant can create a WSDL document from a simple language structure such as a COBOL copybook, or a language structure (copybook) from an existing WSDL document, and supports COBOL, C/C++, and PL/I. It also generates information that is used to enable automatic runtime conversion of the SOAP messages to containers and COMMAREAs, and vice versa.

The assistant generates a WSBIND file. The WSBIND file is used by CICS to perform message transformation (SOAP to application data and vice versa).

- ▶ Rational Developer for System z

Rational Developer for System z facilitates the development of both Java and z/OS-based applications. The XML services for the enterprise (XSE) capability of Rational Developer for System z provides tools for you to adapt COBOL-based applications so that they can consume and produce XML messages. Also with Rational Developer for System z, you can develop Java applications that can act as service consumers or providers for CICS.

Both tools can be used to build web services in various development styles. You can use the tools to build a web service in top-down style, bottom-up style, or meet in the middle:

- ▶ Top-down style

This approach is usually the starting point when we have an existing WSDL document for a web service and we want to either implement or invoke the web service within CICS. We can use either CICS web service assistant or Rational Developer for System z to do this.

- ▶ Bottom-up style

This approach is usually the starting point when we have an existing CICS application that is already in production and has either a COMMAREA or channel-based interface. We now want to expose this application to remote client applications using CICS web services support.

- ▶ Meet in the middle

This hybrid technique often involves the use of a wrapper program that maps between the data format generated by the CICS web service assistant (or Rational Developer for System z) and the desired data format that is used by the existing application. This approach is used in complicated scenarios where COMMAREA fields or language are not supported by the CICS web service assistant.

Support for Message Transmission Optimization Mechanism

In standard SOAP messages, binary objects are base64-encoded and included in the message body, which increases their size by 33%. For large binary objects, the larger payload can significantly affect transmission time. CICS SOAP pipelines can support the Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specifications. These specifications define a mechanism for sending and receiving binary data using SOAP, without incurring the overhead of base64 encoding. CICS supports and controls the handling of MTOM messages in both web service provider and requester pipelines using an MTOM handler program and XOP processing.

See the topic about CICS web services in *CICS and SOA: Architecture and Integration Choices*, SG24-5466 for more information.

CICS support for web services standards:

- ▶ CICS ensures maximum interoperability with other web services implementations by conforming with the Web Services Interoperability Organization (WS-I) Basic Profile 1.1 and WS-I Simple SOAP Binding Profile 1.0.
- ▶ CICS supports the WS-Atomic Transaction and WS-Coordination specifications. By default, web service requests are stateless. This specification and the previous specifications add a layer for transactionality.
- ▶ CICS supports the WS-Security specification. WS-Security is a large specification that deals with the many aspects of security related to web services. It covers everything from passing a user ID, single-field encryption, message encryption, signing, and so on.
- ▶ CICS conditionally complies with WS-Trust, and support is subject to restrictions. WS-Trust (one of the six sub-specifications of WS-Security) allows you to support more forms of identification. CICS, for example, supports only username tokens and X509 certificates.
- ▶ For more information about WS-Security and WS-Trust, see Chapter 6 in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

4.4.3 Java Connector Architecture through CICS Transaction Gateway

The Java Enterprise Edition (Java EE) Connector Architecture (JCA) defines a standard set of APIs and interfaces for connecting from the Java EE platform to heterogeneous Enterprise Information Systems (EIS). The JCA standards enable vendors such as IBM to provide a *JCA resource adapter* to connect and call services in an EIS such as CICS.

Figure 4-10 shows the JCA being used in a managed environment. That is, the application is running in a Java EE environment such as WebSphere Application Server. In this case, management of connections, transactions, and security is managed by the application server. The JCA can also be used in a non-managed environment, in which case the application must manage connections, transactions, and security itself.

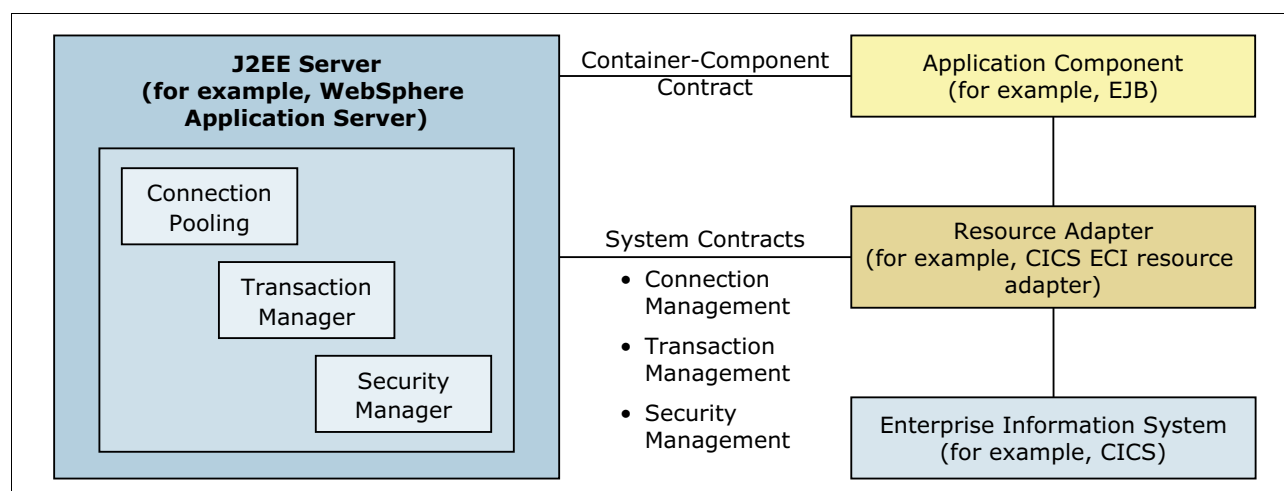


Figure 4-10 Java EE connector architecture

Managed environment preference: We strongly advise that you use a managed environment over a non-managed environment. The application development costs for a non-managed environment are significant and the quality of service is generally not as good as that provided by a managed environment such as WebSphere Application Server.

The Common Client Interface (CCI) defines a common application programming model for interacting with resource adapters and is independent of any specific EIS. Of course, this does not mean that a developer can write exactly the same code to access one EIS (for example, CICS) that is written to access another EIS (for example, an IMS system). However, the generic CCI classes are the same in that they are independent of the EIS, whereas specific EIS classes cater to the differences. For example, the parameters that are used to call a CICS program differ from those that are used to invoke an IMS transaction, but the programming model is the same (independent of the EIS). As a result, you can increase developer productivity if you develop applications to communicate with more than one EIS. The CCI programming interface is similar to other Java EE interfaces, such as the Java Database Connectivity (JDBC) interface or Java Message Service (JMS) interface.

System contracts

The JCA defines a standard set of system-level contracts between a Java EE application server and a resource adapter:

- ▶ A *connection-management* contract provides a consistent application programming model for connection acquisition and enables a Java EE application server to pool connections to a back-end EIS. This contract leads to a scalable and efficient environment that can support a large number of components requiring access to an EIS system.
- ▶ A *transaction-management* contract defines the scope of transactional integration between the Java EE application server and an EIS that supports transactional access. This contract allows a Java EE application server to function as a transaction manager, and control two-phase commit transactions across multiple resource managers (known as global transactions). This contract also supports the LocalTransaction interface, which refers to one-phase commit transactions that are managed internally to a resource manager without the involvement of an external transaction manager. When using the managed environment with JCA V1.6, the transaction level (LocalTransaction or XA) can be specified on a connection basis by using a connection factory property.
- ▶ A *security-management* contract enables secure access to an EIS. This contract provides support for a secure application environment, which reduces security threats to the EIS and protects valuable information resources managed by the EIS. Both of the following items are supported:
 - Container-managed sign-on, in which the Java EE application server is responsible for flowing security context to the EIS
 - Component-managed sign-on, in which the application is responsible for flowing security context to the EIS

When used with WebSphere Application Server for z/OS, the resource adapter enables automatic propagation of security credentials from the application server to CICS. This functionality is known as *thread identity support*.

These system contracts are transparent to the application developers, which means that they do not have to implement these services themselves. In a managed environment, these system contracts are what make the JCA such a powerful solution for integrating existing CICS applications with new Java EE applications running in an application server such as WebSphere Application Server.

Resource adapters

There are two types of resource adapters for the JCA integration, through the CICS Transaction Gateway, with CICS:

cicseci.rar The CICS ECI (external call interface) resource adapter is provided with both the CICS Transaction Gateway for multiplatforms and CICS Transaction Gateway for z/OS. It supports the XA and LocalTransaction interfaces.

cicsepi.rar The CICS EPI (external presentation interface) resource adapter is provided only with CICS Transaction Gateway for Multiplatforms and provided access to 3270-based CICS transactions.

The ECI resource adapter is the simplest to use and the most commonly used CICS Transaction Gateway resource adapter. Support is provided both for synchronous and asynchronous calls. However, asynchronous calls using the CICS ECI resource adapter have their limitations. For example, you cannot make several concurrent calls and then wait for the response. You must take the response of each previous call before making another call.

The JCA resource adapters provided by the CICS Transaction Gateway are effective replacements for the CICS Transaction Gateway base Java classes. Support for the ECI resource adapters is included in the Rational Software Development Platform series of products, but tooling support for direct use of the ECI Java classes is not.

Figure 4-11 shows how the CICS Transaction Gateway enables access to a CICS business logic program.

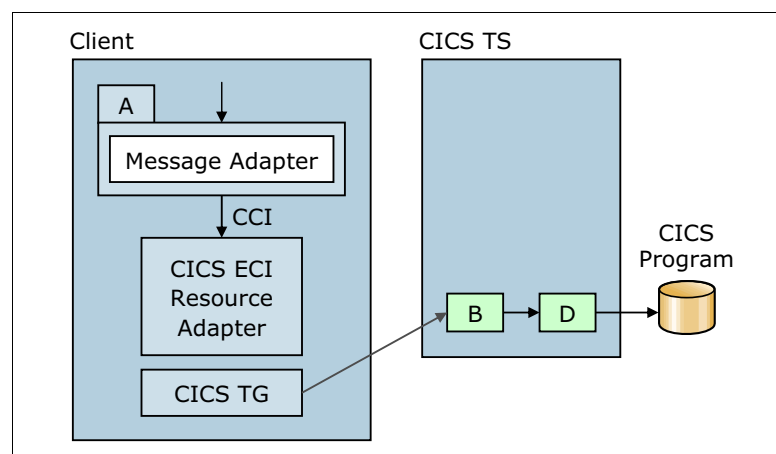


Figure 4-11 Connecting using JCA

A Java EE application uses the CCI programming interface to invoke the CICS ECI resource adapter. The CICS Transaction Gateway ECI classes are packaged with the ECI resource adapter and are used to pass the application request to the CICS Transaction Gateway.

The Java EE application can invoke the CICS business logic program (B) directly if no message transformation is required. In this case, Rational Application Developer can be used to create a Java bean to represent a COMMAREA formatted as COBOL types, with Java methods for getting and setting fields.

A message adapter in CICS is required only if the message is to be transformed. For example, the request is in XML and the CICS business logic program requires a COBOL record format. The length of the message is subject to the normal CICS COMMAREA message length limitation of 32 KB.

The CICS Transaction Gateway is the preferred implementation for JCA connectors to access all CICS servers from WebSphere Application Server, for applications that require a high-performing, secure, and scalable access option with tight integration to existing CICS applications. The CICS Transaction Gateway benefits from ease of installation and flexible configuration options, and requires minimal changes to CICS, and in most cases no changes to existing CICS applications. In addition, the CICS Transaction Gateway supports a range of non-Java clients, including C, C++, COBOL, and .NET.

The JCA is considered a medium coupling architecture, compared with the EJB architecture (high coupling) and the web services architecture (low coupling). The JCA can be used with a diverse range of supported environments and different deployment options. These are described in detail in the “CICS Transaction Gateway” topic in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

4.4.4 JCA through WebSphere z/OS Optimized Local Adapters (WOLA)

WOLA is a functional component of WebSphere Application Server for z/OS that provides an efficient cross-memory mechanism for calls both inbound and outbound to and from WebSphere Application Server for z/OS. It can communicate with external address spaces, which include CICS, batch, IMS, UNIX System Services, and ALCS. Because it avoids the overhead of other communication mechanisms, it is capable of high-volume exchange of messages. Figure 4-12 shows the topology of WOLA and CICS.

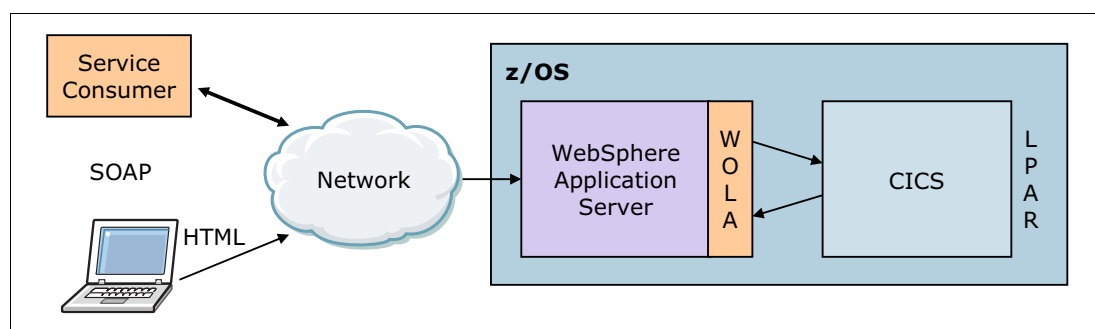


Figure 4-12 WOLA and CICS topology

WebSphere Optimized Local Adapters (WOLA) is a set of runtime components and APIs that are provided with WebSphere Application Server z/OS V7 and later. WOLA offers the following benefits:

- ▶ Enables procedural style calls to and from other subsystems on z/OS.
- ▶ Provides a set of APIs for sending data between different z/OS environments.

- Provides interfaces to connect in and out of WebSphere Application Server for z/OS (bidirectional connectivity).
- Supports the routing of requests between separate WebSphere Application Server servers with the development mode support.

For inbound connections to CICS from WebSphere Application Server, the JCA resource adapter hides most of the WOLA implementation details, which can make Java programming easier.

Benefits of WOLA

WOLA began as a way to allow program access into WebSphere Application Server for high transaction rate batch programs. WOLA offers the following benefits:

- Is integrated with WebSphere Application Server and therefore inexpensive.
- Has low pathlength and thus performs well.
- Is bidirectional.
- For inbound connections to CICS from WebSphere Application Server, the JCA resource adapter (ola.rar) hides most of the WOLA implementation details, which makes Java programming easy.

A standard JCA resource adapter is used for deployment into WebSphere Application Server for z/OS so that Java programs can have a standard Common Client Interface (CCI) to interact with however, it is not the same one that is used by CICS Transaction Gateway. These resource adapters are described in detail in the “WOLA” topic (section 3.3) in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

4.4.5 CICS web support

CICS web support provides an HTTP listener and a message adapter program that can be written using CICS WEB APIs. Figure 4-13, shows how HTTP can be used directly with CICS TS. CICS web support also allows a CICS application to initiate an HTTP request and to receive the response from an HTTP server program, thus providing bidirectional support for the HTTP protocol.

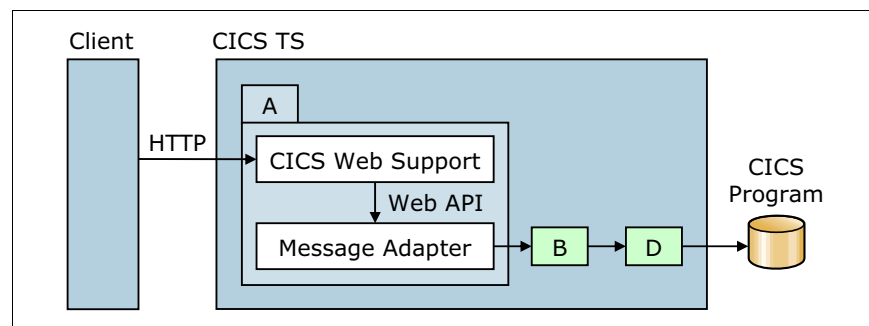


Figure 4-13 Connecting using CICS web support

CICS supports HTTP basic authentication for user ID identification or the more secure Secure Sockets Layer (SSL) encryption and authentication with client and server certificates. CICS web support sets up the transaction and security environment and calls the message adapter. The message adapter uses the CICS WEB APIs to extract the HTTP user data, which is

typically formatted as an HTML form. The message adapter also has access to the SSL certificate and HTTP headers and socket information if required. The message adapter transforms this information and places it into containers or a COMMAREA and calls the business logic program.

Supports HTTP: CICS TS currently supports the HTTP 1.0 and HTTP 1.1 specifications.

If the service requester is a web browser, the response message will typically be formatted as HTML. If the service requester is an application, the response message will normally be formatted as XML. The message adapter can use the CICS XMLTRANSFORM or DOCUMENT APIs to create HTML or XML documents. The response is returned to the client for display or processing.

HTTP is synchronous and stateless. However, if state management is required, CICS provides a utility for storing state data indexed by a state management token that the HTTP client can return on subsequent calls to retrieve the state.

CICS web support also forms the basis for the CICS support of Atom feeds that adhere to the Atom Syndication Format and the Atom Publishing Protocol. CICS exposes resources such as VSAM files and temporary storage queues as Atom feeds without data access programs. For more information about CICS web support and Atom, see the “CICS web support” topic in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

4.4.6 Asynchronous messaging

With WebSphere MQ you can more easily exchange information across separate platforms, integrating existing business applications in the process. WebSphere MQ assures reliable delivery of messages, dynamically distributes workload across available resources, and helps to make programs portable.

WebSphere MQ provides Java Message Service (JMS) APIs and native WebSphere MQ APIs for use by service requesters on a wide variety of platforms, with many options for routing and encrypting messages prior to arriving on WebSphere MQ for z/OS.

Figure 4-14 shows the WebSphere MQ trigger monitor program that is provided by CICS, which can be used to automatically start an appropriate message adapter program when messages arrive. The message adapter uses WebSphere MQ native APIs to receive the message, transform it if required, and call the business logic program. A reply message can be sent by using the reply-to queue that is defined in the message. For efficiency, the message adapter program usually continues to process messages on the inbound queue until it is empty.

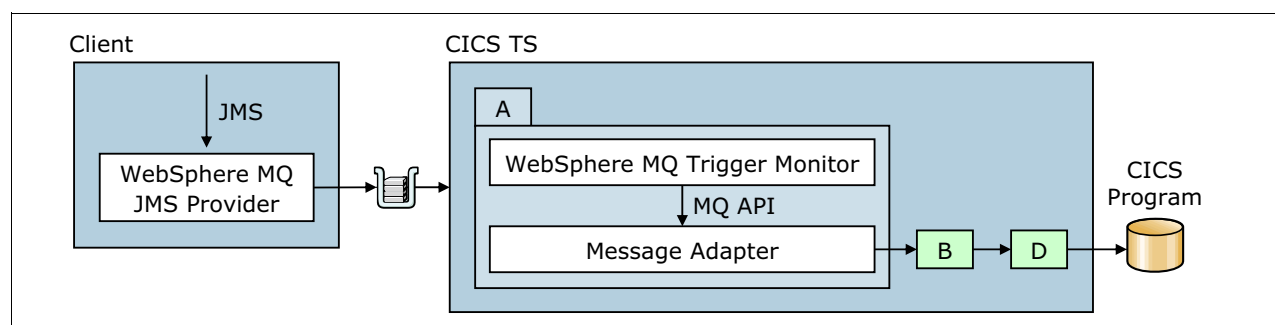


Figure 4-14 Connecting using WebSphere MQ

The WebSphere MQ DPL bridge for CICS provides an alternative option (Figure 4-15). This generic adapter passes a message from a named input queue to a business logic program through the COMMAREA. This way is ideal in the situation where the service requester can format the message into a form acceptable by the business logic program.

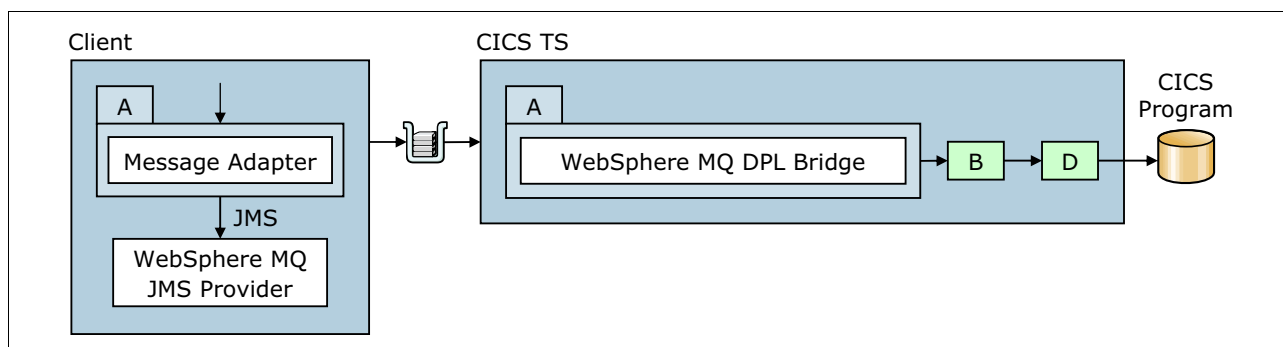


Figure 4-15 Connecting using the WebSphere MQ DPL bridge

When using the WebSphere MQ DPL bridge, the client application writes a structured message to the queue. This message must contain information in a predefined format that the monitoring transaction can use to decide how to handle the message. Several formats are possible, each starting with a block of data referred to as an MQMD header. This field contains control information used by the monitoring transaction like the message format type, along with optional information, such as a reply-queue identifier and a user ID.

For more information about the use of WebSphere MQ with CICS, see the “WebSphere MQ” topic in *CICS and SOA: Architecture and Integration Choices*, SG24-5466.

4.4.7 CICS sockets

The TCP/IP Socket Interface for CICS (also known as *CICS sockets*) is part of z/OS Communications Server and supports peer-to-peer applications in which both ends of the connection are fully programmable (Figure 4-16). CICS sockets is most suitable when you are required to use a protocol not already supported by CICS TS.

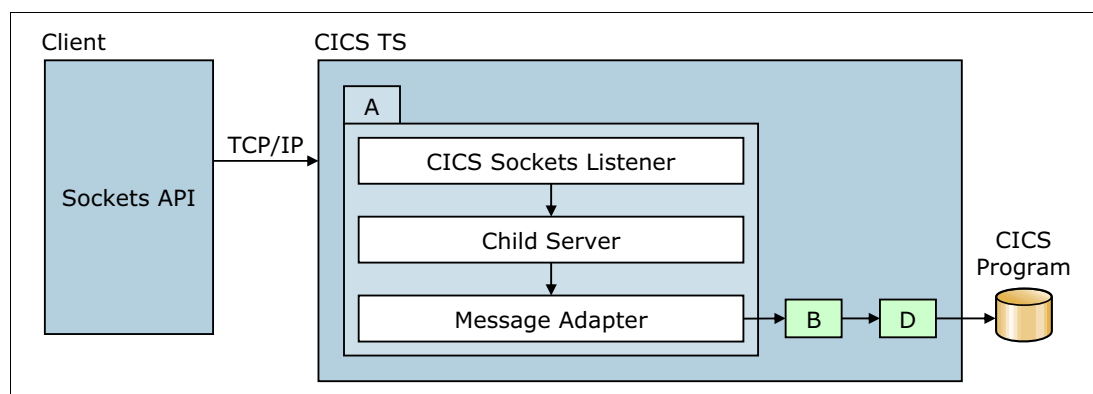


Figure 4-16 Connecting using CICS sockets

CICS sockets is configured and managed using CICS sockets transactions and configuration files rather than CICS systems management facilities.

CICS sockets provide an iterative listener and a concurrent listener, or you can write your own listener to meet your needs. The listener and child server use the CICS sockets APIs to open

connections, send and receive data, and perform general communications control functions. The programs can be written in COBOL, PL/I, assembler language, or C. Client adapters can be written to create new outbound connections.

For further details, see the Communication Server manual, *IP CICS Sockets Guide*, SC31-8518, and see *CICS/ESA and TCP/IP for MVS Sockets Interface*, GG24-4026.

4.4.8 Integration options comparison

CICS provides a range of access methods to support modern connectivity architectures, such as web services and JCA, and other standard transport mechanisms. With the correct external connectors and internal adapters, you can maximize the reuse of your existing mission-critical CICS assets. Table 4-1 compares the connection architectures and standard transport mechanisms discussed in this chapter.

Table 4-1 Common architectures and standard transport mechanism

Connectivity option	Required middleware	Main capabilities	Guidance
Web services	CICS-only solution	▶ Inbound and outbound ▶ Low coupling ▶ Synchronous (HTTP) ▶ Asynchronous (WMQ) ▶ QoS based on transport type ▶ Support for some Web Services standards	Should be first consideration for service enabling CICS applications, particularly when you need to support multiple service requester types or need bidirectional support.
JCA with CICS TG	CICS Transaction Gateway Java EE server (normally WebSphere Application Server)	▶ Inbound to CICS ▶ Medium coupling ▶ Synchronous ▶ High qualities of service (QoS)	Most appropriate solution when service requester is Java EE component and when high QoS required (high availability, transactions, security).
JCA with WOLA	WebSphere Application Server for z/OS	▶ Inbound and outbound ▶ Tight coupling ▶ Synchronous ▶ High QoS	Particularly useful for JCA access to and from CICS, and for high throughput and performance requirements.
CICS web support	CICS-only solution	▶ Inbound and outbound ▶ Medium coupling ▶ Synchronous ▶ Medium QoS	Use with web services, RESTful services, and Atom feeds, or when remote client/server only supports HTTP.
WebSphere MQ for z/OS	CICS-only solution	▶ Inbound and outbound ▶ Medium coupling ▶ Asynchronous, with almost-synchronous capabilities ▶ Assured delivery	Exploit WebSphere MQ for basic messaging and flowing web services.
CICS sockets	z/OS Communications Server	▶ Inbound and outbound ▶ Tight coupling ▶ Synchronous ▶ Limited QoS	Use when remote client/server supports only TCP/IP sockets communication.

Both CICS Transaction Server and WebSphere Application Server are strategic middleware products that interoperate well when using technologies, such as web services, to support end-to-end on demand systems. They exploit and complement z/OS qualities of service, such as high availability and scalability at a low cost per transaction, with a high level of security. In combination, WebSphere Application Server and CICS support almost any mission-critical SOA solution.

4.5 Modernizing data layer using CICS VSAM Transparency

CICS VSAM Transparency (CICS VT) was introduced to help migrating VSAM-based applications from using VSAM files to DB2 without incurring the costs of application redesigns or complex rewrites. CICS VT provides a solution where the actual data is moved from VSAM into DB2, while the application continues to operate unchanged, performing VSAM programming operations, which are intercepted by CICS VT, to make the actual data retrieval calls from DB2.

Using CICS VT, VSAM-based applications can remain in operation while the data layer is being migrated to relational database and thus taking advantage of newer products, such as analytical engines and at the same time capitalizing on the abundance of database management tools and DBA services.

Figure 4-17 shows how CICS VT intercepts all VSAM calls and decides if the data is retrieved from VSAM or DB2. For each VSAM call, a check is made to determine whether the VSAM data moved to DB2. If not, processing continues normally. If the data was moved to DB2, CICS VT uses generated drivers to access the data through SQL calls and returns it to the calling program in the original file format that was expected before the data was migrated.

The design and functionality of CICS VT allows applications to implement a staggered approach to migrating VSAM files to DB2 as opposed to a mass and complex all-or-nothing approach. Each file can be migrated and tested, one at a time, without change to the original applications. After the data is in DB2, future changes to the application can include DB2 access to eventually replace the actual VSAM access calls.

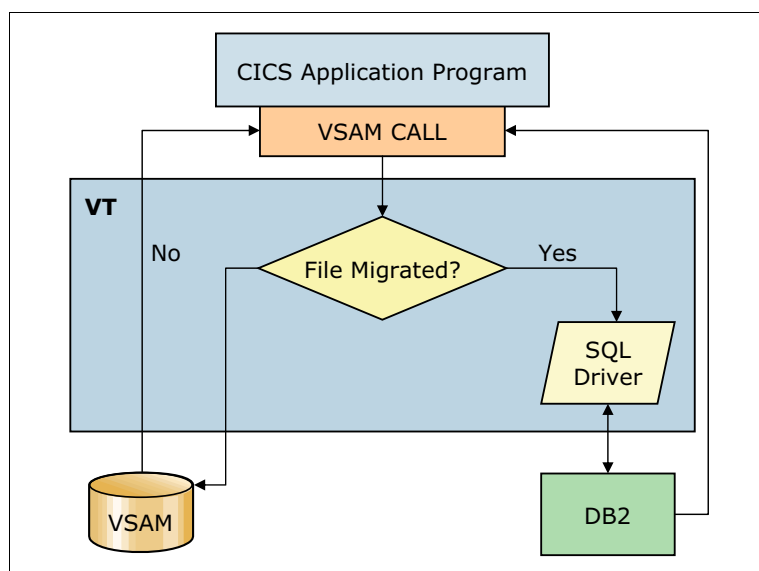


Figure 4-17 CICS application VSAM call using CICS VSAM Transparency

CICS VT provides several tools and components to aid with the conversion process. The mapping component identifies the relationship between the original VSAM file layout and its proposed DB2 format. For VSAM files that map to a single DB2 table, CICS VT provides an automated mapping facility and for more complex mappings, a manual feature provides advanced options to allow a mapping to multiple DB2 tables.

After the mapping phase is complete, the data can be migrated to DB2 by using a combination of CICS VT and DB2 utilities. CICS VT provides a dual-mode facility to allow testing against the original VSAM file and the new DB2 tables during the cutover process. Both modes of access can be tested and verified to ensure the accuracy of the migration before a final cutover.

The mapping process between VSAM files and DB2 tables generates CICS VT run-time drivers, with static SQL calls, which are used by CICS VT to access the DB2 data and return the results to the calling application in the original VSAM file format, thus earning the name VSAM Transparency.

4.6 Modernizing 3270 presentation layer

From the late 1960s to early 1990s, the terminal devices, such as the 3270, were the most popular type of client for CICS applications. The presentation and business layers in these CICS applications were optimized and limited to the capabilities of the terminal, often requiring more than one screen interaction to validate and accumulate the necessary information before finally processing the request against a database.

This type of programming results in some CICS applications combining the presentation and business layers in the same program, which presents challenges when needing to reuse the business layer to support new client types. In this section, we focus on four technologies that enable a 3270 application to be reused:

- ▶ CICS Front End Programming Interface (FEPI)
- ▶ IBM Rational Host Access Transformation Services (HATS)
- ▶ CICS Link3270 Bridge
- ▶ Service Flow Modeler

Figure 4-18 shows the differences between the first three technologies at a high level.

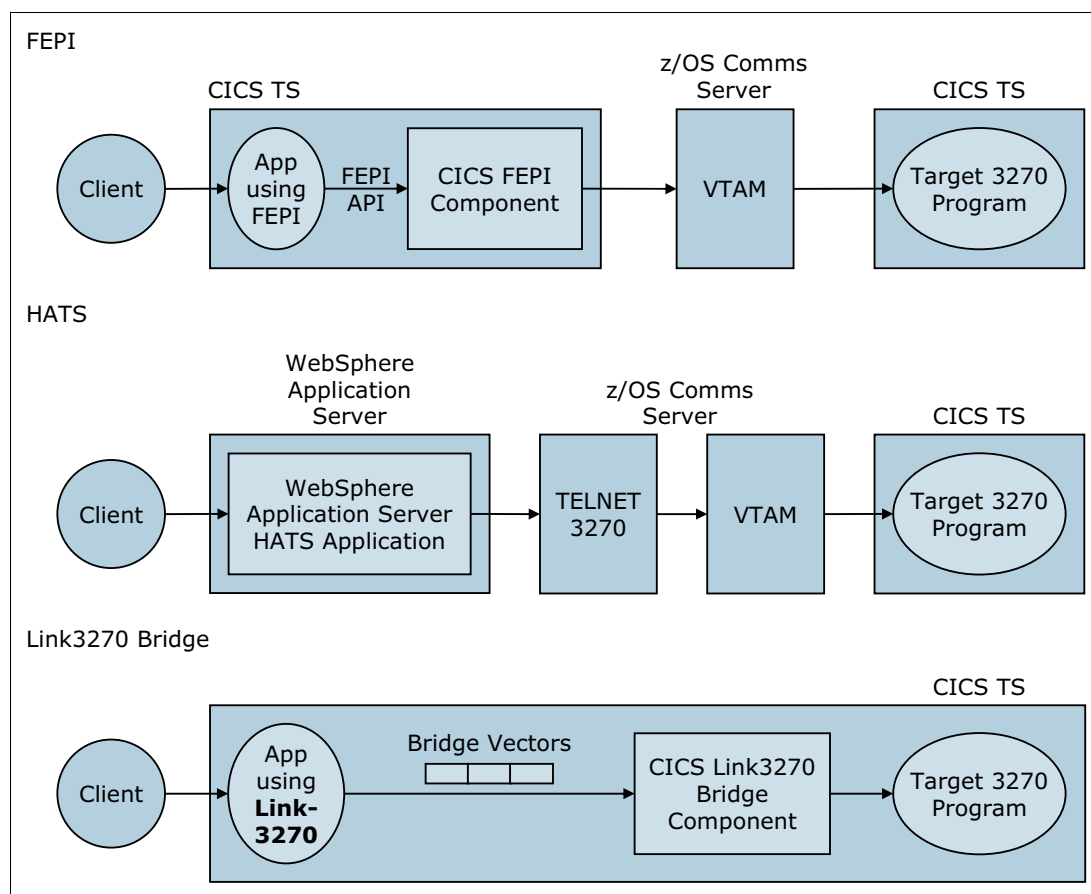


Figure 4-18 Comparison of FEPI, HATS, and the Link3270 Bridge architectures

In Figure 4-18, the client at the top of the diagram connects into CICS TS by using an integration technology such as web services or the CICS Transaction Gateway. A CICS application is invoked, which uses the FEPI API to tell CICS to communicate with the target 3270 program through IBM VTAM® by emulating a 3270 terminal.

The client in the middle of Figure 4-18 connects over HTTP to an application running in WebSphere Application Server on either a distributed platform or z/OS. The application, which is generated through HATS tooling, emulates a 3270 terminal and connects through VTAM to the target 3270 program.

The client at the bottom of Figure 4-18 connects into CICS TS using a technology such as web services or the CICS Transaction Gateway. A CICS application is invoked, which in turn invokes the CICS Link3270 bridge, passing data in the form of bridge vectors. The CICS Link3270 bridge bypasses VTAM and interacts directly with the target 3270 program.

4.6.1 Front End Programming Interface (FEPI)

FEPI is an API that CICS TS offers. It enables you to write CICS application programs that drive other CICS 3270 programs, thus providing a front end to those programs. The interface simulates the terminals that the other programs use. The ability to drive these 3270 programs from another CICS program allows the existing 3270 programs to be used in various ways without changing them.

The CICS FEPI component acts as a virtual terminal. It communicates through VTAM to the target 3270 program, in the same way as a real terminal does. The CICS application that uses the FEPI API sends commands such as Position the cursor at point X on the screen then enter the text ABC. The benefit of this approach is that the target 3270 program is being called through VTAM as it would be from a real terminal. Therefore, the behavior of the target 3270 program will be as it is for any other terminal. For further information about FEPI, see the CICS Transaction Server Information center, and in particular, the *Front End Programming Interface User's Guide*, SC34-7169.

4.6.2 Rational Host Access Transformation Services (HATS)

Rational Host Access Transformation Services (HATS) provides the ability to create a new presentation layer, such as a web front end, to an existing CICS 3270 program. With it users can more easily generate modern presentation layers that map to the input that is expected by the target 3270 program. It also generates a runtime component, which runs in an environment such as WebSphere Application Server. Users can connect to WebSphere Application Server from a web browser. The generated WebSphere Application Server application parses the input from the users, then interacts with the target 3270 program through a Telnet 3270 server, which in turn communicates with VTAM. From the perspective of the target 3270 program in CICS, this program is just another terminal connecting through VTAM.

The HATS solution provides the tools needed to quickly and easily transform 3270 programs to web, portlet, rich client, or mobile device user interfaces, or to create a web service front end to a 3270 program. HATS has a development component named the HATS toolkit. The HATS toolkit has a wizard-based development process for creating HATS applications to transform existing 3270 programs. The development process is similar, regardless of whether users are extending CICS 3270 programs to the web, portal, a mobile browser, a rich client, or as web services. The HATS solution enables tailoring of 3270 applications to a specific set of users, hides unnecessary information, organizes data into tables, and displays only required input fields. Also, the solution can provide drop-down lists of valid values for an input field, change the size and location of text, and provide navigation buttons to reduce data entry errors and increase productivity.

There is no specialized HATS runtime server. For a web interface, all of the necessary runtime information is deployed into an enterprise archive (EAR) file and runs in WebSphere Application Server or WebSphere Portal. For a rich client interface, the necessary runtime information can be generated to run in an environment such as the Eclipse Rich Client Platform, IBM Lotus® Notes® and Lotus Expeditor. The generated code takes care of the interaction with the target 3270 program. For further information about HATS, go to the following website:

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

4.6.3 CICS Link3270 bridge

The CICS Link3270 bridge allows an application in CICS TS to call a CICS 3270 application, and is similar in its aims to FEPI. The difference in the approaches is that the Link3270 bridge removes the need for requests to flow out of CICS TS through VTAM to the target 3270 program. Instead, the Link3270 bridge component interacts directly with the target 3270 program. So for example, when the target 3270 program issues a RECEIVE call to obtain data from the terminal, this call is handled by the Link3270 bridge component, which passes the required data to the target 3270 program. The invoking application passes data to the Link3270 bridge in a format known as bridge vectors. The bridge vectors are used by the Link3270 bridge to build the information requested by the target 3270 program. The benefit of

this approach is the removal of the need to interact through VTAM. The potential drawback is that the target 3270 program is now being driven in a different manor from its original design, and there are some CICS API restrictions to which the target 3270 application must adhere. See the “Link 3270 programming considerations” topic in the *CICS External Interfaces Guide*, SC34-6449, for further details about such restrictions.

The 3270 terminal and user are replaced by an application program, known as the bridge client application. Commands for the 3270 terminal in the 3270 user transaction are intercepted by CICS TS and replaced by a messaging mechanism that provides a bridge between the client application and the 3270 user transaction.

This mechanism provides a link interface that can be accessed using a distributed program link (DPL), an external call interface (ECI) request, or an External CICS TS Interface (EXCI) request. The bridge client requests services of the Link3270 bridge using COMMAREA messages in a prescribed format called bridge vectors, and the Link3270 bridge returns results to the bridge client in formatted messages.

The Link3270 bridge provides an interface to enable 3270-based CICS transactions to be invoked using a LINK request, and without the facilities of a 3270 terminal. As such, it is not a means of providing web access (because there is no direct means of generating HTML), but it is an important enabling technology because it allows a client module to link to existing 3270 transactions. The client module could itself be exposed to external clients that need access to the business logic.

The client application uses the Link3270 bridge to run 3270 transactions by passing a COMMAREA that identifies the transaction to be run and contains the data used by the user application. The response contains the 3270 screen data reply. For further information about the Link3270 bridge, see the CICS Transaction Server information center, in particular the *External Interfaces Guide*, SC34-7168.

4.6.4 Service Flow Modeler (SFM)

HATS includes its own tooling, but the Link3270 bridge and FEPI require a programmer to write a program that builds the interactions between the invoking application and the target 3270 application. Performing this process manually can be complex and time consuming, so it is worth considering tooling that can assist with this process. IBM provides Rational Developer for System z, which contains a component called the Service Flow Modeler, which can be used to simplify the development of applications that need to interact with 3270 programs.

The Service Flow Modeler (SFM) provides the tooling for generating flows that can be deployed into CICS TS. Within CICS TS is a component called the Service Flow Runtime (SFR), which provides the runtime environment for the generated flows.

As an example, perhaps you have two 3270-based terminal applications that you want to call serially to compose a new CICS application, and you want that CICS application to be made available as a web service. The Service Flow Modeler provides tooling to record the terminal interactions with the target 3270 programs, model the new application, and generate runtime code and artifacts that can be deployed into CICS TS. The result is a web service enabled CICS application that uses the Link3270 bridge or FEPI to invoke the 3270 applications that it relies on. Use of this tooling can make development of such applications considerably easier than manual coding.

SFM enables distributed applications to make business requests of existing CICS 3270 and COMMAREA applications as callable services. It enables customer-written applications to integrate seamlessly with business-critical 3270 and COMMAREA applications by generating service flows that contain a web service interface.

SFM provides modern graphical tooling based on the Eclipse platform. The tooling provides a means to simplify the reuse of existing 3270 and COMMAREA applications within business processes by assisting with tasks such as these:

- ▶ Interactions with terminal-based programs can be recorded in a simple manner by following a guided wizard.
- ▶ Application flow, conversion, and integration can be orchestrated within the tooling.
- ▶ The completed business process flow, including terminal interactions, can be generated as code for the Service Flow Runtime in CICS TS.
- ▶ The generated business process can be automatically deployed into CICS TS.
- ▶ Exposing business processes as web services inside CICS TS.

Approaches to running business process: Running the business process inside CICS TS can be important for performance, because a client invoking the business process over the network makes only one invocation to the business process, which can then call multiple CICS programs sequentially. An alternative approach is to run the business process outside of CICS TS and make a call to CICS TS for each CICS program that needs to run, but this approach will make more calls over the network, which can be costly.

For further information about the Service Flow Modeler, see the “Enterprise Service Tools for Web services and SOA” section of the IBM Rational Developer for System z information center:

<http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp>

CICS Service Flow Runtime (SFR)

CICS Service Flow Runtime (SFR) is a run time that enables business processes created with SFM to be deployed and run in a CICS environment. SFR is a strategic solution that is used to avoid being forced into programming-intensive solutions that are prone to error, especially when the host CICS applications are changed for maintenance or upgrade.

SFR uses SFM-generated adapter services to provide the sequencing of 3270 screens in a CICS application. Thus, when a service request comes to CICS from a distributed application, SFR navigates the appropriate 3270 screen sequences, formulates a consolidated response, and sends a single service response to the requester. Figure 4-19 on page 109 shows a conceptual view of a generated service flow deployed into CICS.

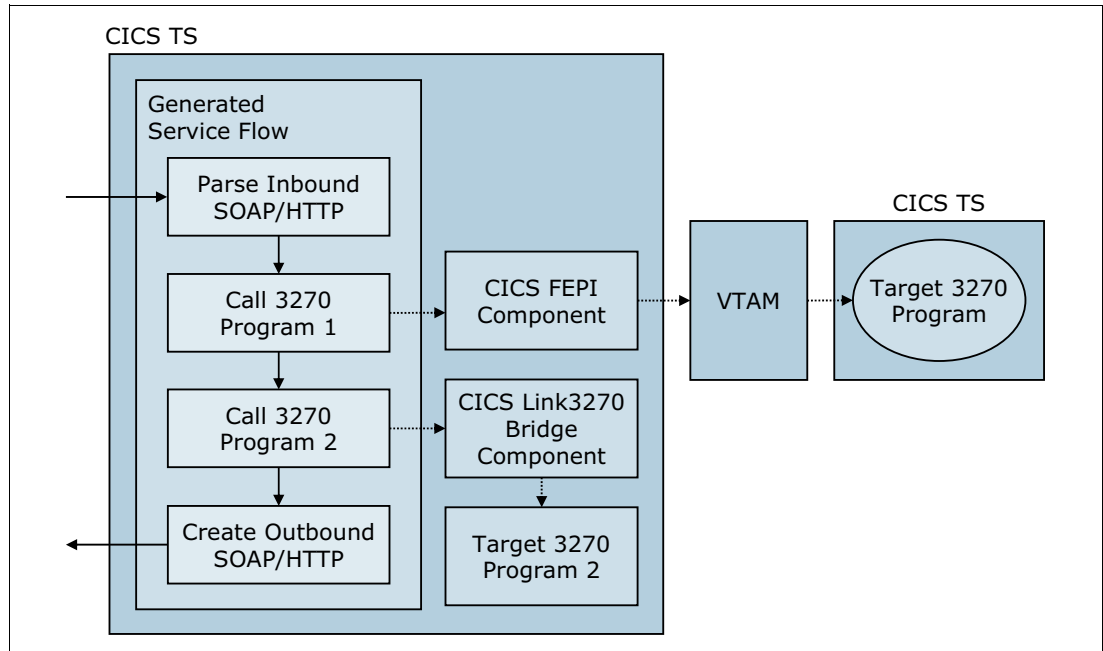


Figure 4-19 A conceptual view of a generated service flow deployed into CICS TS

In Figure 4-19, CICS TS exposes a web service that can be invoked from any web service enabled client application. On receipt of a web service request, the message is parsed by CICS TS and converted into a data structure suitable for use in a CICS program. The service flow then makes a call to a 3270 program by using FEPI. On return from that call, the service flow makes a second call to a 3270 program, this time by using the Link3270 bridge. Finally, a response message is built by the service flow, converted to a web service response message by CICS TS, and returned to the invoking application. For further information, see the CICS TS information center, specifically *CICS Service Flow Runtime User's Guide*, SC34-6913.



Application development lifecycle

This chapter describes the CICS application development lifecycle and how it can be made more productive and efficient by using several tools we describe. All tools discussed in this chapter are Eclipse-based and therefore offer a common user interface and tight integration.

The chapter is divided into three sections:

- ▶ CICS application development lifecycle
- ▶ Procedural application development lifecycle, which applies to most of the procedural programming languages available on CICS, such as COBOL and PL/I
- ▶ Java application development and deployment lifecycle, which covers the architecture of Java applications running on CICS and the associated development tools

5.1 CICS application development lifecycle

Over many years, companies have implemented a huge number of CICS applications. These applications are driving much of today's business in financial institutions, banks, and so on.

Thus, these applications are (in accounting terms) intangible assets of significant value: intellectual property that automates, extends, and encapsulates important aspects of a business. The value of these applications, if viewed from the standpoint of income that is generated over time, is substantial.

Yet software, as with all corporate assets, requires reinvestment to sustain it, for continuous improvement and growth to support changing business models.

Consider debugging a situation in which incorrect values appear in a Java web page that rendered data from an IBM Information Management System (IMS) database that was accessed through a CICS transaction using a web service. How do you track this kind of problem? Where do you start?

Fortunately, the software industry has moved beyond time-sharing-option (TSO) and manual approaches to maintain and support cross-platform (or even complex single-platform) applications. New technologies are saving companies time and money, and, ultimately, improving application quality and lowering defect rates.

Tools in the following categories, then, are key to managing the application development lifecycle and preserving and extending the value of an enterprise's CICS applications:

- ▶ Analysis:
 - CICS Interdependency Analyzer
 - Rational Assets Analyzer
- ▶ Modeling:
 - Rational Software Architect
- ▶ Development:
 - Rational Developer for System z
 - CICS Explorer SDK
 - Rational Team Concert
- ▶ QA testing:
 - Rational Development and Test Environment for System z
 - Debug Tool for z/OS
- ▶ Deployment:
 - CICS Deployment Assistant
 - CICS Configuration Manager
- ▶ Performance analysis and tuning:
 - Application Performance Analyzer for z/OS
 - CICS Performance Analysis for z/OS
 - Workload Simulator for z/OS
- ▶ Service delivery management:
 - IBM OMEGAMON® XE for CICS
 - Fault Analyzer for z/OS
 - Health Center

Figure 5-1 shows phases of the CICS application development lifecycle and the tools that are associated with each phase.

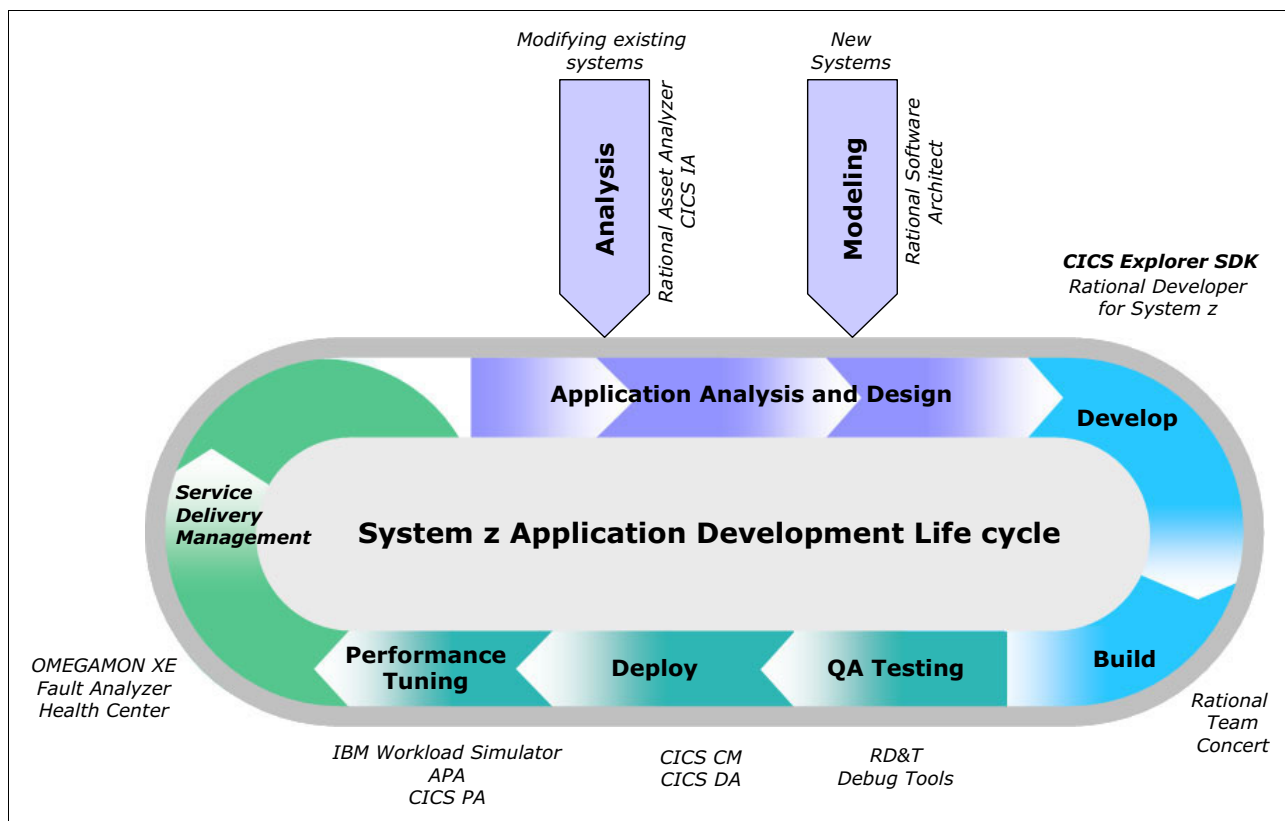


Figure 5-1 CICS application development lifecycle

In the next sections, we describe how each of these products fits into the application development lifecycle. Later in the book, we give details about the capabilities of each product in both the procedural and object-oriented programming paradigms.

Important: There are additional approaches for application development on System z, such as Rational Unified Process for System z, Scrum, and Agile. For more information about System z application development resources, see *z/OS Traditional Application Maintenance and Support*, SG24-7868.

5.1.1 Eclipse-based environment

Eclipse is a platform for building and deploying client applications in which the majority of data manipulation is done by the client application rather than the server. This platform is known as a rich client platform (RCP). With it, you can deploy native graphical user interface (GUI) applications to a variety of desktop operating systems, and present the user with a simple, consistent view.

One feature of Eclipse that makes it such a good choice for CICS system management and development tools is its plug-in architecture, which allows various components to be added that can extend and complement each other. A number of plug-ins exist that can be integrated into Eclipse environments, such as system management interfaces, messaging clients, or development tools. Two popular Eclipse-based products are IBM Rational Developer for System z, which is an integrated development environment for the System z platform, and the

CICS Explorer, which is the latest tool for performing CICS system management and resource deployment activities. The CICS Explorer also acts as an integration point for CICS Transaction Server, CICS Tools, CICS Transaction Gateway, and Problem Determination (PD) Tools.

By using Eclipse as a common base technology, CICS Explorer integrates within Rational Developer for System z to provide system administration capabilities to CICS application developers. For more information about Eclipse, see the following website:

<http://eclipse.org/>

5.1.2 CICS Explorer

CICS Explorer is the smart new interface to CICS. It has a common Microsoft Windows style so anyone can easily learn and use it. It provides a framework of navigators and views backed by a rich CICS data model that is fed by CICSplex System Manager (CICS SM). CICS Explorer can significantly accelerate the transfer of knowledge, skills, and leading practices to the next generation of technical staff and experts.

The CICS Explorer also provides an integration point for CICS tooling (plug-ins), each adding their own special value. The CICS Explorer has a common, intuitive, Eclipse-based environment that architects, developers, administrators, system programmers, and operators can use to efficiently develop and manage new and existing applications. Using integrated help, tutorials, and a direct connection to the CICS Information Center, both experienced and new users will find getting started easy. All of the information about CICS resources and the logic for passing that information to users is executed by CICSplex SM services, in the presence of CICSplex, or, in the case of single CICS regions in smaller developments, by updating CSD definitions.

The CICS Explorer is a great way to accelerate the transfer of knowledge, skills, and leading practices to the next generation of technical staff and experts. Figure 5-2 on page 115 provides a glimpse of the CICS Explorer interface.

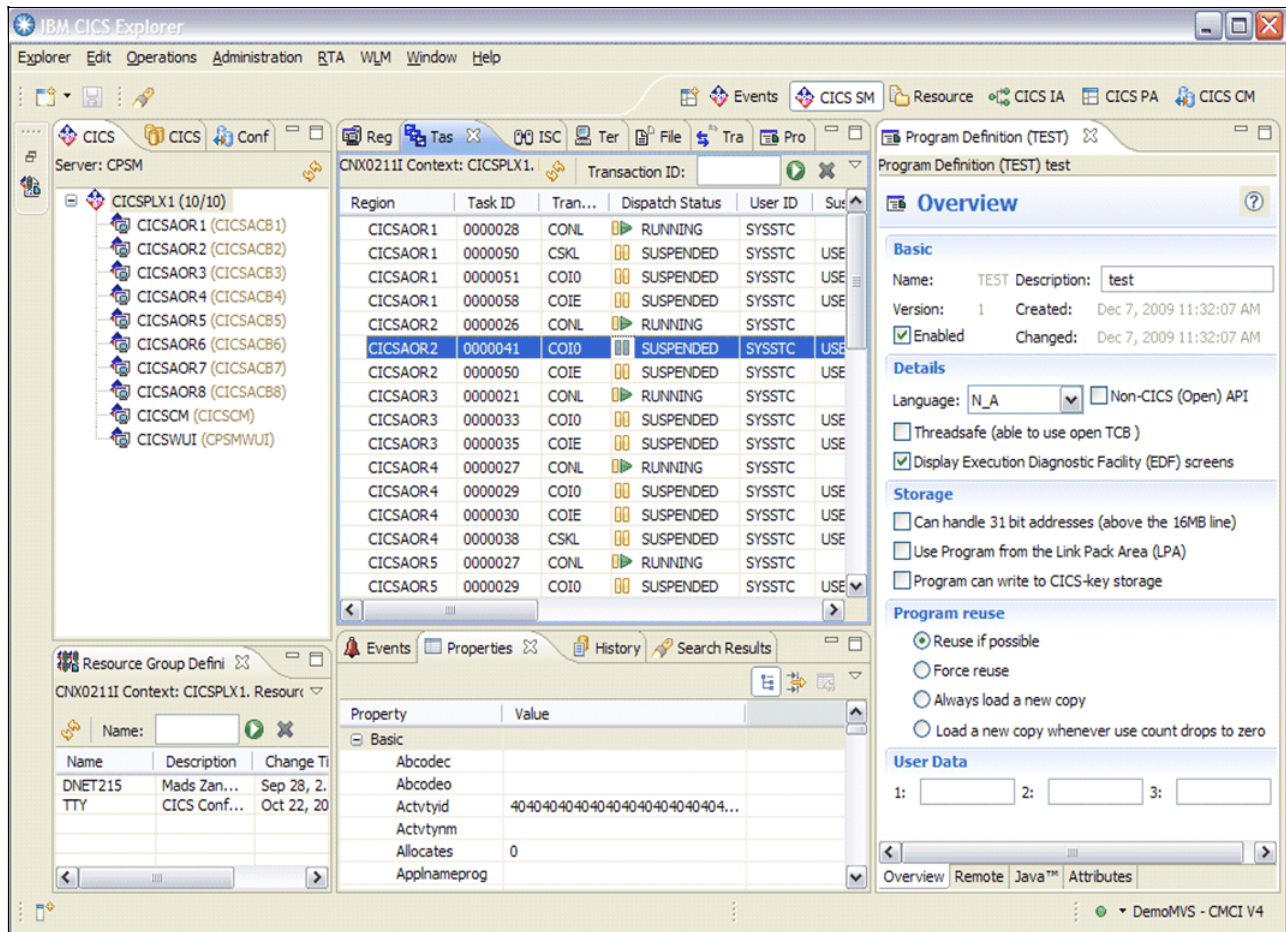


Figure 5-2 Sample CICS Explorer window

The task-oriented views in CICS Explorer provide integrated access to a broad range of data and control capabilities. CICS Explorer also has powerful, context-sensitive resource editors. Integration points for CICS Transaction Server, CICS Tools, CICS Transaction Gateway, Problem Determination Tools, and Rational Tools are extensible by independent software vendors, system integrators, and customers who use the CICS Explorer Software Development Kit. More information about the CICS Explorer and its various plug-ins, see *IBM CICS Explorer*, SG24-7778.

Most of the tools mentioned in this chapter are designed to help ease the burden of building and managing complex System z environments. By helping to improve delivery throughout the application lifecycle, these CICS tools provide increased productivity in the areas of source code analysis and design, development debugging, testing, abend analysis, performance management, deployment to production, and service management. In these ways, they help an enterprise realize its newest and most critical business strategies.

5.2 Procedural application development lifecycle

As shown in Figure 5-1 on page 113, the CICS application development lifecycle, such as can occur following a system modification request or request for new system development, is organized into phases:

- ▶ Application analysis and design
- ▶ Development and build
- ▶ Functional testing
- ▶ Performance testing and tuning
- ▶ Deployment to production
- ▶ Service delivery management

5.2.1 Application analysis and design

When there are new requirements to modify an existing system or to build a new one, IT personnel must analyze the requirements and provide a solution design. The people typically involved in this phase are the architect, business analyst, and development lead, who collectively carry out the following activities:

- ▶ *Analyze* existing application structures, flows, and dependencies to determine where changes are required and how they should be made, and project the impact of such changes. There are two types of analysis:
 - Dynamic application analysis: This analysis is performed by tracing an application's control flow and watching it as it executes. The analysis occurs in real time while the application executes dynamically on a given CICS TS region. It also examines variable values in program storage, file buffers, database records, and temporary storage. This type of real-time monitoring at the source-code level is unrivaled for solving the most difficult software analysis problems and is invaluable for corrective maintenance.
 - Static application analysis: The application source code is analyzed to build structured top-down diagrams and bottom-up element reports. Also created is an easily searchable dictionary that contains all of the application semantics or metadata, which helps facilitate understanding of the existing application code.
- Both types of analysis are required to increase the speed of maintenance of existing applications and also add new functionality.
- ▶ *Design* the new systems by using methods such as *unified language models*, which can be helpful in documenting the design of the new systems, communicating the design to developers for implementation, and facilitating the development cycle by using model-driven techniques that can generate code directly from the models.

Several tools can help carry out these activities, including the following tools:

- ▶ CICS Interdependency Analyzer
- ▶ Rational Assets Analyzer
- ▶ Rational Software Architect

CICS Interdependency Analyzer

CICS Interdependency Analyzer (IBM CICS IA®) is a discovery tool for CICS TS that can be used for dynamic application analysis. It captures application interdependency information while CICS is running. By using CICS IA, it is possible to achieve better reuse, management, and control of your applications through improved understanding of an even wider range of CICS applications, resource flows, and resource inter-relationships.

Rational Assets Analyzer

Rational Assets Analyzer is used as a static code analysis tool for application discovery, code comprehension, and visual understanding of programs. It you follow the trail of a data value as it proceeds from variable to variable to file to variable to database across an application. Rational Asset Analyzer is especially helpful when compared to the time and effort required to accomplish the same tasks by using ISPF and source listings. Rational Asset Analyzer can be integrated with Rational Team Concert source code management to ensure ongoing source code analysis as part of your software development lifecycle.

Rational Software Architect

Rational Software Architect is an integrated design and development tool that is used to rapidly create, evaluate, and communicate software architectures and designs. Rational Software Architect leverages model-driven development (MDD) using Unified Modeling Language (UML) to create well-architected applications and services. Rational Software Architect has the following features that are particularly relevant to MDD:

- ▶ UML 2.0 editor with refactoring support
- ▶ Support for UML 2.0 profiles
- ▶ Patterns infrastructure with pattern library
- ▶ Transformation infrastructure with sample transformations

Patterns, profiles, and transformations together provide the capabilities that are required to customize Rational Software Architect and Rational Developer for System z to support the automation of a development process. Rational Software Architect is the appropriate tool for architects and designers. For more information, see the Rational Software Architect information center:

http://pic.dhe.ibm.com/infocenter/rsahelp/v8/topic/com.ibm.rsa_base.nav.doc/topics/crootintro_rsa_base.html

5.2.2 Development and build

This phase covers the technical development and construction of the CICS systems. The basic activities are to develop, build, and unit test new or modified components to ensure they are working properly before moving them to the functional testing stage. Because modifications are being performed to the existing source code, it is crucial to track those modifications and to keep a history of them, so source code version control is important.

The roles that are typically involved in this phase are the development lead and the developers. The following tools can assist during this phase:

- ▶ Rational Developer for System z
- ▶ Rational Team Concert
- ▶ IBM Debug tool for z/OS

Rational Developer for System z

Rational Developer for System z is used for development, maintenance, and support of business applications running on System z. It supports analyzing, editing, and compiling or assembling source files using integrated, language-sensitive editors for COBOL, PL/I, IBM High Level Assembler (HLASM), C, C++, and Java. It supports job control language (JCL), basic mapping support (BMS), and Message Format Service (MFS). The default editing process installation is direct to z/OS partitioned data sets (PDS libraries), or you can configure the setting to access source through most of the source control management systems on the market, such as Rational Team Concert.

For the development of new systems, the UML models that are created by using Rational Software Architect (to model data structures and programs) can be generated into COBOL programs, copybooks, data definitions, and application logic. This way can help later, when using the UML profiles for COBOL applications in Rational Developer for System z. This then yields an increase in development productivity and ensures that all development activities will conform to the design specifications already outlined inside the UML models.

Source code generation in Rational Developer for System z also features the ability to create new COBOL programs that perform I/O to VSAM and QSAM files that use the new VSAM/QSAM Wizard, or the Database Application Generator functionality that helps you rapidly create a z/OS data access layer for web services (based on a relational database schema or a simplified UML model). The tool generates code for create, read, update, and delete (CRUD) operations and web services definitions.

For more information about the source code generation capability of Rational Developer for System z, see the product information center:

<http://pic.dhe.ibm.com/infocenter/ratdevz/v7r5/index.jsp?topic=%2Fcom.ibm.etools.wdz.uml.transform.doc%2Fconcepts%2Fzapgintro.html>

Rational Team Concert

Rational Team Concert is tool for team collaboration, task management, builds, version control, and configuration management that can be integrated with Rational Developer for System z. On the development and maintenance project management levels, Rational Team Concert can be used to manage development plans and schedules.

IBM Debug tool for z/OS

IBM Debug Tool for z/OS provides debugging capability for applications that are running in a variety of environments, such as IBM CICS, IBM IMS, IBM DB2 stored procedures, and IBM UNIX System Services. Rational Developer for System z and the Debug Tool for z/OS together provide a complete solution that can help reduce application development cycle times and, at the same time, help build more effective applications by unit-testing the programs before they are moved to the functional testing and regression phase.

5.2.3 Functional testing

As new code is introduced, testing new or modified functions to ensure they work properly is essential. In the case of maintenance, full regression testing (which is a type of functional testing usually performed on existing functions of a system after changes have been made) is also needed, to ensure that nothing else in the existing application is broken in the new release. By the end of this phase, a functional testing report is compiled and fed back to the development team so it can fix any issues in the application source code. Thus, both the development and testing phases become an iterative process.

During this phase, testing and development teams work together to identify and resolve all reported bugs. The primary tool that is used in this phase, Rational Development and Test Environment for System z, allows you to create a personal z/OS development and test environment on developer desktops or shared servers to test programs before starting the functional and regression phase. It allows the testing of new applications in the test environment before releasing to production, without affecting shared mainframe environments or processes, and thus frees mainframe development MIPS for production capacity. It can easily integrate with other Rational and IBM tools for additional productivity and cost savings.

5.2.4 Performance testing and tuning

Performance testing and tuning is aimed at confirming the responsiveness and stability of a system under a specific workload. It is particularly important when the applications in use involve large numbers of concurrent users and the response times must not exceed specific limits. Performance testing is an iterative process: the testing results are provided to the development team so it can pinpoint bottlenecks and then tune the source code to avoid them. The tests are then repeated to confirm that the identified issues are resolved.

Performance testing and tuning should iterate until the requirements are met. Roles involved during this phase are performance testers, developers, and system programmers. The following tools are ideal for use in performance testing and tuning:

- ▶ Workload Simulator for z/OS
- ▶ Application Performance Analyzer for z/OS
- ▶ CICS Performance Analyzer for z/OS

Workload Simulator for z/OS

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

Workload Simulator Test Manager (WTM) guides the user through the test process. It can help users develop and manage test cases, automate test runs, and analyze results. For even more flexibility, WTM offers various modes of operation that control the amount interaction the user can observe through Workload Simulator.

Application Performance Analyzer for z/OS

Application Performance Analyzer for z/OS provides functions that help isolate application performance problems. It helps you drill down to the application-source level to find performance bottlenecks that affect online transaction response times, and assists in reducing batch application execution times. Application Performance Analyzer supports Assembler, C, C++, CICS, COBOL, DB2, IMS, Java, PL/I, WebSphere MQ, and WebSphere Application Server.

CICS Performance Analyzer for z/OS

CICS Performance Analyzer for z/OS helps programmers and systems staff identify constraints and improve the performance of their CICS systems. CICS Performance Analyzer for z/OS is a powerful reporting tool that analyzes the System Management Facilities (SMF) records created by the CICS Monitoring Facility (CMF), CICS Statistics, and CICS Server Statistics, and also SMF data from the related subsystems (DB2 and WebSphere MQ). CICS Performance Analyzer for z/OS produces a wide range of reports and extracts to help tune and manage CICS systems.

5.2.5 Deployment to production

In this phase, system programmers migrate all CICS resources and definitions from the testing environment to the production environment, where they will be available to users. Also in this phase, system programmers create and start the CICS regions that will host the applications. Tools to assist with these activities include CICS Deployment Assistant and CICS Configuration Manager.

CICS Deployment Assistant

CICS Deployment Assistant is designed to help you to discover, model, visualize, and deploy new and existing CICS regions. With CICS Deployment Assistant you can *visualize* and *control* all your CICS resources, and automate and standardize your CICS environment. CICS Deployment Assistant is an interactive control system that allows CICS systems programmers to create and maintain a CICS environment easily.

CICS Configuration Manager

CICS Configuration Manager is a tool for administering and maintaining CICS resource definitions. It provides comprehensive audit, reporting, and lifecycle change management control facilities that are useful in building, managing, and deploying complex mainframe CICS applications. CICS Configuration Manager can be used to promote sets of CICS definitions from test regions into production regions in a controlled and manageable manner, thus ensuring successful promotions to the production environment without the need to manually update and test the resource definitions after they are in production.

5.2.6 Service delivery management

After an application is up and running and performing well, service delivery management processes help ensure it continues to run smoothly. Service delivery management is an integrated set of activities aimed at maintaining the availability, cost, and quality of deployed IT services.

Responsibility for service management usually falls on system programmers, but again, there are tools that can help. Products such as IBM Tivoli® OMEGAMON XE for CICS on z/OS and Fault Analyzer for z/OS can help system programmers significantly.

For more information about service management processes, in particular those related to CICS on System z, see the white paper *IBM service management for CICS with System z tools*, which is available at this website:

ftp://public.dhe.ibm.com/software/http/cics/tools/WSW11330-USEN-00_cicstools_itsm_WP_0619D.pdf

Tivoli OMEGAMON XE for CICS on z/OS

OMEGAMON XE for CICS on z/OS provides system monitoring and performance management across System z, including CICS subsystems, with an enhanced, easy-to-use CICS Explorer plug-in. It provides performance and availability information that can be used to avoid downtime and performance issues. OMEGAMON XE provides insight into ongoing issues, which allows the user to choose an informed course of action.

Tivoli Enterprise Monitoring Server gathers data from the Tivoli Enterprise Monitoring Agent, which is installed in a CICS sub-system, and acts as a collection and control point for alerts. Monitoring data and real-time performance data can be displayed with the CICS Explorer Tivoli OMEGAMON XE plug-in.

As Figure 5-3 shows, one difference between OMEGAMON XE for CICS and CICS Performance Analyzer is that OMEGAMON XE relies on real-time monitoring using data that is collected from Tivoli Enterprise Monitoring Agent; CICS Performance Analyzer gathers performance data on a transactional basis from the CICS Service Management Facility (SMF). As OMEGAMON XE gathers real-time data, it can monitor the availability of separate CICS resources and send alerts when one of them is down.

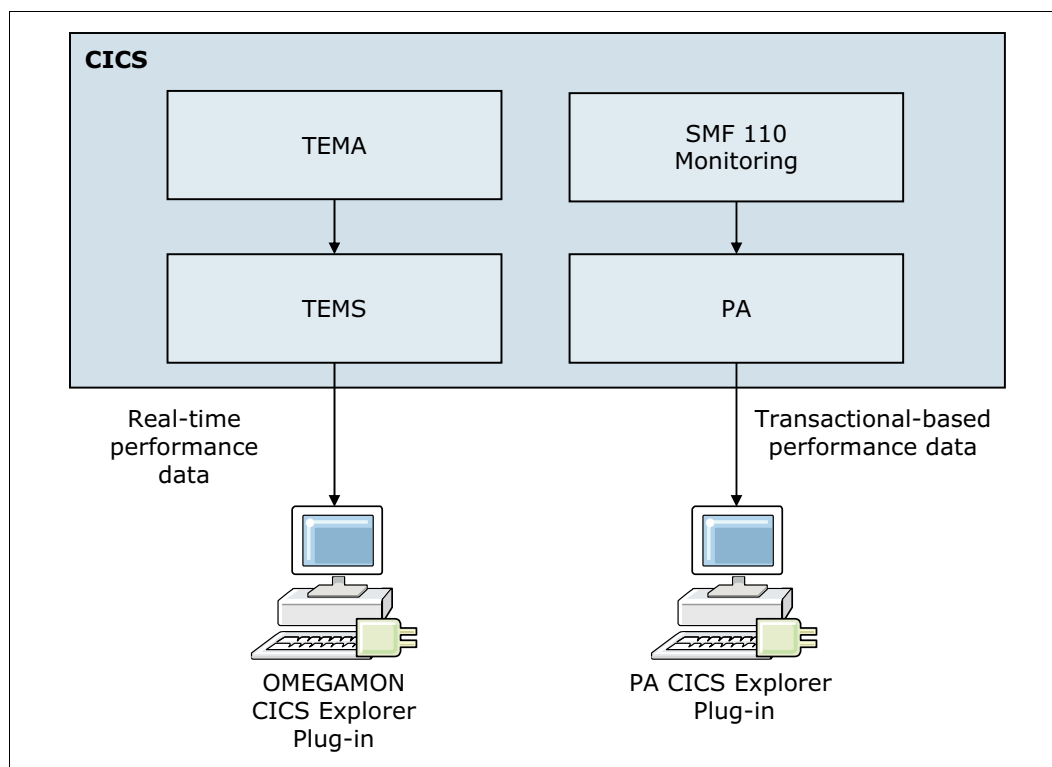


Figure 5-3 Performance data collection in OMEGAMON XE versus CICS PA

The most significant abilities of OMEGAMON XE for CICS can be summarized as follows:

- ▶ Improved time-to-resolution for problems (and higher overall system availability) because fewer steps are needed to find the causes of CICS performance issues
- ▶ Improved visibility of the CICS environment using the CICS Explorer Tivoli OMEGAMON XE plug-in
- ▶ Simplified reporting through grouping CICS regions into CICSplex reporting groups, which can be defined using simple rules or with CPSM definitions, if they are available
- ▶ Execution of the Service Level Analysis component of OMEGAMON XE for CICS on z/OS on a zIIP speciality processor

IBM Fault Analyzer for z/OS

Fault Analyzer for z/OS provides the information that is required to determine the cause, and assist with the resolution of, application and subsystem failures. You can use this tool to assist in composite-application abend analysis, where it can help repair failures quickly by gathering information about an application and its environment at the moment of failure.

Fault Analyzer for z/OS has two modes of operation:

- ▶ **Runtime analysis**

When a CICS transaction fails, Fault Analyzer produces a report that describes the failure, finds the line of code that caused the failure, formats the variables in use at the time, displays the panel involved, and extracts the message and code details from the appropriate publications. All of this information enables you to more quickly identify the root cause of the failure. The abend details are also saved in a direct access storage device (DASD) file for subsequent analysis or online viewing.

- ▶ **SDUMP processing**

If an error in a CICS region causes an SDUMP to be created, Fault Analyzer can analyze it at a system level, showing domain information such as kernel, dispatcher, program manager, and transaction manager. The dump information is displayed in a series of panels that have point-and-shoot hotspots, enabling fast and easy navigation from one functional area to another.

5.3 Java application development and deployment lifecycle

CICS provides the tools and runtime environment to develop and run Java applications in a Java virtual machine (JVM) that is under the control of a CICS region. Java applications can interact with CICS services and applications written in other languages.

CICS uses the IBM 64-bit SDK for z/OS, Java Technology Edition. The SDK contains a Java Runtime Environment that supports the full set of Java APIs and a set of development tools. Another option to use special processors for Java workloads is available in certain System z hardware. These processors are the IBM System z Application Assist Processor (zAAP) and the IBM System z Integrated Information Processor (zIIP) with zAAP on zIIP capability. Both processors can provide additional processing capacity to run eligible Java workloads at a reduced cost. You can find more information about Java on the z/OS platform and download the 64-bit version of the SDK from the following website:

<http://www.ibm.com/servers/eserver/zseries/software/java/>

5.3.1 Basic concepts of Java development in CICS

For the purpose of Java development, CICS provides an Eclipse-based tool and two runtime environments. We describe each of them in detail and explain the basic concepts of the OSGi framework, which underlies CICS Java development.

CICS Explorer SDK

The CICS Explorer SDK is a freely available download for Eclipse-based integrated development environments (IDEs). The SDK provides support for developing and deploying applications that comply with the OSGi Service Platform specification.

The OSGi Service Platform provides a mechanism for developing applications using a component model and deploying those applications into a framework as OSGi bundles. An OSGi bundle is the unit of deployment for an application component and contains version control information, dependencies, and application code. The main benefit of OSGi is that you can create applications from reusable components that are accessed only through well-defined interfaces called OSGi services. You can also manage the lifecycle and dependencies of Java applications in a granular way.

The CICS Explorer SDK supports developing Java applications for any supported release of CICS. The SDK includes the Java CICS (JCICS) library of classes (to access CICS services) and examples to help you get started with developing applications for CICS. You can also use the tool to convert existing Java applications to OSGi.

JVM server

The JVM server is the strategic runtime environment for Java applications in CICS. A JVM server can handle multiple concurrent requests from different Java applications in a single JVM. It reduces the number of JVMs that are required to run Java applications in a CICS region.

To use a JVM server, Java applications must be *threadsafe* and must comply with the OSGi specification. It is the preferred method for running Java workloads in a CICS region and provides the following benefits:

- ▶ Run more than one Java application in a JVM server, which can make running and managing JVMs in a CICS region easier.
- ▶ Run eligible Java workloads on zAAPs, reducing the cost per transaction.
- ▶ Run different types of work in a JVM server, including threadsafe Java programs and web services.
- ▶ Manage the lifecycle of applications in the OSGi framework without restarting the JVM server.
- ▶ More easily port Java applications that are packaged by using OSGi between CICS and other platforms.

The JVM server model is essentially a single, long-running JVM placed under the control of CICS. Its defining principle is the ability to run many CICS tasks concurrently within the same JVM (Figure 5-4).

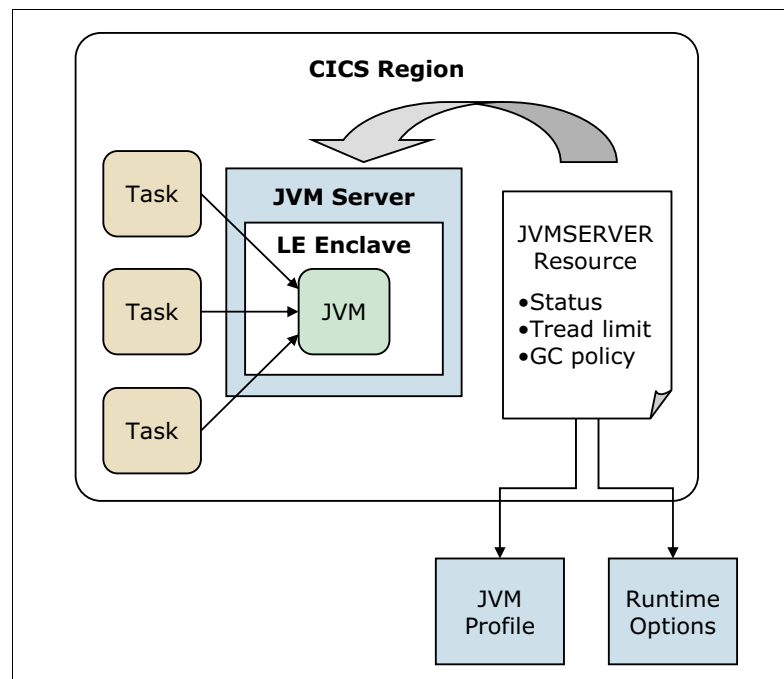


Figure 5-4 JVMSERVER resource

In this model, each CICS task that requires Java support is switched onto a thread within the JVM. The thread is taken under the control of CICS dispatcher and becomes a new CICS OTE TCB type called a T8.

Not only can multiple CICS transactions target the same JVM and run concurrently, but every CICS task in this environment is capable of executing Java code, and perhaps more important, of making CICS calls and accessing CICS resources and data from Java.

As shown in Figure 5-4 on page 123, the JVMSERVER resource acts as a central point for configuration, provides centralized control, and manages the interface between CICS and the underlying JVM. Additional fine tuning and low-level configuration of both the JVM and the underlying Language Environment enclave can be achieved by using the Language Environment runtime options and JVM profile artifacts. Each time the JVMSERVER resource is enabled, it rereads configurations from these files, allowing customers to change the Language Environment enclave characteristics or the JVM settings. Typically, customers want to generate Language Environment storage reports, or tune the JVM heap size for garbage collection.

The OSGi Service Platform

The OSGi Service Platform provides a mechanism for developing applications by using a component model and deploying those applications into an OSGi framework. The OSGi architecture is separated into a number of layers that provide benefits for creating and managing Java applications.

The OSGi framework is at the core of the OSGi Service Platform specification. The OSGi framework is initialized when a JVM server starts. OSGi for Java applications provides the following major benefits:

- ▶ Java applications are more portable, easier to re-engineer, and more adaptable to changing requirements.
- ▶ The plain old Java object (POJO) programming model can be used, giving you the option of deploying an application as a set of OSGi bundles with dynamic lifecycles.
- ▶ You can more easily manage and administer application bundle dependencies and versions.

OSGi provides a set of Java APIs to handle the lifecycle and the versioning of bundles. Bundles can be dynamically installed and uninstalled and their exposed services registered or made unavailable. Multiple bundle versions can coexist in the same OSGi framework, and bundle upgrades can be transparent to the user applications, because continuity of services can be granted during the upgrade.

The OSGi architecture has the following layers:

- ▶ Modules layer
- ▶ Lifecycle layer
- ▶ Services layer

Modules layer

The unit of deployment in this layer is an OSGi bundle. The modules layer is where the OSGi framework processes the modular aspects of a bundle. The metadata that enables the OSGi framework to do this processing is provided in a bundle manifest file.

One key advantage of OSGi is its class loader model, which uses the metadata in the manifest file. There is no global class path in OSGi. When bundles are installed into the OSGi framework, their metadata is processed by the module layer and their declared external

dependencies are checked against the exports and version information declared by other installed modules.

The OSGi framework works out all of the dependencies and calculates the independent required class path for each bundle. This approach resolves the shortcomings of plain Java class-loading by ensuring that the following requirements are met:

- ▶ Each bundle provides visibility only to Java packages that it explicitly exports.
- ▶ Each bundle declares its package dependencies explicitly.
- ▶ Packages can be exported and imported at specific versions, to or from a specific range of versions.
- ▶ Multiple versions of a package can be available concurrently to different clients.

Lifecycle layer

The bundle lifecycle management layer in OSGi enables bundles to be dynamically installed, started, stopped, and uninstalled, *independently* from the lifecycle of the JVM. The lifecycle layer ensures that bundles are started only if all of their dependencies are resolved, reducing the occurrence of `ClassNotFoundException` exceptions at run time. If there are unresolved dependencies, the OSGi framework reports the problem and does not start the bundle.

Each bundle can provide a bundle activator class, identified in the bundle manifest, for the framework to use to start and stop events.

Services layer

The services layer in OSGi intrinsically supports a service-oriented architecture through its non-durable service registry component. Bundles publish services to the service registry, and other bundles can discover these services from the service registry. These services are the primary means of collaboration between bundles.

An OSGi service is a plain old Java object (POJO), published to the service registry under one or more Java interface names, with optional metadata stored as custom properties (name/value pairs). A discovering bundle can look up a service in the service registry by an interface name, and can potentially filter those services based on the custom properties.

Services are fully dynamic, and typically have the same lifecycle as the bundle that provides them.

How CICS exploits OSGi

CICS uses the Equinox version 3.6.1 implementation of the OSGi framework, which supports version 4 of the OSGi Service Platform specification. Through OSGi, CICS implements the dynamic deployment of Java applications into the JVM, as opposed to the traditional class path method, which requires new application classes to be added to the class path and the JVM to be restarted to install or update Java applications.

Bundles are the resources that is used by CICS to control the lifecycle (install, uninstall, start, stop, register services, unregister services) of OSGi bundles. The new CICS Explorer SDK allows you to specify CICS main classes into the OSGi bundle manifest file. When an OSGi bundle is deployed into a JVM server, CICS registers the specified main classes into the OSGi service registry, and the new services become dynamically available to be used by other bundles in the framework or by CICS when a CICS Java program is linked.

Figure 5-5 illustrates how CICS binds an OSGi service when a CICS Java program is run.

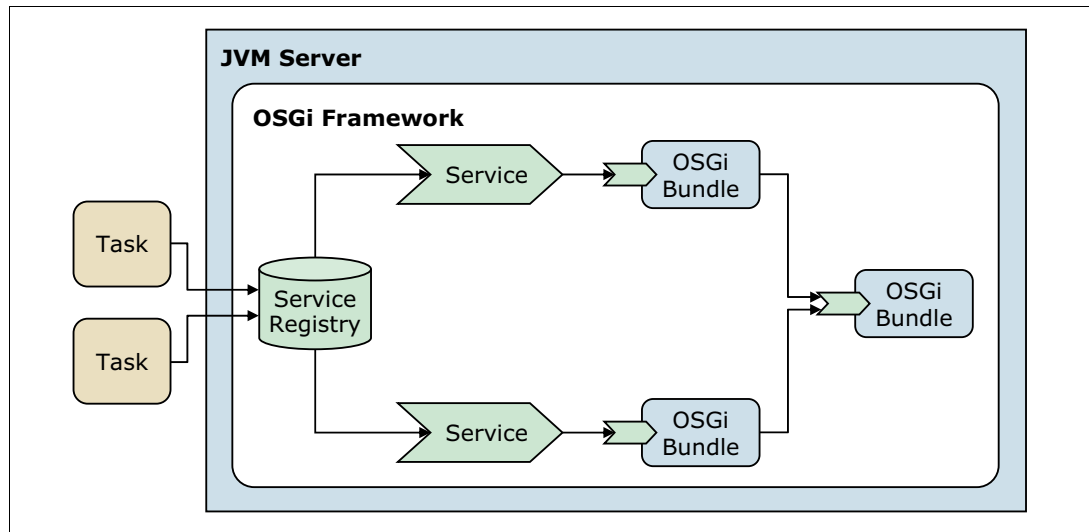


Figure 5-5 The OSGi bundle is looked up in the service registry when a Java program is invoked

CICS relies on OSGi to provide a smooth and continuous Java application upgrade path. Duplicate disabled services, installed by multiple versions of the same bundle, automatically become active when the currently active service is unregistered.

With CICS Explorer SDK, you can specify aliases for duplicate services, so that multiple versions of the same bundle can be simultaneously operational on the same JVM server.

OSGi support in CICS TS 4.2 makes the JVM server a flexible and maintainable platform for the deployment and the execution of available and reliable Java applications.

5.3.2 Java application development on CICS

The CICS Explorer SDK plug-in delivers an end-to-end experience for the development, deployment, and management of CICS Java applications. When the plug-in is installed into the appropriate prerequisite Eclipse-based IDE, users can take advantage of the concise documentation, built-in examples, and simple deployment and have a CICS application running in minutes. Furthermore, the full-function CICS Explorer perspectives allow administrators to create and install the necessary JVM server or pooled JVM environment and CICS resource definitions.

An application that runs in a JVM server is developed as one or more Eclipse plug-in projects, each of which produces an OSGi bundle. An accompanying CICS bundle project is used to package, deploy, and install the finished application into a particular CICS region or CICSplex. This project allows different combinations of code and configurations (for example, a JVM server name) to be used in certain environments. It also allows platform-independent code or libraries to be shared between CICS and other OSGi-enabled server runtime environments such as WebSphere Application Server.

Use the following steps to get a CICS Java application to run:

1. Define your target platform (Figure 5-6).

Select the appropriate CICS TS runtime target to use. This step ensures you do not accidentally use an API that is not available in the intended CICS runtime environment. Optionally, you can customize the target platform by adding any IBM, locally written, or third-party library dependencies you need, such as WebSphere MQ.

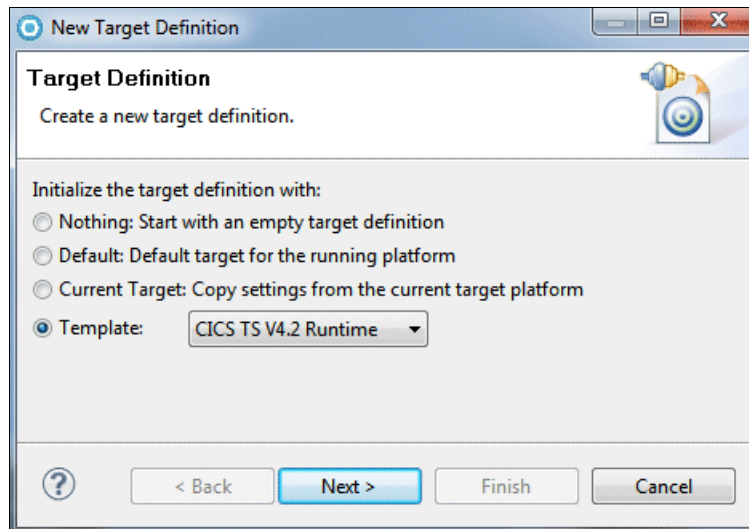


Figure 5-6 Defining the target platform

2. Create your Eclipse plug-in project (Figure 5-7).

Use the standard New Project wizard to create your own OSGi bundle, or choose one of the samples provided. Add any required packages to the OSGi bundle manifest. Your previous choice of target platform ensures only that packages in your CICS server environment are available.

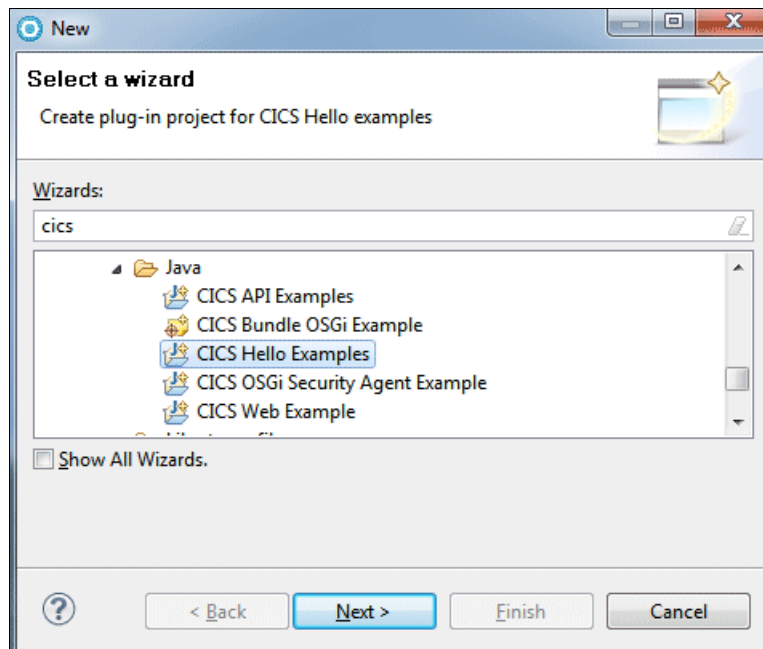
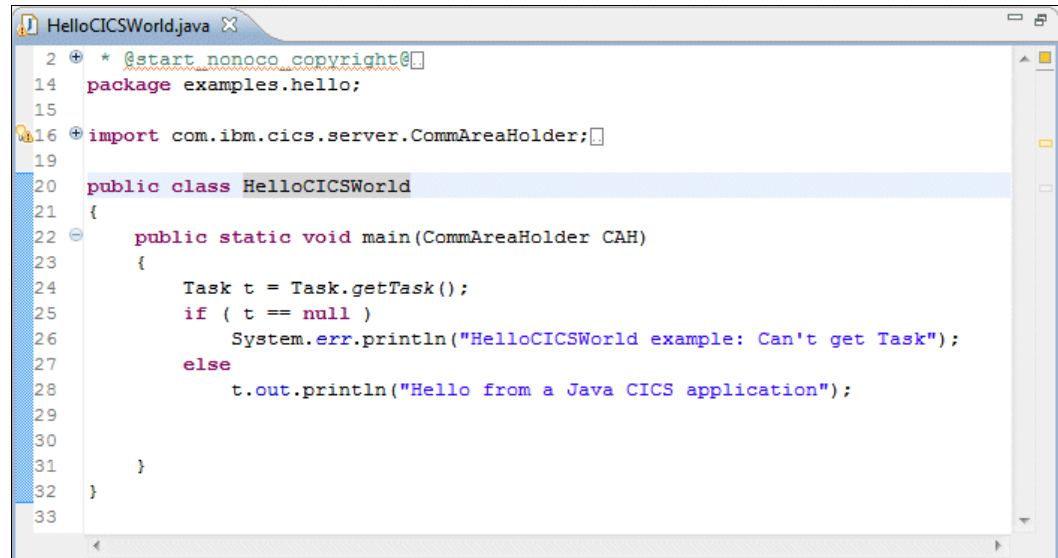


Figure 5-7 Creating a new CICS Java example

3. Write your CICS program (Figure 5-8).

Create a standard CICS main class POJO as with previous versions of CICS. Direct access to Javadoc for all JCICS APIs is available from within the SDK. Declare the new class in the OSGi manifest so that CICS can find it at run time.



```
2  * @start nonoco copyright@
14 package examples.hello;
15
16 import com.ibm.cics.server.CommAreaHolder;
19
20 public class HelloCICSWorld
21 {
22     public static void main(CommAreaHolder CAH)
23     {
24         Task t = Task.getTask();
25         if ( t == null )
26             System.err.println("HelloCICSWorld example: Can't get Task");
27         else
28             t.out.println("Hello from a Java CICS application");
29     }
30 }
31
32
33
```

Figure 5-8 Sample Java code to be executed on CICS

4. Create your CICS bundle (Figure 5-9).

Create a new CICS bundle project then use the Include OSGi Project in Bundle wizard to package your Java application. Use this wizard to decide which Eclipse plug-in projects to include and which JVM server will be used during run time.

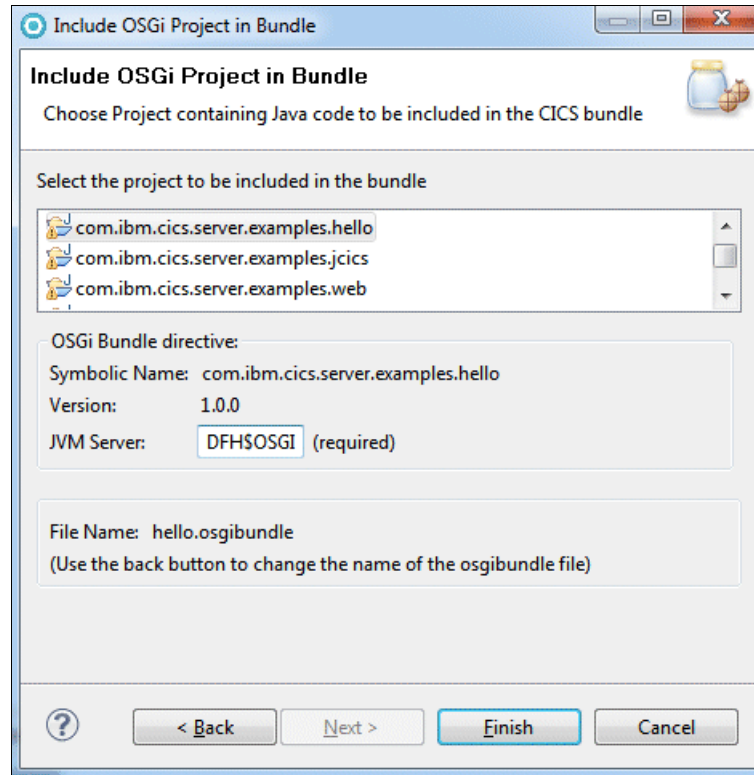


Figure 5-9 OSGi bundle packaging

5. Deploy your CICS application (Figure 5-10).

In the pop-up menu for your CICS bundle project, select **Export to z/OS UNIX File System as Bundle Project** to send it over to CICS and install it with a CICS bundle definition in your CSD or CICSplex System Manager BAS. Open the new OSGi Bundles and OSGi Services views to determine whether the application is ready to run. If any problems occur, check the JVM trace file by using the z/OS UNIX Files view for any error messages.

The screenshot displays the CICS Explorer interface with the following views:

- Region View:** Shows one region, IYCWEGG1, with bundle DFH\$OSGB, status ENABLED, and install time 23-Jun-2011 14:33:22.
- Bundle Parts View:** Shows three bundle parts for DFH\$OSGB: hello, jcics, and web, all enabled.
- OSGi Bundles View:** Shows three bundles: com.ibm.cics.server.examples.hello, com.ibm.cics.server.examples.jcics, and com.ibm.cics.server.examples.web, all active.
- OSGi Services View:** Shows nine services, all active, including examples.ProgramControl.ClassOne through examples.hello.HelloWorld.

Region	Name	Status	Install Time
IYCWEGG1	DFH\$OSGB	✓ ENABLED	23-Jun-2011 14:33:22

Region	Bundle	Bundle Part	Enable Status	Meta Data File	Part Class	Part Type
IYCWEGG1	DFH\$OSGB	hello	ENABLED	hello.osgibundle	DEFINITION	http://www.ib...
IYCWEGG1	DFH\$OSGB	jcics	ENABLED	jcics.osgibundle	DEFINITION	http://www.ib...
IYCWEGG1	DFH\$OSGB	web	ENABLED	web.osgibundle	DEFINITION	http://www.ib...

JVM Server	Symbolic Name	Version	State	Bundle	Bundle Part
DFH\$JVMS	com.ibm.cics.server.examples.hello	1.0.0	✓ ACTIVE	DFH\$OSGB	hello
DFH\$JVMS	com.ibm.cics.server.examples.jcics	1.0.0	✓ ACTIVE	DFH\$OSGB	jcics
DFH\$JVMS	com.ibm.cics.server.examples.web	1.0.0	✓ ACTIVE	DFH\$OSGB	web

JVM Server	Service Name	OSGi Bundle	Version	Service Status	Bundle
DFH\$JVMS	examples.ProgramControl.ClassOne	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.ProgramControl.ClassTwo	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.ProgramControl.ClassThree	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.ProgramControl.ClassFour	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.TDQ.ClassOne	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.TSQ.ClassOne	com.ibm.cics.server.examples.jcics	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.Web.Sample1	com.ibm.cics.server.examples.web	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.hello.HelloCICSWorld	com.ibm.cics.server.examples.hello	1.0.0	ACTIVE	DFH\$OSGB
DFH\$JVMS	examples.hello.HelloWorld	com.ibm.cics.server.examples.hello	1.0.0	ACTIVE	DFH\$OSGB

Figure 5-10 OSGi Bundles and OSGi Services views inside CICS Explorer

If you make any changes, discard the CICS bundle and repeat step 5. If you want to share your application with colleagues, use the Eclipse team support functions to check the relevant projects into Rational Team Concert, SVN, or any standard repository.

The Eclipse-based development environment offers a number of options for customers who are migrating Java applications from earlier versions of CICS, for those who want to maintain coexistence with the pooled JVM run time, or for others who are simply seeking to take advantage of the improved experience of using OSGi.

Applications that are destined for JVM server must be deployed as OSGi bundles. Eclipse offers the following techniques to help:

- ▶ By clicking a few options, you can convert your existing Java project into a plug-in project. The Eclipse Plug-in Development Environment (PDE) checks for unsupported dependencies. You can still run the deployed code in a pooled JVM by adding the OSGi bundle JAR files to the class path in your JVM profile.
- ▶ Use the new project wizard to take an existing JAR file and automatically “inject” the required OSGi manifest file. The code does not need to be recompiled, reducing the need for extensive retesting. Use this technique if you want to redeploy to both pooled JVMs and JVM servers.
- ▶ Use the new project wizard to *wrapper* an existing JAR file and automatically add the required OSGi manifest file. Use this approach if licensing or other restrictions prevent artifact modification.

Also included with the SDK is an Eclipse target platform template for every supported version of CICS TS, defining the correct level of JCICS and JRE. This template ensures that application developers use only those Java APIs that are supported in the intended CICS server. Most important, whether your plans to move to JVM server are long-term or short-term, you can start taking advantage of OSGi today.

5.3.3 Debugging and troubleshooting Java applications running on CICS

The JVM in CICS supports the Java Platform Debugger Architecture (JPDA), which is the standard debugging mechanism that is provided in the Java 2 Platform. This architecture provides a set of APIs that allow the attachment of a remote debugger to a JVM.

You can use any tool that supports JDPA to debug a Java application running in CICS. For example, you can use the Java Debugger (JDB) that is included with the Java SDK on z/OS. However, to attach a JPDA remote debugger, you must set some options in the JVM profile.

To debug Java applications, complete the following steps:

1. Ensure the JVMSERVER resource that is used by your Java application is disabled, or not yet installed.
2. Add the following lines to the JVM profile:
-Xdebug
-Xrunjdwp:transport=dt_socket,server=y,address=27508,suspend=n
Change the address= option to an IP port number on z/OS that is not already being used, that the default CICS region user ID is authorized to open, and to which your workstation can connect.
3. Enable the JVMSERVER resource.
4. Install the BUNDLE resource that contains your CICS Java application.
5. In CICS Explorer SDK, switch to the debug perspective by selecting **Window → Open Perspective → Other → Debug → OK**.
6. Select **Run → Debug Configurations**.

7. In the Debug Configurations window (Figure 5-11), create a new entry under Remote Java Application, filling in the **Project** (your OSGi plug-in project that contains the source to debug), **Host** and **Port** fields (your z/OS system host name, and the port entered in step 2 on page 131).

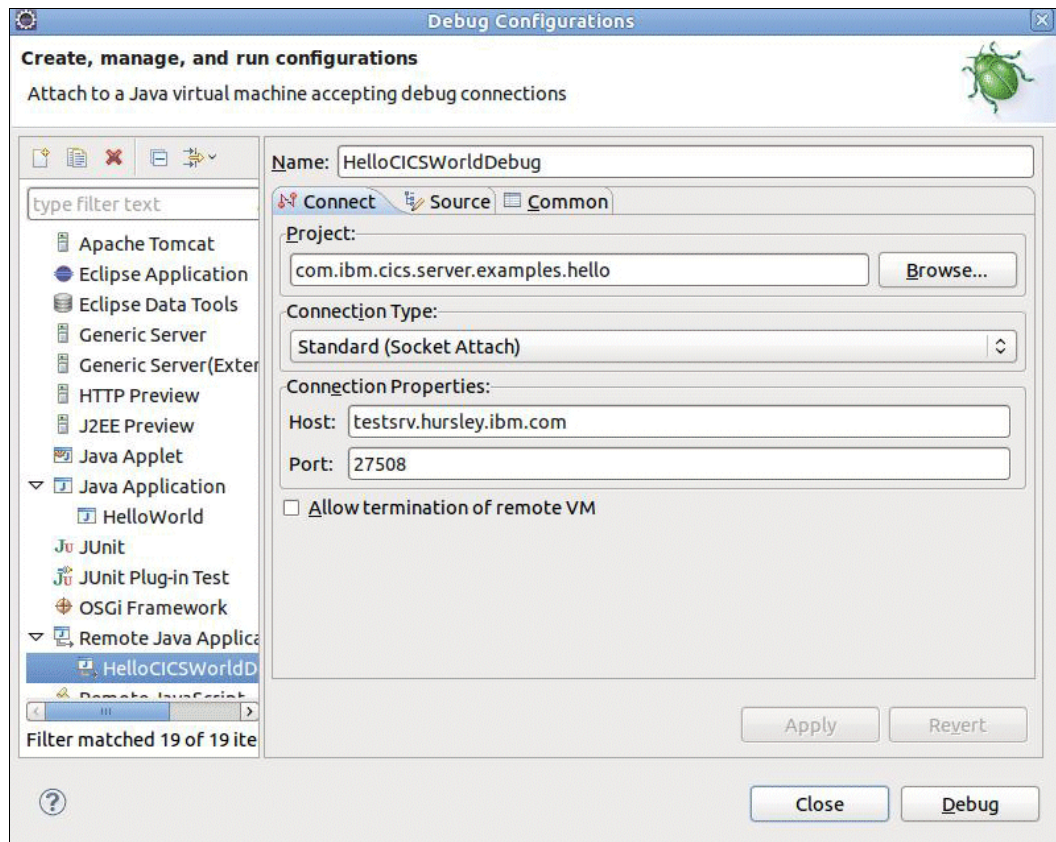


Figure 5-11 Remote debugging for CICS Java application

8. Click **Debug** to start the debug session.
If the Failed to connect to remote VM message is displayed, determine and confirm if the JVMSERVER resource is enabled and that your workstation can reach the z/OS host.
9. Switch to the Plug-in perspective, navigate to the Java class you need to debug, and open the editor for it.
10. Double-click in the margin to create a breakpoint. Each break point appears as a blue dot in the margin.
11. Run your CICS Java application.
12. When the breakpoint is reached, the CICS Explorer SDK will switch automatically to the Debug perspective and you can now step through your code, watch variables, and so forth. If you hover the mouse over a variable in the source, a pop-up window displays the current value.

For more information about remote debugging of Java applications, see this website:

https://www.ibm.com/developerworks/mydeveloperworks/blogs/cicsdev/entry/configuring_cics_and_cics_explorer_sdk_to_remotely_debug_java_applications43?lang=en

5.3.4 CICS diagnostics tools for Java

Many of the usual sources of CICS diagnostic information contain information that applies to Java applications. In addition to the information that is supplied by CICS, a number of interfaces are available that are specific to the JVM and that you can use for problem determination.

When the first JVM is started in a CICS region after initialization, CICS issues message DFHSJ0207, showing the version of Java that is being used.

The Java SDK provides diagnostic tools and interfaces that give you more detailed information about what is happening in the JVM. Messages and diagnostic information from the JVM are written to the `stderr` log file for the JVM. If you encounter a Java problem, always consult this file. For example, if CICS issues a message to indicate that the JVM has abended, the `stderr` log file is the primary source of diagnostic information. Controlling the location for JVM stdout, stderr, and dump output tells you how to control the location of output from the JVM, and how to redirect messages from JVM internals and output from Java applications running in a JVM.

When you develop Java applications for CICS, be sure to consider the requirements for thread safety and transaction isolation in CICS. If a Java application works correctly on its first use but does not behave correctly on subsequent uses, the problem is likely because of isolation issues. In this case, use the CICS JVM Application Isolation Utility as part of your problem determination work to help identify the cause of the problem.

JVM diagnostic tools

You may use the following JVM diagnostic tools and interfaces:

- ▶ Activating and managing tracing for JVM servers: This tool describes how you can use the component tracing that is provided by the CETR transaction to trace the lifecycle of the JVM server and the tasks running in it. JVM servers do not use auxiliary or GTF tracing. Instead, the tracing is written to a file on zFS that is uniquely named for each JVM server.

Additional information is available at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhpj_trace_jvmserver.html

- ▶ Defining and activating tracing for pooled JVMs: This tool describes how you can use the internal trace facility of a pooled JVM through the interfaces provided by CICS. The internal trace facility can provide detailed tracing of entry, exit, and event points within the JVM. This information is output as CICS trace.

Additional information is available at this website:

http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhs1_trace_jvm.html

OSGi diagnostic files

The OSGi framework produces diagnostic files in zFS that you can use to help troubleshoot problems with OSGi bundles and services in a JVM server.

OSGi cache

The OSGi cache is in the following directory of the JVM server:

`$WORK_DIR/applid/jvmserver/configuration/org.eclipse.osgi`

In this path, `$WORK_DIR` is the working directory of the JVM server, `applid` is the CICS APPLID, and `jvmserver` is the name of the JVMSERVER resource. The OSGi cache contains

framework metadata and other information that is required to run the framework. The cache is replaced when the JVM server starts.

OSGi logs

If an error occurs in the OSGi framework, an OSGi log is created in the following directory of the JVM server; and .log is the file extension:

`$WORK_DIR/applid/jvmserver/configuration/`

The OSGi framework continues to write to the log file until it reaches 1000 KB in size. After this, the OSGi framework creates another log file to write out further error messages. You can have up to ten log files in the directory. After the tenth log file is full, the OSGi framework overwrites the oldest log file.

5.3.5 Profiling and tuning CICS Java applications

IBM Monitoring and Diagnostic Tools for Java - Health Center is a low-overhead monitoring tool. It runs along with a Java application, with a small impact on the application's performance. Health Center monitors several application areas, using this information to provide guidance and analysis that help you improve the performance and efficiency of your application. Health Center can save the data that is obtained from monitoring an application and load it again for analysis at a later date.

As shown in Figure 5-12 on page 135, the Health Center profiling view provides visibility, monitoring, and profiling in the following application areas:

- ▶ **Performance**
 - **Java method profiling:** The Health Center uses a sampling method profiler to diagnose applications showing high CPU usage. Its low overhead means there is no need to specify in advance which parts of the application to monitor; the Health Center simply monitors everything. It works without recompilation or byte code instrumentation and shows where the application is spending its time, by giving full call-stack information for all sampled methods.
 - **Lock analysis:** Synchronization can be a big performance bottleneck on multi-CPU systems. Identifying a hot lock or assessing the impact that locking is having on your application is often difficult. Health Center records all locking activity and identifies the objects with most contention. Health Center analyses this information, and uses it to provide guidance about whether synchronization is affecting performance
 - **Garbage collection (GC):** The performance of GC affects the entire application. Tuning GC correctly can potentially deliver significant performance gains. Health Center identifies where garbage collection is causing performance problems and suggests more appropriate command line options.
 - **Threading:** Thread behavior in an application can have a major impact on your application's performance. Thread contention issues can often be difficult to find. Health Center identifies which threads are holding onto locks and which threads are blocked, assisting in the resolution of contention issues.
- ▶ **Memory usage**

Health Center can identify whether your application is using more memory than seems reasonable, or where memory leaks occur. It then suggests solutions to memory issues, and also Java heap sizing guidance.
- ▶ **System environment**

Health Center uses an *environment perspective* to provide details of the Java version, Java class path, boot class path, environment variables, and system properties. This

perspective is particularly useful for identifying problems on remote systems or systems where you do not control the configuration. If Health Center detects misconfigured applications, it will provide guidance on how to fix it.

- ▶ Java class loading:

Health Center provides class loading information, showing exactly when a class was loaded and whether it is cached. This information helps you determine whether your application is being affected by excessive class loading.

- ▶ File input and output

Health Center uses an *I/O perspective* to monitor application input or output (I/O) tasks as they run. You can use this perspective to monitor how many and which files are open, and to help solve problems such as when the application fails to close files.

- ▶ Object allocations

Health Center allows you to view the size, time and code location of an object allocation request that meets specific threshold criteria.

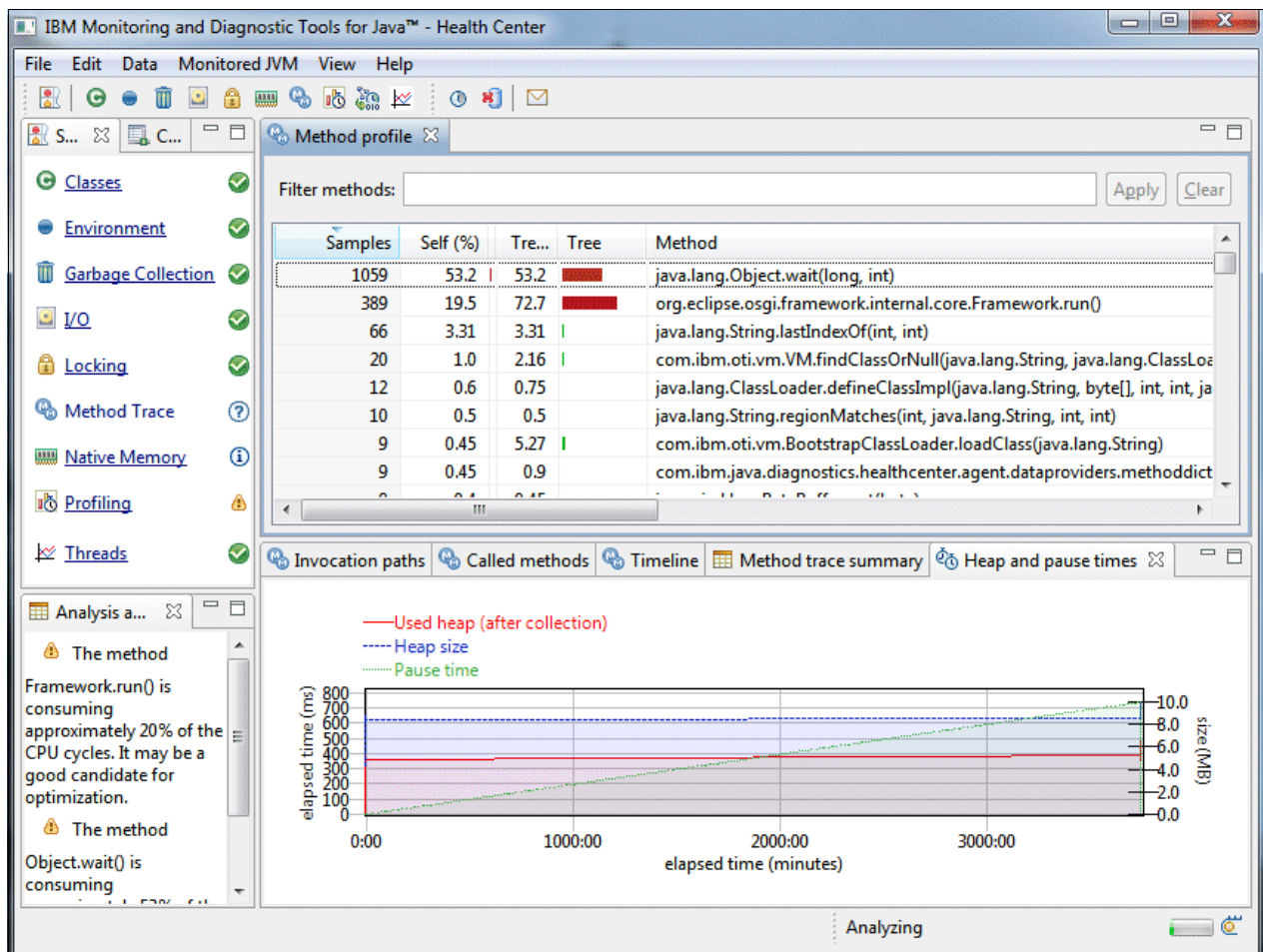


Figure 5-12 Health center profiling view

For more information about the Health Center tool, go to the following website:

<http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/>



Exploring the CICS development tools

The enterprise business paradigm continues to create more complex applications, driving application development to become more expensive.

In this chapter, we explore a comprehensive suite of powerful tools that you can use for enhancing the efficiency of your CICS application development lifecycle and, at the same time, address the growing need for cost-effective application development and testing.

By using these tools, you can more quickly optimize your application development activities and more easily identify and resolve problems. Many features in this suite of tools can help you with daily tasks; learning about these tools and their capabilities can enhance your application development skills.

6.1 Rational tools

In this section, we describe the following tools:

- ▶ Rational Asset Analyzer
- ▶ Rational Developer for System z
- ▶ Rational Development and Test Environment for System z
- ▶ Rational Team Concert

6.1.1 Rational Asset Analyzer

Rational Asset Analyzer is a static application analysis package for understanding programs, application discovery, code comprehension, and visual aspects. Rational Asset Analyzer is server-based and capable of materializing an enterprise-wide, cross-platform, and cross-application view of your software's processing semantics.

Rational Asset Integration is offered in Rational Developer for System z as an Eclipse plug-in with a subset of Rational Asset Analyzer functionality. This functionality is no cost if you own Rational Asset Analyzer and Rational Developer for System z. It has so much functionality that knowing where to start, what to use, and when to use Rational Developer versus Rational Asset Analyzer integration, and so forth can be a daunting task. We divide it into five standard Rational Asset Analyzer integration use case functions and then briefly describe each:

- ▶ Facilitating top-down program understanding
- ▶ Control flow analysis
- ▶ Data flow analysis
- ▶ Impact analysis
- ▶ Documenting your application using Rational Asset Analyzer integration

Facilitating top-down program understanding

Starting with the program source that is loaded into Rational Developer for System z, you can select to open the **Show program diagram** view. Here, you can learn about an individual program from a high level and get a better understanding of program I/O and calls to subroutines. If you are studying a CICS transaction, use the Run unit¹ diagram. If you are studying a z/OS batch job, you can use the Batch job diagram (Figure 6-1) to gain a clearer understanding of these components:

- ▶ Files
- ▶ I/O activity
- ▶ Job step dependencies
- ▶ Calls to subroutines and other run units

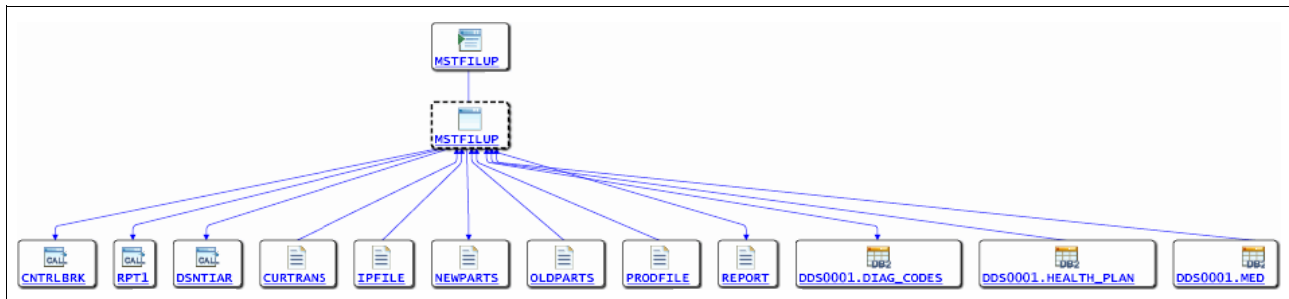


Figure 6-1 Rational Asset Analyzer integration program diagram

¹ A run unit is the operational (load module) level view starting from each program or transaction and ending at GOBACK function for that individual program.

Control flow analysis

The Asset Analyzer control flow diagram is a hierarchical chart of the program's PROCEDURE DIVISION. It is hyper-linked so that a developer can click a paragraph or section, and the editor synchronizes the code with the hierarchical chart. Figure 6-2 shows the program tree view detailing the control flow diagram synchronized to the program source file. Going from a high-level view to a low-level view, you can use the control flow diagram to understand how each program is hierarchically structured from all the PERFORM chains and other branching statements.

Most programmers learn a program in two directions: top-down and bottom-up. The control flow diagram is the top-down view. Without this technology, creating this view is difficult and slow. The Rational Developer for System z editor, which can be considered as a bottom-up view, describes the details of each paragraph. By opening both views simultaneously, you can learn the program logic easily, quickly, and with fewer passes.

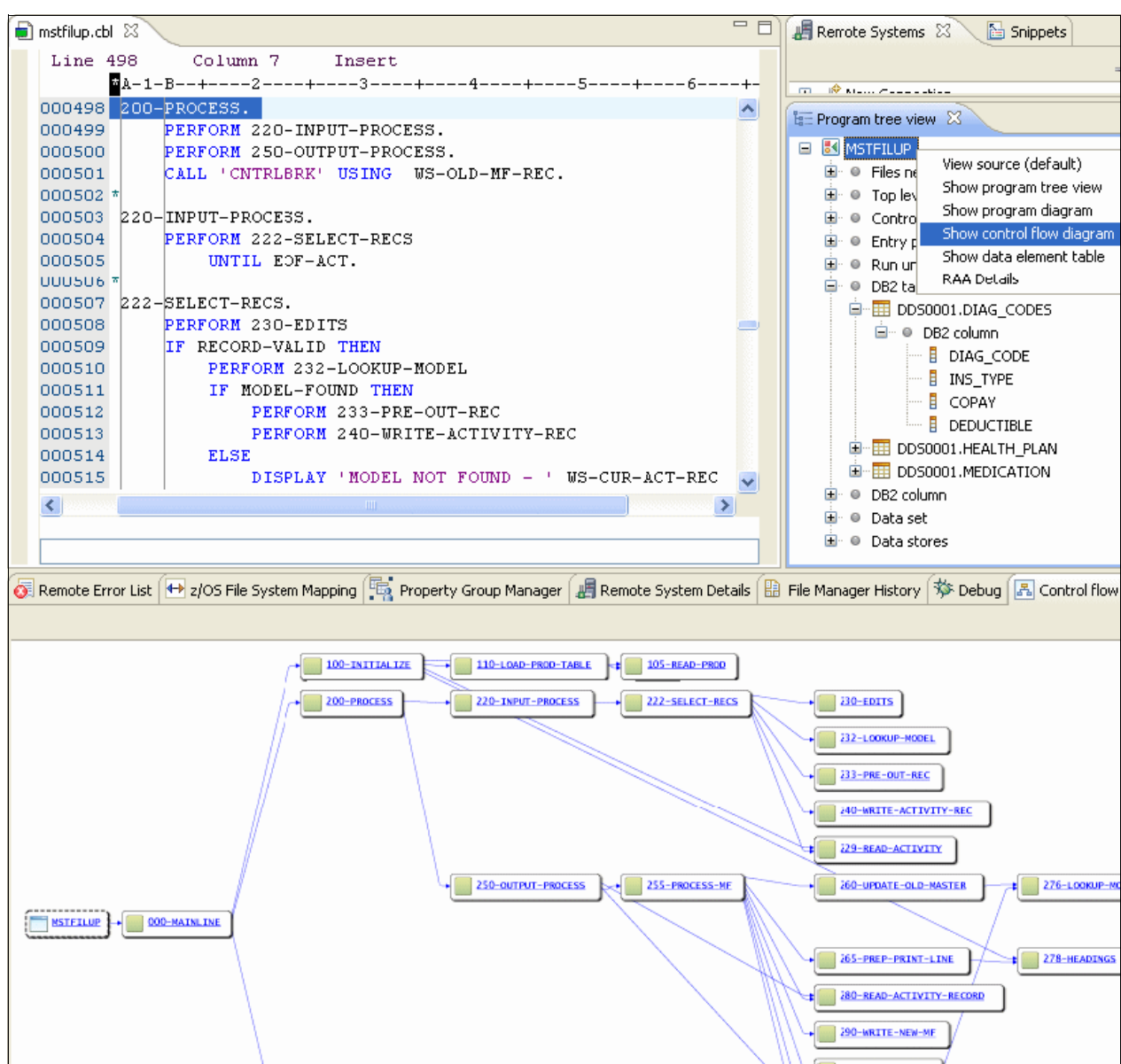


Figure 6-2 Rational Asset Analyzer integration's program tree view

Data flow analysis

A common requirement during application lifecycle is to track the progression of variable values as they are processed, moved, or made part of arithmetic operations, a file, a database, MQ, or screen I/O. We refer to this tracking as *data flow analysis*.

Performing data flow analysis by using ISPF searches, source listing lookups, and compiler cross-references is lengthy and error-prone. By using Rational Asset Analyzer integration, you can perform a data flow analysis, and create a tree that is composed of branches as variables, and read each branch to see if there are offshoots (additional branches) as shown in Figure 6-3.

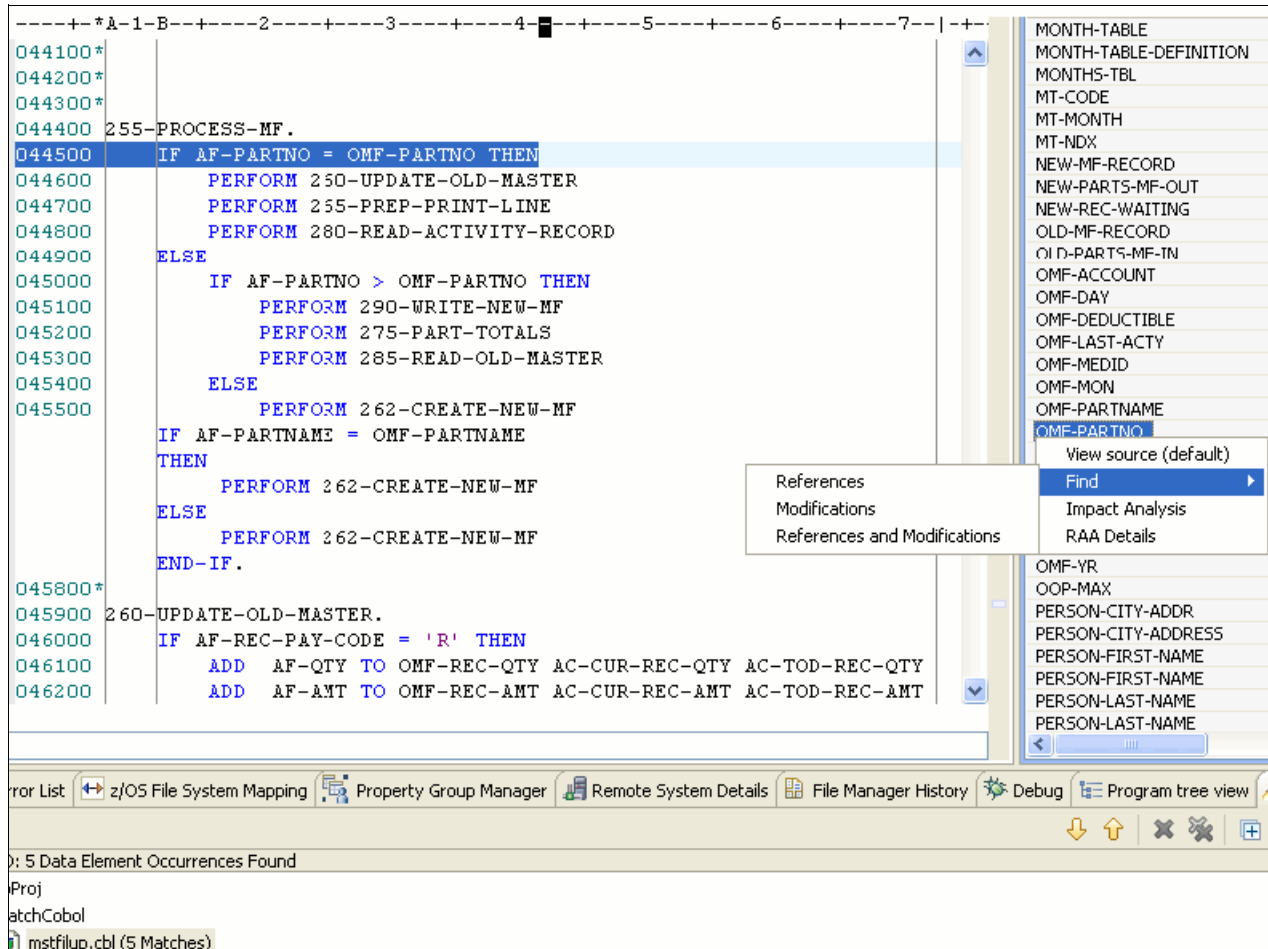


Figure 6-3 Asset Analyzer - data element table

Impact analysis

Another approach for solving the problem of data flow analysis is to use impact analysis. The *impact analysis* feature builds the tree structure as a report. The impact analysis report (Figure 6-4) shows all program elements (all application-scoped elements) that might be affected by a modification or change to a file, program statement, range of program statements, or a DB2 table or column. This type of analysis task is among the most in-depth and challenging types of software maintenance activities. Automating impact analysis by using static analysis tooling is beneficial and highly valuable.

Figure 6-4 shows a report that is produced as an Eclipse view. Each element in this report hyperlinks back to one of the following points:

- ▶ A source code line in a program within the scope of the analysis (see the impacted data elements in Figure 6-4)
- ▶ An element that you can analyze further with either Rational Asset Analyzer integration or Rational Asset Analyzer by using the Rational Asset Analyzer Details option:
 - Figure 6-5 on page 142, which shows a graphical representation of the possible ramifications of changing the variable from a high-level view
 - Figure 6-6 on page 142, which shows the details and an in-depth view of the work needed to analyze the change completely

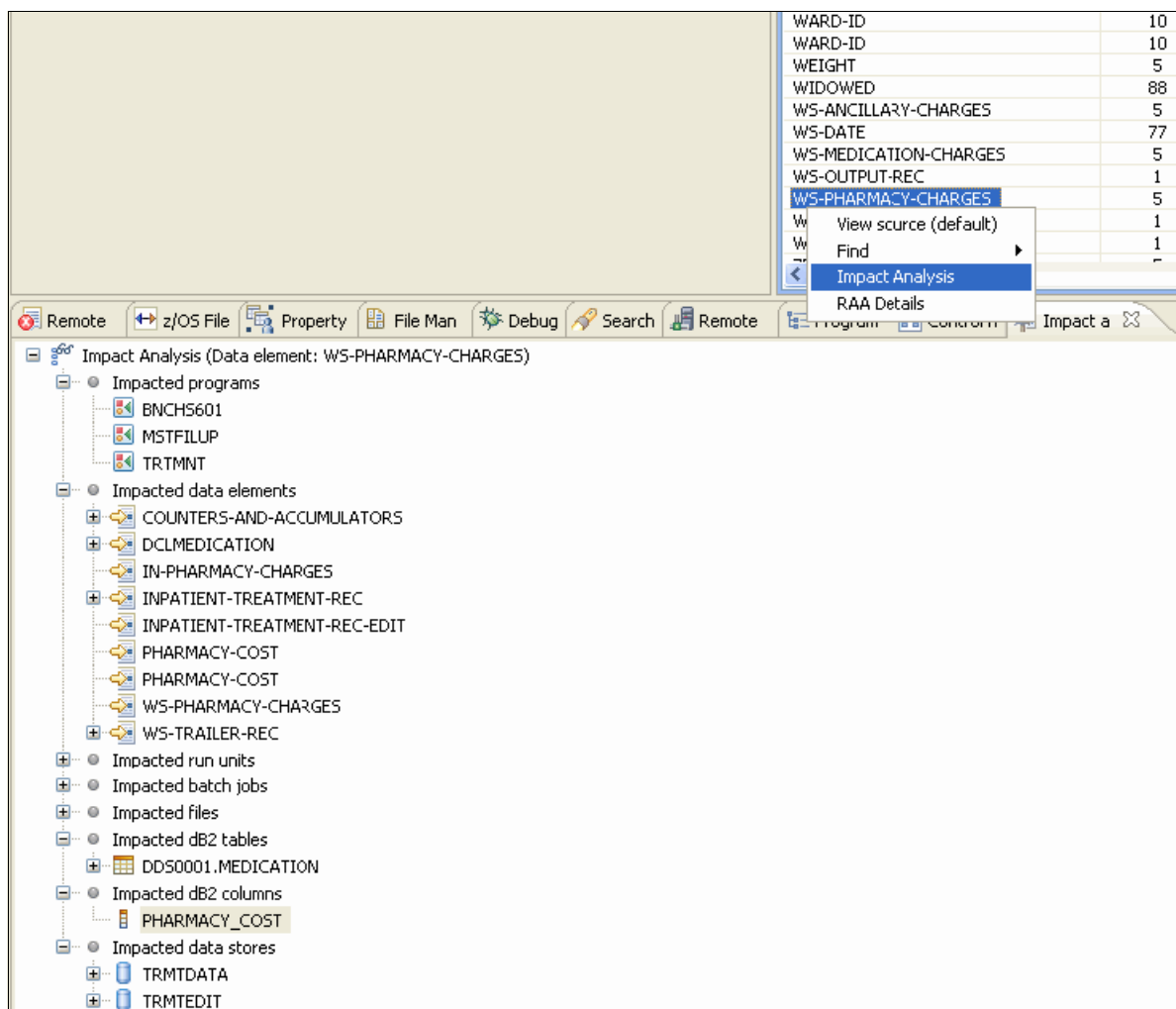


Figure 6-4 Rational Asset Analyzer integration Impact Analysis report

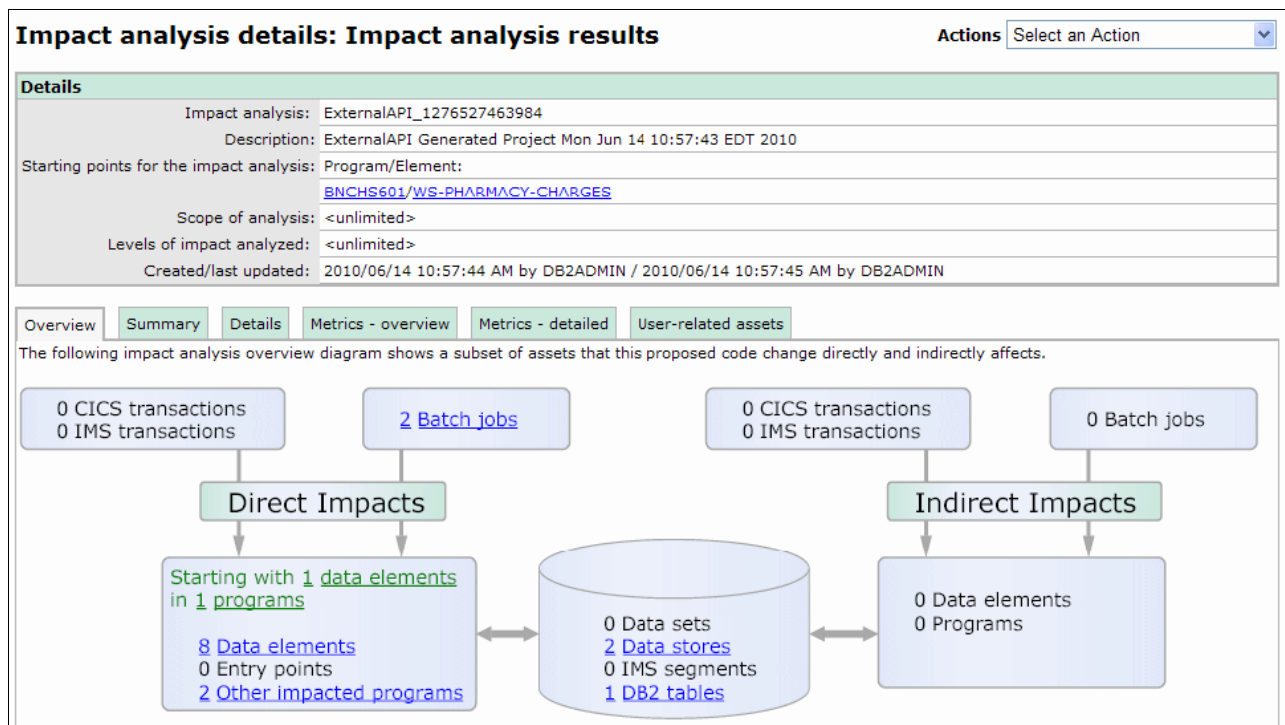


Figure 6-5 Rational Asset Analyzer integration impact analysis results under the Details option

Overview

Summary

Details

Metrics - overview

Metrics - detailed

User-related assets

The following lists the assets that are affected by this proposed code change. You can access the details page for an asset by selecting that asset.

Container (2)					Type	File count	Site
C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol					NTFS	65	RAAV5513
C:/D-Drive/RDZProjectNew/DemoProj/JCL					NTFS	3	RAAV5513

File (8)	Language	Type	Analysis status	Number of lines in file	Source location	Site
BNCHMRK.icl	JCL	Batch job source	Completed	306	C:/D-Drive/RDZProjectNew/DemoProj/JCL/BNCHMRK.icl	RAAV5513
BNCHS601.cbl	COBOL	Program source	Completed	850	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/BNCHS601.cbl	RAAV5513
BNCHTRMT.cpy	COBOL	Included source	Completed	64	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/BNCHTRMT.cpy	RAAV5513
mstfilup.cbl	COBOL	Program source	Completed	862	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/mstfilup.cbl	RAAV5513
MEDICATN.cpy	COBOL	Included source	Completed	28	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/MEDICATN.cpy	RAAV5513
RUNHELLO.icl	JCL	Batch job source	Completed	63	C:/D-Drive/RDZProjectNew/DemoProj/JCL/RUNHELLO.icl	RAAV5513
TRAILER.cpy	COBOL	Included source	Completed	7	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/TRAILER.cpy	RAAV5513
TRTMNT.cbl	COBOL	Program source	Completed	794	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/TRTMNT.cbl	RAAV5513

Site (1)	Description	HTTP IP address	FFS IP address
RAAV5513			

Batch job (2)	Analysis status	Source location	Site
DDS00011	Completed	C:/D-Drive/RDZProjectNew/DemoProj/JCL/RUNHELLO.icl	RAAV5513
DDS0001A	Completed	C:/D-Drive/RDZProjectNew/DemoProj/JCL/BNCHMRK.icl	RAAV5513

DB2 system (1)	Description	Site
WSAA	Inserted by SQL reference resolution	RAAV5513

Run unit (3)	Analysis status	References	Site
BNCHS601	Completed	0	RAAV5513
MSTFILUP	Completed	1	RAAV5513
TRTMNT	Completed	1	RAAV5513

Literal (3)	References
_99	5
990	5
ZERO	178

Program (3)	Language	Analysis status	Number of lines in program	Source location	Site
BNCHS601	COB	Completed	1254	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/BNCHS601.cbl	RAAV5513
MSTFILUP	COB	Completed	862	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/mstfilup.cbl	RAAV5513
TRTMNT	COB	Completed	948	C:/D-Drive/RDZProjectNew/DemoProj/BatchCobol/TRTMNT.cbl	RAAV5513

Figure 6-6 Rational Asset Analyzer integration impact analysis Details tab (partial results and hyperlinks to source files)

Documenting your application using Rational Asset Analyzer integration

In addition to these analysis usage models, developers commonly include several of the Rational Asset Analyzer integration diagrams and reports with their system documentation. All the diagrams are Scalable Vector Graphics (SVG) files, and they retain their hyperlinks back to the source elements from which they were derived.

You can incorporate Rational Asset Analyzer integration as an extremely simple and effective means of collaborating and sharing analysis workflow. You can copy and paste the Impact Analysis detail report (Figure 6-6 on page 142) to any rich-text document processor, such as Microsoft Word, Microsoft Excel, or IBM Symphony®, to provide project teams with an ideal working document for maintenance tasks.

6.1.2 Rational Developer for System z

Rational Developer for System z is an IDE that unities z/OS resources with desktop software for developing, maintaining, supporting, and modernizing your z/OS-based applications. Rational Developer for System z is built on the Eclipse framework. With Figure 6-7 on page 144, you can see that Rational Developer for System z is a comprehensive package with a rich set of modern IDE-based development capabilities:

- ▶ Access to z/OS assets: Organizing, allocating, managing, editing, and working with z/OS data sets and tools through the remote systems view.
- ▶ Development, maintenance, and support of business application functionality (analyzing, editing, and compiling or assembling), and source files by using integrated language-sensitive editors for COBOL, PL/I, IBM High Level Assembler (HLASM), C, C++ and Java, job control language (JCL), basic mapping support (BMS), and Message Format Service (MFS).

The editing process installation is direct to z/OS partitioned data sets (PDS libraries) by default, or you can configure the setting to access source through most of the source control management (SCM) systems on the market including Rational Team Concert.

- ▶ Managing and editing unit test data, such as queued sequential access method (QSAM) or sequential data sets, Virtual Storage Access Method (VSAM) or indexed, relative, and entry sequenced data sets, DB2 tables and views, and Information Management System (IMS) databases. Note that access to DB2 and IMS data is through Java Database Connectivity (JDBC), which lowers the cost of performing routine test data development and maintenance considerably.
- ▶ Access to z/OS Job Entry Subsystem (JES) submitting jobs and JES spool files, organizing and managing batch jobs.
- ▶ Wizards and *declarative development* or software tools that replace low-level coding to create, deploy, and test DB2 stored procedures, Web Services, and Service Component Architecture (SCA), transforming Unified Modeling Language (UML) models to COBOL.
- ▶ Rational Developer for System z can also be packaged with Enterprise Generation Language (EGL) for the development of Web 2.0 and Java 2 Platform, or packaged with Java EE for the development of Java Enterprise Edition applications.

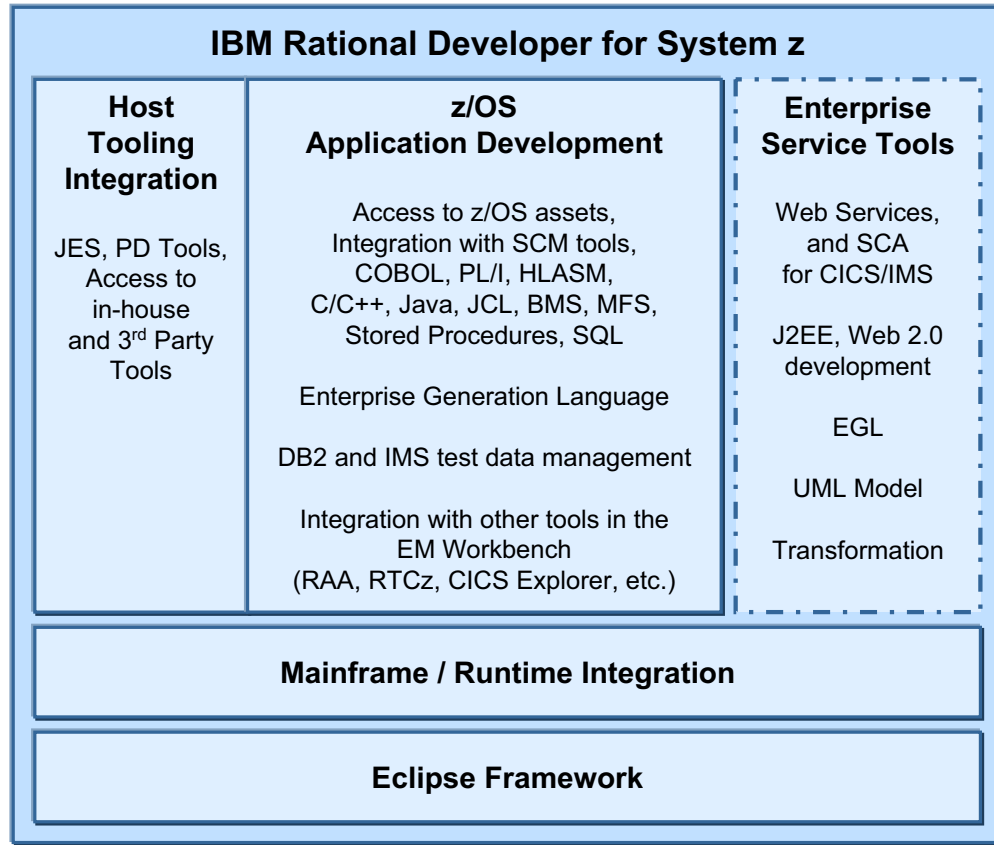


Figure 6-7 Overview of Rational Developer software architecture

Figure 6-8 on page 145 shows several common Rational Developer for System z views that are used during a typical COBOL analysis or edit session. The following items are highlighted in Figure 6-8 on page 145:

1. The program BNCHS601 was opened in the COBOL editor.
2. It is from the DDS0001.TEST.COBOL library (PDS), which is shown in the Remote Systems view.
3. The developer uses the Perform Hierarchy view (and Outline view) to study the program's control flow (perform chain processing)
4. The developer also uses the Outline view (with the Perform Hierarchy view) to study the program's control flow (perform chain processing).
5. The developer moves the mouse pointer over and analyzes the declaration of VALID-RECORD, which is a COBOL 88-level data item as shown and is part of the 01-structure FLAGS-AND-SWITCHES.

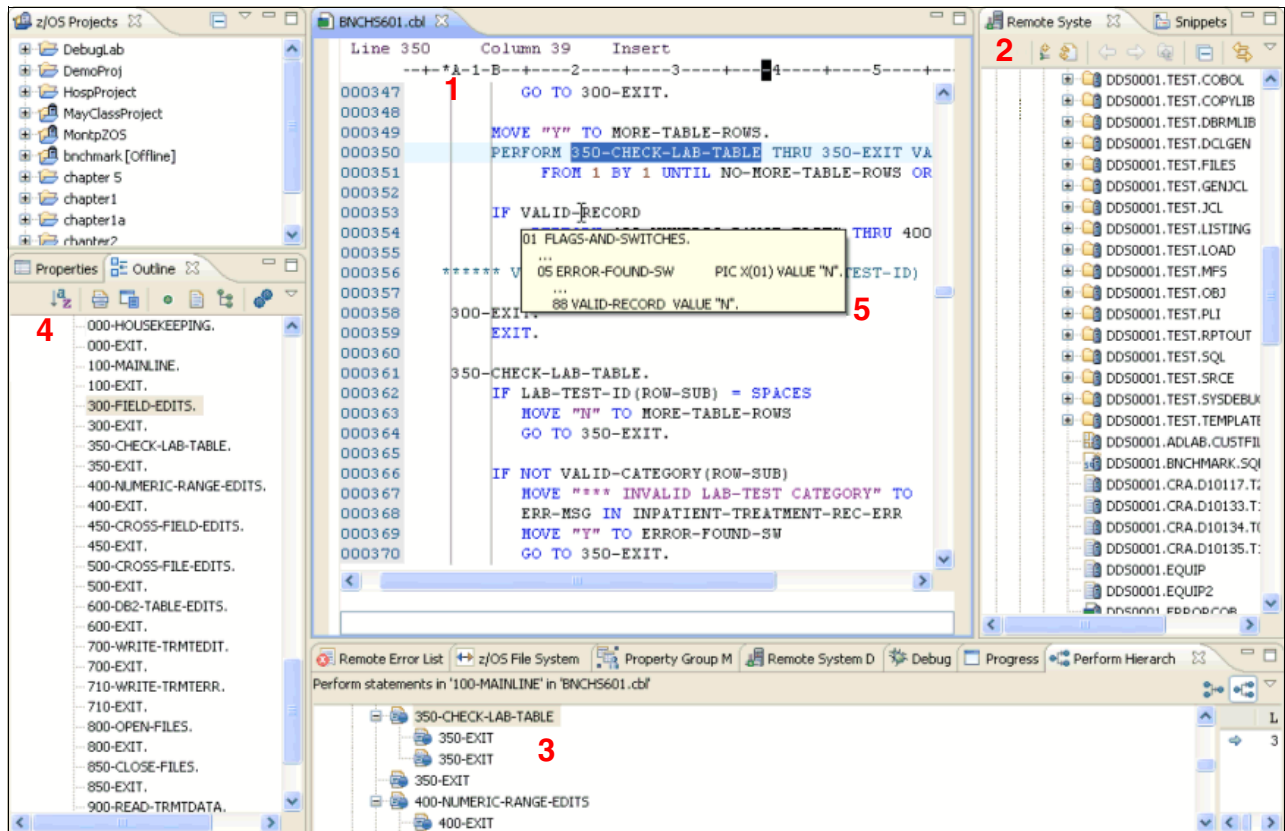


Figure 6-8 Views and editor functionality of a COBOL program loaded into Rational Developer for System z

As an Eclipse-based product, Rational Developer for System z tools and features include source editors, menus, and views and perspectives. You can customize all of these editors, views, perspectives, and menu tools extensively.

Rational Developer for System z Data perspective

The Data perspective allows you to connect to a DB2 database instance on any IBM system that you can access by using a Java Database Connectivity (JDBC) driver, and by entering the proper values for the following variables:

- ▶ Location: External name of the DB2 subsystem
- ▶ Host: IP address of the z/OS running your DB2 instance
- ▶ Port number: Port number of the listener for incoming JDBC access
- ▶ User ID and password: Credentials that provide authorized access to the DB2 subsystem and all data objects

After connecting, you can do all the work that is listed by using rich graphical tools, rather than by using the arduous process of accessing SQL Processor Using File Input (SPUFI) or IBM Query Management Facility™ (QMF™), and coding and executing SQL statements to manipulate test data.

Figure 6-9 shows an example Data perspective session where three DB2 tables are shown (Ward_Data, Hosp_Position, and Emp) in full screen-edit mode.

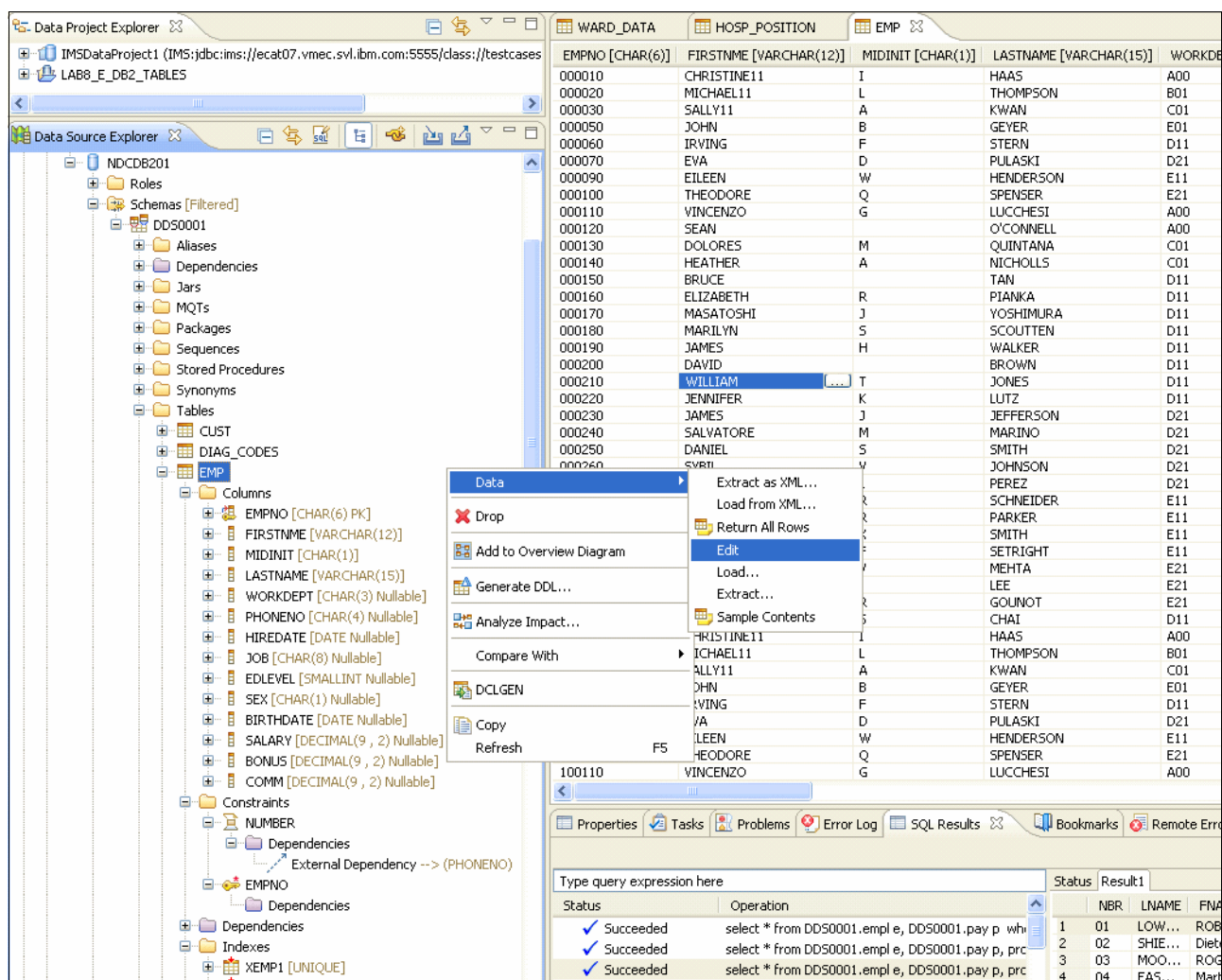


Figure 6-9 The Data perspective: Working with DB2 table data

Because the Data perspective accesses DB2 objects through a JDBC driver, this feature of Rational Developer for System z can save considerable development LPAR host resources compared to using QMF or SPUFI.

There are two options for creating, syntax checking, and running (testing) SQL:

- ▶ Use a graphical SQL statement builder (Figure 6-10 on page 147).
- ▶ You also can use Content Assist (Ctrl+spacebar). Similar to the COBOL editor, PL/I statement building functionality allows you to select the schema and table names and build out your SQL at the language level. This function is useful for developers who have more SQL experience.

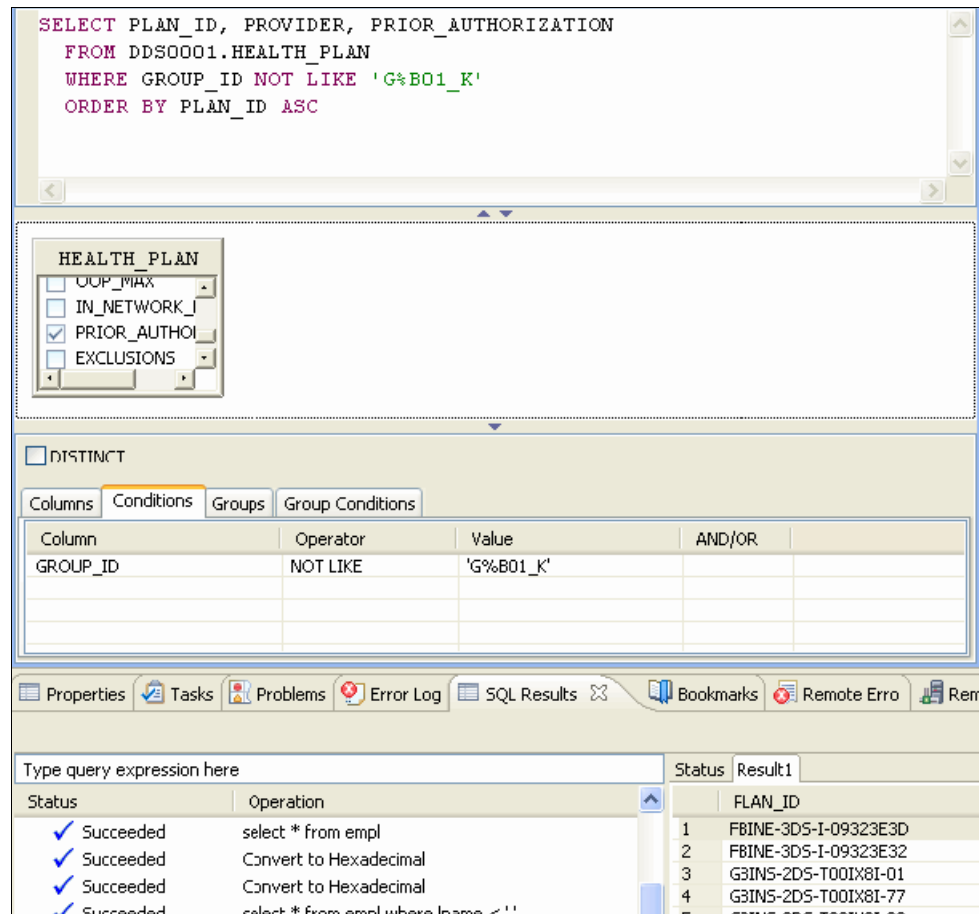


Figure 6-10 SQL statement builder and SQL results

CICS and Rational Developer for System z

Rational Developer for System z provides a set of high-value functions for working with CICS applications, BMS screens, and CICS-DL/I statements, including but not limited to the following functions:

- ▶ Remote syntax checking.
- ▶ Local syntax checking, which requires a local copy of the IBM TX Series for multi platforms. If you use local syntax checking for CICS statements, you potentially reduce your z/OS CICS preprocessing time and costs.
- ▶ Content assist, which is used to develop command-level CICS statements.
- ▶ A CICS BMS map editor (Figure 6-11 on page 148), which you may use to develop or maintain and support BMS files graphically.
- ▶ Integration with the CICS Explorer, which you may use to manage connections, define CICS resources, and perform other CICS system administrator tasks.
- ▶ The ability to debug CICS online applications that are written in COBOL, PL/I, HLASM, and Java by using one of the following tools:
 - IBM Debug Tool, in which case, the debugging occurs under z/OS
 - TX Series, in which case, the debugging occurs on a workstation
- ▶ CICS (EXEC CICS) COBOL program templates, which are skeleton program files that you may customize, export, and roll out to developers from Rational Developer for System z workspace defaults to speed-up the development process and make it less error prone.

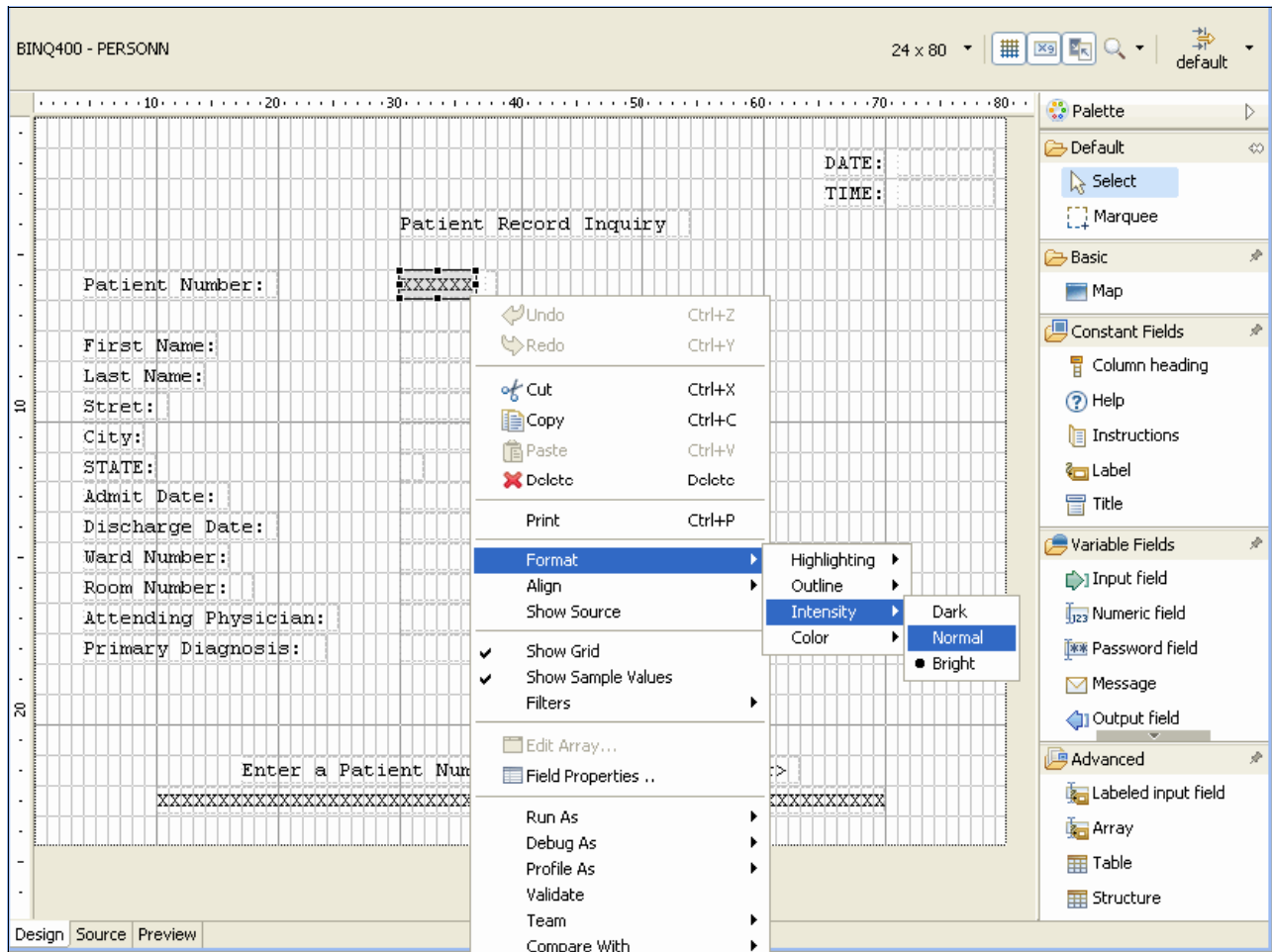


Figure 6-11 The CICS BMS map editor

See chapter 4, IBM Rational Developer for System z, in *z/OS Traditional Application Maintenance and Support*, SG24-7868 for more information.

6.1.3 Rational Development and Test Environment for System z

Rational Development and Test Environment for System z allows mainframe operating systems, middleware, and software to run on Intel and Intel-compatible platforms, therefore, no System z mainframe hardware is needed for the initial set of development activities. Rational Development and Test Environment for System z may be used for development and test of new application function and also maintenance changes. Additionally, Rational Development and Test Environment for System z is licensed for the investigation and verification of new middleware levels (version-to-version upgrades), prototyping new application architectures, demonstration of application function, or even internal employee education.

Rational Development and Test Environment supports various versions of operating systems and middleware, including z/OS, CICS, IMS, DB2, WebSphere, COBOL, PL/I, C++, Java, HLASM, and JCL. Therefore, it can be used with various types of development and test activities including the following types:

- ▶ Compile development-level source code and build systems
- ▶ Analysis and problem determination tools

- ▶ Data import, export, inspection, and alteration utilities
- ▶ Middleware or application setup and configuration tools

Developers also have easier access to modern versions of IBM middleware in the development and test environment without having to wait for system upgrades on the mainframe. Developers can explore new CICS or IMS features, investigate new architectural models, and perform their first series of tests and component regression testing, without worrying about changes coming from other developers and causing unexpected errors. Using the Rational Development and Test Environment for System z, developers can build and test System z applications anywhere and anytime, without affecting the production operations of an organization.

According to the Rational Development and Test Environment for System z version you are using, it includes a preconfigured set of IBM software entitled for development usage specifically in the Rational Development and Test Environment for System z including:

- ▶ z/OS including sub features
- ▶ WebSphere Application Server for z/OS
- ▶ CICS Transaction Server (CICS TS)
- ▶ IMS
- ▶ DB2 for z/OS
- ▶ WebSphere MQ for z/OS
- ▶ IBM Java SDK for z/OS
- ▶ Enterprise COBOL
- ▶ Enterprise PL/I
- ▶ XL C++
- ▶ IBM Rational COBOL Runtime (EGL)
- ▶ IBM Debug Tool

Note: The included IBM software products are provided for development purposes only on an as-is basis. No technical support or APAR/PTF deliverables are provided for the included software as listed in the Supporting Programs section of the product license with your purchase of the Rational Development and Test Environment for System z.

Rational Development and Test Environment for System z offers the following advantages:

- ▶ Provides a high-fidelity test environment comparable to what is available on the mainframe. By using actual z/OS middleware, less retesting and rework is required when moving applications from the Development and Test Environment to the quality assurance or pre-production environment on the mainframe.
- ▶ Executes on an x86 PC, using zero development MIPS on the production mainframe for functional application testing. The development MIPS can be reallocated to other production or testing needs, freeing additional capacity for new workload while reducing line-of-business development costs and chargeback.

Do not run on production workloads: Rational Development and Test Environment for System z is not intended as a replacement for a larger mainframe system. We advise the final testing of software be done in an environment as close as possible to what would be run in production. Thus, Rational Development and Test Environment should not be used to run production workloads of any kind, nor can it be used for more robust development workloads like production module builds, preproduction testing, stress testing, or performance testing. Production builds and their testing must continue to be done on the mainframe, but all development work up to that point can now be offloaded to Rational Development and Test Environment for System z.

6.1.4 Rational Team Concert

IBM Rational Team Concert provides the essentials of a lean collaborative lifecycle management solution with integrated planning, work-item tracking, version control, build management and reporting. The Rational Team Concert tool is designed to increase individual and team productivity for teams that follow agile or traditional development processes.

With support for the Eclipse client, client for Visual Studio, and ISPF client, all developers can share a single repository for building and delivering software. The Web 2.0-based browser interface makes it possible for users to access project areas, browse repository information, update tasks and plans, and track work-item progress.

Rational Team Concert offers the following features to the CICS application lifecycle:

- ▶ Enhance team collaboration

Rational Team Concert provides many features in terms of team collaboration including: integrated planning, work-item tracking, source control, build management, and project health and transparency. Using customizable portal views, team members can access information about projects such as news and events, current build status, work in progress, and changes that are requested. Members can also see what their teammates are working on, view online status of colleagues and access availability of teammates for collaboration.

With Rational Team Concert software, exchanging information directly in the context of the work you are doing is easier. And the software can automatically update stakeholders about any changes to the document or associated work item.

- ▶ Work items tracking

Rational Team Concert creates and tracks the progress of individual work items according to the team process and project rules. This feature enables defects, enhancements, and conversations to flow efficiently across the team, accelerating project progress. As work advances, the Rational Team Concert software captures information (such as who, what, when, and why) that is associated with each work item. This information helps the team members to apply the much-needed context for shared work items. Support for approvals and discussions helps to improve scheduling of code reviews and approvals and helps the team members remain in sync.

- ▶ Rich software configuration management capabilities

Rational Team Concert delivers essential software version control, workspace management, and parallel development support to individuals and teams. It can help improve productivity by automatically tracking changes to artifacts and enabling rollback to any previous version. Teams can find the correct balance between experimentation and transparency through combinations of private and public workspaces. Integrated stream management and server-based sandboxes help individuals solve problems in a controlled environment and easily contribute to the team's regular builds. New distributed source code management (SCM) capabilities help improve collaboration with other lines of business and outside suppliers that need to share component code changes across repositories.

- ▶ Supports agile or traditional processes

Rational Team Concert includes advanced planning capabilities for Scrum, Open Unified Process (OpenUP) and traditional teams, which make it easier for you to support or extend your agile, iterative or traditional process adoption.

► Enables more efficient software builds

The comprehensive build management features with Rational Team Concert can help your team schedule and execute software builds efficiently. Built-in work-item and change-set traceability makes it easy to identify precisely where the build breaks. Comprehensive build reporting automatically provides a detailed record of the team's build activities. Support for continuous build integration allows agile and other teams to benefit from the productivity gains associated with continuous build processes.

Rational Team Concert provides native z/OS build support using JCL or Antz (which is an extension of the Apache Ant build tool that allows you to write Ant XML scripts to define your build for z/OS). Through the use of build definition templates, you can define the following types of builds for z/OS:

- Command line build: You can specify a series of commands that will do all of the required build steps. You also may use REXX syntax for describing build commands.
- Job control language (JCL) build: You can specify the JCL either in the file system or in the build definition. You must submit the specified JCL through Job Monitor, which you install separately.
- Antz builds: You can write an Ant XML script to define your build for z/OS. Individual build steps, such as compilations of separate source codes, are defined in language definitions that you need to configure before a build. Rational Team Concert for System z converts language definitions into Ant macros, and then Apache Ant processes the build script with those macros.

Rational Team Concert uses Rational Build Agent to define and run builds which runs as a daemon process on z/OS.

► Orchestrating change across the product and application lifecycle

Built on Open Services for Life cycle Collaboration (OSLC), Rational Team Concert collaborative change management solutions can provide leading change-management capabilities, including agile, traditional or hybrid real-time planning and tracking, task boards, Gantt charts, product backlogs, burn down charts and dashboards to help improve planning and predictability for a wide range of projects. Rational Team Concert Stakeholder provides basic change-management capability, access to task status and work item information, and is designed for use by customers, partners, and extended participants.

Figure 6-12 shows how work items are created and prioritized in Rational Team Concert.

The screenshot displays the IBM Rational Team Concert interface for viewing and managing work items. The top navigation bar includes links for Dashboards, Project Areas, Work Items (selected), Plans, Source Control, Builds, and Reports. The user is logged in as John Milne. The main content area is titled "Defect 370" and shows a summary of the defect: "RTPOT08A - S0CB in CLCLBCST". The defect is categorized as "New" and is associated with the "EM Workbench Redbook" project area. The left sidebar provides navigation options, including "Create Work Item", "Create Query", and a list of "Recently Viewed" items. The "Details" section on the right shows the defect's attributes: Type (Defect), Severity (Major), Found In (Unassigned), Creation Date (Jul 10, 2010 10:48 A.M.), Created By (John Milne), Project Area (EM Workbench Redbook), Team Area (EM Workbench Redbook Prod Team), and Filed Against (Backlog). The "Quick Information" section shows the defect is subscribed to by John Milne (JM). The "Description" section at the bottom provides a detailed overview of the defect: "overnight master update suite abended in step TRTSRCh. S0CB ABEND in subprogram CLCLBCST of TRMTSRCh."

Defect 370: RTPOT08A - S0CB in CLCLBCST - IBM Rat...

Rational Team Concert

John Milne | Log Out | ?

Dashboards Project Areas **Work Items** Plans Source Control Builds Reports

EM Workbench Redbook

Defect 370

Summary: * RTPOT08A - S0CB in CLCLBCST New

Save

Overview Links Approvals History

Saved: Jul 10, 2010 10:47 A.M.

Details

Type: Defect Owned By: Jon Sayles

Severity: Major Priority: 2 Medium

Found In: Unassigned Planned For: Sprint 1 (1.0)

Creation Date: Jul 10, 2010 10:48 A.M. Estimate: Correction:

Created By: John Milne Time Remaining:

Project Area: EM Workbench Redbook Due Date:

Team Area: EM Workbench Redbook Prod Team

Filed Against: * Backlog

Tags:

Quick Information

Subscribers (1): JM

Description

overnight master update suite abended in step TRTSRCh. S0CB ABEND in subprogram CLCLBCST of TRMTSRCh.

Figure 6-12 IBM Rational Team Concert Work Items view

Figure 6-13 shows how team members can collaborate using discussion boards inside Rational Team Concert.

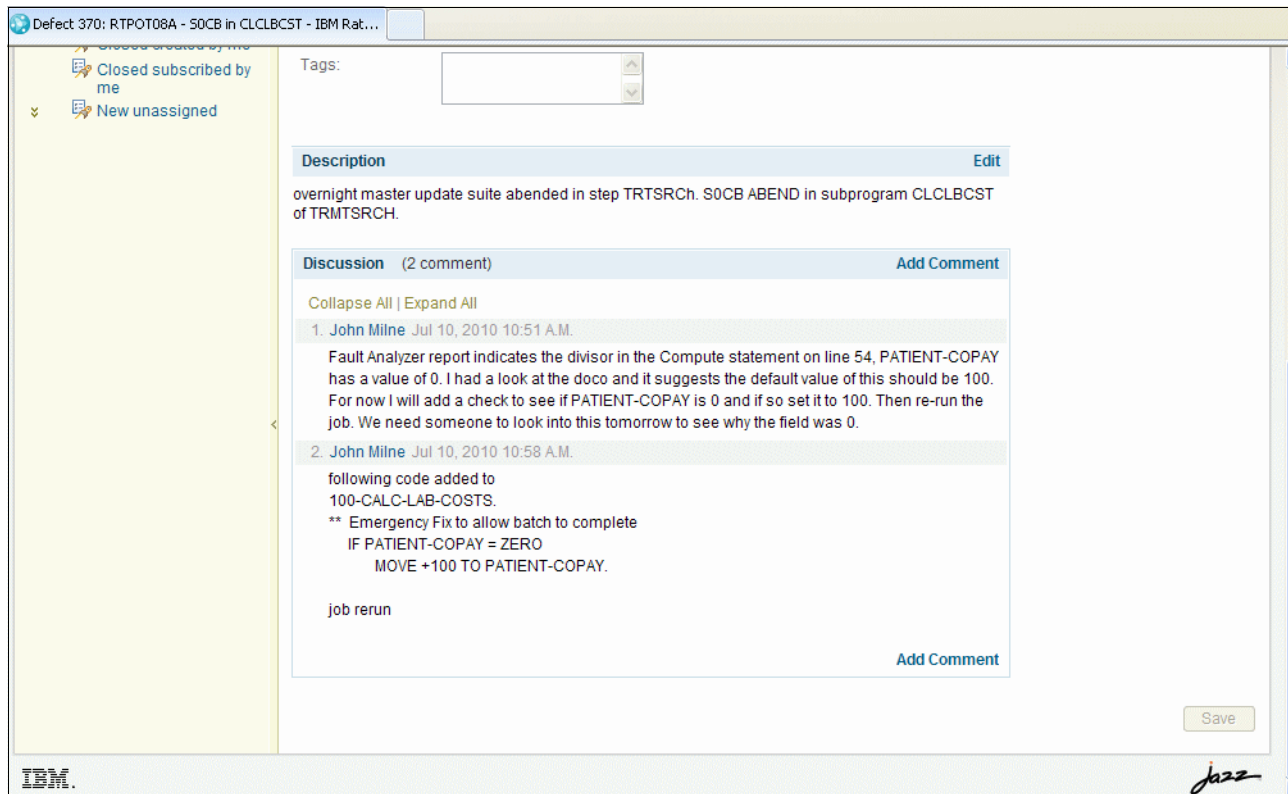


Figure 6-13 Work item discussion inside RTC

6.2 CICS and z/OS tools

This section addresses tools created specifically for CICS:

- ▶ CICS Interdependency Analyzer (IA)
- ▶ z/OS Problem Determination Tools
- ▶ CICS Performance Analyzer for z/OS
- ▶ CICS Application Performance Analyzer for z/OS
- ▶ CICS Configuration Manager for z/OS
- ▶ CICS Deployment Assistant for z/OS
- ▶ IBM Session Manager for z/OS

6.2.1 CICS Interdependency Analyzer (IA)

CICS Interdependency Analyzer for z/OS is a productivity tool that is used to discover and analyze CICS resources and identify relationships between them. CICS IA uses a runtime collector to gather actual usage information related to the execution of CICS applications, giving you a true picture of your application and its interactions. CICS IA accomplishes this by monitoring applications for API and SPI commands, along with optional DB2, IMS, MQ and COBOL calls, to give you a complete picture of your application and the interactions and resources that are referenced, with their interrelationship. Figure 6-14 on page 154 contains a chart showing the collector and reporting structure of CICS IA.

CICS IA collects this dependency information, storing it in a DB2 database that allows you to run queries and reports using an offline reporting interface or using the CICS IA plug-in under the CICS Explorer, which, with other CICS Tools plug-ins, provides a seamless integration between the productivity tools. For example, while you display CICS resources in the CICS Explorer, you can right-click an item and display all resources related to it.

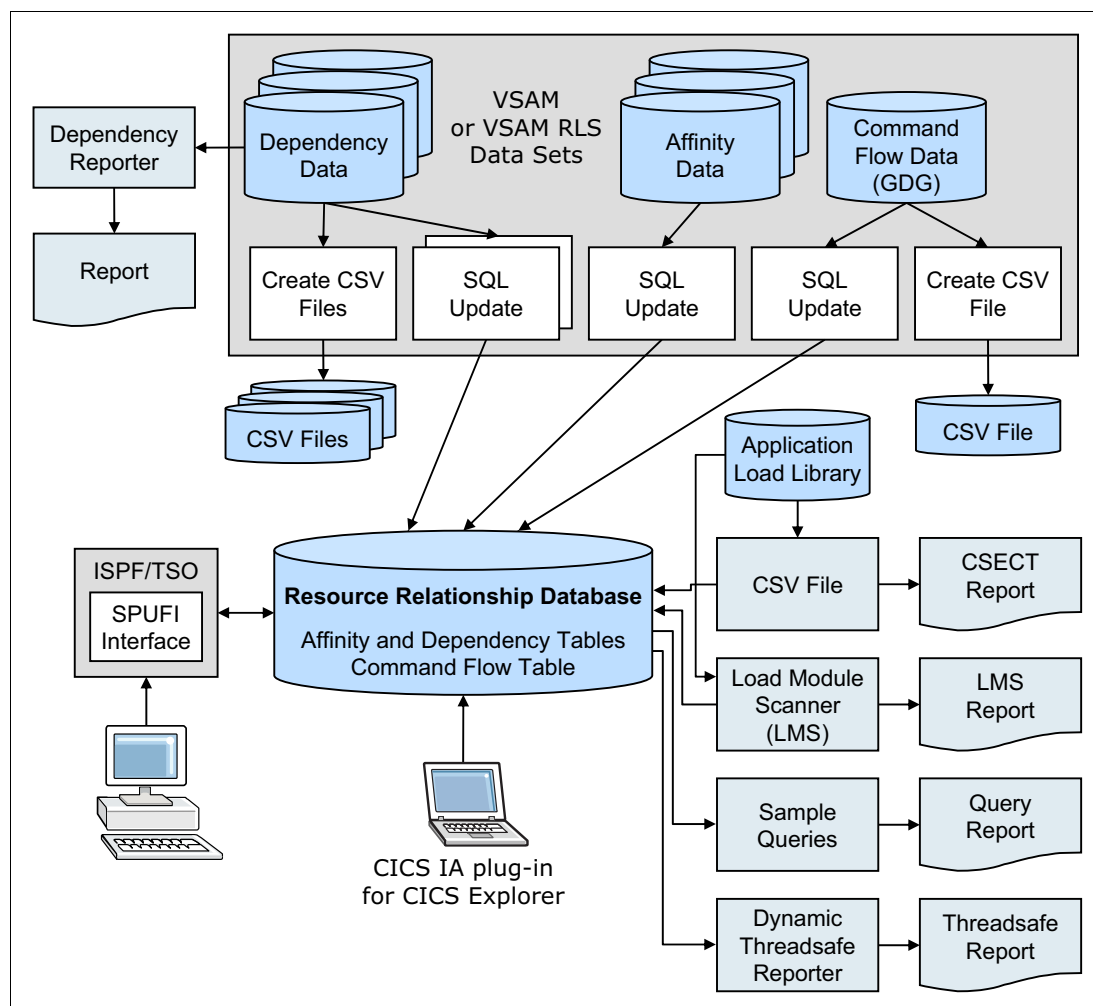


Figure 6-14 CICS Interdependency Analyzer Reporting Structure

As our work environments challenge us to do more with less, CICS IA can help a CICS professional analyze applications for threadsafe considerations to improve the overall execution efficiency. You can run reports to analyze your applications to identify the impacts of a CICS version or release upgrade by identifying what API commands are affected to help with upgrade planning and testing.

With the Command Flow feature of CICS IA, you can capture and view all EXEC CICS, SQL, MQ, and IMS calls in chronological order. This feature can be helpful in diagnosing the flow of your application as it executes, and the ability to identify TCB mode switches to help with tuning and threadsafe analysis.

CICS IA helps you analyze potential improved application design, as in the following examples:

- ▶ Understand cross-system applications and dependencies.
- ▶ Know the resource topology within a particular CICS region.
- ▶ Run comparisons on discovered data.
- ▶ Identify the resource flow following a transaction abend.
- ▶ Understand the effect of opening and closing files.
- ▶ Know the last time a particular resource was used.
- ▶ Analyze resource relationships when planning for dynamic workload balancing with CICSplex SM.

In a CICS environment where workload balancing is being used, CICS IA can provide affinities analysis and can also generate affinity-transaction-group definitions that can be directly input into CICSplex SM.

For a complete system inventory solution, CICS IA should be used with Rational Asset Analyzer to provide application understanding and component details. Rational Asset Analyzer assists your IT personnel with maintenance and extension of existing assets through impact analysis and application understanding.

CICS IA helps you modernize the existing enterprise assets and skills by providing knowledge about the static environments (finding and reusing application code and the connecting components), and the dynamic environments (understanding what code is executing in run-time environments). You can launch Rational Asset Analyzer with an on-screen button while working with CICS IA, making it easier to explore your solution environments.

CICS IA benefits can be summarized as follows:

- ▶ You may identify resources associated with transactions, programs, Basic Mapping Support maps, files, temporary storage queues, transient data (TD) queues, 3270 Bridge facility, web services, Corba Server, and Enterprise Java Beans (EJBs).
- ▶ Reports on DB2, IMS, and WebSphere MQ resources that CICS uses are also generated.
- ▶ A CICS IA plug-in is available for CICS Explorer (and accordingly with Rational Developer for System z, because it includes an integrated copy of CICS Explorer) to offer the same intuitive way to query CICS relationship data, manage queries, and navigate through complex application relationships, now from a standard interface. This way can provide easy integration between the CICS IA plug-in and other CICS tools from the CICS Explorer perspectives, for example, to click a transaction in a CICS Performance Analyzer bar chart and drill down to see its dependencies, relationships, affinities, and more.
- ▶ Sample queries assist you by highlighting those applications that are currently running in previous levels of CICS TS that are most sensitive to CICS API and SPI command changes.
- ▶ Support for CICS resources used by Software AG Natural gives you new insight into the behavior of your Natural applications.

Figure 6-15 shows all views that are available in the new CICS IA plug-in.

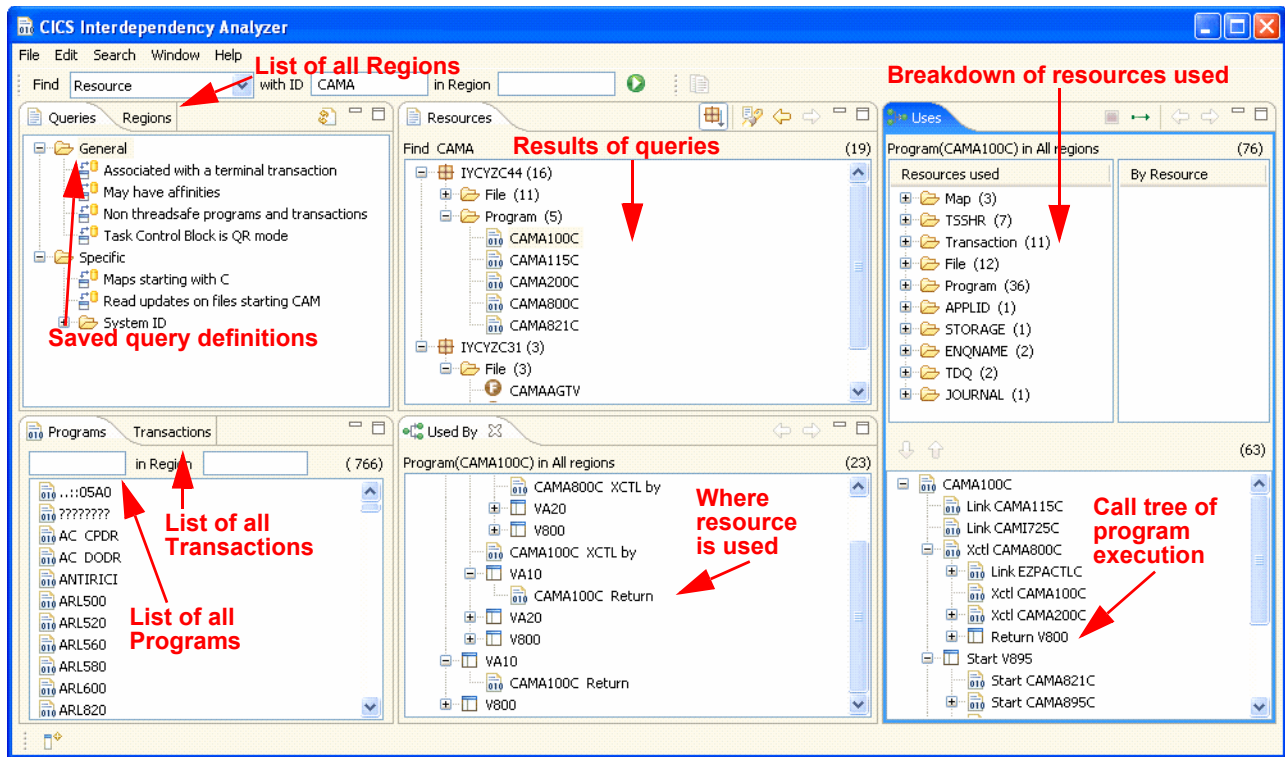


Figure 6-15 Views of CICS IA plug-in

For more information about CICS IA, see *IBM CICS Explorer*, SG24-7778.

6.2.2 z/OS Problem Determination Tools

Problem Determination Tools for z/OS (PD Tools) consist of a core group of IBM products that are designed to work in conjunction with compilers and run times to provide a start-to-finish development solution for the IT professional.

This section introduces the tools, and provides an overview of their integration, with Rational Developer for System z.

Debug Tool for z/OS

IBM Debug Tool is an interactive source-level debugging facility for compiled applications that integrates with Rational Developer for System z within its own Debug Eclipse perspective. Debug Tool and Rational Developer work in a variety of environments (Figure 6-16 on page 157) and provide testing capabilities for most of the major IBM languages and runtime environments.

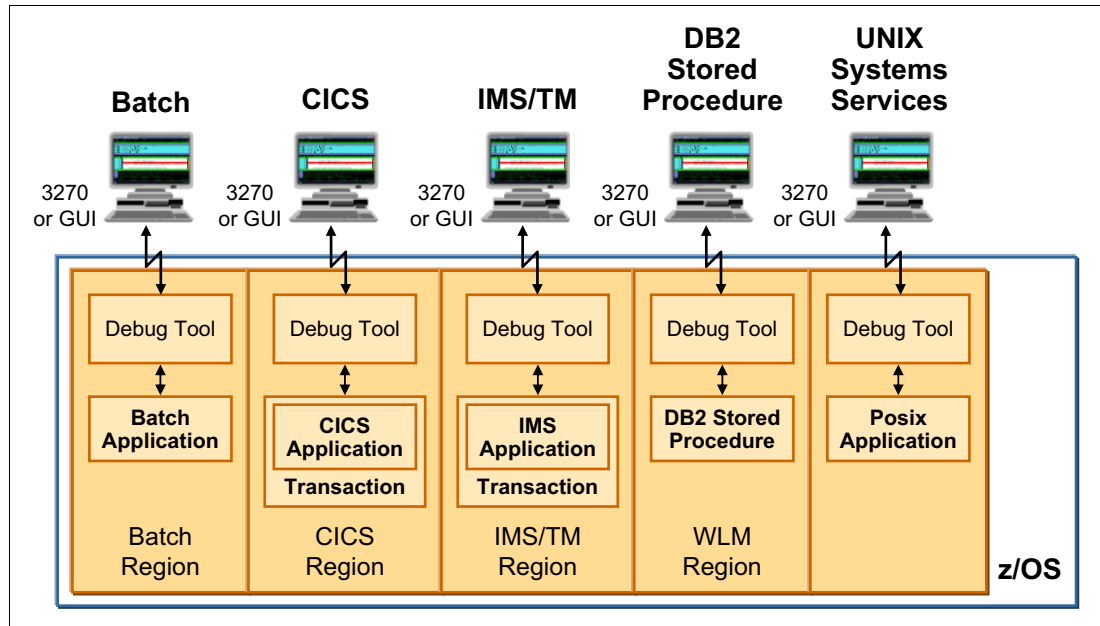


Figure 6-16 Debug Tool supported languages and run times

Using the integration between Rational Developer for System z and the Debug Tool, you can perform the following tasks:

- ▶ Debug online (CICS or IMS TM) and batch applications.
- ▶ Debug multiple languages (COBOL, PL/I, HLASM, Java, and others), including mixed-language and cross-platform application debugging, for example, where COBOL calls HLASM, or Java calls COBOL CICS.
- ▶ Display, monitor, and alter program variables.
- ▶ Set standard types of breakpoints.
- ▶ View data in hex (EBCDIC) or string values.
- ▶ Analyze your application dynamically.

Debug Tool and Rational Developer for System z integration process

Figure 6-17 on page 158 depicts the way that the process works. The Debug Tool has a listener or daemon that runs on z/OS for debug operations that are launched from your Rational Developer for System z workstation.

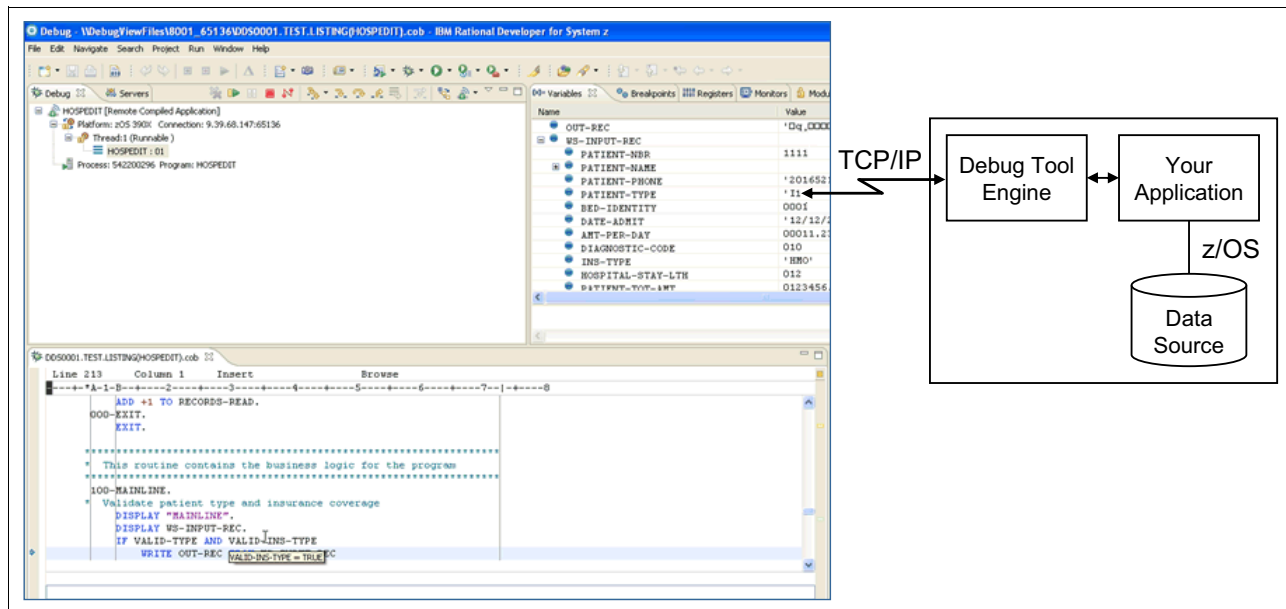


Figure 6-17 Rational Developer for System z and Debug Tool API process: Workstation and z/OS cross-platform debugging environment

This listener, shown as the Debug Tool Engine in Figure 6-17, performs the following tasks:

- ▶ Determines dynamically what action to take on the host based on your Debug session commands:
 - Step
 - Animate (continuous stepping without user intervention)
 - Run to a breakpoint
 - Issue a command
 - Define a breakpoint or variable “watch” monitor
 - Inspect a variable value
- ▶ Executes the application statements in the load module on z/OS.
- ▶ Returns data (the values associated with program variables) in the appropriate Rational Developer for System z Debug Perspective views (Figure 6-18 on page 159) that show the following information:
 - Program statements
 - Breakpoints and several of the dozens of debug runtime functions:
 - Jumping to a line (unconditional branch)
 - Running to a line (executing statement code)
 - Live, real-time browse access to a VSAM file that is used in the program
 - Real-time browse or edit access to a DB2 table’s rows, from tables referenced in your application code’s EXEC SQL statements; with this function you can modify table row data dynamically during Debug and to prototype various statement execution paths
 - Monitored expressions
 - Variable values
 - Information about the load module
 - Information about the registers and storage content

The simple and effective approach of Debug and Rational Developer for System z to dynamic code analysis and application debugging encourages developers to not take shortcuts and to use first-class testing techniques. Debug and Rational Developer for System z not only improve your productivity and help you accomplish more in less time, but they also improve the quality of your application. With Debug and Rational Developer, you identify more bugs earlier in the testing cycles by simplifying the traceability to the source of the problem.

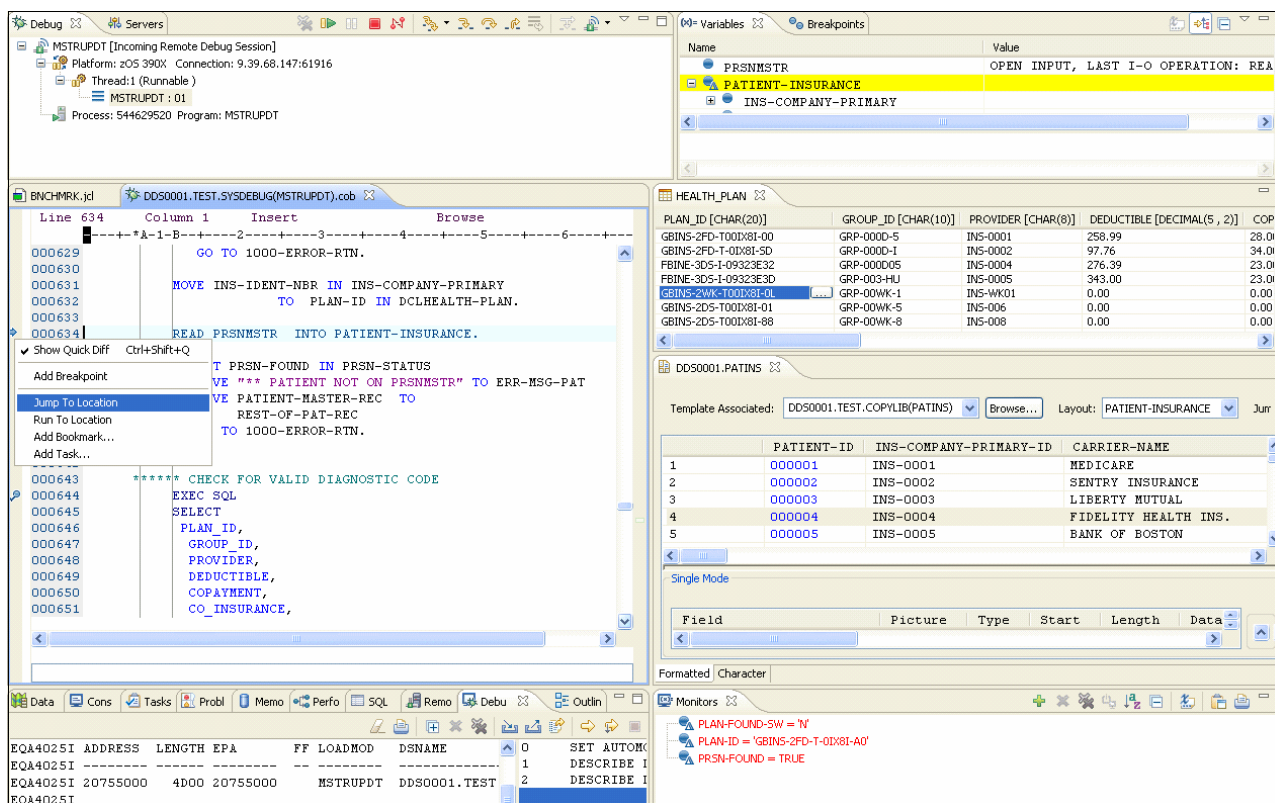


Figure 6-18 Debug Perspective and access to numerous other views related to the program being tested

File Manager for z/OS

A common requirement during application development is to browse, search, analyze, and edit QSAM (sequential) and VSAM (indexed, sequential, and relative record) data sets. You already saw that you use the Data Perspective to work with DB2 database tables.

For QSAM and VSAM, you can use the PD Tools File Manager and, specifically, its integration with Rational Developer for System z through Eclipse.

Figure 6-19 on page 161 shows the browse and edit features of File Manager on a VSAM file, and Figure 6-20 on page 162 shows the search and replace functionality. In these images, note the following functions to view and edit QSAM and VSAM file data:

- ▶ You can view file data as a table of records and as an individual record (through a copybook) in Figure 6-20 on page 162.
- ▶ You can view file data as a stream of characters under the Character tab in Figure 6-19 on page 161.
- ▶ You can open files in full edit or browse-only mode.
- ▶ In Edit, you can add and delete records, and you can change field data values.
- ▶ You can transfer any number of records to your workstation and move around within a file. You control this setting and other settings in the File Manager Preferences page.

- ▶ A File Manager History view documents your File Manager data access and allows you to re-access the files at any point in time.
- ▶ From the Remote Systems Explorer view, you can use the context menu options, after you select a file, for these tasks:
 - Copy data from an existing file with the same characteristics.
 - Test data generation.
 - Assign a copybook as a template to view a file through logical fields.
- ▶ For Find/Replace (Figure 6-20 on page 162), you can search within these types of information:
 - Logical fields as defined in the copybook template
 - Excluded records
 - Non-excluded records
 - A file or fields using a regular expression

All of this functionality uses standard Eclipse development skills. You do not need to learn proprietary editing procedures.

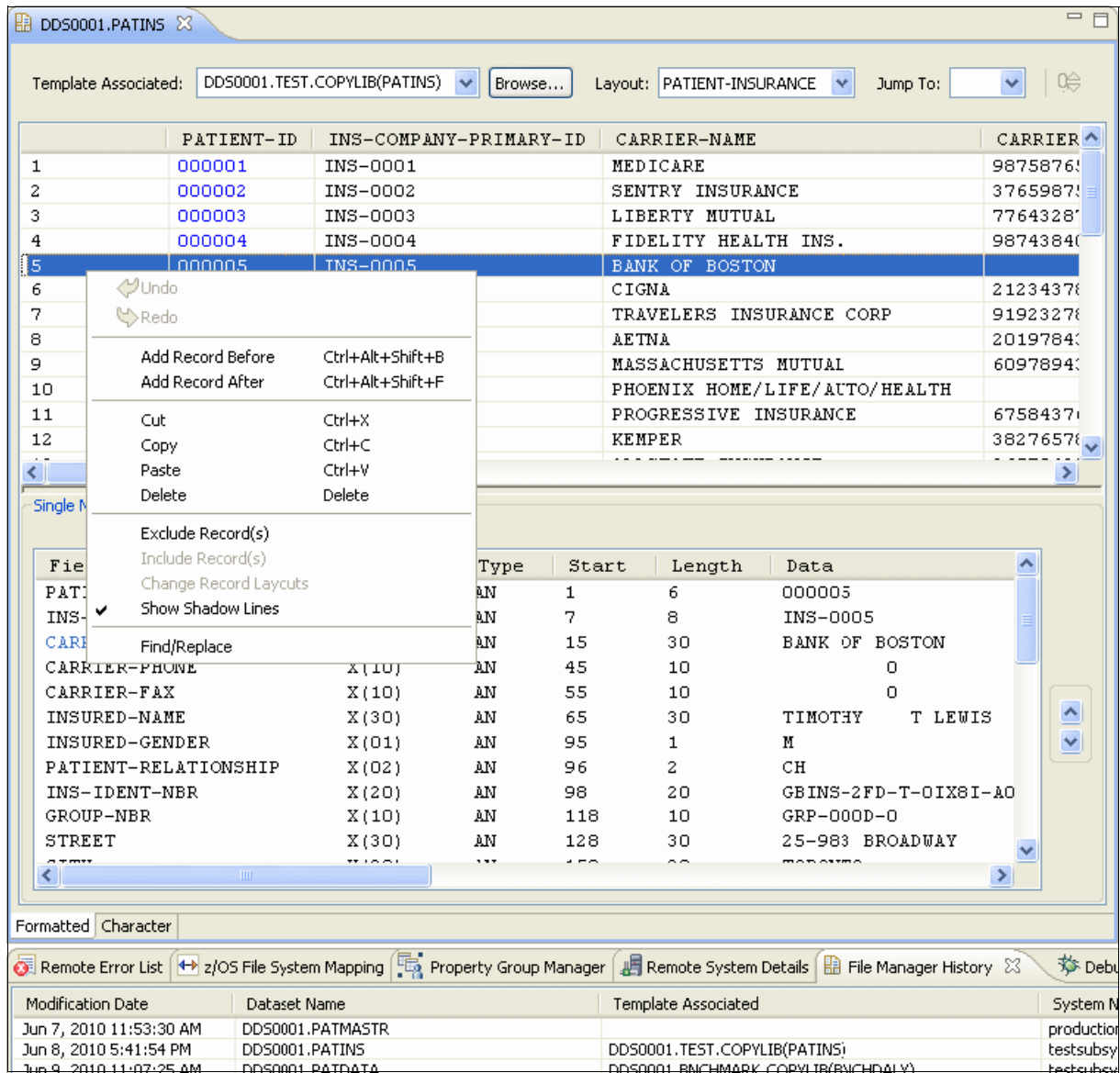


Figure 6-19 File Manager integrated with Rational Developer for System z

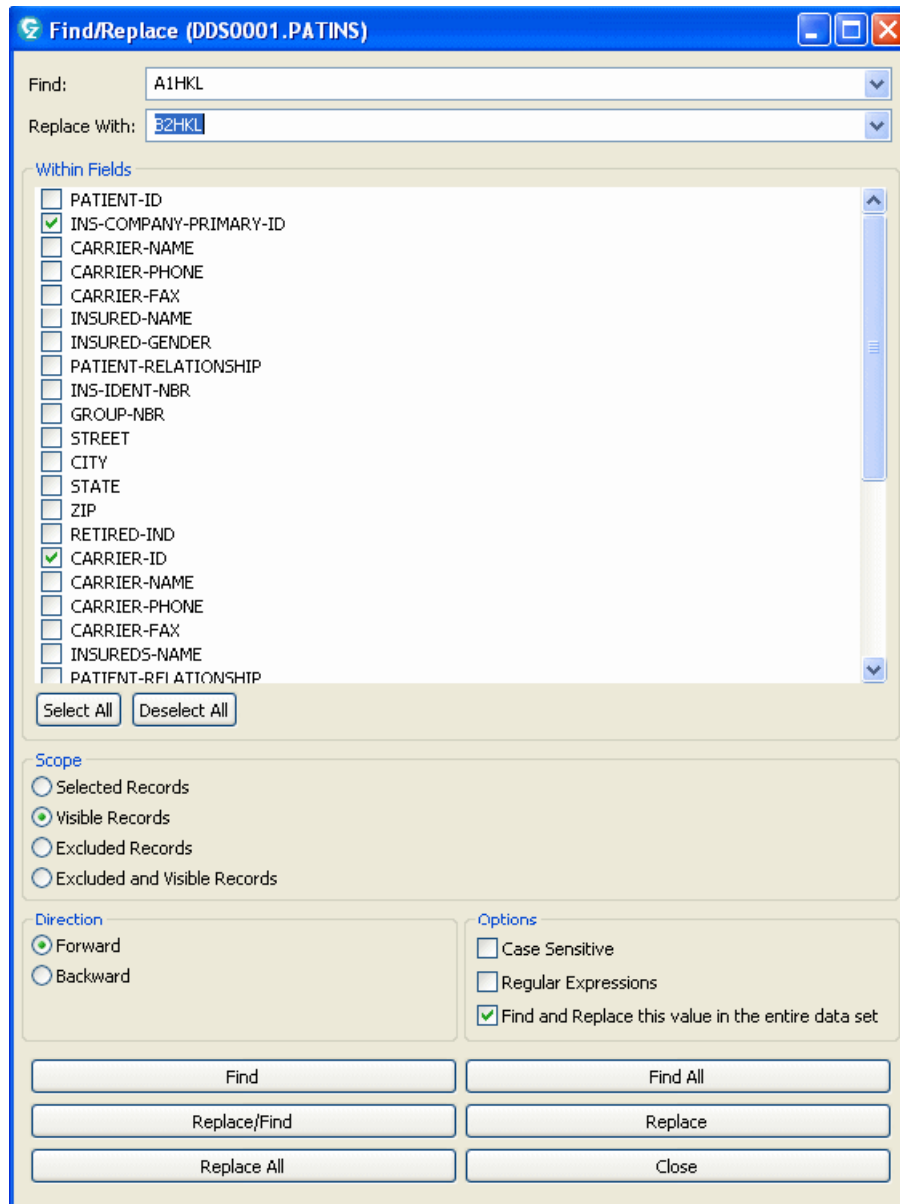


Figure 6-20 Search and replace through logical fields by scope and options

Fault Analyzer for z/OS

Fault Analyzer for z/OS provides the information that you require to determine the cause and assist with the resolution of application and subsystem failures. You can use this tool to assist in composite-application abend analysis. It helps you repair failures quickly by gathering information about an application and its environment at the time of failure.

Fault Analyzer provides the following features:

- ▶ Provides support for analyzing batch, IMS, and CICS online application and system failures with debugging facilities for all of the IBM mainstream online files and databases:
 - IMS DL/I, DB2, VSAM, IBM Identity Management Services (IDMS), and so forth
 - WebSphere Application Server for z/OS system failures
 - WebSphere MQ application failures
 - Batch (QSAM, VSAM, and DB2) application failures

- ▶ Helps you analyze failures when they occur or re-analyze them later.
- ▶ Expands error messages and codes that apply to your failure with interactive re-analysis and includes a feature for using application-specific messages and codes to supplement those messages and codes that are supplied by IBM.
- ▶ Creates a fault history file with an interactive display that helps you track and manage application failures.
- ▶ Starts automatically when an application fails, eliminating the need to recompile programs or change the JCL.
- ▶ Integrates with Rational Developer for System z to enable developers to diagnose application problems without changing their user interfaces.

Figure 6-21 describes how Fault Analyzer operates. When an application abnormally ends (abend), a Fault Analyzer listener on z/OS immediately correlates key information about the abend into a Fault Analysis report (Figure 6-22 on page 164). Fault Analyzer also creates a history file with critical Fault Analysis information and DUMP data, including registers. Developers use this history file when solving abends. Managers use this history file to track the health of an application (traceability).

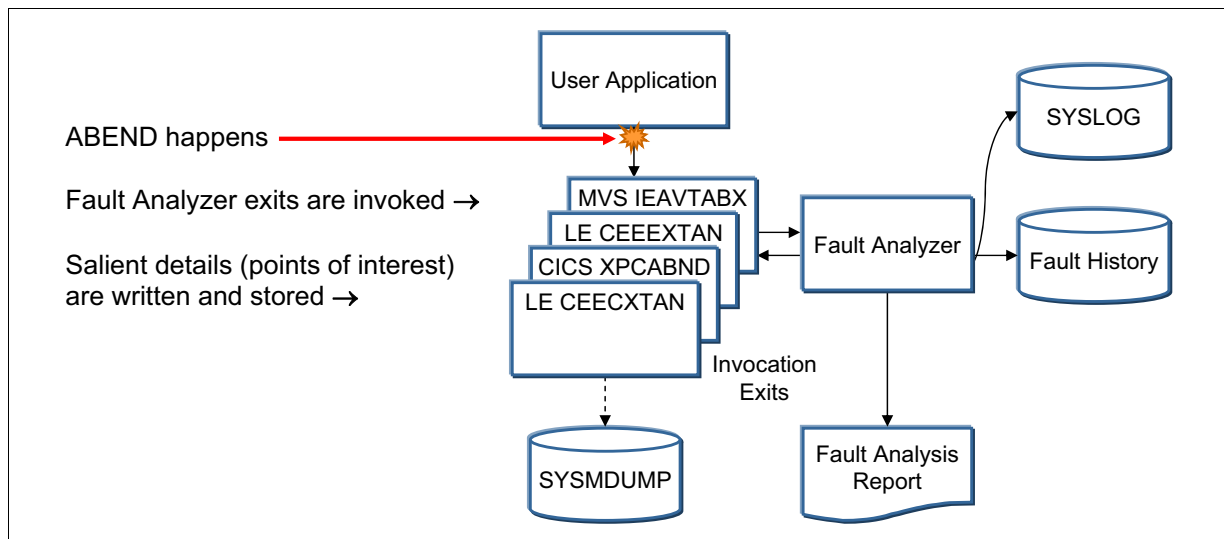


Figure 6-21 Fault Analyzer process at run time showing events and Fault Analyzer artifacts

The Fault Analyzer plug-in, as shown in Figure 6-22 on page 164, within Rational Developer for System z or the CICS Explorer interface provides access to problem reports for diagnosing mainframe application errors and abends. The Fault Analyzer report describes the following types of information:

- ▶ The nature of the abend
- ▶ An abend lookup document, which describes how and, more important, why common runtime abends occur, and which gives a little assistance in identifying the root cause
- ▶ The load module, job, or transaction that was running when the abend occurred
- ▶ The program in which the abend occurred
- ▶ The statement that forced the abend
- ▶ Any variables and variable values that are considered evidence or points of interest in understanding the nature of the abend
- ▶ Hyperlinks back to the program source for ease of use

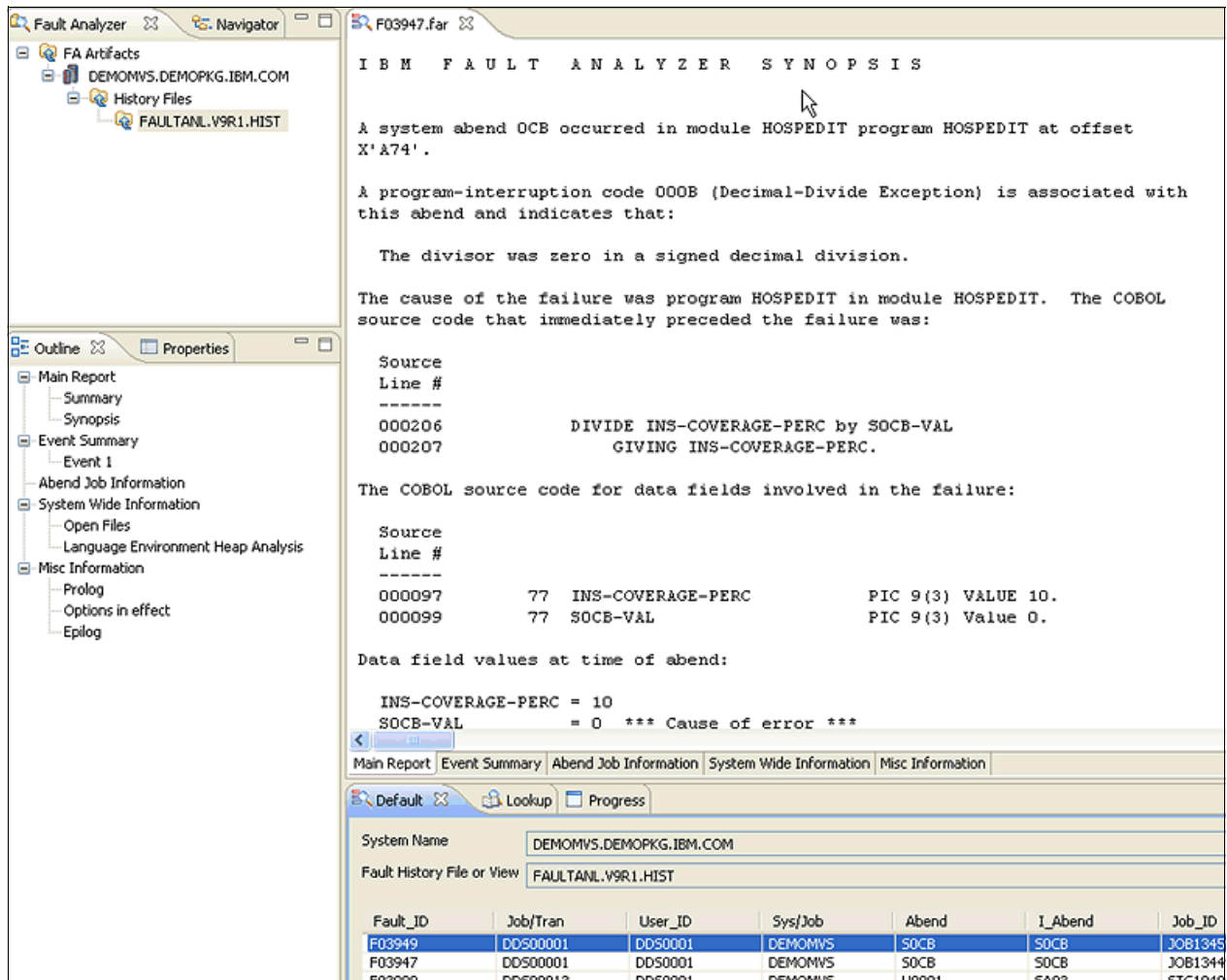


Figure 6-22 Fault Analyzer perspective and report showing why (and where) an abend occurred inside a program and formatting the necessary information to respond to the abend situation

Fault Analyzer allows the less-experienced members of your staff to work productively with application abends.

For more information about problem determination tools, see chapter 6, *z/OS Traditional Application Maintenance and Support*, SG24-7868.

6.2.3 CICS Performance Analyzer for z/OS

IBM CICS Performance Analyzer (CICS PA) for z/OS is a performance reporting and analysis productivity tool that helps the CICS system and application specialists explore historical performance and statistics information about their systems and applications by using SMF data collected from the CICS Monitoring Facility, CICS Statistics and Server Statistics, CICS Transaction Gateway Statistics, IMS, DB2, WebSphere MQ, System Logger, and Omegamon XE.

Figure 6-23 shows the inputs and components in the CICS Performance Analyzer environment.

Many of today's large CICS customers run mission-critical applications that traditionally experience high transaction per second processing rates. CICS PA gives you access to reports and graphics that help you identify, analyze, and resolve performance bottlenecks, along with the ability to generate extracts that can be imported into spreadsheet tools for analysis on your workstations.

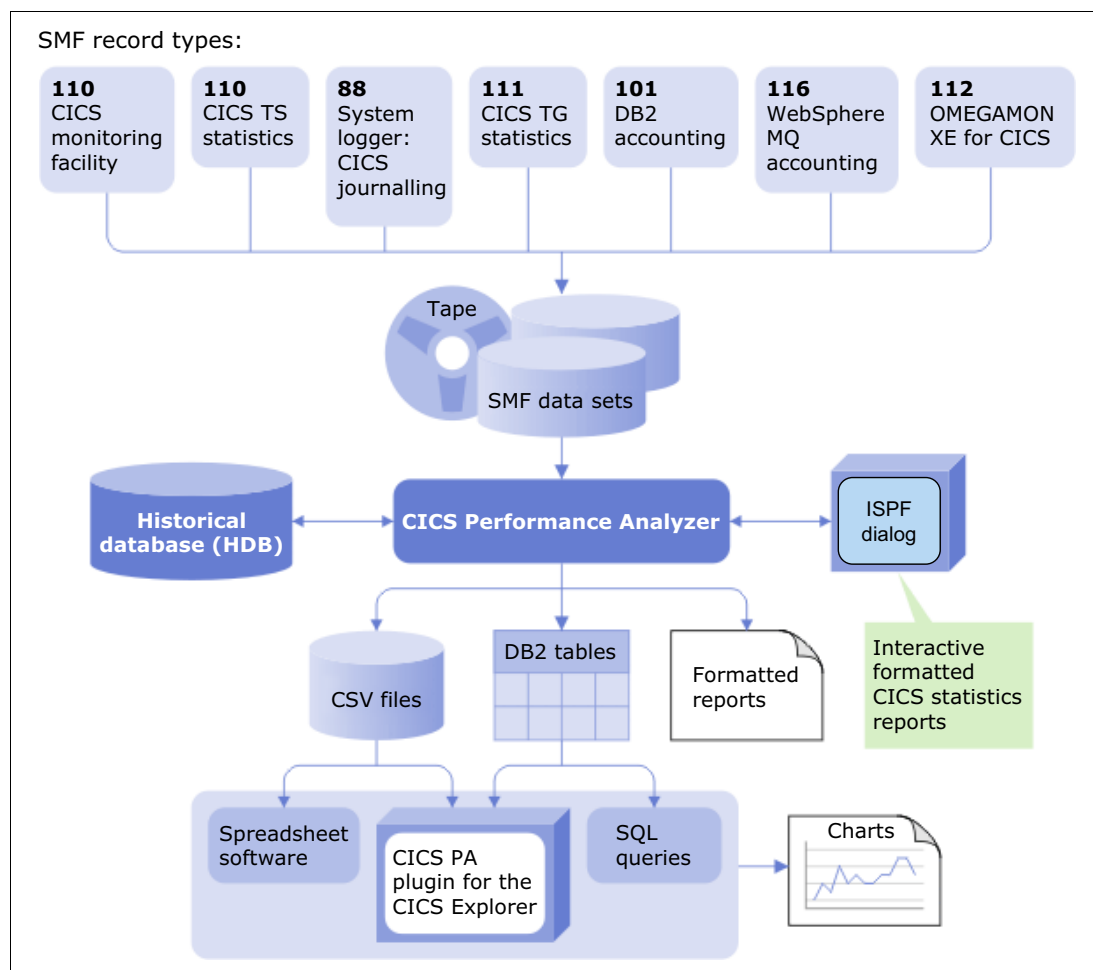


Figure 6-23 CICS Performance Analyzer components

CICS PA provides many advanced features, such as Cross System reports that combine your performance information from multiple CICS regions to give you the complete picture of your multiregion transactions. You may also group transactions, or use transaction profiling to compare styles from before changes are made to after changes, to identify changes over time periods, such as before and after an upgrade.

The CICS PA plug-in, shown in Figure 6-24, provides access to many CICS performance metrics, enabling analysis of CPU, response time, TCB usage for threadsafe, memory, VSAM file, DB2, and WebSphere MQ usage. Contextual linkage is available from the CICS Explorer and CICS Configuration Manager resource views to tabular and graphical performance views, and from performance views to relationship data provided by CICS Interdependency Analyzer. Performance data can be accessed by using downloaded CSV files or by direct access to DB2 data sources.

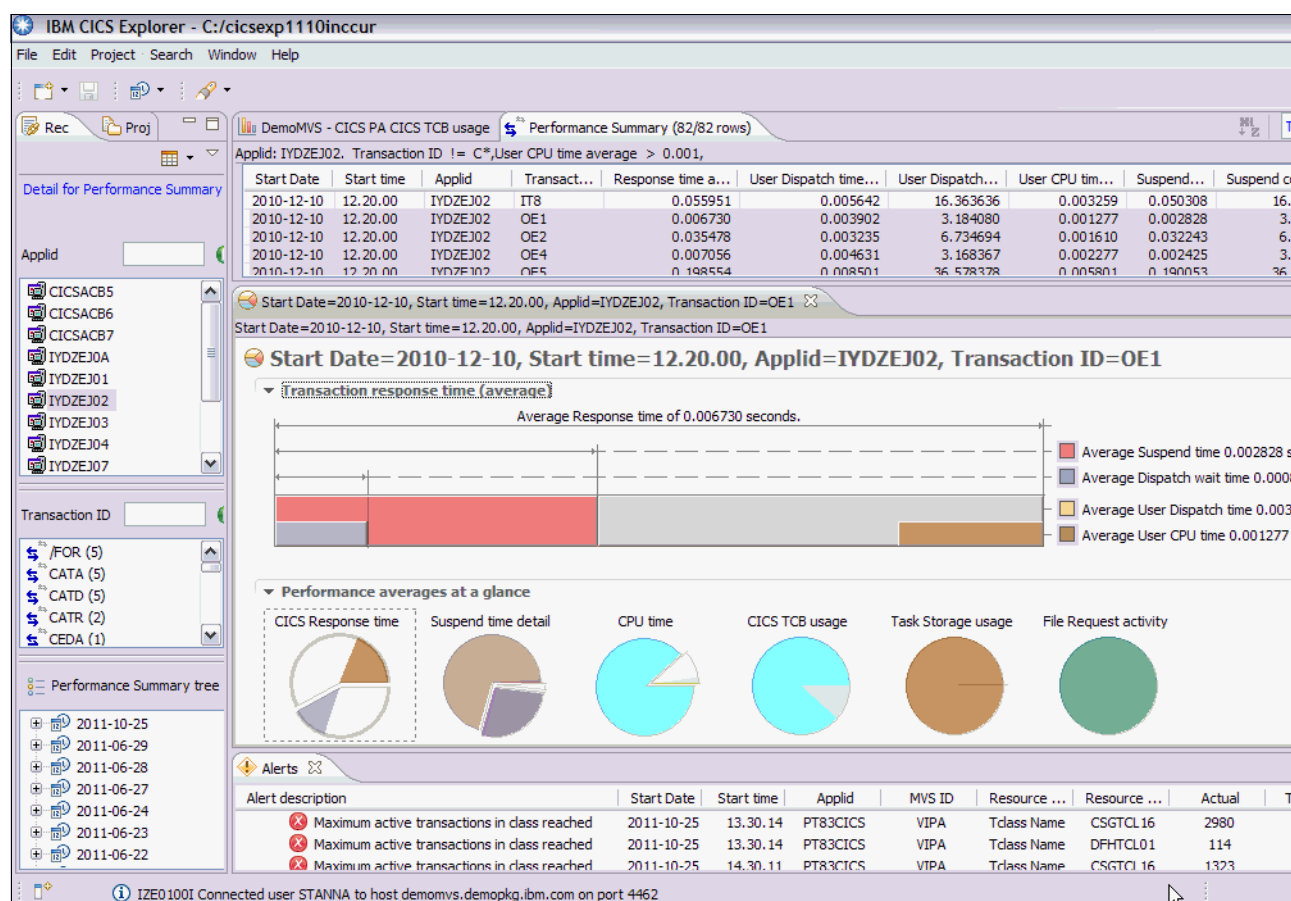


Figure 6-24 View of the CICS Explorer Performance Analyzer plug-in

6.2.4 CICS Application Performance Analyzer for z/OS

In the previous section, we described CICS Performance Analyzer and what it offers to assist analysts while identifying, analyzing, and resolving performance bottlenecks considering the overall system performance.

Eventually, after the system performance bottleneck is identified, the analysts might need to look at the CICS transactions, seeing more detailed reports to learn how to address a given performance issue. CICS Application Performance Analyzer (APA) is a non-intrusive performance measurement and analysis system that helps to resolve issues with transaction performance, providing functions that facilitate the isolation of performance problems in transactions, whether those transactions are applications, subsystems, or tasks. It provides performance statistics on any transaction that analysts want to monitor. These statistics can be the current system data, data for scheduled future jobs, or data collected over a certain time period. By using CICS APA, analysts can immediately focus their activities on tuning specific areas of an application, thereby improving productivity and meeting the challenges

demanded by business applications. Figure 6-25 shows CICS Application Performance Analyzer environment.

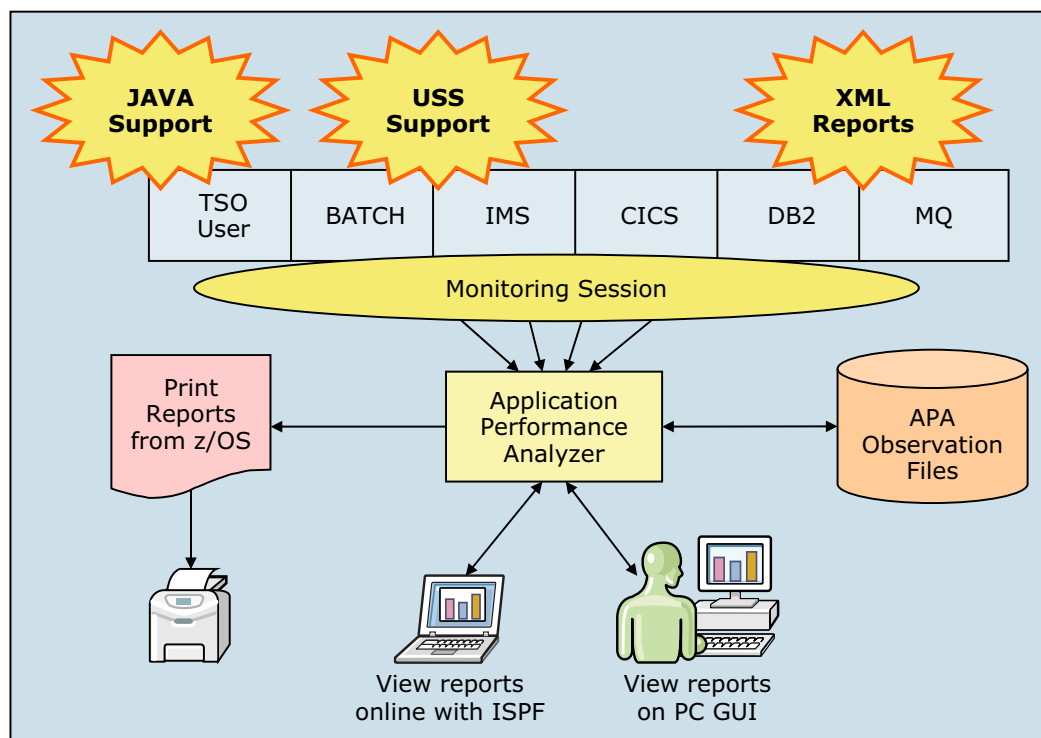


Figure 6-25 CICS Application Performance Analyzer environment

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

In addition to many performance measurements functions in CICS APA, it offers various other capabilities:

- ▶ Optimizes the performance of existing application resources.
- ▶ Increases application understanding during stress and regression testing.
- ▶ Provides maximum flexibility, with support for Assembler, C/C++, COBOL, PL/I, DB2, CICS, IMS, and Web Sphere MQ technologies. APA also provides support for Stored Procedures written in Java and for Java running in a CICS region.
- ▶ Eliminates excessive I/O activity and CPU time to increase response times.
- ▶ Improves response time of online transactions and batch turnaround times.
- ▶ Identifies code bottlenecks during initial testing and isolates performance problems in existing applications.
- ▶ Integrates with the Fault Analyzer and Debug Tool with source mapping slide files.

6.2.5 CICS Configuration Manager for z/OS

In a CICS environment, the CICS systems programmer or administrator has two choices for how to manage application and system resources:

- ▶ CICS System Definition (CSD)
- ▶ CICSplex SM Business Application Services (BAS)

IBM CICS Configuration Manager for z/OS (CICS CM) takes both methods to a higher level of management by providing auditing and reporting, controlled back-out, redundancy and cold start analysis, packaging and approval support all in a secure managed environment.

As the number of CICS regions increases, administrators can experience difficulties trying to migrate application changes as they are promoted through the various development, testing, and production environments. CICS CM simplifies the process by allowing you to deploy changes automatically and dynamically from a single point-of-control.

CICS CM interacts seamlessly with both CSD-defined resources and CICSplex SM BAS resources by automatically translating stored resources into their correct form and structure when passing from one repository type to another. Figure 6-26 shows how CICS CM integrates into the CICS environment.

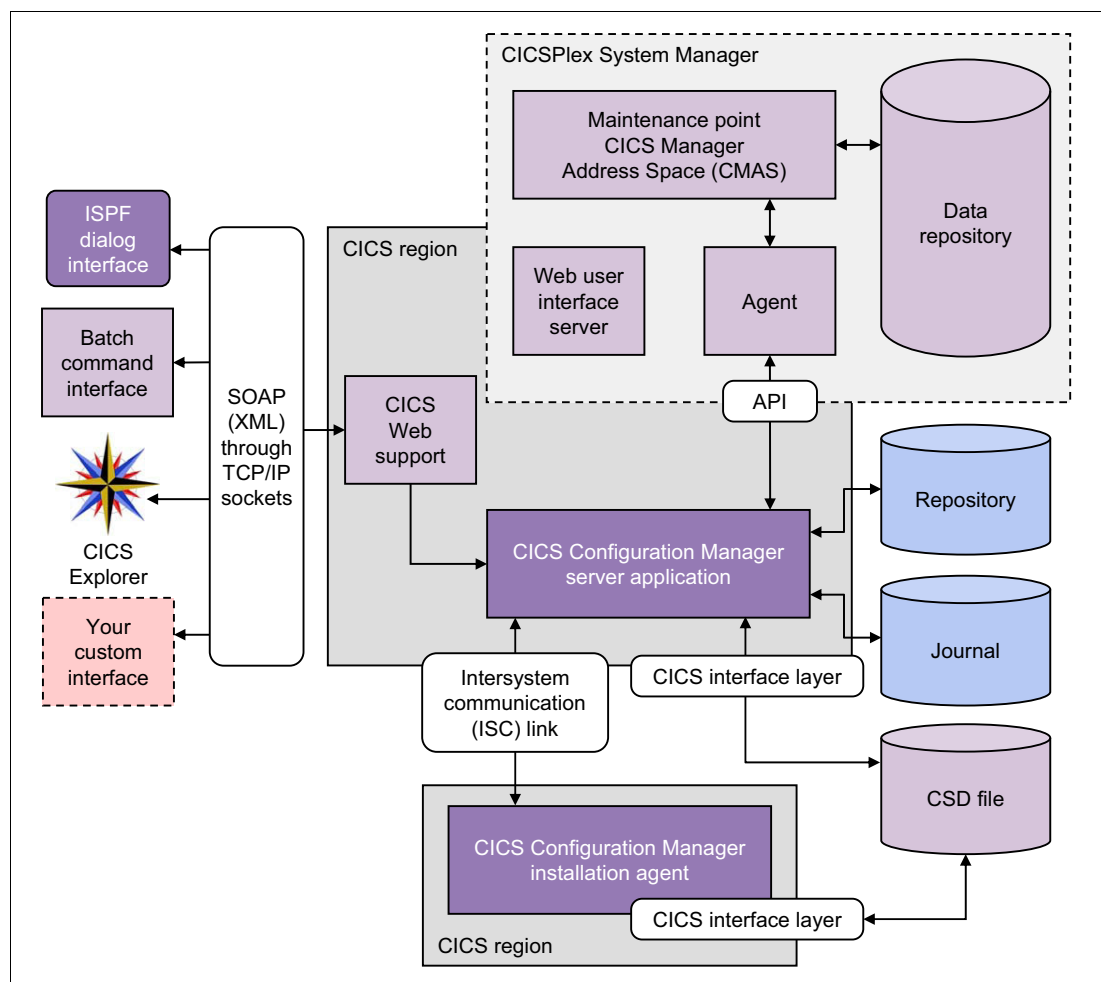


Figure 6-26 CICS Configuration Manager architecture

CICS CM provides many usability features, such as security and standards enforcement, where you limit who can see or modify resources. You can also enforce how changes are made, such as implementing naming or standards for definition attributes. CICS CM also provides features, such as *change packages*, which can identify a set of resource definitions that you want to process together, or it can contain a set of commands that you want to process together, or both. You can process a change package to migrate resource definitions, and the commands to **ADD**, **DELETE**, or **REMOVE** resources during migration between configurations. The following list indicates other usability features:

- ▶ **Migration schemes:** These provide details of **SOURCE** and **TARGET** configuration repositories when promoting tested CICS resources definitions and programs from testing to production environment.
- ▶ **Transformational rules:** Each migration path in a migration scheme can see a set of transformation rules. During migration, CICS Configuration Manager uses these rules to transform candidate resource definitions that apply to the source CICS configuration of a migration path.
- ▶ **Approval profiles:** When CICS Configuration Manager users issue the command to approve or disapprove a change package, they specify the approver role that they are representing, to give full control over every phase of the definition, migration, and installation process. In this way, you can distribute the management of your CICS resources to include development staff into the process.

The CICS Explorer Configuration Manager (CICS CM) plug-in, shown in Figure 6-27 on page 170, provides an Eclipse-based infrastructure to view and manage CICS CM resource definitions across an enterprise. It supports a subset of the function available in CICS CM.

When you open the CICS CM plug-in for the first time, the following default views are displayed:

- ▶ The **Configurations** view displays all CICS CM configurations, both those representing CSDs and those representing CICSplex SM Contexts. The configuration information is retrieved when you connect to CICS CM and contains details of the CSD or context that the configuration represents.
- ▶ The **Lists** view displays all lists in the currently selected configuration. For a configuration that represents a CSD, lists represent the Group Lists in the configuration. For a configuration that represents a CICSplex SM Context, lists represent the ResDescs in the configuration.
- ▶ The **Groups** view displays all groups in either the currently selected configuration or the currently selected list. Again, for a configuration that represents a CSD, groups represent the csdgroups. And for a configuration that represents a CICSplex SM Context, groups represent the ResGroups.
- ▶ The **Search Results** view displays search results on resource definitions.
- ▶ The **History** view shows changes made to definitions.

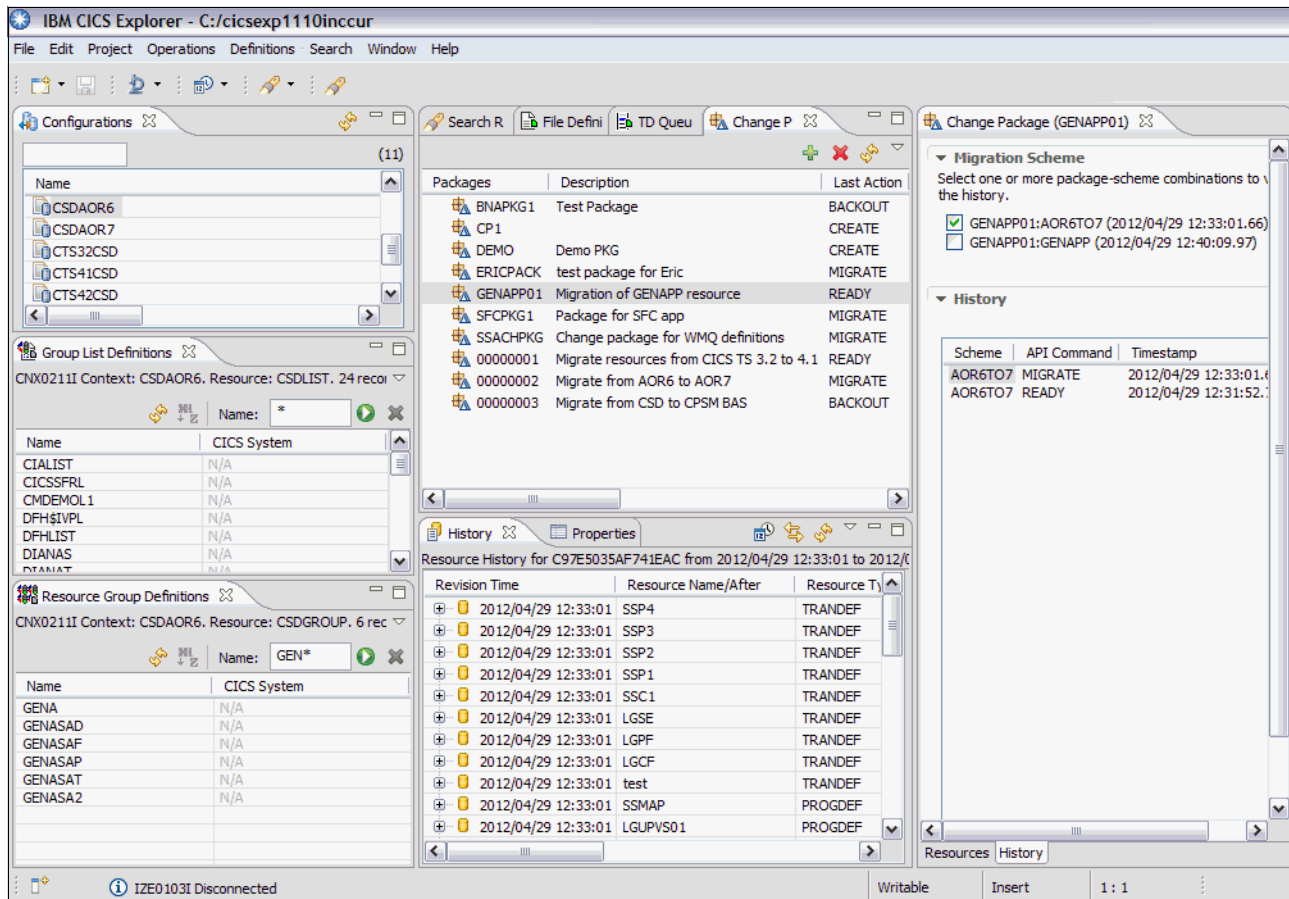


Figure 6-27 View of CICS Explorer Configuration Manager plug-in

6.2.6 CICS Deployment Assistant for z/OS

CICS systems programmers or administrators can use CICS Deployment Assistant (DA) from within the CICS Explorer to discover, model, visualize, create, and deploy CICS regions using the CICS DA plug-in interface.

Using the CICS Explorer, you can view resources in a CICS region or in a CICSplex. Adding the CICS Deployment Assistant extends the scope outside the CICS environment into the z/OS arena, giving you a complete picture of every aspect of your CICS environment and providing you with a complete topology of your CICS environment.

CICS DA provides a discovery feature that uses a two-phase process to discover information about running CICS regions, and then drills down into each CICS region to discover detailed information, such as startup options, connectivity relationships, and major subsystem dependencies. After the discovery process is complete, you can save an offline model in an XML format that can be shared with other team members.

Figure 6-28 shows the CICS DA plug-in inside CICS Explorer displaying a CICS region, with the Plex selection information available on the left, and the regions MSGUSR log at the bottom.

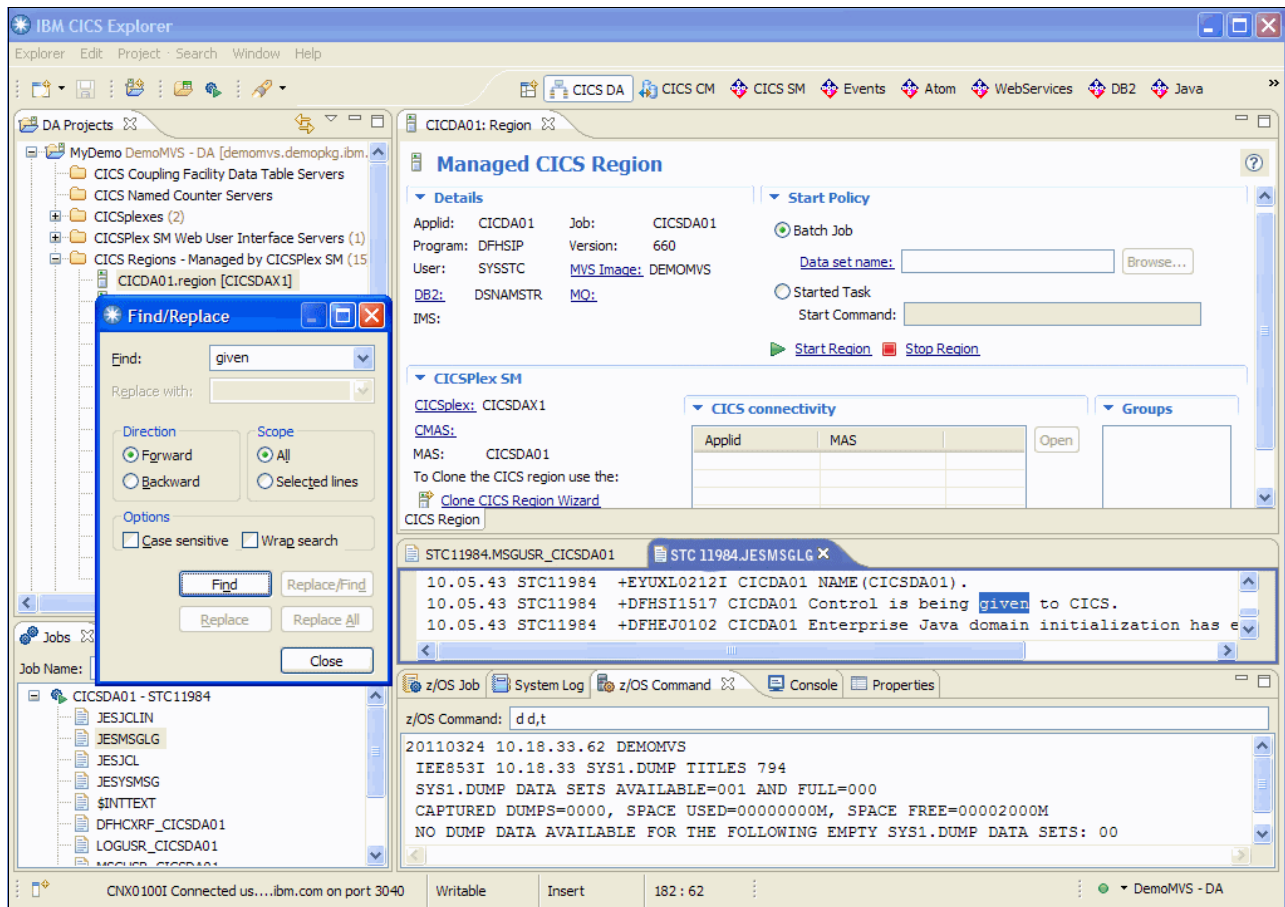


Figure 6-28 CICS Deployment Assistant

The offline models can provide a graphical visualization of your CICS topology that integrates with the CICS Explorer to help you visualize and navigate through the topology using the CICS SM operations and views. CICS DA also provides an operational function so you can start and stop CICS regions, and view their associated job logs and system logs.

By using the automations capabilities of CICS DA, you now have the functionality to clone existing CICS regions and add CICS regions to your CICSplex.

With CICS DA, you can create guided help documents referred to as *cheat sheets*, viewable from the CICS Explorer that you can use to create custom operational process and procedure documentation.

Using a CICS DA plug-in, you can also do the following tasks:

- ▶ Discover connectivity relationships and major subsystem dependencies.
- ▶ View held and running z/OS jobs and the system log.
- ▶ Clone existing CICS regions by creating and submitting JCL directly from CICS Explorer.
- ▶ Add an existing CICS region to a CICSplex.

6.2.7 IBM Session Manager for z/OS

IBM Session Manager is a session management product that provides secure and reliable access to multiple applications from a single 3270 terminal session. The 3270 terminal session can exist on either a Systems Network Architecture (SNA) or a Transmission Control Protocol/Internet Protocol (TCP/IP) network. That terminal session can simultaneously connect to and navigate between multiple Virtual Telecommunication Access Method (VTAM) and TCP/IP applications.

Session Manager offers the following features:

- ▶ Simultaneous access to multiple applications with the ability to switch between applications through escape keys
- ▶ User exit facility that can monitor and modify 3270 data streams, enhance Session Manager's standard actions in various situations, and enable connection to an external security manager using standard System Authorization Facility (SAF) interfaces
- ▶ Personal computer (PC) or Common User Access (CUA) type windowing
- ▶ Saving and recalling individual screens
- ▶ Cutting and pasting between sessions or within the same session
- ▶ Three types of terminal messaging
- ▶ Creating a hardcopy image of the current screen
- ▶ Viewing the contents of a user's screen by another user
- ▶ Providing multiple users with the ability to view the contents of a single user's screen
- ▶ Recording and replaying a sequence of user and terminal interactions
- ▶ Collecting and viewing Session Manager performance statistics and session level response time statistics
- ▶ Reducing network data traffic through 3270 data stream compression techniques
- ▶ Networking two or more Session Manager systems
- ▶ Using a supplied CICS transaction that allows access to Session Manager via CICS
- ▶ Panel and script language for adding logic to installation-specific customizations such as automating the sign-on process to applications and building new applications that combine information from multiple existing applications and present the results on a single panel

We describe several of these features in more detail and how you can take advantage of them in certain situations.

Simple and secure session-management

Many business tasks require users to switch among a variety of applications on different systems. For example, when taking orders, users need to validate client information, check client credit history, and determine whether items are in stock and when delivery is possible. Users might also need to arrange client financing and payment schedules. And because users must keep track of all the applications and systems they work on, becoming proficient at completing complex transactions like these can be difficult and time-consuming.

Figure 6-29 illustrates the IBM Session Manager configuration environment.

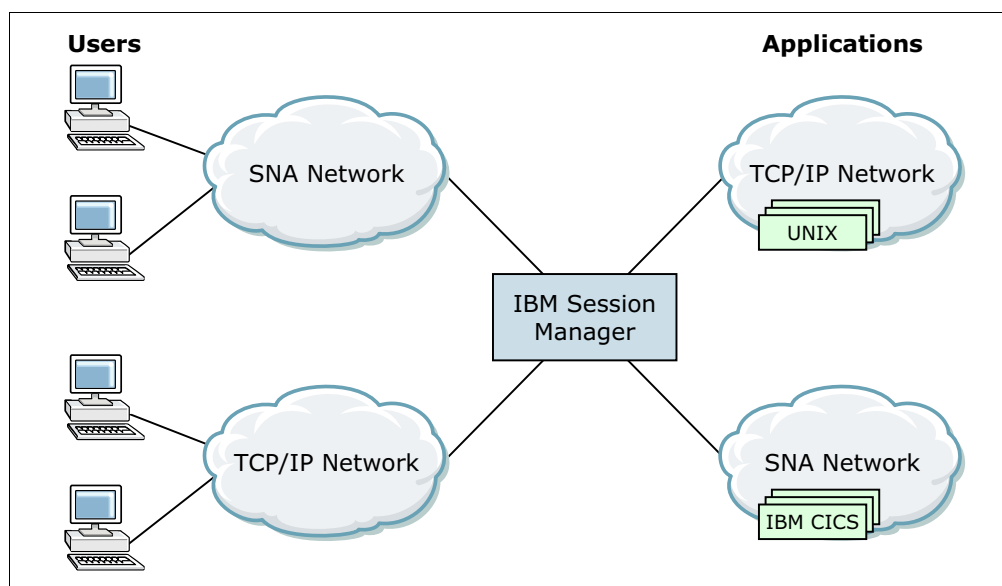


Figure 6-29 IBM Session Manager configuration

IBM Session Manager for z/OS provides IBM VTAM software users, as well as TCP/IP users, with security-rich, highly-available, and user-friendly access to multiple IBM z/OS systems from a single 3270 terminal. A password or passphrase-protected menu means that users are within a few keystrokes of any business application from multiple, concurrent virtual sessions. System details are transparent to users; users can access the information they need without having to know where the applications are or what system the applications run on.

IBM Session Manager provides a consolidated view of a user's business application network from a single workstation, helping to reduce system administration time and provide quick access to a variety of applications. Because users work with applications from one easy-to-use menu, training requirements can be minimized.

Increase ease-of-use and productivity

IBM Session Manager can help users work more efficiently. Users can connect to a critical application or switch between applications with a single keystroke or simple command. With the windows facility, you can view several sessions simultaneously, and position, move, or resize windows anywhere on the display. IBM Session Manager also helps improve communications efficiencies.

Messaging capabilities allow users to send messages to other users and groups, or to another terminal. Users can also copy and paste data from one session into the same or another session, save screen images for later reference with the push-pull facility, and create a permanent copy of the current screen image by entering the specified hardcopy command to help save time and effort.

Highly available access

Even with the superior reliability of IBM System z, chance failures can occur, however infrequently. Previously, if a failure occurred to the system that hosted IBM Session Manager, or if the system needed to be quiesced for maintenance, your users needed to reestablish their application sessions.

Now, the high availability and near-continuous operations support in Session Manager, together with the coupling facility in an IBM Parallel Sysplex environment, means your users simply reconnect to Session Manager and all application sessions are recovered automatically, minimizing lost time and reducing the risk of data loss. As users connect (or reconnect) to Session Manager, the Workload Manager (WLM) balances the distribution of users and sessions across available Session Manager instances.

Multiple Session Manager instances operating in a sysplex can use Session Manager Sysplex Networking to enable a number of facilities such as automatic user reconnection, sysplex group-wide BROADCAST and MSG commands, and sysplex menu provide a single-system view across the sysplex group. Sysplex panels are provided to enable the user to display users and nodes within a sysplex group, to enter sysplex group commands, and to view and navigate the sysplex group audit log. Other facilities that support operations in a Parallel Sysplex environment include workload balancing, configuration sharing, and sysplex group audit log.

Help Desk

The Help Desk facility provides help desk and service desk personnel with quick access to information about session manager users. The Help Desk facility provides a list of users who can be selected individually with rapid access to Session Manager commands such as Query, Message, Spy, or Cancel. The Spy facility, which enables help desk and operations personnel to easily view user problems, can help to reduce cost and effort associated with Incident Management. Additionally, it offers centralized user ID administration and the ability to broadcast messages to users.



Part 3

CICS system design

This part describes the patterns and techniques you can use to create CICS systems that provide the qualities of service that your business requires.



High availability and continuous operation

This chapter describes how the architecture of your CICS TS systems and applications can affect the availability of the services they provide to your customers, staff, business partners, and so on. It includes information to help you build systems that meet availability requirements of your enterprise. These following requirements are typical:

- ▶ High availability: Minimizing the number of unplanned outages and their impact
- ▶ Continuous operation: Minimizing the number of planned outages and their impact
- ▶ Disaster recovery: Providing the capability to resume operations as quickly as possible following a disaster

7.1 Availability requirements

This section provides introduces availability concepts and how they relate to CICS systems:

- ▶ Types of availability requirement
- ▶ Unplanned outages
- ▶ Planned outages

7.1.1 Types of availability requirement

The availability requirements for any service arise from the business requirements of the enterprise. They are often formally documented as service level agreements (SLAs). The categories of requirements are as follows:

- ▶ High availability

This category minimizes the number of unplanned outages and the impact of unplanned outages. Unplanned outages are service interruptions caused by unexpected events such as hardware failures, software failures, and accidents.

- ▶ Continuous operation

This category minimizes the number of planned outages and the impact of planned outages. Planned outages are service interruptions required for expected events such as running batch processes, upgrading hardware or software, and applying preventive maintenance.

- ▶ Disaster recovery

This category provides the capability to resume operations following a disaster which minimizes the time to recovery and the loss of data. Time to recovery obviously includes any delay before activating disaster recovery procedures. Because these procedures can be disruptive and sometimes risky, many installations require a formal process of declaring a disaster before proceeding.

Tip: We suggest you do not concentrate exclusively on production systems. Many enterprises that use CICS, especially large enterprises, need reliable systems for development, test, training, and so on. For example, test system outages can cause significant problems for rapidly fixing defects in production application code.

To meet the availability requirements of your enterprise, you typically need these items:

- ▶ Components that are reliable in themselves
- ▶ Spare capacity that you can use if and when a component fails
- ▶ Processes and technologies for fail-over

Reliable components

CICS TS systems automatically benefit from the exceptional reliability of the IBM System z hardware and software on which they run. For example, System z servers incorporate fault tolerance that minimizes the number of unplanned outages. They can detect problems with critical components such as processor cores, memory, power supplies, and so on, and switch over to spare components without affecting the running programs such as CICS TS transactions. They also incorporate facilities for concurrent maintenance and concurrent capacity growth that largely eliminate the need for planned outages.

Spare capacity

System z features on-demand capabilities that can help you provide spare capacity in a cost-effective way. For example, System z servers can be configured with spare processor capacity that the server activates automatically when and only when it is required. In this way, you can configure systems to transfer workload from a failed server to a surviving server. That surviving server automatically activates extra capacity to handle the extra workload.

Failover and failback

The following two primary configurations are for handling component failures in IT systems:

- ▶ **Active/standby:** This configuration uses two or more components. One component is active, and processing work. The remaining components are standby, and not processing work, until the active component fails.

Failover when the active component fails comprises switching work to one of the spare components. Failback when the component recovers comprises switching the work back to the recovered component.

- ▶ **Active/active:** This configuration uses a cluster of components and distributes work between them. All components in the cluster are active, processing work, until a component fails.

Fail-over when any component fails comprises redistributing work between the surviving components. Fail-back when the component recovers comprises redistributing work to include the recovered component.

Active/active configurations have many advantages. For example, they can require much less hardware to provide the same capacity¹ and they can be used to eliminate planned outages and also to handle component failure and recovery.

However active/active configurations can be much more complex than active/standby. For example, when the component is a transaction processing application, active/active configuration must be able to manage concurrent access to the same data. This way requires sophisticated locking and synchronization capabilities.

A compromise solution for CICS installations is to use an active/active configuration for optimum availability at the primary site and to provide a standby configuration at a remote site for disaster recovery.

7.1.2 Unplanned outages

The goal of high availability is to reduce or, where possible, eliminate the impact of unplanned outages. To address this type of requirement, you must identify the factors that cause or might cause unplanned outages in your installation. This section discusses the types of factor that we advise you consider.

Hardware failure or degradation

This issues is the most obvious for high-availability systems. But, modern enterprise-class IT hardware, especially System z, is reliable. Most installations experience few outages that are caused by System z hardware compared with other causes such as human factors.

You can use a variety of well known techniques and technologies, for example, RAID disks, to reduce the chances that a hardware failure causes a service outage and to enable rapid

¹ Active/active exploits aggregated spare capacity in the cluster. For example, if a cluster of five components can handle the required workload, a cluster of six provides fail-over capability for one component failure. It requires 20% spare capacity instead of the 100% spare capacity that an active/standby configuration requires.

recovery from an outage if one does occur. For a description of options that are available for CICS z/OS systems, see:

- ▶ 7.2, “z/OS Parallel Sysplex and CICSplex” on page 184
- ▶ 7.3, “Data availability” on page 188

System software failures

Modern enterprise-class System z software, especially z/OS itself and z/OS “middleware” such as DB2, WebSphere MQ, and CICS, is reliable. But even the most reliable software *can* fail.

You can take actions to reduce the chances that a system software failure causes a service outage and to enable rapid recovery from an outage if one does occur. Actions that are available for CICS z/OS systems are described in 7.2, “z/OS Parallel Sysplex and CICSplex” on page 184. For more information about handling failures in data management and messaging subsystems, see:

- ▶ 7.3.2, “Application data requiring high availability” on page 188
- ▶ 7.3.3, “WebSphere MQ” on page 191

Application program failures

There are well known development leading practices that apply to all software development. They help minimize programming errors and the outages that they can cause.

In transaction processing systems, such as CICS TS, many different transactions run at the same time, using the same data. These transactions can interact with each other in ways that are difficult to predict. Therefore a particularly valuable task is to test, if possible, CICS applications with realistic data and transaction rates before deploying them into production.

Also you can and should use CICS TS facilities to help protect both CICS TS itself and other transactions running on the same CICS system from application program errors. For more detail, see 7.4, “Application availability” on page 195.

Security and integrity failures

Security and integrity are often considered as separate from availability. But it is important to remember that a security breach can easily cause an outage; a denial of service attack is an obvious example. Equally an integrity failure that results in loss or corruption of data can easily cause an outage.

Consider using security facilities to restrict access even to data that is not sensitive or confidential and to systems that are not accessed from outside your enterprise. This approach can enhance availability.

Human factors

For many large and complex systems, human factors are the biggest single cause of unplanned outages.² Reducing the number and severity of human factors as much as possible is clearly important. To an extent, you can achieve this goal through training, but you must also consider ergonomics.

² Usually the human is one of the staff responsible for running the production system. But not always. One of the authors of this book witnessed a serious IT outage caused by a distinguished visitor mistakenly activating the fire alarm while trying to open the computer-room door.

As far as possible, ensure that your applications have graphical and command-line interfaces that follow these simple rules:

- ▶ The interfaces are easy to understand and use.
- ▶ Any interfaces that might be needed in an emergency are available and are easy to understand and use.
- ▶ Interfaces that create emergencies are not available.³

There is a wide range of reference material about how to achieve good software usability. It can be a valuable resource for building high-availability into your systems.

Non-IT component failures

Many external factors, such as building access control systems, communication links, and so on, can affect system availability and possibly cause outages. IT architects are not usually responsible for these factors. But the factors can sometimes influence IT architecture. For example, you might need to design your applications differently if you need to use an unreliable communication infrastructure.

Disasters

In the context of IT availability, a disaster is understood to be an event that destroys an entire IT installation or makes the installation unusable. These are events such as earthquakes, fires, acts of war or terrorism, and so on.

For disasters, the role of an IT architect is normally limited to planning how services can be restarted at a remote recovery site. The most challenging aspects are usually making the data that is required to restart available at the remote recovery site and redirecting communication from the destroyed site to the remote recovery site.

7.1.3 Planned outages

The high reliability of modern enterprise-class hardware and software, and the use of leading practices in application development can reduce the number of unplanned outages to low levels. The reduced number of unplanned outages can leave planned outages as the most significant interruptions to service availability.

The goal of continuous operation is to reduce or, where possible, eliminate the impact of planned outages. To address this type of requirement, you need to identify the factors that cause or might cause planned outages.

We do not attempt to provide a comprehensive list here, but we advise you to consider at least the items in the remainder of this section.

Hardware preventive maintenance and upgrades

System z servers and system storage incorporate facilities for concurrent maintenance and concurrent capacity growth that largely eliminate the need for planned outages. However, hardware changes typically require temporarily removing or deactivating a component. Because this can cause some reduction in capacity, it is prudent to schedule this activity for times of relatively low load.

Of course, installations that do not require continuous operation can schedule service and upgrades for times when the affected components are not in use anyway.

³ Although this rule might seem obvious, early versions of PC-DOS included the RECOVER command, which, when used, almost invariably creates an emergency.

System software preventive maintenance and upgrades

Some System z system software incorporates facilities for concurrent service. This way can reduce the need for planned outages. More generally, you can temporarily shut down and restart System z subsystems such as DB2 and WebSphere MQ without shutting down other software. By using this approach updating subsystems that CICS is using without shutting down CICS itself is possible. Some CICS application functions are likely to be unavailable while switching to the updated subsystem, but this service degradation is temporary. It does not necessarily cause a complete CICS outage.

System z Parallel Sysplex and CICSplex SM allow you to achieve true continuous operation of CICS systems. See 7.2, “z/OS Parallel Sysplex and CICSplex” on page 184.

Application program fixes and enhancements

CICS itself provides facilities for dynamically loading new and updated application programs without a service interruption. These facilities include the NEWCOPY and PHASEIN options of the **SET PROGRAM** command. With NEWCOPY, if any transactions are already running with previous version of the program the **SET PROGRAM** command fails; with PHASEIN, any new transactions use the new program but any transactions that are already running with a previous version of the program continue to use that previous version.

CICS applications typically interact with each other in complex ways; a change to one application or program can affect and require corresponding changes to other applications and components. CICS cannot by itself ensure that changes to multiple applications or application components are correctly coordinated. For example, when you are replacing a CICS BUNDLE that contains an OSGi bundle, you can install a new bundle at a different version and then update the program to use this without service interruption.

Also CICS applications can interact with applications on other platforms, including separate (not z/OS) operating systems and, in some cases, with systems that are managed by separate organizations. CICS TS V5.1 introduces the concept of a *CICS application*, which is the new base technology for managing the lifecycle of complex applications like this.

Application database changes

You can add new data components, such as DB2 tables, VSAM clusters, and so on without a service interruption.

Additions and changes to DB2 tables do not affect CICS directly but require rebinding applications that use, for example, new or changed columns. Depending on the changes, temporarily disabling these programs might be necessary when changing the table.

Be aware that this flexibility depends on good application design. For example, an application that selects entire rows rather than specified columns requires actual code changes when columns are added to or deleted from the table

To add or change other data, such as VSAM clusters, CICS operations must update the corresponding FILE resource definitions. This process includes temporarily closing the file. While the file is closed, application programs that use it will experience error returns from file accesses. Application design should include testing for these error returns and generating appropriate “retry in xx minutes” responses to their users. This technique can reduce the disruption that an actual system outage causes for users.

Batch

Enterprise application systems often include both online transaction processing (OLTP) and batch processing. In many installations, both types of processing need to access the same

data. This way can prevent them running at the same time; the OLTP system must shut down while the batch jobs run.

This type of application architecture depends on a period, the *batch window*, when OLTP services are not required. This is obviously not compatible with continuous operation. Increasing globalization, web and mobile access, and other factors tend to reduce the batch window even for enterprises that do not already require continuous operation.

A highly effective approach, if possible, is to reduce or eliminate dependencies on batch windows by reworking existing applications so that batch programs can run parallel with OLTP programs. Even if that is not possible, you should design new applications in a way that avoids the need for a batch window.

CICS applications and batch applications that access the same DB2 databases can run concurrently. This behavior requires that the batch applications use DB2 transactional interfaces in an appropriate way, that is, to maintain a consistent view of the data for the OLTP applications and to minimize interference that transactional locks cause to the OLTP applications.

CICS applications and batch applications that access the same VSAM data can run concurrently in a similar way. To achieve this behavior, CICS must use VSAM record-level sharing (RLS) and the batch application must use DFSMS Transactional VSAM Services (DFSMSStvs). If you are converting an existing batch application that does not use DFSMSStvs you must modify the batch application to use transactional interfaces in an appropriate way, as for DB2.

Distributed components

Modern application systems often include interconnected and interdependent components on a variety of platforms. These components communicate with each other using a variety of technologies including web services, asynchronous messaging (such as WebSphere MQ), RESTful HTTP, and so on. In this type of application architecture it can be difficult to upgrade software in one component without disrupting the availability of the overall service or parts of that service.

One common reason for this difficulty is that a software change in one component also changes the format of the data that the software exchanges with other components.

As an example, consider a program that runs on a large number of IBM AIX® machines. The program sends request messages to a CICS application. A change to the CICS application might change the request message format. Without a technique to handle this, it is necessary to upgrade the AIX program at the same time as the CICS application. Until then, the AIX program cannot use the CICS application; the service is down.

There are several techniques to handle this type of problem. Most depend on a version number that changes when the format changes. Depending on the protocol, programs can use this version number either on every request message or when establishing a connection. When the version number that the sender provides does not match the version number that the receiver expects the receiver can respond, for example, in one of the following ways:

- ▶ Return an error to the sending program. The sending program can report the problem to a local operator who can upgrade the program as required.
- ▶ Accept requests that use the old format. To do this, it can transform requests from the old format to the new format before processing. It might also need to transform responses from the new to the old format.
- ▶ Accept requests that use the old format and return an update notification to the sending program.

Tip: Design and implement this type of versioning *before* you need to use it.

Consider the role of an enterprise service bus (ESB) in interconnecting distributing components and enabling continuous availability across planned outages. An ESB can support version contact and routing to facilitate a coupling facility.

7.2 z/OS Parallel Sysplex and CICSplex

This section describes two technologies that can be used to establish the highest possible availability and continuous operation configurations for CICS systems:

- ▶ z/OS Parallel Sysplex
- ▶ CICSplex

7.2.1 z/OS Parallel Sysplex

System z servers and the z/OS operating system combine with specialized components such as coupling facilities and timer synchronization hardware to provide an advanced clustering capability referred to as *Parallel Sysplex*. Parallel Sysplex provides many benefits including almost unlimited scalability, and single system image. In this section, we concentrate on Parallel Sysplex as a technology for high availability and continuous operation.

Parallel Sysplex is also a base technology for the IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS®) family of offerings. GDPS is an integrated, automated application and data availability solution that provides the capability to manage the storage subsystem (or multiple subsystems) and remote copy configuration across heterogeneous platforms, automate Parallel Sysplex operational tasks, and perform failure recovery from a single point of control, thereby helping to improve application availability. GDPS is independent of the transaction manager (such as CICS TS, IMS, or WebSphere) or database manager (such as DB2, IMS, and VSAM) being used, and is enabled by means of key IBM technologies and architectures:

- ▶ Base or Parallel Sysplex
- ▶ Tivoli NetView for z/OS
- ▶ System Automation for z/OS
- ▶ Disk control units such as IBM System Storage® DS8000® that support:
 - Metro Mirror architecture for GDPS/PPRC
 - z/OS Global Mirror architecture for GDPS/XRC
 - Global Mirror architecture for GDPS/Global Mirror
- ▶ Peer-to-Peer Virtual Tape Server (PtP VTS) that supports Virtual Tape Server Remote Copy architecture

Figure 7-1 on page 185 is a simplified picture of a Parallel Sysplex consisting of three LPARs. Each LPAR runs its own copy of the z/OS operating system and its own WebSphere MQ queue managers, DB2 data managers, and other subsystems. The picture also shows several CICS regions that are running in each LPAR. All these CICS regions can access shared data on the same direct access storage devices and the services provided by a coupling facility.

Many basic high availability options are available, including RAID disks for the shared data and duplexing the coupling facility, or configuring it to use non-volatile memory. But for high availability and continuous operations, the most important capability of the Parallel Sysplex is the ability to shut down an individual LPAR and start a new LPAR without disrupting the overall availability of the system.

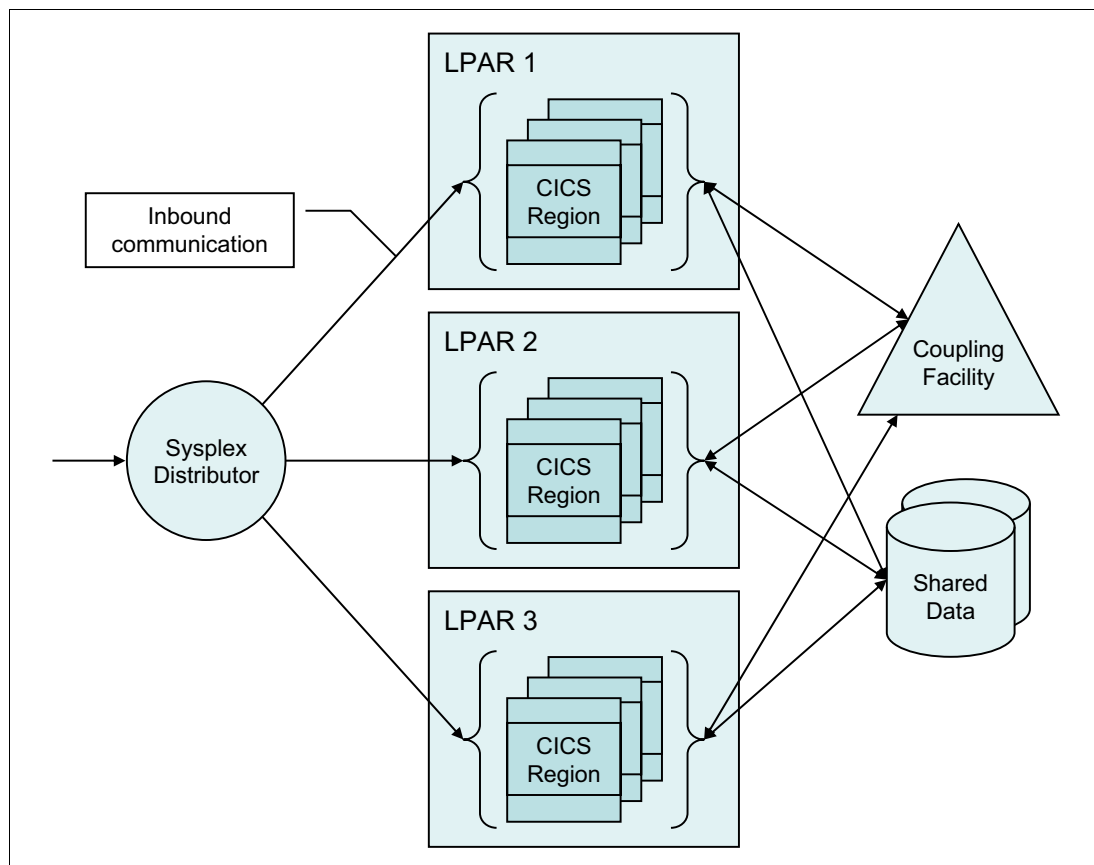


Figure 7-1 Parallel Sysplex

When a hardware or software problem causes a complete LPAR to fail, the sysplex distributor automatically redistributes new connections across the surviving LPARs. The surviving LPARs continue to provide the service. When the original problem is resolved, the failed LPAR restarts, and again sysplex distributor redistributes new inbound communication to include the restarted LPAR.

Tip: If persistent TCP/IP connections are used, then to ensure connections are evenly distributed to a restarted LPAR, the architecture should ensure that existing connections are periodically recycled.

This capability also allows you to upgrade server hardware or software or both. The following steps describe how:

1. Shut down LPAR 1. The remaining LPARs, 2 and 3, continue processing inbound communication so that there is no interruption to the service.
2. Upgrade the hardware or software, or both, for LPAR 1. This step could be, for example, to apply preventive service. Or, it could be to replace a complete System z server with a newer and more powerful model.

3. Restart the upgraded LPAR. Both the upgraded LPAR 1 and the original LPARs 2 and 3 now share processing of the inbound communication.
4. Repeat steps 1 on page 185 - 3, but this time shut down and upgrade LPAR 2.
5. Repeat for LPAR 3.

If your CICS application is designed so that multiple CICS regions running the application can share their workload in this way, you can use Parallel Sysplex to help achieve both high availability and continuous operation. See 7.2.2, “CICSplex” on page 186, which has more detail about running multiple CICS regions.

7.2.2 CICSplex

Multiple CICS regions can communicate with each other and cooperate to handle inbound work requests. This specialized type of cluster is referred to as a *CICSplex*. CICSplex provides many benefits including superior scalability, single system image, and so on. However, in this section, we concentrate on CICSplex as a technology for high availability and continuous operation.

Figure 7-2 is a simplified picture of a CICSplex consisting of five CICS regions. Separate regions in a CICSplex can perform separate roles, as the figure shows:

- ▶ One terminal owning region (TOR) is the connection point for terminals and distributes work to:
- ▶ Three application owning regions (AORs) run the applications that perform the work.
- ▶ One file owning region (FOR) provides transactional access to VSAM data

The figure also shows a DB2 region, which provides access to relational data. The CICSplex uses the DB2 region but the DB2 region is not part of the CICSplex.

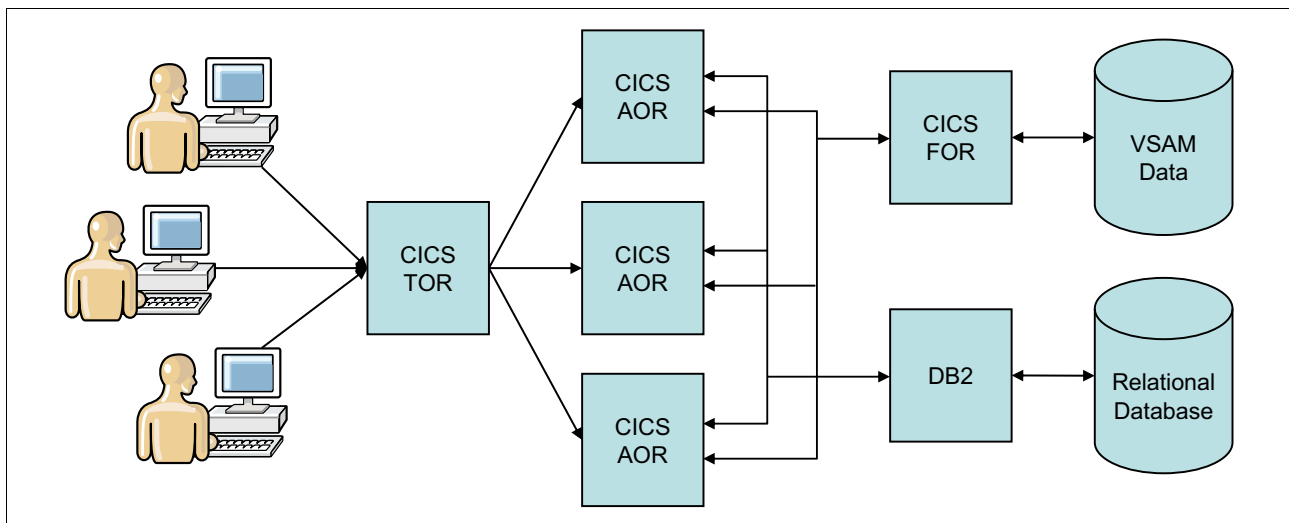


Figure 7-2 CICSplex

There are many other possible CICSplex configurations. You can, for example, include regions that perform other roles not shown in the figure, you can include multiple TORs and other CICS regions that act as communication “front ends,” and so on. Also not shown in the figure, CICS provides management options so that you can control the set of multiple regions from a single console.

For high availability and continuous operation, the following capabilities of the CICSplex are the most important:

- ▶ You can stop and start individual CICS regions without disrupting the overall availability of the services your CICS applications provide.
- ▶ You can isolate different components of your application systems in different CICS regions; see 7.4.3, “Isolating applications in separate CICS regions” on page 198.

Stopping and starting CICS regions in a CICSplex

When a hardware or software problem causes a CICS AOR to fail, the TOR or other communication front-end region can automatically redistribute inbound work between the surviving AORs. The surviving AORs continue to provide the service. When the original problem is resolved, the failed AOR restarts, the communication front end redistributes work to include the restarted AOR.

You can achieve similar resilience for other CICS regions, such as communication front end regions. However implementation details can vary depending, for example, on how external communication links connect to the CICS communication front end regions.

Two options: You have two options for distributing work within a CICSplex. You can develop your own distribution algorithms to suit unique characteristics of your applications or workload. Alternatively, you can use the standard workload management capabilities of CICSplex SM, which is part of CICS TS.

This same capability also allows you to upgrade CICS TS software or your applications or both. The following steps illustrate how you do this:

1. Select one AOR or group of AORs and suspend routing of new work to it. The remaining AORs continue processing new work so that service is not interrupted.

The standard CICS TS WLM facilities, include a useful capability that accommodates workload *affinities* where the same AOR must process successive requests from the same originator. You can suspend routing new work to a selected AOR or group of AORs but continue to honour affinity requirements.
2. Wait for all existing work to complete on the selected AOR or group of AORs. When complete, shut them down.
3. Upgrade CICS TS or application software, or both, for the selected AOR or group of AORs. This upgrade might be, for example, to apply preventive service, or it might be to replace a version of CICS TS with a newer version.
4. Restart the upgraded AORs. All AORs, including the newly upgraded AORs now share processing of new work.
5. Repeat steps 1 - 4 until all AORs are running with the upgraded software.

CICSplex technology complements z/OS Parallel Sysplex technology. Different CICS regions in a CICSplex can run in the same z/OS LPAR or in different LPARs. It extends the options for continuous operation by adding sophisticated application software upgrade capabilities.

7.3 Data availability

This section describes techniques and technologies you can use to enhance the availability of the data that your CICS systems and applications use. It contains the following topics:

- ▶ CICS data requiring high availability
- ▶ Application data requiring high availability
- ▶ WebSphere MQ

7.3.1 CICS data requiring high availability

Some CICS data sets must be available during restart after CICS fails:

- ▶ CICS system logs
- ▶ CICS system definition (CSD)
- ▶ CICS global and local catalogs

Use mirrored or other RAID disks for these. You may use the z/OS Data Facility Storage Management Subsystem (DFSMS) as a convenient way to manage this type of requirement together with other data storage requirements such as target access time. For detailed information about using DFSMS, including examples for CICS TS, see *z/OS V1R11.0 DFSMS Implementing System-Managed Storage z/OS*, SC26-7407-06.

7.3.2 Application data requiring high availability

In this section, we describe technologies and techniques that you might want to use to enhance the availability of application databases.

DB2 and data sharing groups

You use DB2 facilities to control the availability of DB2 data: where DB2 stores the data, what backup and restore capabilities you require, and so on. We describe the availability of the DB2 data managers that CICS uses to access the data.

In a Parallel Sysplex environment, DB2 can provide data sharing capabilities through the use of a DB2 data sharing group. A data sharing group consists of several DB2 data managers that can all access the same tables. The data managers can run in the same or in separate z/OS logical partitions. This way of running allows the various regions of a CICSplex to share data in the same DB2 tables.

From an availability perspective, a data sharing group has the important advantage of when one data manager in the group fails, other data managers can provide continued access to the data.

A facility referred to as *DB2 group attach* can provide additional availability benefits.

- ▶ Without DB2 group attach, each CICS region is configured to use a specified DB2 data manager.
- ▶ With DB2 group attach, each CICS region is configured to use any data manager in the data sharing group (with some constraints; see “DB2 in-doubt resolution” on page 189).

DB2 data sharing group support allows you to configure a CICSplex in various ways, as described in the following examples:

- ▶ One data manager in each logical partition, without CICS DB2 group attach

All CICS regions in a partition connect to their specified DB2 data manager. That DB2 manager must be running in the same logical partition.

When a CICS region in partition “A” fails, then any other CICS region in partition A continues to use its data manager without interruption. When a partition fails, any CICS region in another partition continues to use its data manager without interruption.

When a CICS region in partition A restarts in a partition “B” then it cannot access its DB2 data manager in partition A. It cannot access DB2 data unless and until its data manager also restarts in partition B.

When the DB2 data manager in partition A fails, then all CICS regions in partition A must wait for that data manager to restart before they can access the DB2 data.

- ▶ One data manager in each logical partition, with CICS DB2 group attach

All CICS regions in a partition connect to whatever DB2 data manager is running in the same logical partition.

This configuration is similar to the previous configuration except that a CICS region in one partition can fail and then restart in any partition where there is a data manager in the same data sharing group. It can use any data manager in the data sharing group to access the DB2 data.

- ▶ More than one data manager in each logical partition, with CICS DB2 group attach

Each CICS region in a partition connects any one of the DB2 data managers running in the same logical partition.

This configuration is similar to the previous configuration except in the following situations:

- When a CICS region in one partition fails and then restarts in the same region it might connect to a different DB2 data manager. This step can delay resolution of in-doubt units of recovery (see “DB2 in-doubt resolution”).
- When a DB2 data manager in one partition fails, CICS regions in that partition can connect to another data manager in the same partition and continue to access the DB2 data.

DB2 in-doubt resolution

DB2 uses locks to safeguard transactional integrity. For example, while a first transaction is updating a table, locks prevent a second transaction from “seeing” half-complete table updates. DB2 releases these locks during transaction-commit or transaction-rollback. DB2 data elements (such as table rows) locked for in-doubt units of work stay locked until the in-doubts are resolved.

There can be in-doubt units of work when the connection between a CICS region and a DB2 data manager fails. CICS and DB2 can resolve these in-doubts only when the same CICS region reconnects to the same DB2 data manager. Until that time, DB2 data elements remain locked, which can affect the availability of other CICS transactions that access the table.

Tip: When designing a CICS application that uses SQL, remember that the transaction locks that DB2 obtains can impact application availability when a data manager that is part of a data sharing group fails.

When you configure CICS to use CICS DB2 group attach, you allow CICS to reconnect to a different DB2 data manager after a failure. This behavior is not a problem when there are no

in-doubts; CICS can proceed with connection to any DB2 data manager. It is also not a problem when the same DB2 data manager is available; CICS can proceed with connection to the same data manager.

To handle the cases when there are in-doubts and the same data manager is not available, CICS provides the RESYNCMEMBER attribute of the DB2CONN definition. This attribute selects between the following options:

- ▶ If set to YES, CICS waits for the previous DB2 data manager to become available again and then reconnects to that data manager. This setting causes a delay before the CICS region can resume access to DB2 data but ensures that CICS resolves in-doubts as soon as possible.
- ▶ If set to NO, CICS connects to an available DB2 data manager, leaves the in-doubt transactions unresolved, and issues the warning message DFHDB2064. This setting ensures that the CICS region can resume access to DB2 data as soon as possible but causes an additional delay before CICS can resolve in-doubts.

DL/I

You use Information Management System (IMS) facilities to control the availability of DL/I data; that is, where IMS stores the data, what backup and restore capabilities you require, and so on. We describe the availability of the IMS/DB data manager that CICS uses to access the data.

In a Parallel Sysplex environment, you can use IMS data sharing. This sharing allows multiple subsystems within a sysplex to have concurrent access, with data integrity, to IMS databases. It works in a similar way to DB2 data sharing groups; CICS regions in an LPAR all connect to an IMS data manager in the same LPAR. The data manager can be either an IMS DBCTL subsystem or an IMS DB online subsystem. For a multiple LPAR solution, data managers use the IMS database recovery control (DBRC) together with the IMS resource lock manager (IRLM) subsystem, and the coupling facility to coordinate access to the shared data

CICS includes a capability to reconnect CICS regions to an IMS region when it restarts following a failure. However CICS does not support recovering from an IMS data manager failure by connecting to a different data manager. That is, CICS does not provide an IMS equivalent to the RESYNCMEMBER attribute of the DB2CONN definition.

VSAM

For VSAM clusters, CICS provides a transactional interface from a single CICS region. You can use a CICS file owning region (FOR) to provide the transactional interface to multiple application owning regions (AORs) running in a single LPAR.

VSAM record level sharing (RLS) provides a transactional interface to multiple CICS regions, including regions running on different LPARs. This is important for availability because a multiple LPAR CICSplex provides better resilience than a single LPAR. VSAM RLS uses an SMSVSAM region in each LPAR. AORs in the region connect to the SMSVSAM region instead of to an FOR.

In either case, you can use resilient storage devices for the VSAM data. DFSMS is a convenient way to manage this data; see 7.3.1, “CICS data requiring high availability” on page 188. Also consider using CICS VSAM Recovery, which provides various sophisticated facilities, including facilities for high availability and disaster recovery.

For some installations, DB2 can provide a better overall solution than VSAM. For these installations, CICS VSAM Transparency is a convenient way to migrate existing CICS VSAM data to DB2, without changing the CICS applications that access the data.

Other files and data sets

CICS applications can use z/OS files and data sets directly without using external resource managers such as DB2, WebSphere MQ, or IMS. These files and data sets include HFS and zFS files, and BDAM data sets.

CICS does not provide special high-availability options for the files and data sets. However you can use resilient storage devices for them. DFSMS is a convenient way to manage this; see 7.3.1, “CICS data requiring high availability” on page 188.

7.3.3 WebSphere MQ

In this section, we describe technologies and techniques that you might want to use to enhance the availability of systems that include CICS regions and WebSphere MQ queue managers.

How external queues affect availability: Persistence and expiry

Applications that communicate by using WebSphere MQ put messages onto queues and get messages from queues. WebSphere MQ maintains the queues and the messages on them independently from the communicating programs. This way allows the sending applications to put messages, while the receiving application (and even the system where that receiving application runs) is down. Similarly, receiving applications can get messages while the sending applications or systems are down.

With this capability, WebSphere MQ becomes particularly useful for *fire-and-forget* applications where the sending application does not need an immediate response from the receiving application or does not need a response at all. For these types of applications, you can use WebSphere MQ *persistent* messages. WebSphere MQ always writes persistent messages to disk so that it can assure delivery even after an outage.

Tip: You can use WebSphere MQ to achieve exceptionally high availability for some services. A fire-and-forget client application can use WebSphere MQ persistent messages for requests to a server such as CICS. The client is not aware of planned or unplanned server outages; it can send its requests anyway. The service is always available to the client, even when the server is down.

WebSphere MQ can also be useful for *request-response* applications where the sending application waits for the response message. For this type of application, the sender times-out the request message. That is, if it does not receive a response within a reasonable time, it assumes that the request never arrived. Therefore, you can use *nonpersistent* messages. You should also use WebSphere MQ *message expiry* so that WebSphere MQ can discard the message and the response when the sender times-out the request.

WebSphere MQ local and remote queues

WebSphere MQ makes a distinction between a *local queue* and a *remote queue*. When an application connects to a WebSphere MQ queue manager, queues which that queue manager owns are called local queues; applications can put to (MQPUT) and get from (MQGET) a local queue. Queues that other queue managers own are referred to as *remote queues*; applications can put to a remote queue but cannot get from it.

A CICS application that processes input messages from a WebSphere MQ queue must run in a CICS region that connects to the WebSphere MQ queue manager that owns the queue. You can improve availability by running the application in multiple CICS regions; when one region fails, surviving regions continue to process the messages.

If your application uses transactional interfaces, a CICS region can fail while processing WebSphere MQ messages. WebSphere MQ returns these messages to the queue so that another CICS region can process them.

Figure 7-3 shows an example configuration with three CICS regions connecting to the same WebSphere MQ queue manager. The queue manager owns a local queue. An application program can run in all three of these regions and get (MQGET) messages from the same local queue.

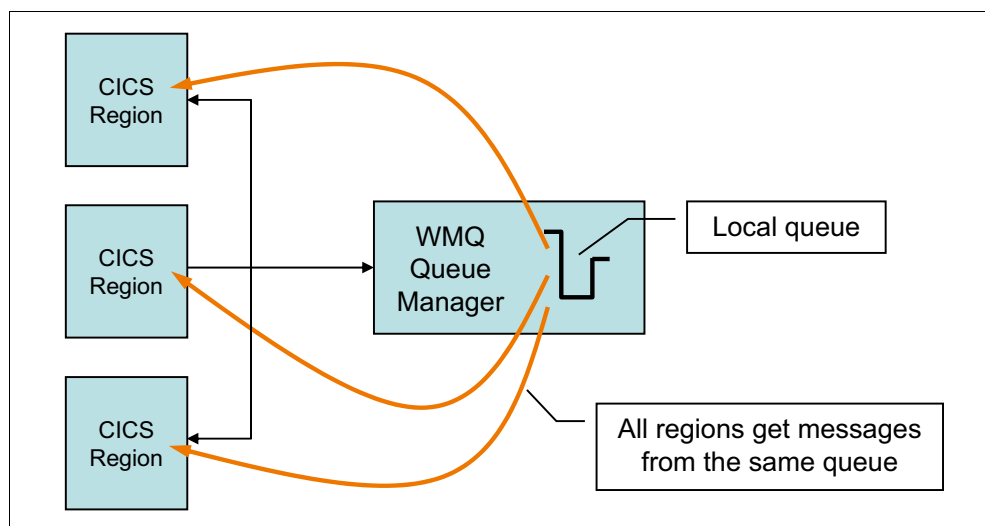


Figure 7-3 WebSphere MQ: sharing a local queue

Sharing queues: By sharing WebSphere MQ queues, you can configure CICS so that an application failure or a CICS region failure does not interrupt the application service. But you cannot easily control the order that applications process WebSphere MQ messages when you have more than one instance of the application running at the same time.

If the order of messages is critical then you must configure your application so that only one instance runs at one time. For availability, arrange to start a “standby” instance of the application when an active instance fails.

WebSphere MQ queue sharing groups, shared queues, and private queues

In a Parallel Sysplex, you can configure multiple WebSphere MQ queue managers as a *queue sharing group* (QSG). Queue managers in a QSG support two types of local queue: a *shared queue* and a *private queue*. The queue managers in a QSG cooperate to maintain and access shared queues in one or more coupling facilities. In this way, all the queue managers own a shared queue.

Different queue managers in a QSG can run in the same LPAR or in separate LPARs. An application running in any CICS region that connects to any queue manager in the QSG can put to and get from a shared queue that the QSG owns.

Figure 7-4 shows an example configuration with three CICS regions running in separate LPARs. Each CICS region connects to a queue manager. The queue managers are all in the same QSG and own a shared queue, which resides in a coupling facility. An application program can run in all three of the CICS regions and get (MQGET) messages from the same local queue.

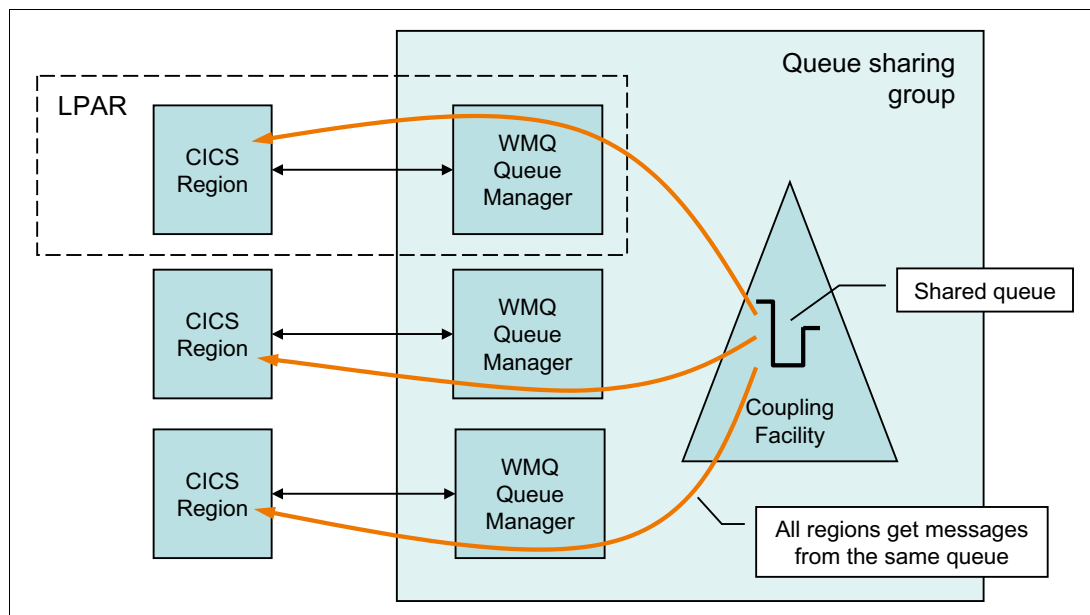


Figure 7-4 WebSphere MQ - sharing a queue in a QSG

Shared queues extend the high-availability and continuous-operation benefits of sharing queues. By using shared queues, you can configure an application so that an LPAR failure, or a planned LPAR outage for a software or hardware upgrade, does not interrupt the application service.

Notes:

- ▶ WebSphere MQ shared queues are not suitable for CICS applications that must process messages in the order they arrive at the queue.
- ▶ A queue manager in a QSG can own private queues and also shared queues. You can configure private queues so that every queue manager owns a queue with the same name. An application in a CICS region that connects to any queue manager in the QSG can access a private queue. But, the same application that is running in a CICS region that connects to a different queue manager, accesses a *different* private queue.

MQ group attach

A CICS facility known as an *MQ group attach* can provide additional availability benefits. Without MQ group attach each CICS region is configured to use a specified WebSphere MQ queue manager. With MQ group attach, each CICS region is configured to use any queue manager in the queue sharing group (with some constraints; see “WebSphere MQ in-doubt resolution” on page 194).

MQ queue sharing group support allows you to configure a CICSplex in a variety of ways:

- One queue manager in each logical partition, without CICS MQ group attach

All CICS regions in a partition connect to their specified queue manager. That queue manager must be running in the same logical partition.

When a CICS region in partition A fails, any other CICS region in partition A continues to use its queue manager without interruption. When a partition fails, any CICS region in another partition continues to use its queue manager without interruption.

When a CICS region in partition A restarts in a partition B, it cannot access its queue manager in partition A. It cannot access WebSphere MQ facilities unless and until its queue manager also restarts in partition B.

When the queue manager in partition A fails, all CICS regions in partition A must wait for that queue manager to restart before they can access WebSphere MQ facilities.

- One queue manager in each logical partition, with CICS MQ group attach

All CICS regions in a partition connect to whatever queue data manager is running in the same logical partition.

This configuration is similar to the previous configuration except that a CICS region in one partition can fail and then restart in any region where there is a queue manager in the same queue sharing group. It can use any queue manager in the queue sharing group to access WebSphere MQ facilities, including any shared queue or queues.

- More than one queue manager in each logical partition, with CICS MQ group attach

Each CICS region in a partition connects any one of the queue managers that runs in the same logical partition.

This configuration is similar to the previous configuration except in the following situations:

- When a CICS region in one partition fails and then restarts in the same region it might connect to a different queue manager. This way can delay resolution of in-doubt units of recovery (see “MQ group attach”).
- When a queue manager in one partition fails, CICS regions in that partition can connect to another queue manager in the same partition and continue to access the same shared queues and other WebSphere MQ facilities.

WebSphere MQ in-doubt resolution

There can be in-doubt units of work when the connection between a CICS region and a WebSphere MQ queue manager fails. If you have at least version 4.1 of CICS TS and version 7.1 of WebSphere MQ, you can configure CICS to use the WebSphere MQ group UR facility. With group UR, CICS establishes transactions with a whole queue sharing group instead of with a single queue manager. After a connection fails, a CICS region can reconnect to any queue manager in the same queue sharing group because any of them can resolve the outstanding in-doubt units of work.

With older versions of CICS and WebSphere MQ, a CICS region can resolve in-doubt units of work only when it reconnects to the same queue manager. To use MQ group attach with these older versions, you must set the RESYNCMEMBER attribute of the MQCONN definition to YES or NO. This steps works the same way as the RESYNCMEMBER attribute of the DB2CONN definition (see “DB2 in-doubt resolution” on page 189).

Like DB2, WebSphere MQ uses locks to safeguard transactional integrity. However, WebSphere MQ transaction locks normally affect individual messages. So, if a first transaction puts a message (MQPUT call), a second transaction cannot get (MQGET call) that particular message before the first transaction commits, although it can get other messages from the same queue.

Therefore, when one CICS region fails, the same application executing in surviving CICS regions can continue to provide its service. The service remains available even though some messages are stuck until in-doubts are resolved. WebSphere MQ group UR ensures that the “stuck time” is kept to a minimum.

Tip: For optimum availability of shared queues with CICS TS and WebSphere MQ, use CICS MQ group attach and WebSphere MQ group UR.

7.4 Application availability

This section describes techniques and technologies that you can use in your application design and development to enhance the availability of the services that your CICS applications provide.

7.4.1 Transaction deadlocks

A CICS transaction *deadlock* occurs if each of two transactions (for example, A and B) need exclusive use of some resource (for example, a particular record in a data set) that the other already holds. Transaction A waits for the resource to become available. However, if transaction B cannot release it because it, in turn, is waiting for some resource held by transaction A, both are deadlocked and the only way of breaking the deadlock is to cancel one of the transactions, thus releasing its resources.

Whether a deadlock occurs depends on the relative timing of the acquisition and release of the resources by several concurrent transactions. Application programs can continue to be used for some time before meeting circumstances that cause a deadlock. It is important to recognize and allow for the possibility of deadlock early in the application program design stages.

CICS can detect many, but not all, deadlock situations. It abends a transaction that is about to enter a deadlock. IMS and DB2 data managers also have deadlock detection capabilities.

More generally, you should design applications to avoid deadlocks. All applications that update (modify) multiple resources must do so in the same order. For example, if a transaction updates more than one record in a data set, it can do so in ascending key order. A transaction that is accessing more than one file must always do so in the same predefined sequence of files.

7.4.2 Application errors: Damage limitation

You can minimize the number and impact of programming errors in your applications by following established software development leading practices. However, completely eliminating all of them is unlikely. Therefore, a sensible approach is to use CICS facilities to help protect both CICS itself and other transactions that are running on the same CICS system from any application program errors.

This section describes CICS TS facilities that can help you limit the damage that application programming errors can cause. The intention is not to prevent single transactions from failing; the intention is to allow a single transaction to fail without affecting other transactions.

Reentrant programming

A *non-reentrant* program is a program that modifies itself. For example, some Assembler language programs include an area where they save registers; these programs are non-reentrant. A *reentrant* program does not modify itself.

Although writing a non-reentrant CICS application program is possible, doing so is not advisable. Non-reentrant CICS application programs can produce unpredictable results for various reasons, including the following examples:

- ▶ When one transaction waits, a second transaction can start using the same program code. In this way, several separate transactions can be updating the same storage (that is, the program itself) at the same time.
- ▶ In some situations, CICS loads a fresh copy of an application program. The next time a transaction uses the program, any previous updates to the program are not there.

Although this apparently random behavior is obviously undesirable, non-reentrant programs have another problem: if a program can modify itself then *any* program can modify it. You can prevent these problems as follows:

1. Ensure that all your application programs are reentrant. The z/OS compilers have a RENT option that helps application programmers by identifying instructions that make a program non-reentrant.
2. Ensure all your application program runtime code is marked as reentrant. Use the link editor RENT option to do this.
3. Have CICS load reentrant programs into storage that application programs cannot modify. Do this step by using RENTPGM=PROTECT in the CICS system initialization table (SIT).

Runaway control

A typical CICS transaction completes within a fraction of a second. Some complex transactions can take much longer because they process significant amounts of data from relational databases, WebSphere MQ queues, and so on. But even these more complex transactions usually spend a short time processing each item of data before they need to wait for the next one.

Therefore, a CICS application program spends short periods of CPU time between waits. A program that spends a relatively long time before waiting is probably in a “runaway” loop that fails to end because of a program or data error. This type of error is wasteful of compute resources and can prevent CICS from doing other work.

You can use the CICS runaway task timer to limit the time CICS allows an application program to run without waiting (and so, allowing CICS to do other work). You set the default runaway task timer limit in the CICS system initialization table (SIT). You can override this default for individual transactions in the TRANSACTION resource definition.

If you do not specify a default runaway task limit, CICS uses five seconds as the default. However, five seconds is a long time on a processor that can execute multiple instructions in a cycle that is a fraction of a billionth of a second. Be sure to consider what is a reasonable limit for your CICS applications. Most likely you can and should specify a value much lower than five seconds; it is possible that you can safely specify the minimum value: 0.5 seconds.

Thread-safe programming

Some CICS application programs cannot run parallel in a multiprocessor environment. These are known as *quasi-reentrant* (QR) programs. Typically, they update storage without using locks or other methods to prevent parallel access by other transactions. This is allowable if the storage is not accessed by other transactions. However, if other transactions share the same storage, parallel updates by multiple transactions can cause storage corruption that is difficult to diagnose.

CICS is able to run these programs safely by using a single z/OS task, the QR TCB.

Modern CICS applications use *threadsafe* programming. Threadsafe programs avoid updating storage that other transactions share. When they cannot avoid updating shared storage, they use locks or other methods to prevent multiple transactions from interfering with each other and causing storage corruption.

Threadsafe programming is superior to quasi-reentrant programming because it allows CICS to use modern multicore processors more efficiently. It is also superior because it simplifies the interaction between parallel transactions. This approach can significantly enhance availability by reducing the incidence of difficult-to-diagnose storage corruption.

Storage protection

CICS can use the storage protection facilities of System z servers to prevent application programs from accidentally overwriting CICS control blocks. To do this task, CICS allocates storage for application programs with a storage key (known as a *user key*), which differs from the key of CICS control blocks (known as *CICS-key*). The server hardware prevents access to a storage area unless the code is executing with an access key that matches the key for that storage area.

The use of this storage protection facility is optional. You specify whether to use it or not in the CICS system initialization table (SIT). We strongly advise that you do use it. Storage protection can significantly enhance availability by reducing the incidence of CICS control block corruption. This type of corruption can cause CICS region outages.

Deliberate overwriting: This facility does not prevent an application program from deliberately overwriting CICS control blocks. CICS cannot prevent an application obtaining the access key that allows it to modify CICS storage.

Transaction isolation

“Storage protection” explains how you can prevent applications from accidentally corrupting CICS control blocks. CICS also provides the *transaction isolation* facility. You can use this facility to prevent an application program that is processing one transaction from corrupting the storage allocated to another transaction.

The use of this transaction isolation facility is optional. You specify whether to use it or not in the CICS system initialization table (SIT). Alternatively, you can specify whether you want to use it or not for a particular transaction in the TRANSACTION resource definition. We strongly advise that you do use it. Preventing accidental transaction data overwrites significantly improves the reliability and availability of CICS regions.

Policies

“Runaway control” on page 196 describes a way that you can limit a particular undesirable behavior in application programs. CICS Transaction Server Version 5 extends this capability to include a wider variety of limits and more options for how you want CICS to react when an application exceeds a limit. This capability is referred to as *policies*.

You can define a policy to limit each of the following items for a user task:

- ▶ The amount of task or user storage below the line (24-bit), above the line (31-bit), or above the bar (64-bit).
- ▶ The number of requests for task or user storage below the line (24-bit), above the line (31-bit), or above the bar (64-bit)
- ▶ The number of program link requests
- ▶ The number of database requests (SQL commands)
- ▶ The number of file access operations: read, read update, write, rewrite, delete, start browse, read next, or read previous requests.
- ▶ The amount of CPU time consumed by a task.

For each limit, you can specify the action CICS takes when a user task exceeds the limit. The action can be to issue a message, abend the task, or issue an event. For development and test systems, having CICS emit a message can be helpful. The message can help identify the task's normal behavior. You can then set appropriate limits for production systems. For production systems, consider having CICS abend the task.

We advise that you use policies to limit potentially disruptive application behavior on your production systems. You should configure the policies to abend a task that exceeds its limits; this approach can significantly enhance the availability of the service as a whole by sacrificing individual transactions that go wrong.

7.4.3 Isolating applications in separate CICS regions

In 7.4.2, “Application errors: Damage limitation” on page 195, techniques are described that you can use to help protect CICS systems against the effects of application programming errors. In a CICSplex, you can achieve even better availability if you run different applications in separate CICS regions. This section describes this capability.

Efficient resource utilization

An installation commonly uses CICS for a variety of applications. Also CICS systems serve a variety of users. This variety uses processor and other resources much more efficiently than using separate dedicated systems. There are several reasons for how this variety uses processor and other resources more efficiently than using separate dedicated systems. One is that each dedicated system needs its own spare capacity to allow for load spikes; the spare capacity in a consolidated system is available to all its applications and users.

System z hardware and software have sophisticated workload management and virtualization technologies that share resources efficiently between regions running widely different workloads. This approach allows you to allocate separate workloads to separate CICSplex regions based on business considerations, including availability requirements. You do not need to worry about any impact on efficient resource utilization.

Introducing application changes

A key benefit of a CICSplex is that the services that the CICSplex provides continue even after one region stops, either for a planned outage or because of an unexpected problem. See 7.2.2, “CICSplex” on page 186, which explains how you can use this benefit to upgrade application software without an outage. You can use the same process to reduce unplanned outages.

Application programming errors that testing does not detect can become apparent when the production system begins using the new code. Often this is because test systems cannot exactly reproduce production data or production system usage patterns, in particular:

- ▶ Production databases can include incorrect data that previous versions of the application did not notice. These data errors or inconsistencies can cause the new application code to fail.
- ▶ Production system users can do things, including make mistakes, that application programmers and testers do not anticipate.

In a CICSplex, you can initially introduce new application code on a small subset of your CICS regions. You can run your CICSplex this way for a time until you are confident that it is reliable in production use. During this *transition* period, errors in the new code cause minimum service interruption.

Protecting critical services and users

Your CICS system might provide critical services that have much more stringent availability requirements than others. There might also be critical users with much more stringent availability requirements than others. You can help meet this type of requirement by dedicating one or more CICS regions to an application that provides critical services. When the same application serves both critical and less critical users, you can route requests from critical users to one or more dedicated regions.

Dedicating regions in this way can provide enhanced availability for critical services and critical users as follows:

- ▶ It can reduce the number of failures.

A smaller number of transactions or a smaller number of separate programs usually generates a smaller number of failures.

This effect is unlikely to be large because the overall number of failures for all users and services should be small anyway. But it can provide a worthwhile benefit for critical parts of the service.
- ▶ It can reduce the time to recovery.

The usual way a CICSplex handles a region failure is to reroute work requests from the failed region to one or more surviving regions. This extra work causes some disruption in the surviving regions while they acquire resources for the extra workload. A dedicated region is likely to have a smaller footprint so that fail-over of a dedicated region is less disruptive and faster.



Scalability and workload management

In this chapter, we describe the scalability options and workload managements options that CICS provides. We cover the options that are available to influence the scalability in positive and in negative ways. We discuss the following topics:

- ▶ Vertical scalability, scalability within a single CICS region
- ▶ Horizontal scalability within an LPAR, multiple CICS regions in one z/OS image
- ▶ Horizontal scalability using multiple CICS regions in multiple LPARs

We also describe the following topics:

- ▶ Options that are available to manage workloads in a single region and in a multiple CICS region sysplex.
- ▶ How client requests are distributed between CICS regions, managed by the CICS workload management capabilities and z/OS Workload Manager in cooperation.
- ▶ Options that are available to allow the same data to be accesses concurrently from multiple CICS regions.

8.1 Building a scalable enterprise system

We describe the components that are necessary to build a scalable and a workload manageable enterprise system with focus on all the capabilities that CICS provides. These capabilities are available in the simplest single-region CICS system and a CICS system running in a multi LPAR System z environment. Ongoing improvements to the CICS architecture and support have added to the ability of CICS to scale vertically and horizontally:

- ▶ Vertical scaling: The ability of CICS to scale within a single CICS region
- ▶ Horizontal scaling: The ability to distribute workloads between more interconnected CICS regions, regarded as one entity known as a *CICSplex*.

A CICSplex is a group of CICS regions that share workloads, are managed, operated and monitored as a single system. The single system image is provided by the CICS Explorer and CICSplex System Manager, features of CICS TS.

8.1.1 Scalable architecture

In support of a highly scalable infrastructure, CICS is making improvements in the following areas:

- ▶ Virtual storage:
 - By moving internal control blocks into 64 bit storage
 - Exploiting the 64-bit JVMSERVER for Java applications
- ▶ Multiprocessor exploitation:
 - Further exploitation of the OTE environment
 - More threadsafe function
 - Increased maximum number of tasks in a single CICS region
 - Increased TCB limit

8.2 Single region considerations

The minimal CICS configuration is a single CICS region with the capacity to handle the complete workload, as illustrated in Figure 8-1.

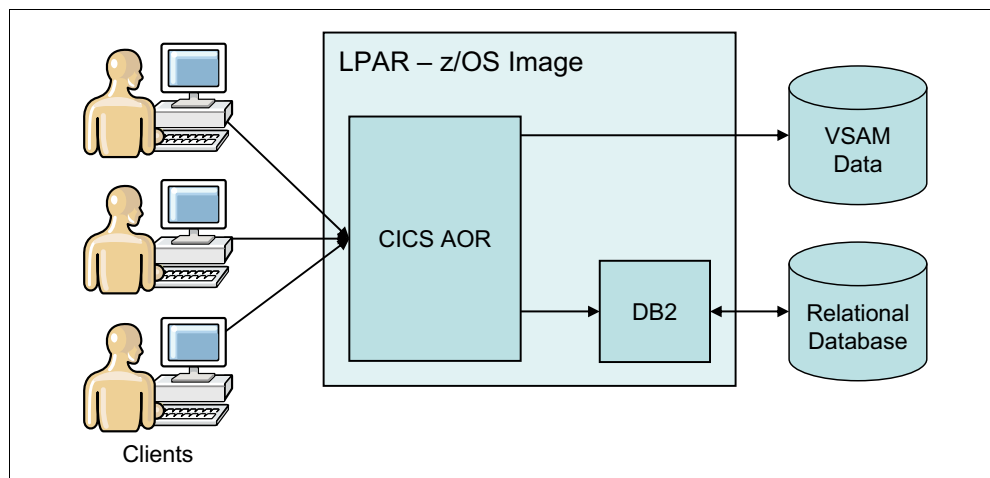


Figure 8-1 Single CICS region

The scalability of a single region CICS system has typically been limited by two factors:

- ▶ The amount of virtual storage available to applications.
- ▶ The amount of work that can be executed on the CICS TCB, also known as the QR TCB.

CICS offers different methods to handle these constraints.

8.2.1 Virtual storage constraint relief

For the virtual storage constraints, the following methods provide help:

- ▶ Maximum number of concurrent tasks in the region, defined by the MXT= SIT parameter. The current default is 500, and the allowed range is 10 - 2000. The value controls the number of tasks in the region, and thereby controls the amount of virtual storage being used. If this value is set to high, CICS enters the *Short on Storage* condition, where tasks go into wait conditions or even be abended.
- ▶ CICS TS V4.1 moves internal data and control blocks into 64-bit virtual storage, releasing 31-bit storage to be used by application programs. Currently the following CICS areas are moved into 64-bit storage:
 - Internal trace table
 - TS Main
 - Message Tables
 - Pooled JVMs
 - JVM Servers
 - Transaction dump table
 - I/O Buffers for XML parse
 - Channels and Containers
 - Loader control blocks
 - CICSplex SM Global Work Area

Policies: CICS TS V5.1 introduces policies to define a threshold for the amount of virtual storage an application is allowed to allocate. A customer-defined action, for example abending the task, will be taken when the threshold is reached.

8.2.2 CPU contention

The following methods are used to control the CPU usage:

- ▶ Interval control value for runaway tasks (ICVR).

The ICVR system initialization parameter specifies the default runaway-task time interval in milliseconds as a decimal number. You can specify zero, or a number in the range of 500 - 2700000, in multiples of 500. CICS rounds down values that are not multiples of 500. This is the RUNAWAY interval that is used by transactions defined with RUNAWAY=SYSTEM.
- ▶ The RUNAWAY value can be defined on the TRANSACTION resource definition to allow separate transactions to have different runaway values.

Consumption thresholds: CICS TS V5.1 introduces policy-based CPU consumption thresholds, in addition to the RUNAWAY option.

- QR TCB constraint relief

For an application to fully benefit from a multiprocessor System z, it should be written in a threadsafe way. See Chapter 3, “CICS application architecture” on page 49 for information about threadsafe.

The majority of CICS API commands are threadsafe. For the exact list see the information center for the CICS version in question. To reduce the constraints for the QR TCB, CICS introduced a number of other TCBs, on which work can be executed. Also the majority of the CICS API is threadsafe. That, together with the application itself being threadsafe, will allow the task to run on what is known as an Open TCB and give additional relief of the QR TCB, and better exploitation of a multiprocessor System z. An extra value of being threadsafe is the avoidance of the TCB switching, reducing the overall CPU consumption.

Tip: An important tool to analyze applications for threadsafeness is the CICS Interdependency Analyzer. The tool helps in finding non threadsafe commands in programs. The result of the analysis can then be viewed in the CICS Explorer.

8.2.3 Managing the workload in a single region

The ability to manage the workload in a single region is limited to the following tasks:

- Setting the SIT MXT value to an appropriate value

This way controls the amount of storage being used in the region.

- Assigning priority to the transaction.

Prioritization is a method of giving specific tasks preference in being dispatched. Priority is specified in the TERMINAL definition (TERMPRIORITY), a transaction in a TRANSACTION definition (PRIORITY), and in the priority field of the user segment of the external security manager (OPPRTY).

Overall priority is determined by summing the priorities in all three definitions for any given task, with the maximum priority being 255:

`TERMPRIORITY+PRIORITY+OPPRTY <= 255`

- Assigning a transaction class to the transaction ID

A TRANCLASS resource defines the characteristics of a transaction class. By putting your transactions into transaction classes, you can control how CICS dispatches tasks. For example, you can separate transactions into those that are heavy resource users and those that are of lesser importance. You can then use the attributes on the TRANCLASS definition to control the number of active tasks allowed from each transaction class.

8.3 Multiregion considerations

When scaling within a single region has reached the limit, or the level of scaling is affecting the overall performance, horizontal scaling, which is adding more regions, must be considered. Horizontal scaling provides further benefits. In configuring horizontal scaling, the further benefit of configuring for higher availability can also be considered. These considerations are described in Chapter 7, “High availability and continuous operation” on page 177.

Moving into a multiregion scalability configuration, several new questions must be asked. For example, does the application use facilities for creating affinity to a single region and how can that be solved? Some of the concerns are related to multi-CICS regions in the same LPAR and some are related to CICS regions that are deployed in multiple LPARs. The following areas must be considered:

- ▶ Distributing the client requests between the LPARs
- ▶ Accessing DB2 data access
- ▶ Accessing DBCTL data
- ▶ Accessing VSAM data

These considerations raise a new question: Will the LPARs be on the same physical System z or will the LPARs be on two separate System z systems, being members in a Parallel Sysplex? Parallel Sysplex configuration is described in Chapter 7, “High availability and continuous operation” on page 177. In this chapter, we describe only the concerns that are related to scalability. Figure 8-2 shows a single LPAR configuration; CICS is configured for scalability.

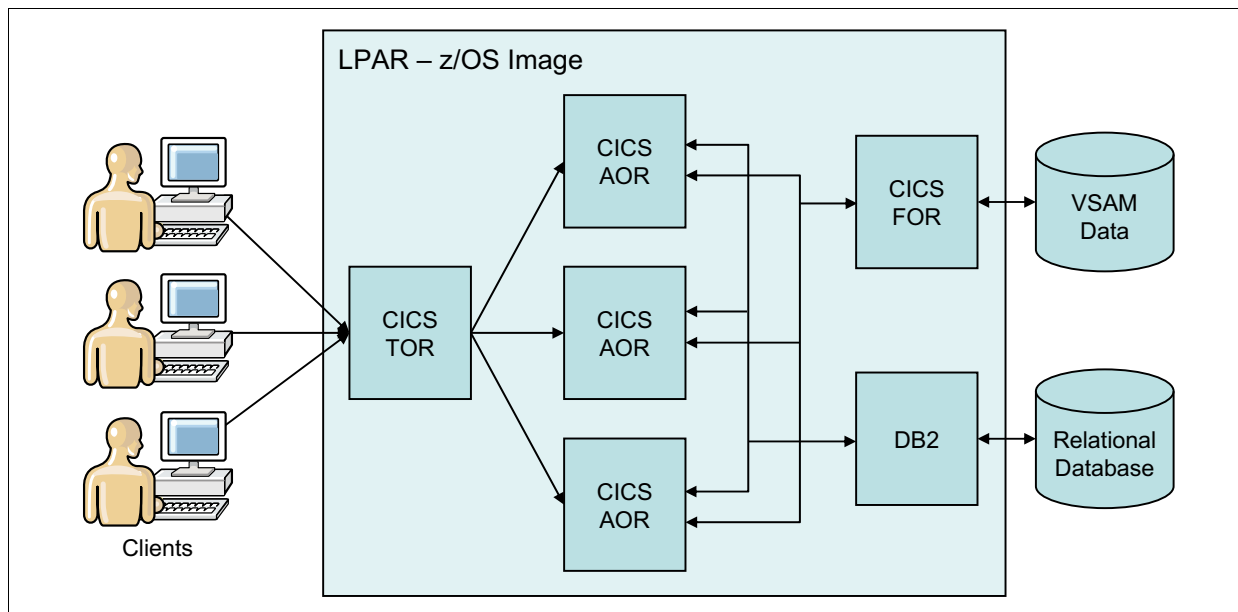


Figure 8-2 Single LPAR scalability

8.3.1 Client access

In this configuration, we move the reception of client requests to a CICS region named *CICS TOR*, shown in Figure 8-2 on page 205. The functionality of this region is to do these operations:

- ▶ Receive requests from the clients.
- ▶ Distribute the work request to one of the available CICS AORs.
- ▶ Receive response back from the CICS AOR.
- ▶ Send response to the client.

8.3.2 Data access

Depending on the type of data store being used, certain aspects must be considered.

Database access

CICS can connect only to database managers that reside in the same z/OS image as CICS itself, which is true for both DB2 and IMS DBCTL. This is not an issue with a single LPAR configuration. Several CICS regions can connect to the same DB2 subsystem, as shown in Figure 8-2 on page 205.

WebSphere MQ

For CICS to connect to a WebSphere MQ queue manager, the queue manager must reside on the same z/OS image as CICS itself, and CICS can connect to only one queue manager at any one time, but several CICS regions can connect to the same queue manager.

VSAM

Accessing VSAM from multiple regions is not possible in the same way; we have more options to use:

- ▶ Move all VSAM files to a new CICS region, we refer to as a *file owning region (FOR)*. All business applications that are running in the AOR will access the VSAM data by using function-shipping from the AOR to the FOR.
- ▶ VSAM Record Level Sharing. This method is similar to using the FOR, but the VSAM data sets are accessed through an SMSVSAM server, which is the owner of the VSAM data sets.
- ▶ CICS VSAM Transparency (CICS VT) is a product that allows you to move VSAM data to DB2, and still access the data from the application, as though it were VSAM data sets, using the CICS FC API unchanged. In this case, without modifying the application the data integrity and transactionality will be handled by the DB2 database system.

For the VSAM examples, CICS VSAM Recovery for z/OS may be considered to enable recovery from physical and logical corruption.

CICS VR information: See the following website for more information:

<http://www.ibm.com/software/http/cics/vr/>

When CICS VT is the solution, the recovery issues are handled by the DB2 subsystem.

CICS VT information: See the following website for more information:

<http://www.ibm.com/software/http/cics/vt/>

These options are shown in Figure 8-3.

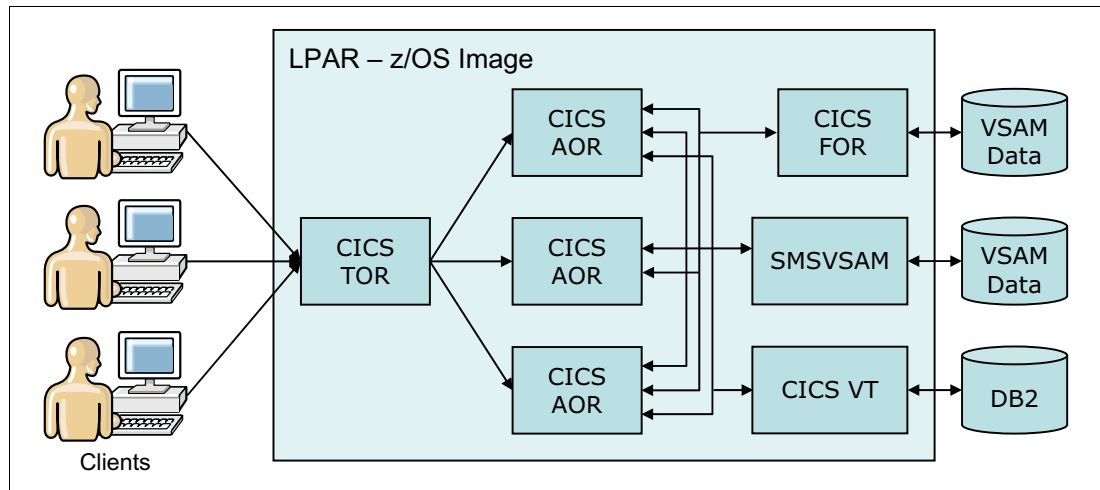


Figure 8-3 Single LPAR VSAM access

Temporary storage

The use of temporary storage must also be considered. This technique is commonly used to pass data from one transaction to another within a pseudo conversation by using temporary storage. Temporary storage data is available only to tasks that run in the same region as the task that originally created the data. Methods to overcome this situation are similar to the issue with VSAM files:

- ▶ A temporary storage owning region (TSOR) can be built to hold all the temporary storage data, which then can be accessed by using *function shipping*. See Figure 8-4 on page 208. A TSOR is an ordinary CICS region, with the single purpose of hosting all temporary storage records. Temporary storage requests from all other CICS regions are function shipped to this region.
- ▶ A Temporary storage server can be established to hold all the temporary storage data. A temporary storage server is accessible from multiple CICS regions and owns the temporary storage queues. The temporary storage server itself is not a CICS region, but a z/OS address space that utilizes the coupling facility to host the temporary storage data. This configuration is typically used in a multi-LPAR configuration.
- ▶ An alternative to temporary storage can be used to pass data between tasks. A good alternative is the CHANNEL and CONTAINER support.
- ▶ Transactions requiring access to the same temporary storage data, can be routed to the same region, using CICS workload management techniques.

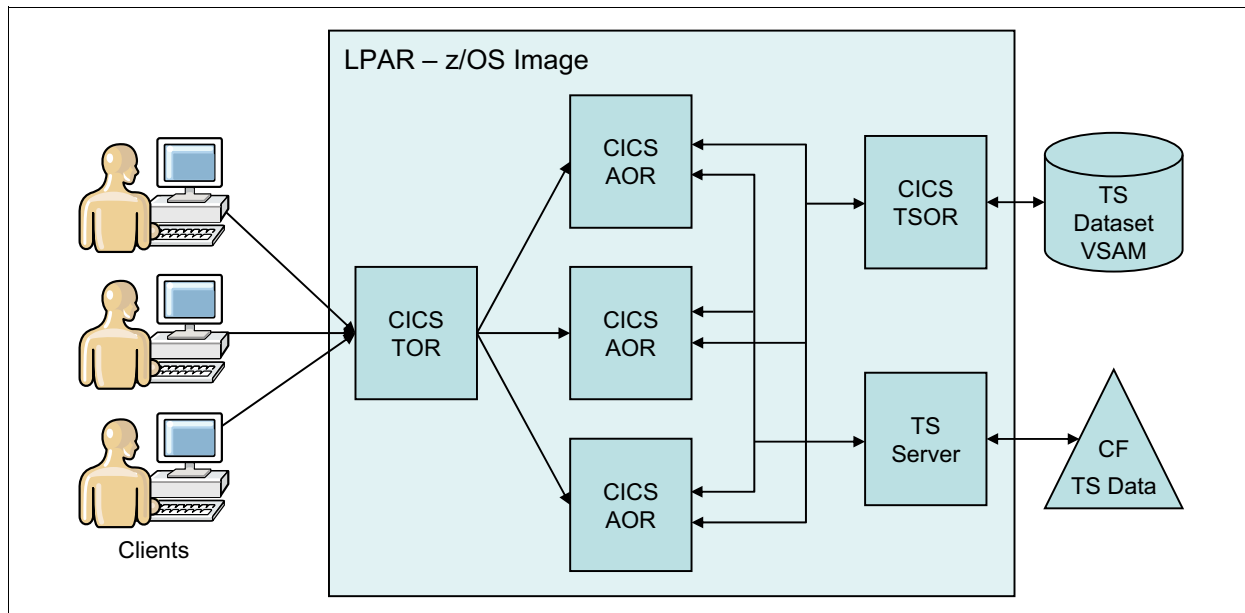


Figure 8-4 Temporary storage server

8.3.3 Other multiregion considerations

There might be existing application logic, which is based on the fact that the application runs in only one CICS region. This leads to other multiregion considerations:

- ▶ Building unique sequence numbers
This issue can be resolved using a Named Counter server.
- ▶ GETMAINing CICS shared storage
- ▶ ENQ/DEQ logic
This issue can be resolved by using Global CICS ENQ/DEQ.

Named counter servers

CICS provides a facility for generating unique sequence numbers for use by application programs in a Parallel Sysplex environment. This facility is controlled by a named counter server, which maintains each sequence of numbers as a named counter.

Although all named counter values are held internally as doubleword unsigned binary numbers, the CICS API provides both a fullword (COUNTER) and doubleword (DCOUNTER) set of commands, which you should not mix.

In addition to the CICS named counter API, CICS provides a call interface that you can use from a batch application to access the same named counters. This interface can be important where you have an application that uses both CICS and batch programs, and both must access the same named counter to obtain unique numbers from a specified range. The call interface does not depend on CICS services, therefore it can also be used in applications that run under any release of CICS.

Named counter recovery

Named counters are stored only in the coupling facility. Applications that use named counters might therefore need to implement recovery logic to handle the possible impact of any coupling facility problems.

Shared data access

None of the proposed alternatives have the same performance characteristics as shared storage, however they may be considered for applications where the high availability of the routing capability exceeds the performance requirement. For applications with high performance requirements, a static routing or a routing based on affinity might be the solution.

Shared data tables

Shared data tables provide efficient use of data in memory. This means that considerable performance benefits are achieved at the cost of some additional use of storage.

This overview of the use of storage assumes that you understand the distinction between various types of storage, such as real and virtual storage, and address space and data space storage. Most of the storage used is data space storage, which is virtual storage separate from address space virtual storage.

Shared data tables use virtual storage as described here. Record data is stored in data spaces DFHDT003, DFHDT004, DFHDT005, and so on, with new data spaces being allocated as required. The total record data storage at loading time is basically the total size of all records (without keys, which are stored in table-entry storage) plus a small amount of control information. Data space storage is acquired in units of 16 MB, and allocated to individual tables in increments of 128 KB. Storage is then sub-allocated in page-aligned frames that are large enough to contain the maximum record length for the table. Data table frames are loosely equivalent to VSAM control intervals, and normally hold a set of records with similar keys. Where possible, each new record is stored in the same frame as the existing record with the closest lower key.

If many records are increased in length after loading, or new records are added randomly throughout a large part of the file, the amount of storage will be increased, possibly up to twice the original size.

Coupling facility data tables

Coupling facility data tables (CFDTs) provide a method of file data sharing, using CICS file control, without the need for a file-owning region, and without the need for VSAM RLS support. CICS CFDT support provides rapid sharing of working data within a sysplex, with update integrity.

The data is held in a coupling facility, in a table that is similar to a shared user-maintained data table. You must define the resources required for CFDTs in a z/OS coupling facility resource management (CFRM) policy.

Because read access and write access have similar performance, this form of table is useful for scratchpad data. Typical uses might include sharing scratchpad data between CICS regions across a sysplex, or sharing of files for which changes do not have to be permanently saved. There are many requirements for scratchpad data, and most can be implemented by using CFDTs.

CFDTs are useful for grouping data into separate tables, where the items can be identified and retrieved by their keys, as in the following examples:

- ▶ You can use a field in a CFDT to maintain the next free order number for use by an order processing application.
- ▶ You can maintain a list of the numbers of lost credit cards in a CFDT.

To an application, a CFDT appears much like a sysplex-wide user-maintained data table, because it is accessed in the same way, by using the file control API. However, in a CFDT there is a maximum key-length restriction of 16 bytes.

Coupling facility data table models

CFDTs can use the contention model or the locking model. You specify the model that each table uses on its file resource definition, so different tables can use different models.

The contention model gives optimal performance but generally requires programs written to exploit it. The reason is because the CHANGED condition code (which indicates that the data is changed since the application program issued a read-for-update request) is specifically for this model. Programs that are not written for the contention model might not be able to handle this condition correctly. The CHANGED response can occur on a REWRITE or DELETE command. There is also a situation with the contention model in which the NOTFND response can be returned on a REWRITE or DELETE.

This model is nonrecoverable: CFDT updates are not backed out if a unit of work fails.

The locking model is API-compatible with programs that conform to the UMT subset of the file control API (this subset is almost the full file control API). This model can be either nonrecoverable or recoverable:

- ▶ **Nonrecoverable**

Locks do not last until sync point, and CFDT updates are not backed out if a unit of work fails.

- ▶ **Recoverable**

CFDTs are recoverable after a unit-of-work failure or a CICS region failure (updates made by units of work that were in-flight at the time of the CICS failure are backed out).

The recoverable locking model supports indoubt and backout failures. If a unit of work fails when backing out an update to the CFDT, or if it fails indoubt during sync-point processing, the locks are converted to retained locks and the unit of work is shunted.

8.3.4 Global CICS ENQ/DEQ

The ENQ and DEQ commands are used to serialize access to a shared resource.

Serialize: In this context, *serialize* means that access to the resource is controlled in the way that only one CICS task can access the resource at any given time. Subsequent access attempts will wait until the previous task does a DEQ.

In earlier releases of CICS, these commands were limited to the scope of CICS tasks running in the same region, and could not be used to serialize access to a resource shared by tasks in different regions. Now, if the ENQs and DEQs are supported by appropriate ENQMODEL resource definitions, they can have sysplex-wide scope.

Global CICS ENQ/DEQ extends the CICS application programming interface to provide an enqueue mechanism that serializes access to a named resource across a specified set of CICS regions that are contained within a sysplex.

Because Global CICS ENQ/DEQ eliminates the most significant remaining cause of inter-transaction affinity, it enables better exploitation of parallel sysplex. It also reduces the need to provide inter-penetrations affinity rules to dynamic routing mechanisms, such as CICSplex workload management, thus reducing the system management cost of exploiting parallel sysplex.

Implementation

Global CICS ENQ/DEQ uses z/OS global resource serialization (GRS) services to achieve locking that is unique across multiple z/OS images in a sysplex.

Benefits

Sysplex enqueue provides the following benefits:

- ▶ Eliminates one of the most common causes of inter-transaction affinity.
- ▶ Enables better exploitation of a parallel sysplex providing better price/performance, capacity, and availability.
- ▶ Reduces the need for inter-transaction affinity rules in dynamic and distributed routing programs, lowering the systems management cost of using parallel sysplex.
- ▶ Enables serialization of concurrent updates to shared temporary storage queues, performed by multiple CICS tasks across the sysplex.
- ▶ Helps to prevent interleaving of records that are written by concurrent tasks in different CICS regions to a remote transient data queue.
- ▶ Allows the single-threading and synchronization of tasks across the sysplex.

Tip: ENQ/DEQ is not designed for the locking of recoverable resources. That process is done by the corresponding resource owners.

8.3.5 Considerations for multi-LPAR CICS regions

Next we describe the special requirements that scaling out of a single LPAR image have:

- ▶ Networking options
- ▶ Database data sharing
- ▶ WebSphere MQ shared queues
- ▶ Use of a z/OS coupling facility

8.3.6 Communication options

For networking, we must consider the two types of networks that are supported by CICS. They have similar functionality implemented in different ways.

SNA/VTAM considerations

This section describes how the generic resource support that is provided by the z/OS Communications Server is used by CICS to balance terminal sessions between available TORs. Figure 8-5 on page 212 shows an example where the terminals request a session with a generic application identifier (APPLID) named *TOR*. z/OS Communication Server resolves this generic name, to one of the two unique resource names: *TOR1* or *TOR2*.

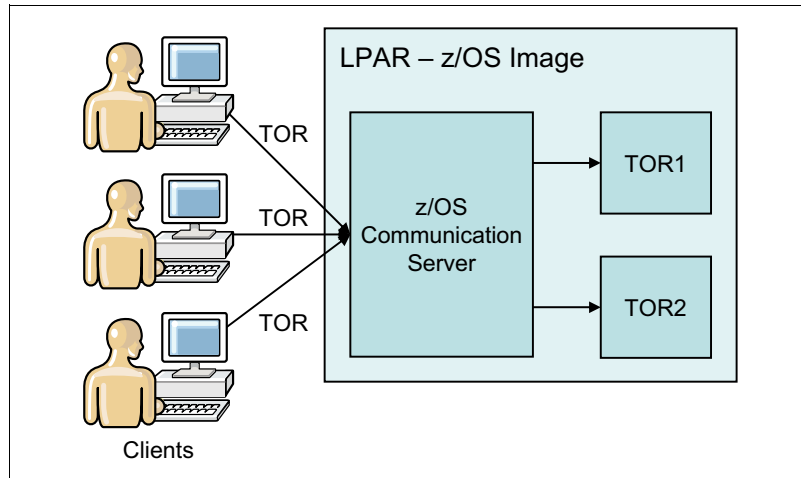


Figure 8-5 VTAM generic resource

A CICS system may register as a Communications Server generic resource. It may then be known either by its unique APPLID or by the generic resource name, which is shared by a number of CICS systems, all of which are registered to the same generic resource.

TCP/IP considerations

Two options are available with TCP/IP:

- ▶ Port sharing
- ▶ Sysplex distributor and DVIPAs

Port sharing

One of the options available with TCP/IP is port sharing. Figure 8-6 illustrates the configuration to use this support. The configuration allows multiple CICS regions to listen to the same port from the same IP stack. In this example, all CICS regions named ROR (for example ROR1, ROR2, and so on) are listening on port 5555, and all requests addressing this IP stack's port 5555 will be distributed between the active regions, in a round-robin fashion, based on the number and health of sockets in each CICS region.

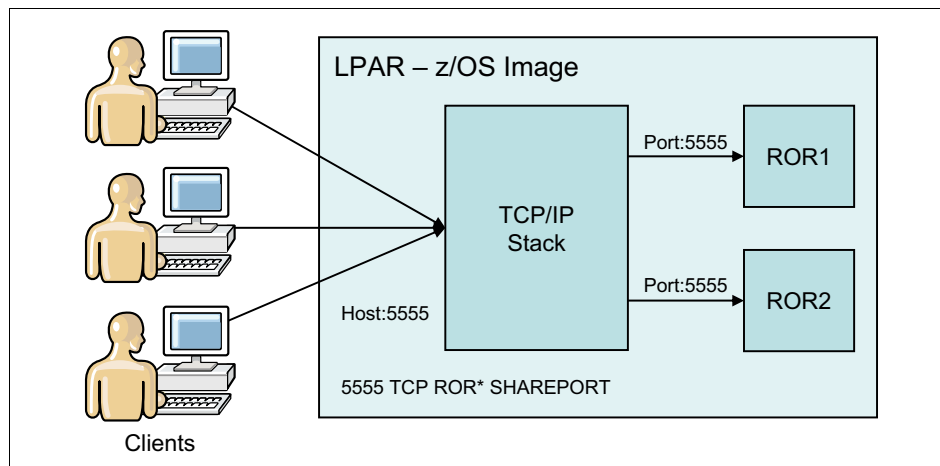


Figure 8-6 TCP/IP SHAREPORT configuration

Sysplex distributor and DVIPAs

A second option that can be used is Dynamic Virtual IP Address (DVIPA). This option must be used if the CICS region resides in separate LPARs or connects to separate IP stacks in the same LPAR.

Sysplex distributor is a function that allows an IP workload to be distributed to multiple server instances within the sysplex but without requiring changes to clients or networking hardware, and without delays in connection setup. With this function, you can implement a dynamic VIPA as a single network-visible IP address that is used for a set of hosts belonging to the same sysplex cluster. A client on the IP network sees the sysplex cluster as one IP address, regardless of the number of hosts in the cluster. With Sysplex Distributor, clients receive the benefits of workload distribution that are provided by both the z/OS Workload Manager (WLM) and the quality of service (QoS) Policy Agent. In addition, Sysplex Distributor ensures high availability of the IP applications that are running on the sysplex cluster by providing continued operation if a LAN fails, or an entire IP stack leaves the XCF distributed DVIPA.

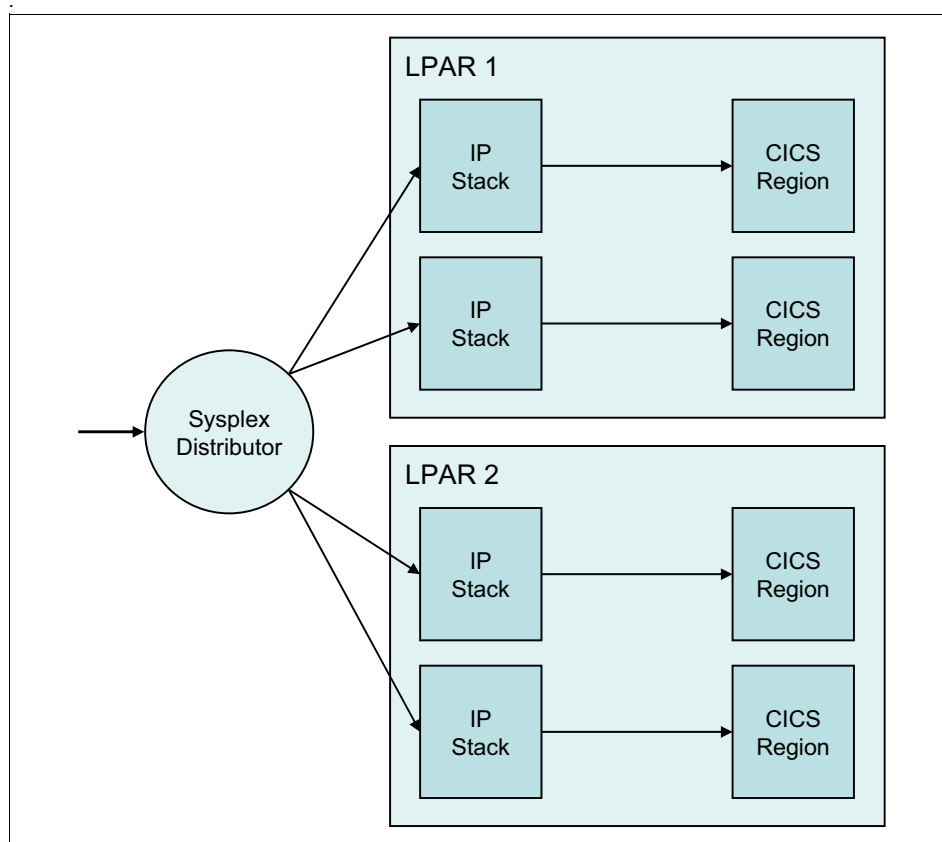


Figure 8-7 Sysplex distributor configuration

Note: Sysplex distributor can be used in conjunction with port sharing to provide a combined high availability approach across CICS regions in the same LPAR and across a sysplex in different LPARs.

DB2 data sharing

DB2 for z/OS takes advantage of data sharing technology in a Parallel Sysplex to provide applications with full concurrent read and write access to shared DB2 data. Data sharing allows users on multiple DB2 subsystems, members of a data sharing group, to share a single copy of the DB2 catalog, directory, and user data sets.

Data sharing provides improvements to availability and capacity without affecting existing applications. For the same DB2 data to be concurrently available from more LPARs, a DB2 data sharing configuration needs to be available for the CICS regions. As shown in Figure 8-8, CICS connects to the local DB2 member of the data sharing group. All members can access the same data, maintaining full integrity.

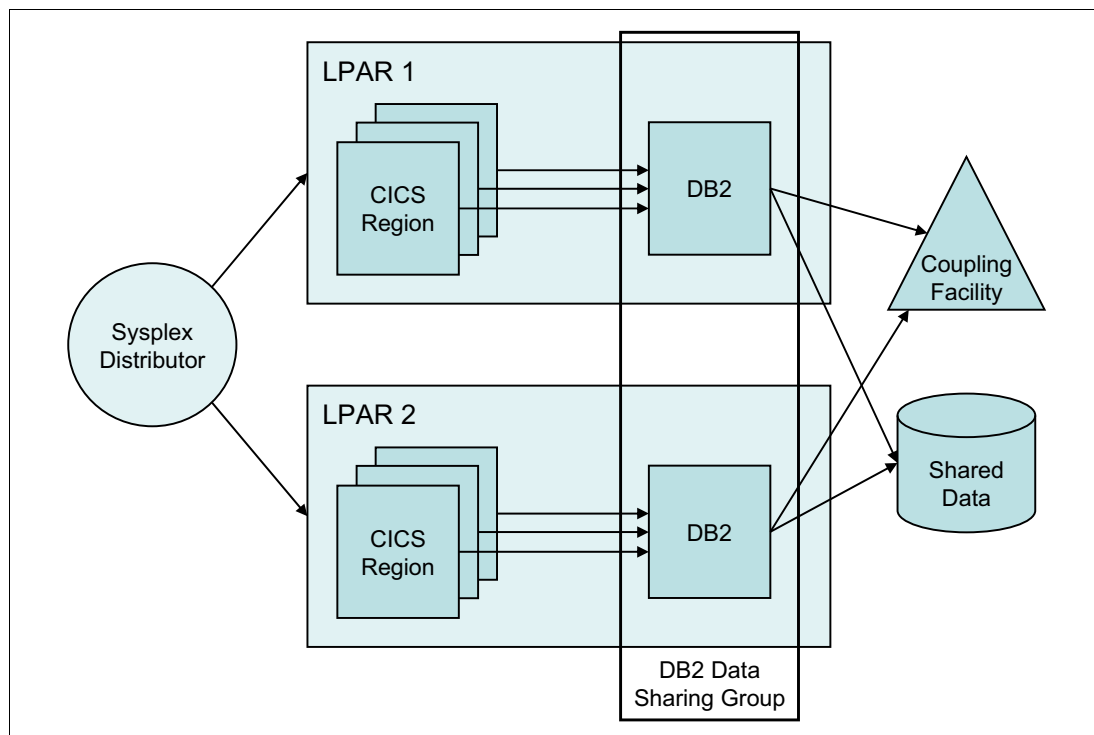


Figure 8-8 DB2 data sharing group

WebSphere MQ shared queues

CICS can connect only to the WebSphere MQ queue manager that lives in the same z/OS image as CICS itself. To overcome this restriction, we must use a function of WebSphere MQ known as *shared queues*. Figure 8-9 shows a configuration with two queue managers, MQM1 and MQM2 members of the same queue sharing group, sharing the same queues in the coupling facility. CICS in LPAR1 connects to WebSphere MQ by using queue manager MQM1, and CICS in LPAR2 connects to the same WebSphere MQ sharing group by using queue manager MQM2, but being able to access the queues.

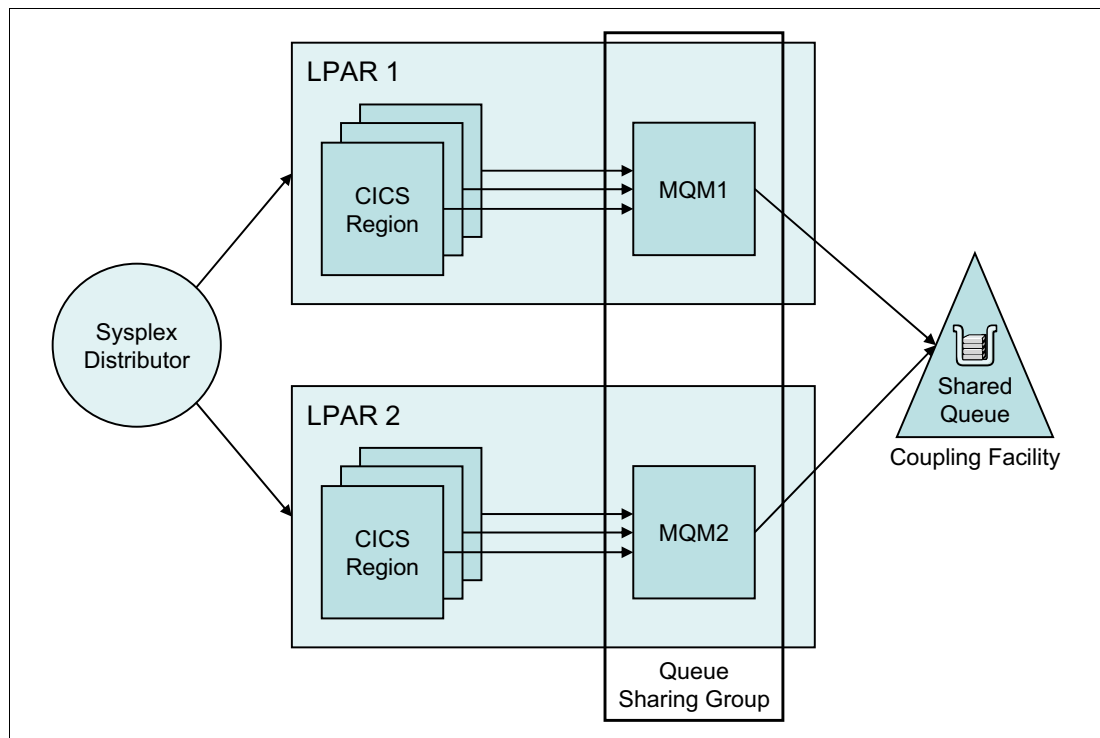


Figure 8-9 WebSphere MQ shared queue configuration

8.3.7 Workload management in a CICSplex

CICS workload management optimizes processor capacity in your enterprise. CICSplex System Manager (CPSM), an integrated part of CICS TS can be used to distribute tasks, the workload, between regions defined as AORs. Different routing criteria can be used by CPSM to decide the best fit target AOR CPSM:

- ▶ In cooperation with z/OS Workload Manager, the regions that are most likely to fulfill the response time goals is selected.
- ▶ The transaction has affinity to a CICS region
- ▶ Workload separation is used.

Fulfilling response time goals

To fulfill the response time goals, CICS workload management optimizes the processor capacity in the enterprise.

Workload management achieves this goal by dynamically routing transactions and programs to the CICS region that is the most appropriate at the time, taking into account any transaction affinities that exist. Workload management can dynamically route the following items:

- ▶ Transactions invoked at a terminal
- ▶ Eligible transactions that are invoked by using EXEC CICS START commands that are associated with a terminal
- ▶ Eligible transactions that are invoked by using EXEC CICS START commands that are not associated with a terminal
- ▶ Distributed program links (DPL), including the following items:
 - CICS Web support
 - CICS Transaction Gateway
 - EXCI calls
 - CICS client ECI calls
- ▶ Any function that issues an EXEC CICS LINK PROGRAM request Link3270 requests
- ▶ Transactions that are associated with CICS business transaction services (BTS) activities

Alternatively, if you want work to run always in a specified region, you can use static routing.

The CICS systems involved in dynamic routing may act as one of the following regions:

- ▶ Requesting region

The CICS system where the request is initiated.

- For terminal-initiated transactions and for inbound client DPLs, the requesting region is typically a terminal-owning region (TOR).
- For terminal-related EXEC CICS START commands, non-terminal-related EXEC CICS START commands, peer-to-peer DPLs, CICS BTS activities, and Link3270 bridge requests, the requesting region is typically an application-owning region (AOR).

- ▶ Routing region

The CICS region that decides where to route the transaction or program.

- For terminal-initiated transactions and terminal-associated EXEC CICS START commands, the routing region is typically a TOR
- For DPLs, non-terminal-related EXEC CICS START commands and CICS BTS activities, and for Link3270 bridge requests, the routing region is typically an AOR.
- For enterprise bean invocations, the routing region is a CICS listener region.

- ▶ Target region

The CICS system in which the transaction or program will run. For all dynamically-routed transactions, programs, BTS activities and enterprise bean invocations, the target region is typically an AOR.

The routing can be based on the following items:

- ▶ User, terminal, processtype, and affinity attributes that are associated with the work requests, which may be the transactions, programs, or BTS processes and activities
- ▶ The work requests themselves

When you establish a workload, you are associating the work itself and the CICS systems, acting as requesting, routing, and target regions, to form a single, dynamic entity. Within this entity, you can route the work to the following areas:

- ▶ To a target region that is selected on the basis of its availability. This type of routing, known as *workload balancing*, allows you to balance work activity across all of the target regions that are associated with a workload.
- ▶ To a subset of the target regions based on specific criteria. This type of routing, known as *workload separation*, allows you to separate transaction and program occurrences and direct them to different target region subsets, where activity is balanced across the target regions within the subset.

Affinity types

Two types of affinity affect dynamic routing:

- ▶ Inter-transaction affinity
- ▶ Transaction-system affinity

Inter-transaction affinity occurs among a set of transactions that share a common resource or coordinate their processing. Transaction-system affinity is an affinity between a transaction and a particular CICS region, where the transaction interrogates or changes the properties of that CICS region. Any affinity restricts CICS workload manager in selecting a target region based on the load, and where possible affinities should be avoided. Chapter 3, “CICS application architecture” on page 49 describes how to write applications without affinity.

8.3.8 Workload separation

Workload separation is the workload management function that causes transactions that meet predesignated selection criteria to be routed to specific target scopes.

The target scope for a separated workload item might vary from a single application-owning region (AOR) to a large AOR group comprising many CICS regions. If an AOR group is the target, the routing algorithm will be applied to select the most suitable region from those defined to it.

The criteria you use to separate transactions or programs can be based on the following items:

- ▶ The terminal ID and user ID that are associated with a transaction or program occurrence
- ▶ The process type that is associated with the CICS BTS activity
- ▶ The transaction
- ▶ To a selected target region based on its affinity relationship and lifetime

This type of routing, based on the transaction affinity of the target region, allows you to route specific transaction occurrences to the same target region for a designated period of time.

Workload routing and workload separation can be active concurrently in the same or different workloads associated with a CICSplex.

8.3.9 Preparing for workload management

Before benefitting from the workload management capabilities of CICS, several actions must be performed in CPSM:

- ▶ Creating system definitions for the routing region
- ▶ Creating system definitions for the target regions
- ▶ Adding regions to the corresponding system groups
- ▶ Creating workload management specifications

CICS Deployment Assistant can be used to discover running CICS regions and their connectivity to other subsystems such as DB2 and WebSphere MQ. It also can assist in building the CICSplex infrastructure and setting up the workload management infrastructure.

Another important aspect of setting up a workload management environment is to understand application resource usage and affinities. CICS Interdependency Analyzer has functionality to perform these tasks.



Security and auditing

In this chapter, we describe the business requirements for secure enterprise applications and how these can be addressed with the security and auditing capabilities provided by CICS. We describe capabilities that allow us to protect against the following items:

- ▶ Access from clients who are not authenticated
- ▶ Non authorized access to CICS resources
- ▶ Exposure of confidential data

We describe the value of running applications inside CICS, taking advantage of the security capabilities provided by the System z architecture.

In addition, we discuss capabilities that allow us to create audit logs for authorized access to resources and also access that was denied by CICS or RACF.

9.1 Security requirements

Applications require different levels of security. Certain criteria must be considered. Implementing unnecessary security measures might impose an unwanted amount of performance overhead. Not implementing sufficient security mechanisms might, however, put the business at risk.

CICS supports a large variety of mechanisms to allow applications to run in a secure environment, and to ensure messages can be received and sent by using trusted and secure protocols, guaranteeing confidentiality, and integrity.

A complete security solution puts mechanisms in place to achieve the following objectives:

- ▶ Authentication
- ▶ Identification
- ▶ Authorization
- ▶ Integrity
- ▶ Confidentiality
- ▶ Auditing
- ▶ Non-repudiation

9.2 Authentication and identification

Authentication is the process of validating the identity that is claimed by the accessing entity. Authentication is performed by verifying authentication information that is provided with the claimed *identity*. The authentication information is generally referred to as the accessor's *credentials*. A credential can be the accessor's name and password. It can also be a *token* that is provided by a trusted party, such as a Kerberos ticket or an X.509 certificate.

Identification is the ability to assign an identity to the entity that is accessing the system. Typically, the identity is used to control access to resources. Depending on the security model in which the identification is performed, the identity might come from the authentication credentials, asserted from another server, or it might be entered by a human operator.

For CICS to run a transaction, an identification, also known as a *user ID*, is required. The user ID might originate from different sources. These are discussed in 9.2.2, "Identification" on page 221.

9.2.1 Considerations

Consider the following authentication-related questions:

- ▶ Does the service requester need to authenticate?

The answer can be decided for specific services rather than having a general rule for the application. It might be appropriate to run read-only services by using a generic user ID, whereas more sensitive services might need the requester to authenticate. In CICS, this split can be made by running secured services on a different pipeline to unsecured services.

- ▶ Who authenticates the service requester? CICS or an intermediary server?

CICS can authenticate service requesters directly, or an intermediary might be able to provide an authentication service to CICS. In this case, the intermediary server authenticates the service requester and then flows an *asserted* identity to CICS.

- Will you use transport-based or SOAP message based authentication?

You might choose to use only transport-based security to secure your CICS web services environment in these circumstances:

- No intermediaries are used in the web service environment. Or, if there are intermediaries, you can guarantee that after the data is decrypted, it cannot be accessed by an untrusted node or process.
- The transport is based only on HTTP.
- Performance is your primary concern.
- The web services client is a stand-alone Java program.

WS-Security can be applied only to clients that run in a web services environment that supports the WS-Security specification (for example, WebSphere Application Server).

You might choose to use WS-Security (possibly in addition to transport-level security) in these circumstances:

- Intermediaries are used, some of which might be untrusted.

Security credentials that flow in the SOAP message can pass through any number of intermediaries. Protecting confidential information in the actual SOAP message can avoid the overhead of encrypting and decrypting through SSL at every intermediary node.

- Multiple transport protocols are used.

WS-Security works across multiple transports and is independent of the underlying transport protocol.

- You might choose to implement your own security procedures and processing by writing a custom message handler program that can process secure SOAP messages in the pipeline.
- If you chose SOAP message-based security, what token type will be used?
 - Username tokens, X.509 certificates, and ICRX tokens can be processed directly by the CICS-supplied security handler.
 - You will most likely have to configure the CICS-supplied handler to call an STS if other token types are used.
 - You can also use an STS to process nonstandard token types or you can write a custom security handler.

9.2.2 Identification

Consider the following identification-related question: How will you assign the RACF user ID for running the CICS business logic program?

- It might come from a human operator signing into a CICS region at the start of a terminal session. The operator is asked to provide a user ID and password. The user ID remains associated with the terminal until the operator signs off.
- It might come from the authentication credentials.
- It might be asserted by an intermediate server, either as a Username token, X.509 certificate, or a ICRX.
- It might be assigned by a user-written message handler that runs as part of the pipeline processing.
- It might be hard-coded in CICS.

User IDs

When a human operator signs into a CICS region at the start of a terminal session, the person is asked to provide a user ID and a password (Figure 9-1). The user ID remains associated with the terminal until the terminal operator signs out. Transactions executed from the terminal and requests made by those transactions are associated with that user ID.

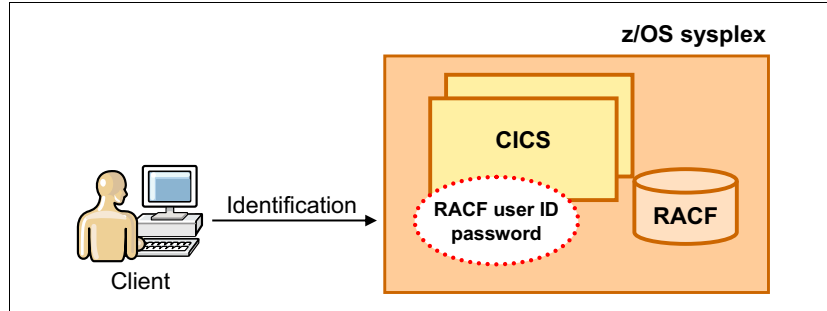


Figure 9-1 Terminal operator identification

For connections from web users, there are other ways in which the user of a CICS transaction can be identified to include the following situation:

- ▶ A web browser user is challenged to provide Hypertext Transfer Protocol (HTTP) basic authentication information (a user ID and a password). The transaction that services the client's request and other requests made by that transaction are associated with that user ID.
- ▶ A client program that is communicating with CICS using the SSL supplies a client certificate to identify itself. The security manager maps the certificate to a user ID. The transaction that services the client's request and other requests made by that transaction are associated with that user ID.

In addition to these transport-level authentication mechanisms, web service clients can also pass authentication data in the SOAP message.

CICS special user IDs

CICS uses two particular user IDs in addition to those that identify individual users:

- ▶ Region user ID: The CICS region user ID is used for checking the authorization when the CICS system, rather than an individual user of the system, requests access to system resources, such as CICS data sets and other servers.
- ▶ Default user ID: When a user does not sign in, CICS assigns a default user ID to the user. It is specified in the system initialization table (SIT) parameter DFLTUSER. In the absence of more explicit identification, it identifies TCP/IP clients that connect to CICS, providing little authority to the default user ID.

Passwords and password phrases

In z/OS Release 8, RACF added the infrastructure for password phrases, as an alternative to passwords greater than eight characters. A password phrase can be any character string in the range of 9 - 100 characters, including blanks. CICS TS V4.2 introduced support for this new feature. This addition enables applications that exploit this new infrastructure to specify an authentication value longer than eight characters to better fit in the heterogeneous world; traditional applications continue to use the traditional password.

The use of password phrases, compared to passwords, improves the system security. Password phrases are much harder to attack, and easier for the user to remember.

Identity assertion

Modern enterprise information processing systems typically consist of multiple software components, for example, WebSphere Application Server running on a distributed platform, and CICS and DB2 running on z/OS. This approach often introduces the challenge of how to provide secure access between middleware components that use disparate security technologies. As stated previously, any user of CICS who requires access to data will usually need to provide a valid credential for authentication and subsequent authorization through RACF security manager. But what if the user does not have a RACF user ID?

The solution often is to provide a form of *identity assertion*, where the distributed identity is mapped to a RACF identity and then asserted to CICS without a password check. However, no matter what solution is used for identity assertion, the process of mapping distributed user identities to a RACF identity is typically a one-way function, resulting in the loss of the original distributed identity after the mapping occurs. Although effective, such mapping solutions have several issues, including lack of end-to-end accountability, inflexibility, and loss of control.

z/OS identity propagation

z/OS identity propagation is a newer form of identity assertion provided by z/OS V1R11. Together with new functions in CICS TS V4.1, and WebSphere DataPower or CICS Transaction Gateway, it supports a cross-platform, end-to-end security solution, providing for identity assertion, control, and auditing.

Identity propagation addresses the issues associated with previous identity assertion solutions by allowing the z/OS security administrator to create a set of flexible rules, stored in the RACF database, ensuring that the distributed identity persists after the mapping stage and remains visible for operational support and auditing.

9.2.3 CICS and identity propagation: The end-to-end security solution

Figure 9-2 shows the current CICS solutions for identity propagation

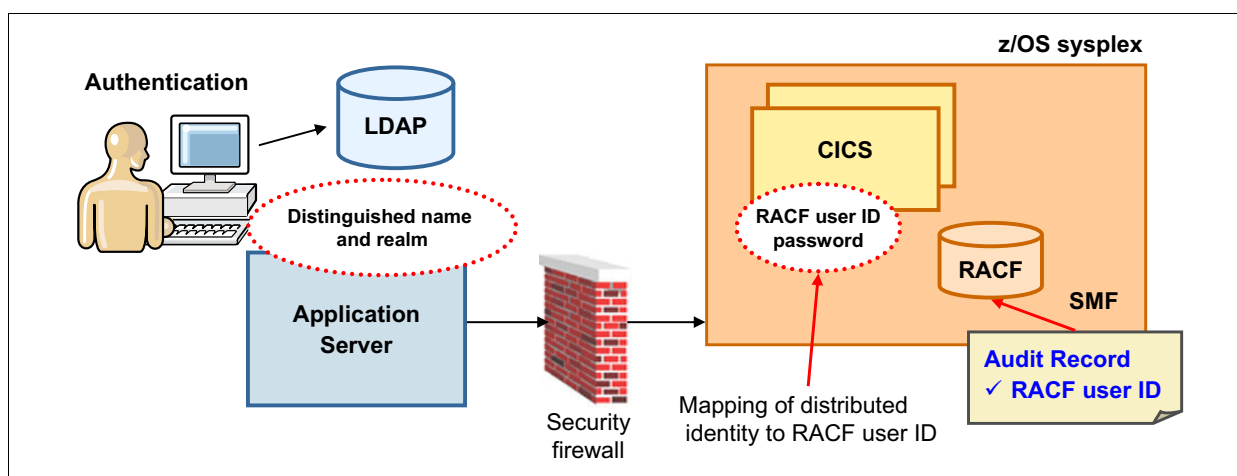


Figure 9-2 CICS identity propagation

When considering IBM z/OS, any application that requires access to data usually must provide a valid credential for authentication and subsequent authorization through the IBM RACF security manager. The security strengths that z/OS provides can then become an issue if it is necessary to authorize requests that originate from a remote security registry that is unknown to RACF. The solution was often to provide a form of identity assertion, where a RACF identity can be asserted without a password check. This scenario typically requires a

level of predefined trust between the two systems. With IBM CICS TS, various techniques are possible:

- ▶ Use of a predefined link user ID
- ▶ Dynamic mapping of Secure Sockets Layer (SSL) client certificate identities to RACF user IDs
- ▶ Delegation through third-party identity management products, such as IBM Tivoli Federated Identity Manager.

No matter what solution is used for identity assertion, the process of mapping distributed user identities to a RACF identity has, until now, been a one-way function, resulting in the loss of the original distributed identity after the mapping occurs. Although effective, such mapping solutions have several issues, including lack of end-to-end accountability, inflexibility, and loss of control. Identity propagation, illustrated in Figure 9-3, addresses these issues by allowing the z/OS security administrator to create a set of flexible rules, stored in the RACF database, which ensure that the distributed identity persists after the mapping stage and remains visible for operational support and further auditing if required.

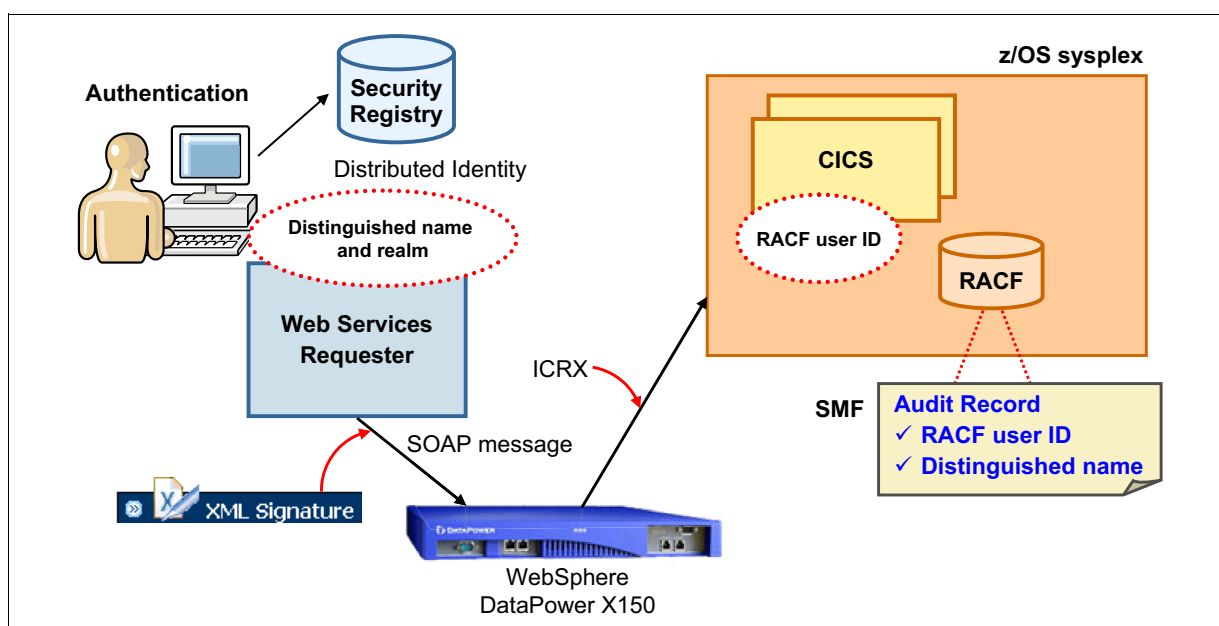


Figure 9-3 Implementation of identity propagation

The identity propagation function that is provided with RACF in z/OS V1R11 can be exploited when using CICS TS V4.1 with a set of enabling APARs: PK83741, PK95579, and PM01622, along with an additional APAR PK98426 for IBM CICSplex SM. If web service requests are being sent to CICS, then an IBM WebSphere DataPower appliance can be used to propagate the distributed identity to CICS so it can be mapped to a RACF user ID. Alternatively, if Java EE components are using the CICS Java EE Connector Architecture (JCA) resource adapter to call CICS applications, CICS Transaction Gateway V8 can be used to propagate the distributed identity.

9.3 Authorization

Authorization is the process of checking whether an identity that was already authenticated should be given access to a resource that it is requesting. A typical implementation of authorization is to pass to the access control mechanism a *security context* that contains the identity that is authenticated.

Consider the following authorization-related question. Does the business application need to run with the terminal operator's or the service requester's identity? If it is not necessary to run the CICS task under the terminal operators user ID, the task can run under what is known as the CICS default user ID, and the terminal operator has no need to sign in to CICS. If it is not necessary to run the CICS task with the service requester's identity, a RACF user ID can be specified in a URIMAP (which avoids running the web service with the CICS default user ID).

If running the CICS task with the terminal operators or service requester's identity is necessary, note the following information:

- ▶ In the terminal operator case, the terminal user needs to sign in to CICS specifying a user ID and password.
- ▶ In the service requester case where no intermediaries are used, CICS authorization processing can be based on the RACF user ID that is associated with the security token that is used by the requester to authenticate.
- ▶ In the service requester case where intermediaries are used, CICS authorization processing can be based on the asserted identity token that is passed by the intermediary.

Surrogate authorization checking should be used to ensure that the intermediary has the correct authority to start work on behalf of the asserted identity.

9.3.1 Securing access to resources

In a CICS environment, the assets you normally want to protect are the application programs and the resources that are accessed by the application programs. CICS security includes a mechanism that allows authorities to be given to users dependent of their role. The roles are typically divided into system administrator, system operator, and client:

- ▶ System administrator
 - Needs authority to run CICS system programming commands and use CICS transactions used to perform administrative functions.
 - Controlled by transaction and command security.
- ▶ System operator
 - Needs authority to run CICS comands and CICS transactions used to operate and control the running CICS systems.
 - Controlled by transaction and command security.
- ▶ Client
 - Needs authority to run one or more client applications.
 - Controlled by transaction and resource security.

To prevent disclosure, destruction, or corruption of these assets, you need a security concept to control the access to the CICS region and to the many CICS components. You can limit the activities of a CICS user to only those functions that the user is authorized to use, by implementing one or more of the following CICS security mechanisms:

- ▶ **Transaction security:** Ensures that users who attempt to run a transaction are entitled to do so.
- ▶ **Resource security:** Ensures that users who use CICS resources, such as files and transient data queues, are entitled to do so.
- ▶ **Command security:** Ensures that users who use CICS system programming commands are entitled to do so.
- ▶ **Surrogate security:** Ensures that a surrogate user is authorized to act on behalf of another user.

Transaction security

When CICS security is active, requests to attach transactions and requests by transactions to access resources are associated with a user ID. When a user makes a request, CICS calls the external security manager, for example, RACF, to determine if the user ID has the authority to make the request.

In Figure 9-4, the terminal operator enters a CICS transaction ID, TRN1. The user ID is checked against an external security manager, for example RACF, to validate the correct authority.

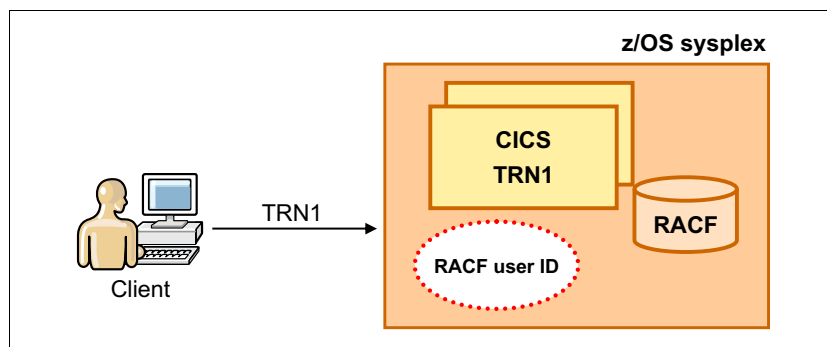


Figure 9-4 Transaction security

If the user ID does not have the correct authority, CICS denies the request. In many cases, a user is a human operator who is interacting with CICS through a terminal or a workstation. However, the user can also be a web browser user or, in a web services solution, a program executing on a client system.

Resource security

CICS uses RACF to control access to resources that you can access through a CICS application program. The resources and the associated CICS system initialization parameter that you use to specify the RACF class name is described in the CICS Transaction Server V4.2 information center:

<http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp>

Figure 9-5 shows an access to a CICS-defined file, FILEA, with resource security active, in which case an external security manager will be called to validate that the user has the appropriate authority to access the file.

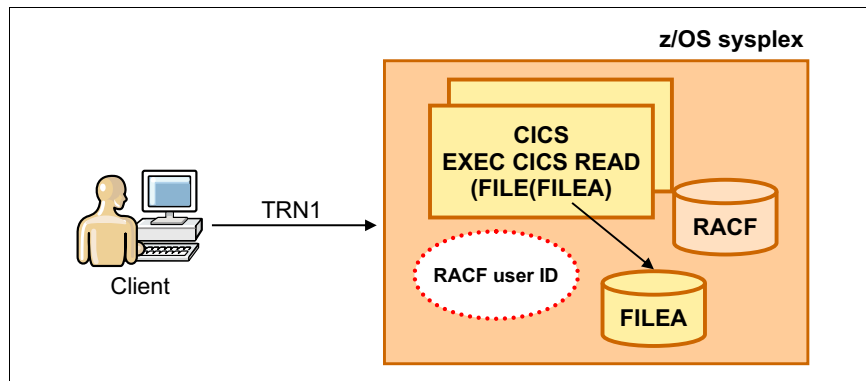


Figure 9-5 Resource security

In the case of authorization failures (for example if the user associated with a CICS task is not authorized to access the resource specified on a CICS command), CICS returns the NOTAUTH condition to the application program.

Command security

CICS command security applies to system programming commands, that is, commands that require the special CICS translator option, SP. Security checking is performed for these commands, when they are issued from a CICS application program, and for the equivalent commands that you can issue with the CEMT master terminal transaction.

Command security operates in addition to any transaction or resource security that you define for a transaction. For example, if a user is permitted to use a transaction called FILA, which issues an EXEC CICS INQUIRE FILE command that the user is not permitted to use, CICS issues a *not authorized* (NOTAUTH) condition in response to the command, and the command fails.

Surrogate user security

A surrogate user is a RACF-defined user who is authorized to act on behalf of another user (the original user). CICS uses surrogate user security in a number of different situations.

Situations where surrogate user checking applies

A surrogate user is one who has the authority to start work on behalf of another user. A surrogate user is authorized to act for that user without knowing that other user's password. To enable surrogate user checking, XUSER=YES must be specified as a system initialization parameter.

CICS performs surrogate user security checking in a number of situations, with the surrogate user facility of an external security manager (ESM) such as RACF. If surrogate user checking is in force, it applies to the following items:

- ▶ The CICS default user
- ▶ PLT post-initialization processing
- ▶ Preset terminal security
- ▶ Started transactions

- ▶ The user ID that is associated with a CICS business transaction services (BTS) process or activity that is started by a RUN command
- ▶ The user ID that is associated with a transient data destination
- ▶ The user ID that is supplied as a parameter on an EXCI call
- ▶ The user ID that is supplied on the AUTHID and COMAUTHID attributes of the DB2CONN and DB2ENTRY resource definitions
- ▶ The user ID that is supplied on the USERID attribute of URIMAP resource definitions

9.3.2 Securing access to DB2

The CICS DB2 environment has four main stages at which you can implement security checking:

- ▶ When a CICS user signs on to a CICS region
CICS sign-on authenticates users by checking that they supply a valid user ID and password.
- ▶ When a CICS user tries to use or modify a CICS resource that is related to DB2
This could be a DB2CONN, DB2ENTRY or DB2TRAN resource definition; or a CICS transaction that accesses DB2 to obtain data; or a CICS transaction that issues commands to the CICS DB2 attachment facility or to DB2 itself. At this stage, you can use CICS security mechanisms, which are managed by RACF or an equivalent external security manager, to control the CICS user's access to the resource.
- ▶ When the CICS region connects to DB2, and when a transaction acquires a thread into DB2
Both the CICS region and the transaction must provide authorization IDs to DB2, and these authorization IDs are validated by RACF or an equivalent external security manager.
- ▶ When a CICS user tries to use a CICS transaction to execute or modify a DB2 resource
This stage can be a plan, or a DB2 command, or a resource that is needed to execute dynamic SQL. At this stage, you can use DB2's security checking, which is managed either by DB2 itself, or by RACF or an equivalent external security manager, to control the CICS user's access to the resource.

DB2 multilevel security and row-level security

DB2 Version 8 introduced support for multilevel security. CICS does not provide specific support for multilevel security, but you can use CICS in a multilevel-secure environment if you are careful with the configuration.

When multilevel security is implemented at row level (row-level security) for data in DB2 Version 8 or later, the RACF SECLABEL class is activated, and a set of security labels is defined for users and for the DB2 table rows. The RACF options SETR MLS and MLACTIVE are not required to be active. You can use DB2 row-level security without impact on the rest of the z/OS system.

CICS is able to access DB2 rows secured in this way. For CICS, you need to ensure that the RACF user profile for a CICS user who needs access to the DB2 rows is defined in RACF to include a default SECLABEL.

When a CICS user signs on to a CICS region, with SEC=YES specified in the SIT, RACF associates the default SECLABEL with the RACF access control environment element (ACEE) for the user. The DB2ENTRY definition (or DB2CONN definition if the pool is being used) needs to specify AUTHTYPE=USERID or AUTHTYPE=GROUP, which ensures the ACEE is passed on to DB2 for further security checking. An individual CICS user can therefore have only one associated SECLABEL.

9.3.3 Intercommunication security

You can connect a number of CICS regions together by using intercommunication; for example, intersystem communication over SNA (ISC over SNA), which uses an SNA access method such as ACF/Communications Server, to provide the required communication protocols. The basic security principles apply to interconnected systems, but the resource definition is more complex and there are additional security requirements.

Intercommunication bind-time security

The first requirement is for a session to be established between the two systems. This step does not, of course, happen on every request; a session, when established, is usually long-lived. Also, the connection request that establishes the session can, depending on the circumstances, be issued either by the remote system or by your CICS system. However, the establishment of a session presents the first potential security exposure for your system.

Your security concern is to prevent unauthorized remote systems from being connected to your CICS system, that is, to ensure that the remote system is really the system that it claims to be. This level of security is known as *bind-time security*. For ISC over SNA connections, it is also known as systems network architecture (SNA) session security. This level can be applied when a request to establish a session is received from, or sent to, a remote system.

Intercommunication link security

Each link between systems is given an authority-defined by a user ID.

Authorization: Users cannot access any transactions or resources over a link that is itself unauthorized to access them. This means that each user's authorization is a subset of the link's authority as a whole.

To limit the remote system's access to your transactions and resources, you use *link security*. Link security is concerned with the single user profile that you assign to the remote system as a whole. As with user security in a single-system environment, link security governs the following types:

- ▶ Transaction security: This type controls the link's authority to attach specific transactions.
- ▶ Resource security: This type controls the link's authority to access specific resources. This applies for transactions, executing on any of the sessions from the remote system, that have RESSEC(YES) specified in their transaction definition.
- ▶ Command security. This type controls the link's authority for the commands that the attached transaction issues. This security applies for transactions, executing on any of the sessions from the remote system, that have CMDSEC(YES) specified in their transaction definition.
- ▶ Surrogate user security. This type controls the link's authority to START transactions with a new user ID, and to install resources with an associated user ID.

APPC (LU6.2) session security

One of the ISC over SNA protocols that CICS uses is for advanced program-to-program communication (APPC), which is the CICS implementation of the LU6.2 part of the SNA architecture. CICS treats APPC sessions, connections, and partners as resources, all of which have security requirements. CICS provides the following security mechanisms for the APPC environment:

- ▶ Bind-time (or session) security prevents an unauthorized remote system from connecting to CICS.
- ▶ Link security defines the complete set of CICS transactions and resources that the remote system is permitted to access across the connection.
- ▶ User security checks that a user is authorized both to attach a CICS transaction and to access all the resources and SPI commands that the transaction is programmed to use.

Multiregion operation (MRO)

Another means of using intercommunication is MRO. It is available for links between CICS regions in a single sysplex, independent of the systems network architecture (SNA) access method.

IP interconnectivity (IPIC) security

The security mechanisms for IPIC connections are similar to those that are provided for APPC (LU6.2) connections, although they are implemented differently:

- ▶ Bind-time security prevents an unauthorized remote system from connecting to CICS. On IPCONNs, bind security is enforced by the exchange of Secure Sockets Layer (SSL) client certificates.
- ▶ Link security defines the complete set of CICS transactions and resources that the remote system is permitted to access across the IPCONN.
- ▶ User security checks that a user is authorized both to attach a CICS transaction and to access all the resources and SPI commands that the transaction is programmed to use. User security is a subset of link security: a user cannot access a resource, even if it is included in the set defined as accessible by his user ID, if it is not also included in the set of resources accessible by the link user ID.

9.3.4 Securing network access

A number of RACF facilities that restricts access to network resources helps protect against unauthorized access to CICS from the Internet.

TCP/IP resources as protected by RACF

These resources can be protected by profiles in the SERVAUTH class, the resource manager in charge is the z/OS TCP/IP stack. In this section, we explain how to control TCP/IP application access to the TCP/IP stacks, ports, and networks.

Network access

The z/OS TCP/IP stack has an option to activate the SAF protection of a resource that identifies an IP network or IP address. So, RACF, or any other external security manager, can be used to control access of applications to and from a specific IP address. The resource itself is defined by a statement within the TCP/IP profile data set. This statement defines a resource name and maps it to a specific IP address or network.

A sample definition to protect a TCP/IP stack is shown in Example 9-1

Example 9-1 NETACCESS example

```
NETACCESS INBOUND OUTBOUND
          10.11.12.13 NETWORK1
          DEFAULT     ALLOTHER
ENDNETACCESS
```

In this example, two resources are defined through the NETACCESS statement in the TCPIP.PROFILE data set:

- ▶ NETWORK1 represents the 10.11.12.13 IP address.
- ▶ ALLOTHER applies to all other IP addresses, also known as the default zone.

In this example, the controlled resources include both inbound and outbound traffic.

If the RACF SERVAUTH class is active, and the resource profile is defined with UACC(NONE), any user ID attempting access to a network that is defined within the NETACCESS statement must be permitted READ access to the resource. For example, to communicate with 10.11.12.13 on z/OS system MVSIV using the TCP/IP stack that was started as TCPIP2, the user ID must have permission to the following resource:

EZB.NETACCESS.MVSIV.TCPIP2.NETWORK1

The resource name has a predefined syntax with the following qualifiers:

- ▶ EZB.NETACCESS is a fixed qualifier.
- ▶ The system ID where the protection is to operate. MVSIV in this example. The TCP/IP stack started task where the protection is to operate. TCPIP2 in this example.
- ▶ The resource name that is defined in the NETACCESS statement in the PROFILE.TCPIP data set. NETWORK1 in this example.

Port access

Similar to network access, port access implements access control for TCP/IP ports by using a resource name of the following form:

EZB.PORTACCESS.MVSIV.TCPIP2.SERVER1

Again, the first two qualifiers are predefined. Corresponding to this statement, the PORT reservation statements in the PROFILE.TCPIP data set would have to specify the resource name such as that shown in Example 9-2.

Example 9-2 PORT statement using SAF

```
PORT
21 TCP * SAF SERVER1
23 TCP * SAF SERVER1
```

For any task to use TCP ports 21 or 23, permission to the resource profile is required. The TCP/IP port SAF resource protection applies both to UDP and TCP communications.

Stack access

Continuing the pattern already established, the following syntax can be used when identifying a TCP/IP stack as a SAF resource (remember that it can be up to eight TCP/IP stacks executing concurrently in a single z/OS image):

EZB.STACKACCESS.MVSIV.TCPIP2

9.4 Integrity

Integrity ensures that transmitted or stored information is not altered in an unauthorized or accidental manner. Typically it is a mechanism to verify that what is received over a network is the same as what was sent.

Consider the following integrity-related questions:

- ▶ Does the integrity of the data warrant protection?

If SOAP messages do not contain critical data, or if the messages are transmitted only within an internal secure network, then flowing unsigned messages might be reasonable.

A single CICS region can process signed and unsigned messages. For example, you can define pipelines that expect to receive signed messages and other pipelines that will reject signed messages.

- ▶ Are intermediaries used?

SSL/TLS provides integrity of the data only during the message transmission. XML signatures might be necessary to protect message integrity within every intermediary node.

- ▶ Does CICS need to deal with signed messages?

It might be appropriate for an intermediary server to terminate an HTTPS session and forward the request to CICS across a secured network as an HTTP request.

Equally, it might be appropriate for an intermediary server to validate an XML digital signature and forward an unsigned message to CICS. In this case, it is still possible for the service requester's identity to flow with the unsigned message so that it can be used for CICS resource authorization checking.

9.4.1 System z security and integrity capabilities

The System z architecture together with z/OS and CICS provides the following levels of securing data from being accessed by non authorized users:

- ▶ The System z virtualization known as LPARs are protected from unauthorized access from other LPARs.
- ▶ The z/OS architecture prevents individual regions, address spaces, within an LPAR from accessing data belonging to other regions. The integrity feature of z/OS prevents unauthorized access to protected storage with a region.
- ▶ CICS internally uses storage keys and storage protection to protect tasks from accessing other tasks storage.

System z virtualization

With the Common Criteria Evaluation Assurance Level 5 (EAL5) awarded by International Standards Organization, System z has the highest security rating or classification for any commercially available server. The EAL5 ranking can provide customers the assurance and confidence that they can run many different applications containing confidential data on one System z, which is divided into partitions that keep each application's data secure and distinct from the others. This isolation allows images that support separate operating systems to run in separate partitions on a single mainframe.

z/OS system integrity

Specifically, z/OS system integrity is defined as the inability of any program not authorized by a mechanism under the installation's control to circumvent or disable store or fetch protection, access a resource protected by the z/OS Security Server (RACF), or obtain control in an authorized state, that is, in a supervisor state, with a protection key less than 8, or Authorized Program Facility (APF) that is authorized.

z/OS can help you protect your system, data, transactions, and applications from accidental or malicious modification. This is one of the many reasons IBM System z remains the industry's premier data server for mission-critical workloads.

9.4.2 CICS storage protection and transaction isolation

Storage protection protects CICS code and control blocks from applications. Transaction isolation protects tasks from each other.

Storage protection

The z/OS subsystem storage protection facility helps you to prevent CICS code and control blocks from being overwritten accidentally by your application programs. It does not provide protection against deliberate overwriting of CICS code or control blocks. CICS cannot prevent an application from obtaining the necessary access (execution key) to modify CICS storage. The use of storage protection is optional. You choose whether you want to use storage protection facilities by means of CICS system initialization parameters.

Transaction isolation

Transaction isolation extends this storage protection to provide protection for transaction data. Accidental overwriting of the transaction data by an application program of another transaction can affect the reliability and availability of your CICS system and the integrity of the data in the system.

9.5 Confidentiality

Confidentiality ensures that an unauthorized party cannot obtain the meaning of the transferred or stored data. Typically, confidentiality is achieved by encrypting the data.

Consider the following confidentiality-related questions:

- ▶ Does the sensitivity of the data warrant encryption?
If HTTP messages, being native HTTP or SOAP that uses HTTP, do not contain sensitive data, or if the messages are transmitted only within an internal secure network, then flowing unencrypted messages might be reasonable.
- ▶ Are intermediaries used?
SSL/TLS provides privacy of the data only during the message transmission. Protecting confidential information in a SOAP message using XML encryption might be necessary to protect message confidentiality within every intermediary node.
Be aware that XML encryption might make content-based routing of SOAP messages difficult because the intermediary server will not be able to read all parts of the message body.

- Does CICS need to deal with encrypted messages?

An appropriate approach might be for an intermediary server to terminate an HTTPS session or to decrypt XML-encrypted messages and forward unencrypted messages to CICS.

- Does CICS call an STS?

You should use SSL/TLS to keep the connection to the STS secure.

9.6 Cryptography

Cryptography is the scientific discipline for the study and development of *ciphers*, in particular, encryption and decryption algorithms. These cryptographic procedures are the essential components that enable secure communication to take place across networks that are not secure. SSL/TLS encryption uses both *symmetric* and *asymmetric* keys.

- Symmetric (secret) key

Secret key cryptography means that the sender and receiver share the same (symmetric) key, which is used to encrypt and decrypt the data, as illustrated in Figure 9-6.

The secret key encryption and decryption process is often used to provide privacy for high-volume data transmissions.

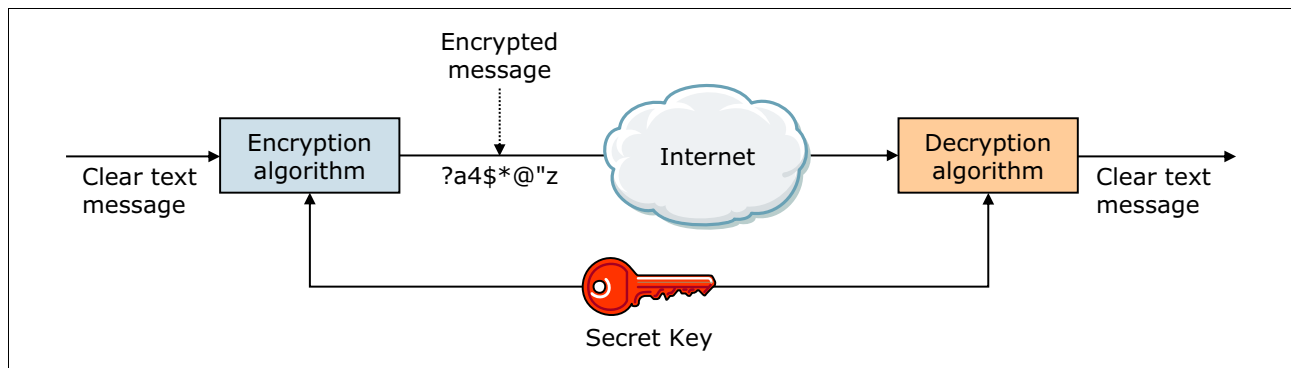


Figure 9-6 Use of secret key

► Asymmetric (public/private) key

Public/private key cryptography uses an asymmetric algorithm. The private key is known only by its owner and is never disclosed. The corresponding public key can be known by anyone. The public key is derived from the private key, but it cannot be used to deduce the private key. Either key of the pair can be used to encrypt a message, but decryption is only possible with the other key. This use is illustrated in Figure 9-7.

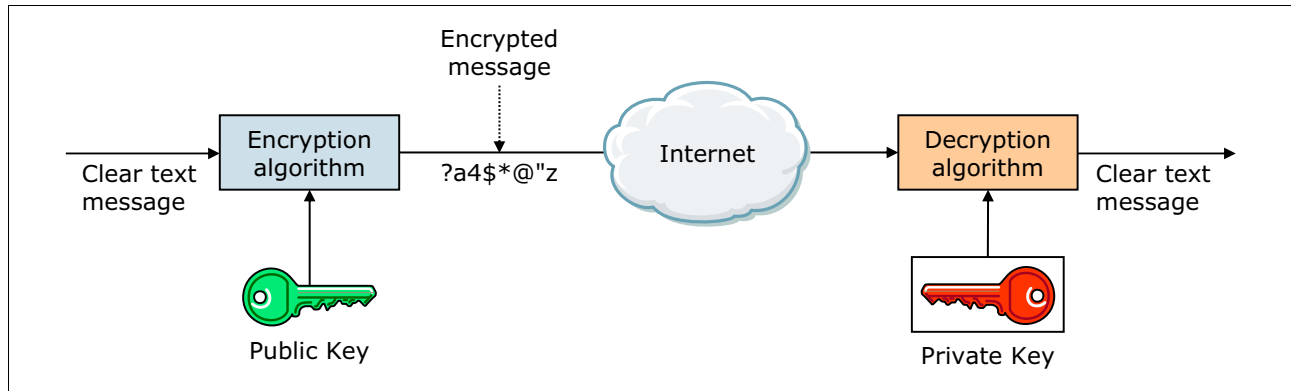


Figure 9-7 Use of private/public key

Most of the CICS integration options mentioned in this book support security solutions that are based on these cryptographic procedures, including SSL/TLS.

System z cryptographic added value

System z cryptographic hardware supports four cryptographic capabilities:

- Data confidentiality: Encrypting or decrypting data by using symmetric, asymmetric, or both algorithms.
- Message integrity: Message authentication, modification detection, non repudiation
- Financial functions: Using symmetric algorithms to protect personal identification numbers (PINs) that are associated with credit cards and financial transactions
- Key management: Security and integrity of keys

In this book we discuss cryptography only in support of data confidentiality and message integrity.

Use of clear key, secure key or protected key

IBM Cryptographic hardware can use clear key, secure key or protected key. No matter which you choose, there is no difference in the cryptographic algorithms and the resulting ciphertext is the same if the underlying key is the same. The difference is the protection that is provided for the key value that protects the data. Value is used to perform the cryptographic operation.

A secure key never exists, in the clear, outside of the tamper-resistant boundary of the secure hardware. When a secure key leaves that tamper-resistant boundary (to be stored in a data set) it is first encrypted under the master key.

A protected key never exists, in the clear, in system storage. A protected key is encrypted under a wrapping key that is uniquely created each time the LPAR is activated or reset. That wrapping key is stored in the hardware storage area (HSA) and cannot be accessed by an application or the operating system.

Integrated Cryptographic Services Facility

Hardware cryptography is available through the use of an Integrated Cryptographic Services Facility (ICSF) card and associated software on System z.

Transport Layer Security (TLS) protocol

CICS supports two security protocols that can be used to provide secure communication over the Internet. The first is the Secure Sockets Layer (SSL) 3.0 protocol. The second is the Transport Layer Security (TLS) 1.0 protocol, which is based on SSL 3.0.

The primary goal of SSL/TLS is to provide privacy (confidentiality) and data integrity between two applications that are communicating over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

When a TLS client and server first start communicating, a *handshake* occurs. During the handshake, the client and server agree on which version of the TLS protocol they will use, select a cipher suite, optionally authenticate each other, and use public key encryption techniques to generate shared secrets.

SSL/TLS requires a server X.509 certificate, which is stored in the server's certificate key ring. The certificate is used as part of the handshake server authentication process. The client validates the server certificate. Successful server authentication requires that the Certificate Authority (CA) that signed the server certificate be considered trusted by the client. To be considered trusted, the certificate of the CA must be in the key ring of the client.

SSL/TLS optionally uses a client X.509 certificate that is used as part of the handshake client authentication process. To use client authentication, the client must have a client X.509 certificate. The server validates the client certificate. Successful client authentication requires that the certificate authority (CA) that signed the client certificate be considered trusted by the server. To be considered trusted, the certificate of the CA must be in the key ring of the server.

CICS uses z/OS System SSL (a component of z/OS Communications Server) to support both the SSL and TLS protocols.

Application Transparent Transport Layer Security (AT-TLS)

AT-TLS consolidates TLS implementation in one location, in a way that exploiters can then use these centralized services without having to implement the TLS protocol themselves. AT-TLS is based on z/OS System SSL, and transparently implements these protocols in the TCP layer of the stack.

For TCP/IP access to CICS, you can either use the CICS-provided SSL/TLS support or AT-TLS. The CICS-provided support requires more configuration but allows authentication information from the client X.509 certificates to be used for authentication and identification purposes within CICS.

9.7 Auditing and non-repudiation

With auditing, you capture and record security-related events (such as a user signing on to or off of a system) so that you can analyze them later, perhaps after a breach of your security has occurred.

Consider the following auditing-related questions:

- ▶ Does the service request need to be audited?

Typical information that needs to be captured is the RACF user ID that is used for running the CICS task, the operation, and the nature of the data update.

- ▶ Do you need an audit log that contains the original distributed identity?

Consider the z/OS identity propagation support that provides for identity assertion, control, and auditing as part of the built-in processing within CICS, RACF, and DataPower.

- ▶ Will you create an audit trail in CICS or in an intermediary server, or both?

Perhaps you need an audit trail in all the servers that are involved in processing a request.

DataPower is an ideal place to perform auditing because it provides a configurable solution that supports multiple format log records that can be stored on the appliance or transferred off-device.

9.7.1 Auditing unauthorized access to CICS resources

Except when processing certain security commands, CICS issues security authorization requests with the logging option. This means that RACF writes SMF type 80 log records to SMF. Which events are logged depends on the auditing in effect. For example, events requested by the AUDIT or GLOBALAUDIT operand in the resource profile, or by the SETROPTS AUDIT or SETROPTS LOGOPTIONS command, can be logged.

In addition to the SMF TYPE 80 log record, RACF issues an ICH408I message to consoles designated to receive messages for route code 9.

9.7.2 Auditing updates to CICS resource definitions

The resource signature, the combination of the definition and the installation signatures, can be used to audit and manage resources by capturing details when the resource is defined, installed, and last changed.

Being able to display information about when the resource was defined, installed, and last changed helps with problem determination. The details improve the auditing and tracing of resources and can be displayed in the CICS Explorer views, CICSplex SM views, CEDA panels, using EXEC CICS INQUIRE SPI commands and CEMT INQUIRE commands.

Only the latest update to a CICS resource definition is logged. If a history of updates is required, a product like CICS Configuration Manager must be used.

9.7.3 Auditing updates to CICS application programs

As important as it might be to audit changes to CICS resources definitions that are used by an application, consider keeping audit changes made to the CICS application programs themselves.

9.7.4 Non-repudiation

Non-repudiation means that a sender and a receiver of data are able to provide legal proof to a third party that the sender did send the information and that the receiver received the identical information. Neither side is *able to deny*.

A solution for non-repudiation must provide proof of the integrity and origin of data, and an authentication mechanism that with high assurance can be claimed to be genuine. The most common method of asserting the origin of data is through the use of digital signatures and a form of public key infrastructure (PKI).



Part 4

Appendix and related publications

This part contains information about supported standards and related publications



Supported standards

This appendix provides a summary of the key industry and open standards that are supported by CICS Transaction Server for z/OS V5.1:

- ▶ “Networking standards” on page 242
- ▶ “Web service standards” on page 242
- ▶ “Security standards” on page 243
- ▶ “Other standards and frameworks” on page 243

For the most recent details, see the topic about supported standards in the relevant CICS TS information center, for your release.

Networking standards

Table A-1 Networking standards

Standard	Description
HTTP v	CICS web support is conditionally compliant with the HTTP/1.1 specification.
Atom	CICS supports the Atom format and Atom publishing protocol described in RFC 4287 and RFC 5023.
TCP/IP	CICS supports both IPv4 and IPv6.
Sockets	IBM Communication Server provides a CICS TCP/Programming interface for CICS. This provides a variant of the Berkeley Software Distribution 4.3 sockets interface.
APPC architecture	CICS implements the APPC architecture and supports a limited number of the APPC option sets.

Web service standards

Table A-2 Web service standards

Standard	Description
Web service standards	▶ SOAP 1.1 and 1.2 ▶ Web Services Addressing 1.0 ▶ WS-I Basic Profile 1.1 ▶ Web Services Description Language (WSDL) Version 1.1 and 2.0 ▶ Web Services Atomic Transaction (WSAT) Version 1.0 ▶ Web Services Coordination Version 1.0 ▶ SOAP Message Transmission Optimization Mechanism (MTOM) ▶ WS-I Basic Profile 1.1 ▶ SOAP Message Transmission Optimization Mechanism (MTOM) ▶ XML-binary Optimized Packaging (XOP) ▶ SOAP 1.1 Binding for MTOM 1.0 ▶ Web Services Trust Language (WS-Trust)
Web services security	▶ X.509 Certificate Token Profile 1.0 ▶ SOAP Message Security ▶ UsernameToken Profile 1.0

Security standards

Table A-3 Security standards

Standard	Description
Secure Sockets Layer	SSL 3.0
Transport Layer Security	TLS 1.0
WS-Security	See Table A-2 on page 242
User authentication tokens	RACF passwords, passtickets, and password phrases

Other standards and frameworks

Table A-4 Framework

Standard	Description
Java (run time)	IBM 64-bit SDK for z/OS V7
Java (development)	Any IBM SDK Java Technology Edition V6 or later
JavaServer Pages	JSP 2.2 ^a
Servlet	Servlet 3.0*
OSGi	CICS supports the OSGi Service Platform Core Specification V4.3 for the development and deployment of Java applications.
SCA Assembly Model 1.0	CICS conditionally complies with the SCA Assembly Model specification
XML	Extensible Markup Language Version 1.0

a. Provided through the embedded WebSphere Liberty Profile Support

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.

- ▶ *CICS and SOA: Architecture and Integration Choices*, SG24-5466
- ▶ *CICS/ESA and TCP/IP for MVS Sockets Interface*, GG24-4026
- ▶ *CICS Service Flow Runtime User's Guide*, SC34-6913
- ▶ *CICS Transaction Server from Start to Finish*, SG24-7952
- ▶ *CICS Web Services Workload Management and Availability*, SG24-7144
- ▶ *IBM CICS Explorer*, SG24-7778
- ▶ *Threadsafe Considerations for CICS*, SG24-6351
- ▶ *z/OS Traditional Application Maintenance and Support*, SG24-7868

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:

- ▶ *CICS Application Programming Guide*, SC34-7158
http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/topic/com.ibm.cics.ts.applicationprogramming.doc/dfhp3_pdf.pdf
- ▶ *CICS External Interfaces Guide*, SC34-6449
- ▶ *External Interfaces Guide*, SC34-7168
- ▶ *Front End Programming Interface User's Guide*, SC34-7169
- ▶ *IBM service management for CICS with System z tools*
ftp://public.dhe.ibm.com/software/http/cics/tools/WSW11330-USEN-00_cicstools_itsm_WP_0619D.pdf
- ▶ *IP CICS Sockets Guide*, SC31-8518
- ▶ *Options for Integrating CICS Applications in an SOA* white paper:
ftp://public.dhe.ibm.com/software/http/cics/pdf/Options_for_Integrating_CICS_Applications_in_and_SOA.pdf
- ▶ *z/OS V1R11.0 DFSMS Implementing System-Managed Storage z/OS*, SC26-7407-06

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Rational Developer for System z information center:
<http://publib.boulder.ibm.com/infocenter/ratdevz/v8r0/index.jsp>
- ▶ Business rule mining process in the Rational Asset Analyzer information center:
http://publib.boulder.ibm.com/infocenter/rassan/v6r0/index.jsp?topic=%2Fcom.ibm.raa.analyze.doc%2Fcommon%2Fbrm_process.html
- ▶ *Options for Integrating CICS Applications in an SOA* white paper:
ftp://public.dhe.ibm.com/software/http/cics/pdf/Options_for_Integrating_CICS_Applications_in_and_SOA.pdf
- ▶ Rational Host Access Transformation Services (HATS), go to the following website:
<http://www.ibm.com/software/awdtools/hats/>
- ▶ Eclipse:
<http://eclipse.org/>
- ▶ Java on the z/OS platform:
<http://www.ibm.com/servers/eserver/zseries/software/java/>
- ▶ Rational Software Architect information center:
http://pic.dhe.ibm.com/infocenter/rsahelp/v8/topic/com.ibm.rsa_base.nav.doc/topics/crootintro_rsa_base.html
- ▶ Source code generation capability of Rational Developer for System z:
<http://pic.dhe.ibm.com/infocenter/ratdevz/v7r5/index.jsp?topic=%2Fcom.ibm.etoos.wdz.uml.transform.doc%2Fconcepts%2Fzapgintro.html>
- ▶ Information about Java on the z/OS platform and to download the 64-bit version of the SDK:
<http://www.ibm.com/servers/eserver/zseries/software/java/>
- ▶ Remote debugging of Java applications:
http://www.ibm.com/developerworks/mydeveloperworks/blogs/cicsdev/entry/configuring_cics_and_cics_explorer_sdk_to_remotely_debug_java_applications43?lang=en
- ▶ Activate and manage tracing for JVM servers:
http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhpj_trace_jvmserver.html
- ▶ Defining and activating tracing for pooled JVMs:
http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhs1_trace_jvm.html
- ▶ IBM Monitoring and Diagnostic Tools for Java - Health Center:
<http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/>
- ▶ CICS VSAM Recovery for z/OS:
<http://www.ibm.com/software/http/cics/vr/>
- ▶ CICS VSAM Transparency for z/OS:
<http://www.ibm.com/software/http/cics/vt/>

- ▶ Rational Software Architect product overview:
http://pic.dhe.ibm.com/infocenter/rsahelp/v8/index.jsp?topic=/com.ibm.rsa_base.nav.doc/topics/crootintro_rsa_base.html
- ▶ Remote debugging of Java applications:
https://www.ibm.com/developerworks/mydeveloperworks/blogs/cicsdev/entry/configuring_cics_and_cics_explorer_sdk_to_remotely_debug_java_applications43?lang=en
- ▶ Source code generation in Rational Developer for System z:
<http://pic.dhe.ibm.com/infocenter/ratdevz/v7r5/index.jsp?topic=%2Fcom.ibm.etools.wdz.uml.transform.doc%2Fconcepts%2Fzapgintro.html>
- ▶ Activating and managing tracing for JVM servers
http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhpj_trace_jvmserver.html
- ▶ Defining and activating tracing for pooled JVMs
http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp?topic=%2Fcom.ibm.cics.ts.java.doc%2Ftopics%2Fdfhs1_trace_jvm.html
- ▶ IBM Monitoring and Diagnostic Tools for Java - Health Center
<http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/>
- ▶ CICS Transaction Server V4.2 information center:
<http://pic.dhe.ibm.com/infocenter/cicsts/v4r2/index.jsp>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Architect's Guide to IBM CICS on System z

(0.5" spine)
0.475" <-> 0.873"
250 <-> 459 pages



Architect's Guide to IBM CICS on System z



**Discover the value to
your business of CICS
on System z**

**Understand how to
design and develop
CICS applications**

**Explore the qualities
of service CICS can
provide**

IBM CICS Transaction Server (CICS TS) has been available in various guises for over 40 years, and continues to be one of the most widely used pieces of commercial software. This IBM Redbooks publication helps application architects discover the value of CICS Transaction Server to their business.

This book can help architects understand the value and capabilities of CICS Transaction Server and the CICS tools portfolio. The book also provides detailed guidance on the leading practices for designing and integrating CICS applications within an enterprise, and the patterns and techniques you can use to create CICS systems that provide the qualities of service that your business requires.

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

SG24-8067-00

ISBN 0738437441