

Considerations for CICS Web Services Performance

Performance advantages of MTOM

Recommendations and tools

Security scenarios



Chris Rayns
Isabel Arnold
Trevor Clarke
Mark Cocker
Graham Hannington
Ivan Hargreaves
Mick Harris
Dieter Hechtberger
David Knibb
Christopher Law
Anna Maciejkowicz



International Technical Support Organization

Considerations for CICS Web Services Performance

May 2009

Archived

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

First Edition (May 2009)

This edition applies to Version 3, Release 2, CICS Transaction Server.

© Copyright International Business Machines Corporation 2009. 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	ix
Trademarks	x
Preface	xi
The team that wrote this book	xii
Become a published author	xiv
Comments welcome	xv
Part 1. Introduction	1
Chapter 1. CICS Web Services overview	3
1.1 Terminology	4
1.2 Performance scenarios	4
1.3 Performance elements	5
1.4 System configuration	5
1.5 Web services	6
1.5.1 Properties of a Web service	7
1.5.2 Core standards	8
1.5.3 Additional standards	11
1.6 SOAP	12
1.6.1 The envelope	12
1.6.2 Communication styles	17
1.6.3 Encodings	17
1.6.4 Messaging modes	18
1.7 WSDL	18
1.7.1 WSDL document	19
1.7.2 WSDL document anatomy	20
1.7.3 WSDL definition	24
1.7.4 WSDL bindings	30
1.8 Summary	31
1.9 Support for Web services in CICS TS V3	32
1.9.1 What is provided in CICS TS V3.1	32
1.9.2 What is new in CICS TS V3.2	34
Chapter 2. Web Services performance considerations	37
2.1 Transport considerations	38
2.1.1 CICS connector choices	38
2.1.2 Web service transports	39
2.2 XML parsing and payload size	40

2.2.1	Payload size	40
2.2.2	XML complexity	41
2.2.3	Using MTOM/XOP for binary data	42
2.3	Security and performance	44
2.3.1	Transport level security	45
2.3.2	WS-Security	46
2.3.3	WS-Security with DataPower	46
2.3.4	Security summary	46
2.4	CICS TS V3.2 performance improvements	48
2.4.1	HTTP enhancements	48
2.4.2	WS-Security improvements	49
2.4.3	Code page conversion	49
2.4.4	64-bit containers	49
2.5	System z10 hardware	50
Part 2.	CICS Tools overview	51
Chapter 3.	RMF	53
3.1	RMF overview	54
3.2	Using RMF	55
3.3	Using RMF with CICS	55
3.4	Defining reporting classes using WLM	56
3.5	RMF monitors	61
3.6	Starting RMF	62
3.7	Sample RMF definitions	62
3.8	Workload Activity Report	64
3.9	Measuring CICS Web Services performance using RMF	65
Chapter 4.	CICS PA	71
4.1	Overview	72
4.2	Analyzing CICS Web Services performance	81
4.3	Charting CICS Web Services performance	85
Chapter 5.	IBM Tivoli Monitoring	87
5.1	Tivoli Management Services components	88
5.1.1	Tivoli Enterprise Monitoring Server	88
5.1.2	Tivoli Enterprise Portal Server	89
5.1.3	Tivoli Enterprise Portal	90
5.1.4	Tivoli Enterprise Monitoring Agents	91
5.1.5	Tivoli Data Warehouse	91
5.1.6	Warehouse Proxy Agent	92
5.1.7	Warehouse Summarization and Pruning Agent	92
5.1.8	Tivoli Management Services Engine	92
5.2	Tivoli Management Services resources	93

5.2.1 Workspaces	93
5.2.2 Navigator.....	94
5.2.3 Links	95
5.2.4 Views.....	96
5.2.5 Query	96
5.2.6 Attributes.....	96
5.2.7 Situations	98
5.2.8 Actions	99
5.3 Additional information	101
Chapter 6. ITCAM FOR SOA	103
6.1 ITCAM for SOA overview	104
6.1.1 Composite application management.....	104
6.1.2 Basic architecture	106
6.1.3 Supported application environments.....	108
6.2 Product components	108
6.2.1 Data collector	109
6.2.2 Tivoli Enterprise Monitoring Agent	112
6.2.3 ITCAM for SOA topology support	113
6.2.4 Product features	115
6.3 ITCAM for SOA installation on z/OS	122
6.3.1 ITCAM for SOA management agent for z/OS.....	123
6.3.2 Enabling the CICS data collector	134
Chapter 7. OMEGAMON XE for CICS on z/OS	137
7.1 OMEGAMON XE for CICS components	138
7.1.1 OMEGAMON XE for CICS Monitoring Agent	138
7.1.2 OMEGAMON II for CICS Menu System	142
7.1.3 OMEGAMON II for CICS CUA interface: Optional	146
7.1.4 OMEGAMON for CICS component in monitored CICS regions ...	147
7.2 OMEGAMON XE for CICS features	149
7.2.1 Resource monitoring.....	149
7.2.2 Service level analysis	151
7.2.3 Bottleneck Analysis.....	158
7.2.4 Transaction history	162
7.2.5 CICS Web Services monitoring.....	169
7.2.6 Resource Limiting	173
Part 3. Performance scenarios	175
Chapter 8. Environment overview	177
8.1 The example application	178
8.2 Installing and setting up the application	178
8.3 MTOM scenario overview	179

8.3.1	Why we use MTOM/XOP	179
8.3.2	Modifications to the catalog application	180
8.4	MTOM scenario preparation	180
8.4.1	System properties	180
8.4.2	Definition checklist	181
8.4.3	Generate workload with varying binary data size	185
8.5	Security scenarios overview	187
8.6	Security scenario preparation	196
8.6.1	Software checklist	196
8.6.2	Definition checklist	196
8.6.3	Basic CICS security configuration	198
8.6.4	CICS Web Services configuration	199
Chapter 9.	MTOM scenario	209
9.1	Introduction to MTOM/XOP	210
9.1.1	XOP processing considerations	211
9.1.2	Catalog application changes for MTOM/XOP	211
9.2	Results	212
9.2.1	MTOM provider results	212
9.2.2	MTOM requester results	217
9.2.3	Effects of switching workload to System z10	222
9.3	Conclusions	225
Chapter 10.	Security scenarios	229
10.1	Scenario overview	230
10.1.1	Our environment	230
10.1.2	Workload simulation	231
10.1.3	Performance metrics	232
10.2	Preparing the scenarios	233
10.2.1	Preparing the CICS XML digital signature scenario	233
10.2.2	Preparing the DataPower X.509 certificate scenario	240
10.2.3	Preparing the DataPower UsernameToken scenario	244
10.3	Running the scenarios	248
10.3.1	Starting the simulation	248
10.3.2	Breakdown of results	250
10.3.3	Investigation of the limits	251
10.4	Results	259
10.4.1	CICS XML digital signature	259
10.4.2	DataPower with X.509 certificate	264
10.4.3	DataPower with UsernameToken	268
10.4.4	Comparison of CPU time	272
10.4.5	Comparison of response time	274
10.5	Conclusions	275

Chapter 11. Using IBM Tivoli Monitoring Tools	277
11.1 Configuration	278
11.1.1 Creating a situation	278
11.1.2 Adding a dynamic workspace link	282
11.2 CPU constraint scenario	289
11.3 File constraint scenario	294
Part 4. Appendixes	301
Appendix A. The modified catalog manager application	303
Introduction to the catalog manager application	304
Providing channel and container interface to the base application	306
COMMAREA structure	309
Data separation of the structure	311
EFH0X02 copybook	313
Creating the separator program EFH0XSEP	314
Extending the catalog manager module	318
Running the 3270 application with channels and containers	323
The Web service interface to catalog manager	324
Adding the image retriever functionality	328
Creating the image server	328
Extensions to the channel container copybook EFH0XS02	330
Extension to the catalog manager	330
Extensions to the Web service wrapper	331
Extending the application with a Web service requester	333
Deploying the image server Web service provider	334
Creating the image client Web service requester	334
Changes to the base application to link to requester	336
Appendix B. XSL to add UsernameToken in DataPower	337
Appendix C. Additional material	341
Locating the Web material	341
Using the Web material	342
System requirements for downloading the Web material	342
How to use the Web material	342
Related publications	343
IBM Redbooks publications	343
Other publications	343
Online resources	344
How to get IBM Redbooks publications	344
Help from IBM	344

Archived

Notices

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:


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

Trademarks

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

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

CICS®
DataPower®
DB2®
IBM®
OMEGAMON II®
OMEGAMON®
RACF®

Rational®
Redbooks®
Redbooks (logo) ®
System z10™
System z9®
System z®
Tivoli®

VTAM®
WebSphere®
z/OS®
z9®
zSeries®

The following terms are trademarks of other companies:

SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

J2EE, Java, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

Preface

The Web services support in CICS® Transaction Server Version 3 enables your CICS programs to be Web service providers or requesters. CICS supports a number of specifications including SOAP Version 1.1 and Version 1.2, and Web services distributed transactions (WS-Atomic Transaction).

This IBM® Redbooks® publication reviews CICS Web Services performance in two particular areas:

- ▶ MTOM/XOP: Here we focus on performance benefits that can be achieved by taking advantage of the MTOM/XOP standard to transmit large binary data objects. The single inquiry function of the CICS catalog manager application has been modified to return an image of the requested item in addition to its details. This function is exposed through a Web service.
- ▶ Security: There are a number of ways to secure your CICS Web Services messages. Here we look at two technologies: WS-Security and DataPower®. We compare the performance and relative merits of using WS-Security with and without DataPower, and discuss what factors might influence your choice of technology.

We highlight tools that can help you to understand the performance profile of Web service interactions with CICS, such as;

- ▶ RMF z/OS® Resource Measurement Facility (RMF)
- ▶ IBM CICS Performance Analyzer for z/OS (CICSPA)
- ▶ ITCAM for SOA
- ▶ OMEGAMON® XE for CICS on z/OS

This book considers performance by using different scenarios including Security, MTOM/XOP, and the use of the IBM Tivoli® Monitoring tools to help identify problems that can affect the performance of Web Services. Specifically, we use ITCAM for SOA and OMEGAMON XE for CICS on z/OS to show how these tools can be of benefit to identify the cause when the performance of a Web service in CICS becomes unacceptable.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the Hursley Center, IBM Hursley Laboratory, UK.

Chris Rayns is an IT Specialist and the CICS/Security Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively on all areas of z/OS security. Before joining the ITSO, Chris worked in IBM Global Services in the United Kingdom as a CICS IT Specialist.

Isabel Arnold is a CICS Technical Sales Specialist in Stuttgart, Germany. She has six years of experience in the Information Technology field, of which one year was focused on mainframe technology. She holds a German Diploma from University of Corporate Education Stuttgart and a BSc with Honours degree in Information Technology Management from Open University London. Her areas of expertise include CICS application transformation and integration and development tools, Java™, and J2EE™.

Trevor Clarke is a CICS Performance Specialist in Hursley, England. He has 30 years of experience in the IT industry, working both for IBM and as a customer. For most of this period he has worked with CICS in technical support, application development, and performance roles, and also as an educator. His areas of expertise also include DB2® and networking. He holds a BSC Mathematics degree from Southampton University.

Mark Cocker is a Senior Software Engineer working at the IBM Hursley Laboratory in England. He has 17 years experience working in CICS development, the Change Team, the IBM Design Center, and now in CICS Technical Strategy. He holds a degree in Information Systems Management from Bournemouth University, England. His areas of expertise include communication methods such as Web services. Mark has written extensively on and presents regularly the latest CICS TS capabilities and product strategy.

Graham Hannington is a Technical Writer at Fundi Software, in Perth, Western Australia, where CICS CM and CICS PA are developed. He has 18 years of experience in information development. He wrote the CICS CM user's guide and has developed a SupportPac for CICS PA. His areas of expertise include technical writing and illustration; XML-related technologies such as XSLT, XML schema, and XML DOM programming; and VBScript and VBA programming.

Ivan Hargreaves is a CICS Developer in Hursley, England. He has 15 years of experience in the IT industry; this includes running his own company, as well as working as a Software Developer for various UK companies. Ivan has spent the last eight years at IBM. For most of this period he has worked in CICS, initially as a tester before becoming a developer of CICS Tools and eventually a CICS Web Services developer. His areas of expertise include Web Services (particularly WS-Security), C++, and Java. He holds an M.Eng (hons) degree in Computer Systems Engineering from the University of Wales (Bangor).

Mick Harris is a Senior Software Engineer in Raleigh, North Carolina. He has 25 years experience in the IT industry. He worked as a customer involved in systems programming before going on to a software development role with IBM and Candle. He worked in the CICS change team and has held both development and change team roles for a number of OMEGAMON and Tivoli products; primarily the CICS monitoring products, OMEGAMON II® for CICS and OMEGAMON XE for CICS.

Dieter Hechtberger is a Certified IT Specialist at IBM, working as a Tivoli Technical Sales team leader on the CEMAAS System z® SW team based in Vienna, Austria. He has 26 years of experience in mainframe computing. His areas of expertise include availability and performance management with a focus on data center management tools like Omegamon and ITCAM. Dieter holds a Masters degree in Mathematics from the Technical University in Vienna.

David Knibb is a Software Engineer at the IBM Hursley Laboratory in England. He has two years experience working to support the CICS software testing team within IBM, where his primary field is Automated Test Execution. He holds a BSc in Computer Science from the University of Durham.

Christopher Law is a CICS System Tester in Hursley, England. He has nine years experience in the IT industry, all working for various areas in IBM. He started with four years in the support organization, firstly on Z/OS, then progressing to Linux® support. Since then, he has worked on MQ, WebSphere® Application Server (messaging) and now most recently on CICS. He graduated from Loughborough University with a BSc in Mathematics and Computation.

Anna Maciejkowicz is a Software Engineer at IBM Hursley Laboratory, England. She holds a degree in Computer Science and Mathematics from Manchester University, England. Since joining IBM as a graduate two years ago she has worked in CICS interoperability testing, specifically Web services, CICS Tools and J2EE technologies.

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

Richard Conway
International Technical Support Organization, Hursley Center

Robert Haimowitz
International Technical Support Organization, Hursley Center

Fraser Bohm, Software Developer
IBM Hursley

Chris Baker, Software Developer
IBM Hursley

Chris Walker, Software Developer
IBM Hursley

Yvonne Scott-Matute
IBM US

Charlie Wiese, CICS Level 2 Support
IBM US

Nigel Williams, Certified IT specialist
IBM Montpellier

Phil Wakelin, CICS Transaction Gateway Technical Planner
IBM Hursley

James O'Grady, CICS Tester
IBM Hursley

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review IBM Redbooks publications form found at:
ibm.com/redbooks

- ▶ Send your comments in an e-mail to:
redbooks@us.ibm.com

- ▶ Mail your comments to:

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



Part 1

Introduction

In this part of the book, we review CICS Web Services, and then highlight the reasons why we are interested in performance.

Archived

CICS Web Services overview

In this chapter, we define the scope of this book and introduce the tools used in the performance scenarios.

1.1 Terminology

The following terms, referred to in the book, are defined here for ease of reference:

- ▶ *Service provider*: An entity that provides a Web Service
- ▶ *Service requestor*: An entity responsible for requesting a service from a service provider

1.2 Performance scenarios

This book focuses on the performance of CICS Web Services using a number of different scenarios. See Chapter 9, “MTOM scenario” on page 209, and Chapter 10, “Security scenarios” on page 229. Note that the performance of CICS Web Services in any system is influenced by other factors including but not limited to:

- ▶ CICS region configuration
- ▶ System workload
- ▶ Network topology
- ▶ z/OS configuration
- ▶ Hardware configuration

CICS Transaction Server for z/OS V3.2 contains enhancements to improve the performance of CICS Web Services, including optimized support for codepage conversion. Other performance enhancements include:

- ▶ Support for SOAP Message Transmission Optimization Mechanism (MTOM)
- ▶ HTTP headers using containers instead of temporary storage
- ▶ Additional commands made threadsafe

Useful information regarding CICS performance can be found at the *CICS Transaction Server for z/OS, Version 3 Release 2 Information Center*, at

<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>

Then refer to the *CICS Transaction Server for z/OS V3.2* section on Improving performance.

Establishing optimal performance of any software product is important because of the interdependent nature of IT infrastructure. Each software product might have a role to play in any number of business applications, so optimal settings can vary across installations. See Chapter 2, “Web Services performance considerations” on page 37 for some general performance considerations. The scenarios covered by this book result in specific recommendations and best practice guidelines that can prove helpful in configuring CICS Web Services for optimal performance.

1.3 Performance elements

The main performance elements addressed by this book are:

- ▶ Response time: This can be defined as the interval between a user command and the receipt of a result from the system. It is important because it has a direct impact on the end to end elapsed time experienced by applications.
- ▶ CPU time: This can be defined as the amount of time taken by the processor to execute a set of instructions. It is important because efficient CPU usage benefits all users of the CPU. In addition, CPU time is used to determine the financial cost of applications in many installations.
- ▶ Throughput: This can be defined as the amount of work that can be processed in a given time period. It is important because efficient throughput benefits the overall system workload, allowing more work to be processed.

Configuring these elements to their optimum settings benefits all installations and can be important factors in meeting any service obligations such as those specified in Service Level Agreements (SLAs).

1.4 System configuration

We used the following hardware and software to run the scenarios described in this book:

- ▶ One LPAR on a 2094-S18 with eight GB of storage and four shared CPs.
- ▶ z/OS 1.9. For more information, go to:
<http://publib.boulder.ibm.com/infocenter/zos/v1r9/index.jsp?topic=/com.ibm.zos.r9/zosr9home.html>

- ▶ z/OS 1.9 Resource Management Facility (hereafter referred to as RMF).
For more information, go to:
<http://publib.boulder.ibm.com/infocenter/zos/v1r9/index.jsp?topic=/com.ibm.zos.r9.e0ze100/rmf.htm>
- ▶ CICS Transaction Server for z/OS V3.2 (hereafter referred to as CICS TS).
For more information, go to:
<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>
- ▶ CICS Performance Analyzer for z/OS V2.1 (hereafter referred to as CICS PA). For more information, go to:
<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>
(Then refer to the CICS Performance Analyzer section.)
- ▶ IBM Tivoli Omegamon XE for CICS on z/OS V4.1 (hereafter referred to as Omegamon XE for CICS). For more information, go to:
<http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon.cics.doc/welcome.htm>
- ▶ IBM Tivoli Composite Application Manager for SOA for z/OS V7.1 (hereafter referred to as ITCAM for SOA). For more information, go to:
<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.itcamsoa.doc>
- ▶ Websphere DataPower Integration Appliance XI50 (hereafter referred to as Websphere DataPower). For more information, go to:
<http://www.ibm.com/software/integration/datapower/xi50/>

The results obtained were representative of the environment used. To assist with extrapolating results for different environments, IBM provides *Large System Performance Ratios (LSPRs)* that contain data on Internal Throughput Rates (ITRs) for various processor configurations running z/OS. LSPR is available at:

<http://www.ibm.com/systems/z/advantages/management/lspr/>

In the following sections we provide more details of these tools and the scenarios in which they were used to analyze the performance of CICS Web Services.

1.5 Web services

Web services perform encapsulated business functions, ranging from simple request-reply to full business process interactions. These services can be new applications or just wrapped around existing business functions to make them network-enabled. Services can rely on other services to achieve their goals.

1.5.1 Properties of a Web service

All Web services share the following properties:

- ▶ Web services are self-contained.

On the client side, no additional software is required. A programming language with XML and HTTP client support is enough to get you started. On the server side, merely an HTTP server and a SOAP server are required.

- ▶ Web services are self-describing.

Using Web Services Description Language (WSDL), all the information required to implement a Web service as a provider, or to invoke a Web service as a requester, is provided.

- ▶ Web services can be published, located, and invoked across the Web.

This technology uses established lightweight Internet standards such as HTTP. It leverages the existing infrastructure.

- ▶ Web services are modular.

Simple Web services can be aggregated to more complex ones, either using workflow techniques or by calling lower-layer Web services from a Web service implementation. Web services can be chained together to perform higher-level business functions. This shortens development time and enables best-of-breed implementations.

- ▶ Web services are language-independent and interoperable.

The client and server can be implemented in different environments. Theoretically, any language can be used to implement Web service clients and servers.

- ▶ Web services are inherently open and standard-based.

XML and HTTP are the major technical foundations for Web services. A large part of the Web service technology has been built using open-source projects. Therefore, vendor independence and interoperability are realistic goals.

- ▶ Web services are loosely coupled.

Traditionally, application design has depended on tight interconnections at both ends. Web services require a simpler level of coordination that allows a more flexible reconfiguration for an integration of the services in question.

- ▶ Web services provide programmatic access.

The approach provides no graphical user interface; it operates at the code level. Service consumers have to know the interfaces to Web services, but do not have to know the implementation details of services.

1.5.2 Core standards

Web services are built upon four core standards, as described next.

Extensible Markup Language

Extensible Markup Language (XML) is the foundation of Web services. However, because much information has already been written about XML, we do not describe it in this book. You can find information about XML at:

<http://www.w3.org/XML/>

SOAP

Originally proposed by Microsoft®, SOAP was designed to be a simple and extensible specification for the exchange of structured, XML-based information in a decentralized, distributed environment. As such, it represents the main means of communication between the three actors in an SOA: the service provider, the service requester, and the service broker. A group of companies, including IBM, submitted SOAP to the W3C for consideration by its XML Protocol Working Group. There are currently two versions of SOAP: Version 1.1 and Version 1.2.

The SOAP 1.1 specification contains three parts:

- ▶ An envelope that defines a framework for describing message content and processing instructions. Each SOAP message consists of an envelope that contains an arbitrary number of headers and one body that carries the payload. SOAP messages might contain faults; faults report failures or unexpected conditions.
- ▶ A set of encoding rules for expressing instances of application-defined data types.
- ▶ A convention for representing remote procedure calls and responses.

A SOAP message is, in principle, independent of the transport protocol that is used, and can, therefore, potentially be used with a variety of protocols, such as HTTP, JMS, SMTP, or FTP. Right now, the most common way of exchanging SOAP messages is through HTTP.

The way SOAP applications communicate when exchanging messages is often referred to as the message exchange pattern (MEP). The communication can be either one-way messaging, where the SOAP message only goes in one direction, or two-way messaging, where the receiver is expected to send back a reply.

Due to the characteristics of SOAP, it does not matter what technology is used to implement the client, as long as the client can issue XML messages. Similarly, the service can be implemented in any language, as long as it can process XML messages.

Note: The authors of the SOAP 1.1 specification declared that the acronym SOAP stands for Simple Object Access Protocol. The authors of the SOAP 1.2 specification decided not to give any meaning to the acronym SOAP.

Web Services Description Language

Web Services Description Language (WSDL) describes Web services as abstract service endpoints that operate on messages. Both the operations and the messages are defined in an abstract manner, while the actual protocol used to carry the message and the endpoint's address are concrete.

WSDL is not bound to any particular protocol or network service. It can be extended to support many different message formats and network protocols. However, because Web services are mainly implemented using SOAP and HTTP, the corresponding bindings are part of this standard.

The WSDL 1.1 specification only defines bindings that describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET and POST, and MIME. The specification for WSDL 1.1 can be found at:

<http://www.w3.org/TR/wsdl>

WSDL 2.0 provides a model as well as an XML format for describing Web services. It enables you to separate the description of the abstract functionality offered by a service from the concrete details of a service description. It also describes extensions for Message Exchange Patterns, SOAP modules, and a language for describing such concrete details for SOAP1.2 and HTTP.

There are eight Message Exchange Patterns defined. CICS TS V3.2 supports four of them:

► In-Only:

A request message is sent to the Web service provider, but the provider is not allowed to send any type of response to the Web service requester.

► In-Out:

A request message is sent to the Web service provider, and a response message is returned. The response message can be a normal SOAP message or a SOAP fault.

► In-Optional-Out:

A request message is sent to the Web service provider, and a response message is optionally returned to the requester. The response message can be a normal SOAP message or a SOAP fault.

► Robust-In-Only:

A request message is sent to the Web service provider, and no response message is returned to the requester unless an error occurs. In this case, a SOAP fault message is sent to the requester.

The other four Message Exchange Patterns that CICS TS V3.2 does not support are:

- Out-Only
- Robust-Out-Only
- Out-In
- Out-Optional-In

The specification for WSDL 2.0 can be found at:

<http://www.w3.org/TR/wsd120>

Web Services Interoperability

Web services can be used to connect computer systems together across organizational boundaries. Therefore, a set of open non-proprietary standards that all Web services adhere to maximizes the ability to connect disparate systems together.

The Web Service Interoperability (WS-I) group is an organization that promotes open interoperability between Web services regardless of platform, operating systems, and programming languages. To promote this cause, the WS-I has released a basic profile that outlines a set of specifications to which WSDL documents and Web services traffic (SOAP over HTTP transport) must adhere in order to be WS-I compliant. The full list of specifications can be found at the WS-I Web site:

<http://www.ws-i.org/>

IBM is a member of the WS-I community, and CICS support for Web services is fully compliant with the WS-I basic profile 1.0.

1.5.3 Additional standards

There are other Web services specifications that are now supported by CICS. For a list of the limitations of CICS support, refer to *CICS Web Services Guide*, SC34-6838.

Web Services Atomic Transaction

This specification, commonly known as WS-Atomic Transaction, defines the atomic transaction coordination type for transactions of a short duration. Together with the Web Services Coordination specification, it defines protocols for short term transactions that enable transaction processing systems to interoperate in a Web services environment. Transactions that use WS-Atomic Transaction have the properties of atomicity, consistency, isolation, and durability (ACID).

Web Services Security: SOAP Message Security

This specification is a set of enhancements to SOAP messaging that provides message integrity and confidentiality. The specification provides three main mechanisms that can be used independently or together:

- ▶ The ability to send security tokens as part of a message, and for associating the security tokens with message content
- ▶ The ability to protect the contents of a message from unauthorized and undetected modification (message integrity)
- ▶ The ability to protect the contents of a message from unauthorized disclosure (message confidentiality)

Web Services Trust Language

This specification, commonly known as WS-Trust, defines extensions that build on Web Services Security to provide a framework for requesting and issuing security tokens, and broker trust relationships.

SOAP Message Transmission Optimization Mechanism

SOAP Message Transmission Optimization Mechanism (MTOM) is one of a related pair of specifications that define how to optimize the transmission and format of a SOAP message. MTOM defines:

- ▶ How to optimize the transmission of base64 binary data in SOAP messages.
- ▶ How to implement optimized MIME multipart serialization of SOAP messages in a binding, independent way.

- ▶ The implementation of MTOM relies on the related XML-binary Optimized Packaging (XOP) specification. As these two specifications are so closely linked, they are normally referred to as MTOM/XOP.

1.6 SOAP

In this section we focus mainly on SOAP 1.1.

1.6.1 The envelope

A SOAP message is an *envelope* containing zero or more *headers* and exactly one *body*:

- ▶ The envelope is the root element of the XML document, providing a container for control information, the addressee of a message, and the message itself.
- ▶ Headers contain control information, such as quality of service attributes.
- ▶ The body contains the message identification and its parameters.
- ▶ Both the headers and the body are child elements of the envelope element.

Figure 1-1 shows a simple SOAP request message:

- ▶ The header tells *who* must deal with the message and *how* to deal with it. When the actor is next or when actor is omitted, the receiver of the message must do what the body says. Furthermore, the receiver must understand and process the application-defined `<TranID>` element.
- ▶ The body tells *what* has to be done: Dispatch an order for quantityRequired 1 of itemRefNumber 0010 to customerID CB1 in chargeDepartment ITS0.

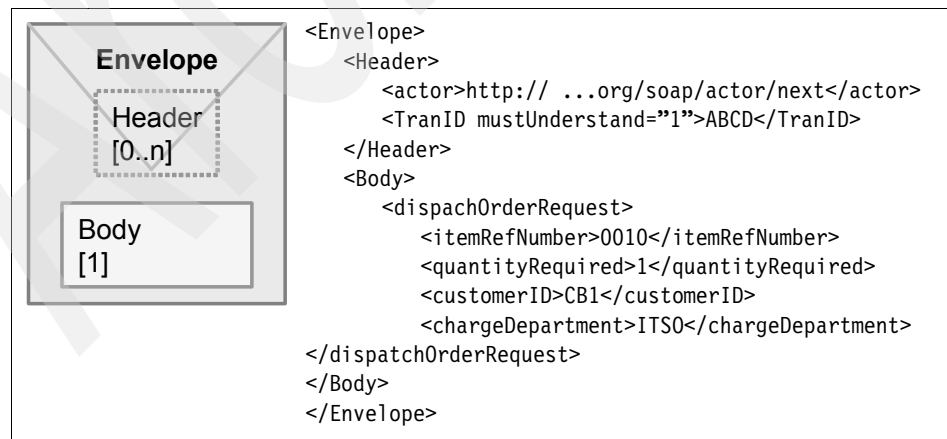


Figure 1-1 Example of a simple SOAP message

Namespaces

Namespaces play an important role in SOAP messages. A namespace is simply a way of adding a qualifier to an element name to ensure that it is unique.

For example, we might have a message that contains an element `<customer>`. Customers are fairly common so it is very likely that many Web services would have customer elements. To ensure that we know what customer we are talking about, we declare a namespace for it, for example, as follows:

```
xmlns:itso="http://itso.ibm.com/CICS/catalogApplication"
```

This identifies the prefix `itso` with the declared namespace. Then whenever we reference the element `<customer>`, we prefix it with the namespace, for example, `<itso:customer>`. This identifies it uniquely as a customer type for our application. Namespaces can be defined as any unique string. They are often defined as URLs, because URLs are generally globally unique, and they have to be in URL format. These URLs do not have to physically exist, though.

The WS-I Basic Profile 1.0 requires that all application-specific elements in the body must be namespace qualified to avoid collisions between names.

Table 1-1 shows the namespaces of SOAP and WS-I Basic Profile 1.0 used in this book.

Table 1-1 SOAP namespaces

Namespace URI	Explanation
http://schemas.xmlsoap.org/soap/envelope/	SOAP 1.1 envelope namespace
http://schemas.xmlsoap.org/soap/encoding/	SOAP 1.1 encoding namespace
http://www.w3.org/2001/XMLSchema-instance	Schema instance namespace
http://www.w3.org/2001/XMLSchema	XML Schema namespace
http://schemas.xmlsoap.org/wsdl	WSDL namespace for WSDL framework
http://schemas.xmlsoap.org/wsdl/soap	WSDL namespace for WSDL SOAP binding
http://ws-i.org/schemas/conformanceClaim/	WS-I Basic Profile

SOAP envelope

The Envelope is the root element of the XML document representing the message; it has the following structure:

```
<SOAP-ENV:Envelope .... >
  <SOAP-ENV:Header>
    <SOAP-ENV:HeaderEntry.... />
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    [message payload]
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In general, a SOAP message is a (possibly empty) set of headers plus one body. The SOAP envelope also defines the namespace for structuring messages. The entire SOAP message (headers and body) is wrapped in this envelope.

Headers

Headers are a generic and flexible mechanism for extending a SOAP message in a decentralized and modular way without prior agreement between the parties involved. They allow control information to pass to the receiving SOAP server and also provide extensibility for message structures.

Headers are optional elements in the envelope. If present, the Header element *must* be the first immediate child element of a SOAP Envelope element. All immediate child elements of the Header element are called *header entries*.

There is a predefined header attribute called SOAP-ENV:mustUnderstand. The value of the mustUnderstand attribute is either 1 or 0. The absence of the SOAP mustUnderstand attribute is semantically equivalent to the value 0.

If the mustUnderstand attribute is present in a header entry and set to 1, the service provider must implement the semantics defined by the element:

```
<Header>
  <thens:TranID mustUnderstand="1">ABCD</thens:TranID>
</Header>
```

In the example, the header entry specifies that a service invocation must fail if the service provider does not support the ability to process the TranID header.

A SOAP intermediary is an application that is capable of both receiving and forwarding SOAP messages on their way to the final destination. In realistic situations, not all parts of a SOAP message might be intended for the ultimate destination of the SOAP message, but, instead, might be intended for one or more of the intermediaries on the message path.

Therefore, a second predefined header attribute, `SOAP-ENV:actor`, is used to identify the recipient of the header information. In SOAP 1.2 the actor attribute is renamed `SOAP-ENV:role`. The value of the SOAP actor attribute is the URI of the mediator, which is also the final destination of the particular header element (the mediator does not forward the header).

If the actor is omitted or set to the predefined default value, the header is for the actual recipient, and the actual recipient is also the final destination of the message (body). The predefined value is:

`http://schemas.xmlsoap.org/soap/actor/next`

If a node on the message path does not recognize a `mustUnderstand` header and the node plays the role specified by the actor attribute, the node must generate a SOAP `MustUnderstand` fault. Whether the fault is sent back to the sender depends on the message exchange pattern in use. For request/response, the WS-I BP 1.0 requires the fault to be sent back to the sender. Also, according to WS-I BP 1.0, the receiver node must discontinue normal processing of the SOAP message after generating the fault message.

Headers can carry authentication data, digital signatures, encryption information, and transactional settings. They can also carry client-specific or project-specific controls and extensions to the protocol; the definition of headers is not just up to standards bodies.

Note: The header must not include service instructions (that would be used by the service implementation).

Body

The SOAP `Body` element provides a mechanism for exchanging information intended for the ultimate recipient of the message. The `Body` element is encoded as an immediate child element of the SOAP `Envelope` element. If a `Header` element is present, then the `Body` element *must* immediately follow the `Header` element. Otherwise it *must* be the first immediate child element of the `Envelope` element.

All immediate child elements of the `Body` element are called *body entries*, and each body entry is encoded as an independent element within the SOAP `Body` element. In the most simple case, the body of a basic SOAP message consists of an XML message as defined by the schema in the `types` section of the WSDL document. It is legal to have any valid XML as the body of the SOAP message, but WS-I conformance requires that the elements be namespace qualified.

Error handling

One area where there are significant differences between the SOAP 1.1 and 1.2 specifications is in the handling of errors. Here we focus on the SOAP 1.1 specification for error handling.

SOAP itself predefines one body element, which is the `fault` element used for reporting errors. If present, the `fault` element must appear as a body entry and must not appear more than once. The children of the `fault` element are defined as follows:

- ▶ `faultcode` is a code that indicates the type of the fault. SOAP defines a small set of SOAP fault codes covering basic SOAP faults:
 - `soapenv:Client`, indicating that the client sent an incorrectly formatted message
 - `soapenv:Server`, for delivery problems
 - `soapenv:VersionMismatch`, which can report any invalid namespaces specified on the `Envelope` element
 - `soapenv:MustUnderstand`, for errors regarding the processing of header content
- ▶ `faultstring` is a human-readable description of the fault. It must be present in a fault element.
- ▶ `faultactor` is an optional field that indicates the URI of the source of the fault. The value of the `faultactor` attribute is a URI identifying the source that caused the error. Applications that do not act as the ultimate destination of the SOAP message must include the `faultactor` element in a SOAP fault element.
- ▶ `detail` is an application-specific field that contains detailed information about the fault. It must not be used to carry information about errors belonging to header entries. Therefore, the absence of the `detail` element in the `fault` element indicates that the fault is not related to the processing of the body element (the actual message).

For example, a `soapenv:Server` fault message is returned if the service implementation throws a `SOAP Exception`. The exception text is transmitted in the `faultstring` field.

Although SOAP 1.1 permits the use of custom-defined `faultcodes`, the WS-I Basic Profile only permits the use of the four codes defined in SOAP 1.1.

1.6.2 Communication styles

SOAP supports two different communication styles:

- | | |
|-----------------|---|
| Document | Also known as <i>message-oriented</i> style: This is a very flexible communication style that provides the best interoperability. The message body is any legal XML as defined in the types section of the WSDL document. |
| RPC | The <i>remote procedure call</i> is a synchronous invocation of an operation which returns a result; it is conceptually similar to other RPCs. |

1.6.3 Encodings

In distributed computing environments, *encodings* define how data values defined in the application can be translated to and from a protocol format. We refer to these translation steps as *serialization* and *deserialization*, or, synonymously, *marshalling* and *unmarshalling*.

When implementing a Web service, we have to choose one of the tools and programming or scripting languages that support the Web services model. However, the protocol format for Web services is XML, which is independent of the programming language. Thus, SOAP encodings tell the SOAP runtime environment how to translate from data structures constructed in a specific programming language into SOAP XML and vice versa.

The following encodings are defined:

- | | |
|----------------------|--|
| SOAP encoding | The <i>SOAP encoding</i> enables marshalling/unmarshalling of values of data types from the SOAP data model. This encoding is defined in the SOAP 1.1 standard. |
| Literal | The <i>literal</i> encoding is a simple XML message that does not carry encoding information. Usually, an XML Schema describes the format and data types of the XML message. |

1.6.4 Messaging modes

The two styles (RPC and document) and the two common encodings (SOAP encoding and literal) can be freely intermixed to produce what is called a SOAP messaging mode. Although SOAP supports four modes, only three of the four modes are generally used, and further, only two are preferred by the WS-I Basic Profile:

- ▶ **Document/literal:** Provides the best interoperability between language environments. The WS-I Basic Profile states that all Web service interactions should use the Document/literal mode.
- ▶ **RPC/literal:** Possible choice between certain implementations. Although RPC/literal is WS-I compliant, it is not frequently used in practice. There are a number of usability issues associated with RPC/literal.
- ▶ **RPC/encoded:** Early Java implementations supported this combination, but it does not provide interoperability with other implementations and is not recommended
- ▶ **Document/encoded:** Not used in practice.

You can find the SOAP 1.1 specification at the following URL:

<http://www.w3.org/TR/2000/NOTE-SOAP-20000508>

The SOAP 1.2 specification is provided at the following URL:

<http://www.w3.org/TR/SOAP12>

1.7 WSDL

This section introduces Web Services Description Language (WSDL) 1.1. WSDL uses XML to specify the characteristics of a Web service: what the Web service can do, where it resides, and how it is invoked. WSDL can be extended to allow descriptions of different bindings, regardless of what message formats or network protocols are used to communicate.

WSDL enables a service provider to specify the following characteristics of a Web service:

- ▶ Name of the Web service and addressing information
- ▶ Protocol and encoding style to be used when accessing the public operations of the Web service
- ▶ Type information: Operations, parameters, and data types comprising the interface of the Web service, plus a name for this interface

1.7.1 WSDL document

A WSDL document contains the following main elements:

Types	A container for data type definitions using some type system, usually XML Schema.
Message	An abstract, typed definition of the data being communicated. A message can have one or more typed parts.
Port type	An abstract set of one or more operations supported by one or more ports.
Operation	An abstract description of an action supported by the service that defines the input and output message and optional fault message.
Binding	A concrete protocol and data format specification for a particular port type. The binding information contains the protocol name, the invocation style, a service ID, and the encoding for each operation.
Port	A single endpoint, which is defined as an aggregation of a binding and a network address.
Service	A collection of related ports.

Note that WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats and supports the XML Schema Definition (XSD) specification.

WSDL 1.1 introduces specific binding extensions for various protocols and message formats. There is a WSDL SOAP binding, which is capable of describing SOAP over HTTP. It is worth noting that WSDL does not define any mappings to a programming language; rather, the bindings deal with transport protocols. This is a major difference from interface description languages, such as the CORBA Interface Definition Language (IDL), which has language bindings.

You can find the WSDL 1.1 specification at the following URL:

<http://www.w3.org/TR/wsdl>

1.7.2 WSDL document anatomy

Figure 1-2 on page 21 shows the elements comprising a WSDL document and the various relationships between them.

The diagram should be interpreted in the following way:

- ▶ One WSDL document contains zero or more services. A service contains zero or more port definitions (service endpoints), and a port definition contains a specific protocol extension.
- ▶ The same WSDL document contains zero or more bindings. A binding is referenced by zero or more ports. The binding contains one protocol extension, where the style and transport are defined, and zero or more operations bindings. Each of these operation bindings is composed of one protocol extension, where the action and style are defined, and one to three messages bindings, where the encoding is defined.
- ▶ The same WSDL document contains zero or more port types. A port type is referenced by zero or more bindings. This port type contains zero or more operations, which are referenced by zero or more operations bindings.
- ▶ The same WSDL document contains zero or more messages. An operation usually points to an input and an output message, and optionally to some faults. A message is composed of zero or more parts.
- ▶ The same WSDL document contains zero or more types. A type can be referenced by zero or more parts.
- ▶ The same WSDL document points to zero or more XML Schemas. An XML Schema contains zero or more XSD types that define the different data types.

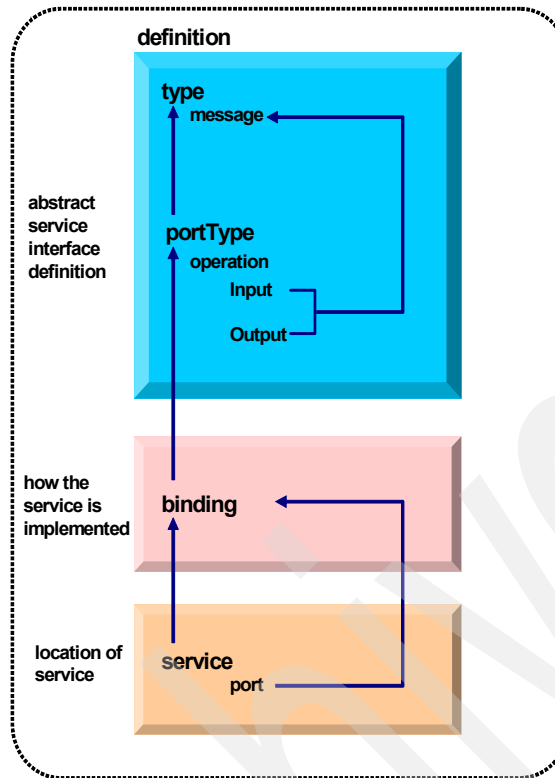


Figure 1-2 WSDL elements and relationships

WSDL example

We now give an example of a simple, complete, and valid WSDL file. As this example shows, even a simple WSDL document contains quite a few elements with various relationships to each other. Example 1-1 contains the WSDL file, which we analyze in detail later in this section.

Example 1-1 Complete WSDL document

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:reqns="http://www.exampleApp.dispatchOrder.Request.com"
  xmlns:resns="http://www.exampleApp.dispatchOrder.Response.com"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.exampleApp.dispatchOrder.com"
  targetNamespace="http://www.exampleApp.dispatchOrder.com">
  <types>
    <xsd:schema xmlns:tns="http://www.exampleApp.dispatchOrder.Request.com"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      attributeFormDefault="qualified"
```

```

        elementFormDefault="qualified"
        targetNamespace="http://www.exampleApp.dispatchOrder.Request.com"
        xmlns:reqns="http://www.exampleApp.dispatchOrder.Request.com">
<xsd:element name="dispatchOrderRequest" nillable="false">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element name="itemReferenceNumber" nillable="false">
        <xsd:simpleType>
          <xsd:restriction base="xsd:short">
            <xsd:maxInclusive value="9999"/>
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="quantityRequired" nillable="false">
        <xsd:simpleType>
          <xsd:restriction base="xsd:short">
            <xsd:maxInclusive value="999"/>
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
<xsd:schema xmlns:tns="http://www.exampleApp.dispatchOrder.Response.com"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  attributeFormDefault="qualified"
  elementFormDefault="qualified"
  targetNamespace="http://www.exampleApp.dispatchOrder.Response.com">
<xsd:element name="dispatchOrderResponse" nillable="false">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element name="confirmation" nillable="false">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="20"/>
            <xsd:whiteSpace value="preserve"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
</types>
<message name="dispatchOrderResponse">
  <part element="resns:dispatchOrderResponse" name="ResponsePart"/>

```



```

</message>
<message name="dispatchOrderRequest">
  <part element="reqns:dispatchOrderRequest" name="RequestPart"/>
</message>
<portType name="dispatchOrderPort">
  <operation name="dispatchOrder">
    <input message="tns:dispatchOrderRequest" name="DFH0XODSRequest"/>
    <output message="tns:dispatchOrderResponse" name="DFH0XODSResponse"/>
  </operation>
</portType>
<binding name="dispatchOrderSoapBinding" type="tns:dispatchOrderPort">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="dispatchOrder">
    <soap:operation soapAction="" style="document"/>
    <input name="DFH0XODSRequest">
      <soap:body parts="RequestPart" use="literal"/>
    </input>
    <output name="DFH0XODSResponse">
      <soap:body parts="ResponsePart" use="literal"/>
    </output>
  </operation>
</binding>
<service name="dispatchOrderService">
  <port binding="tns:dispatchOrderSoapBinding" name="dispatchOrderPort">
    <soap:address
      location="http://myserver:54321/exampleApp/services/dispatchOrderPort"/>
  </port>
</service>
</definitions>

```

Namespaces

WSDL uses the XML namespaces listed in Table 1-2.

Table 1-2 WSDL namespaces

Namespace URI	Explanation
http://schemas.xmlsoap.org/wsdl/	Namespace for WSDL framework.
http://schemas.xmlsoap.org/wsdl/soap/	SOAP binding.
http://schemas.xmlsoap.org/wsdl/http/	HTTP binding.
http://www.w3.org/2000/10/XMLSchema	Schema namespace as defined by XSD.

Namespace URI	Explanation
(URL to WSDL file)	The <i>this namespace</i> (tns) prefix is used as a convention to refer to the current document. Do not confuse it with the XSD <i>target namespace</i> , which is a different concept.

The first three namespaces are defined by the WSDL specification itself; the next definition references a namespace that is defined in the SOAP and XSD standards. The last one is local to each specification.

1.7.3 WSDL definition

The WSDL definition contains types, messages, operations, port types, bindings, ports, and services.

Also, WSDL provides an optional element called `wsdl:document` as a container for human-readable documentation.

Types

The *types* element encloses data type definitions used by the exchanged messages. WSDL uses XML Schema Definition (XSD) as its canonical and built-in type system:

```
<definitions .... >
  <types>
    <xsd:schema .... /> (0 or more)
  </types>
</definitions>
```

The XSD type system can be used to define the types in a message regardless of whether or not the resulting wire format is XML. In our example we have two schema sections; one defines the message format for the input and the other defines the message format for the output.

In our example, the types definition, shown in Example 1-2, is where we specify that there is a complex type called `dispatchOrderRequest`, which is composed of two elements: a `itemReferenceNumber` and a `quantityRequired`.

Example 1-2 Types definition of our WSDL example for the input

```
<types>
  <xsd:schema xmlns:tns="http://www.exampleApp.dispatchOrder.Request.com"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    attributeFormDefault="qualified"
```

```

        elementFormDefault="qualified"
        targetNamespace="http://www.exampleApp.dispatchOrder.Request.com"
        xmlns:reqns="http://www.exampleApp.dispatchOrder.Request.com">
<xsd:element name="dispatchOrderRequest" nillable="false">
  <xsd:complexType mixed="false">
    <xsd:sequence>
      <xsd:element name="itemReferenceNumber" nillable="false">
        <xsd:simpleType>
          <xsd:restriction base="xsd:short">
            <xsd:maxInclusive value="9999"/>
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="quantityRequired" nillable="false">
        <xsd:simpleType>
          <xsd:restriction base="xsd:short">
            <xsd:maxInclusive value="999"/>
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
.
.
</types>

```

Messages

A *message* represents one interaction between a service requester and a service provider. If an operation is bidirectional at least two message definitions are used in order to specify the transmissions to and from the service provider. A message consists of one or more logical parts.

```

<definitions .... >
  <message name="nmtoken"> (0 or more)
    <part name="nmtoken" element="qname" (0 or 1) type="qname" (0 or
1)/>
    (0 or more)
  </message>
</definitions>

```

The abstract message definitions are used by the operation element. Multiple operations can refer to the same message definition.

Operations and messages are modeled separately in order to support flexibility and simplify reuse of existing definitions. For example, two operations with the same parameters can share one abstract message definition.

In our example, the messages definition, shown in Example 1-3, is where we specify the different parts that compose each message. The request message `dispatchOrderRequest` is composed of an element `dispatchOrderRequest` as defined in the schema in the parts section. The response message `dispatchOrderResponse` is similarly defined by the element `dispatchOrderResponse` in the schema. There is no requirement for the names of the message and the schema-defined element to match; in our example we did this merely for convenience.

Example 1-3 Message definition in our WSDL document

```
<message name="dispatchOrderResponse">
  <part element="resns:dispatchOrderResponse" name="ResponsePart"/>
</message>
<message name="dispatchOrderRequest">
  <part element="reqns:dispatchOrderRequest" name="RequestPart"/>
</message>
```

Port types

A *port type* is a named set of abstract operations and the abstract messages involved:

```
<definitions .... >
  <portType name="nmtoken">
    <operation name="nmtoken" .... /> (0 or more)
  </portType>
</definitions>
```

WSDL defines four types of operations that a port can support:

- | | |
|-------------------------|---|
| One-way | The port receives a message. There is an <i>input</i> message only. |
| Request-response | The port receives a message and sends a correlated message. There is an <i>input</i> message followed by an <i>output</i> message. |
| Solicit-response | The port sends a message and receives a correlated message. There is an <i>output</i> message followed by an <i>input</i> message. |
| Notification | The port sends a message. There is an <i>output</i> message only. This type of operation could be used in a publish/subscribe scenario. |

Each of these operation types can be supported with variations of the following syntax. Presence and order of the input, output, and fault messages determine the type of message:

```
<definitions .... >
  <portType .... > (0 or more)
    <operation name="nmtoken" parameterOrder="nmtokens">
      <input name="nmtoken" (0 or 1) message="qname"/> (0 or 1)
      <output name="nmtoken" (0 or 1) message="qname"/> (0 or 1)
      <fault name="nmtoken" message="qname"/> (0 or more)
    </operation>
  </portType >
</definitions>
```

Note that a request-response operation is an abstract notion. A particular binding must be consulted to determine how the messages are actually sent:

- ▶ Within a single transport-level operation, such as an HTTP request/response message pair, which is the preferred option
- ▶ As two independent transport-level operations, which can be required if the transport protocol only supports one-way communication

In our example, the portType and operation definitions, shown in Example 1-4, are where we specify the port type, called `dispatchOrderPort`, and a set of operations. In this case, there is only one operation, called `dispatchOrder`.

We also specify the interface that the Web service provides to its possible clients, with the input message `DFH0X0DSRequest` and the output message `DFH0X0DSResponse`. Because the input element appears before the output element in the operation element, our example shows a request-response type of operation.

Example 1-4 Port type and operation definitions in our WSDL document example

```
<portType name="dispatchOrderPort">
  <operation name="dispatchOrder">
    <input message="tns:dispatchOrderRequest" name="DFH0X0DSRequest"/>
    <output message="tns:dispatchOrderResponse" name="DFH0X0DSResponse"/>
  </operation>
</portType>
```

Bindings

A *binding* contains:

- ▶ Protocol-specific general binding data, such as the underlying transport protocol and the communication style for SOAP.
- ▶ Protocol extensions for operations and their messages.

Each binding references one port type; one port type can be used in multiple bindings. All operations defined within the port type must be bound in the binding. The pseudo XSD for the binding looks like this:

```
<definitions .... >
  <binding name="nmtoken" type="qname"> (0 or more)
    <!-- extensibility element (1) --> (0 or more)
    <operation name="nmtoken"> (0 or more)
      <!-- extensibility element (2) --> (0 or more)
      <input name="nmtoken" (0 or 1) > (0 or 1)
        <!-- extensibility element (3) -->
      </input>
      <output name="nmtoken" (0 or 1) > (0 or 1)
        <!-- extensibility element (4) --> (0 or more)
      </output>
      <fault name="nmtoken"> (0 or more)
        <!-- extensibility element (5) --> (0 or more)
      </fault>
    </operation>
  </binding>
</definitions>
```

As we have already seen, a port references a binding. The port and binding are modeled as separate entities in order to support flexibility and location transparency. Two ports that merely differ in their network address can share the same protocol binding.

The extensibility elements `<!-- extensibility element (x) -->` use XML namespaces in order to incorporate protocol-specific information into the language- and protocol-independent WSDL specification.

In our example, the binding definition, shown in Example 1-5, is where we specify our binding name, `dispatchOrderSoapBinding`. The connection must be SOAP HTTP, and the style must be document. We provide a reference to our operation, `dispatchOrder`, and we define the input message `DFH0X0DSRequest` and the output message `DFH0X0DSResponse`, both to be SOAP literal.

Example 1-5 Binding definition in our WSDL document example

```
<binding name="dispatchOrderSoapBinding" type="tns:dispatchOrderPort">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="dispatchOrder">
    <soap:operation soapAction="" style="document"/>
    <input name="DFH0X0DSRequest">
      <soap:body parts="RequestPart" use="literal"/>
    </input>
```

```

        <output name="DFH0X0DSResponse">
            <soap:body parts="ResponsePart" use="literal"/>
        </output>
    </operation>
</binding>

```

Service definition

A *service* definition merely bundles a set of ports together under a name, as the following pseudo XSD definition of the service element shows.

```

<definitions .... >
    <service name="nmtoken"> (0 or more)
        <port .... /> (0 or more)
    </service>
</definitions>

```

Multiple service definitions can appear in a single WSDL document.

Port definition

A *port* definition describes an individual endpoint by specifying a single address for a binding:

```

<definitions .... >
    <service .... > (0 or more)
        <port name="nmtoken" binding="qname"> (0 or more)
            <!-- extensibility element (1) -->
        </port>
    </service>
</definitions>

```

The binding attribute is of type QName, which is a qualified name (equivalent to the one used in SOAP). It refers to a binding. A port contains exactly one network address; all other protocol-specific information is contained in the binding.

Any port in the implementation part must reference exactly one binding in the interface part.

The <!-- extensibility element (1) --> is a placeholder for additional XML elements that can hold protocol-specific information. This mechanism is required because WSDL is designed to support multiple runtime protocols.

In our example, the service and port definition, shown in Example 1-6, is where we specify our service, called `dispatchOrderService`, which contains a collection of our ports. In this case, there is only one that uses the `dispatchOrderSoapBinding` and is called `dispatchOrderPort`. In this port, we specify our connection point as, for example, `http://myserver:54321/exampleApp/services/dispatchOrderPort`.

Example 1-6 Service and port definition in our WSDL document example

```
<service name="dispatchOrderService">
  <port binding="tns:dispatchOrderSoapBinding" name="dispatchOrderPort">
    <soap:address
      location="http://myserver:54321/exampleApp/services/dispatchOrderPort"/>
    </port>
  </service>
```

1.7.4 WSDL bindings

We now investigate the WSDL extensibility elements supporting the SOAP transport binding.

SOAP binding

WSDL includes a binding for SOAP 1.1 endpoints, which supports the specification of the following protocol-specific information:

- ▶ An indication that a binding is bound to the SOAP 1.1 protocol
- ▶ A way of specifying an address for a SOAP endpoint
- ▶ The URI for the `S0APAction` HTTP header for the HTTP binding of SOAP
- ▶ A list of definitions for headers that are transmitted as part of the SOAP envelope

Table 1-3 lists the corresponding extension elements.

Table 1-3 SOAP extensibility elements in WSDL

Extension and attributes		Explanation
<soap:binding ...>		Binding level; specifies defaults for all operations.
	transport="uri" (0 or 1)	Binding level; transport is the runtime transport protocol used by SOAP (HTTP, SMTP, and so on).
	style="rpc document" (0 or 1)	The style is one of the two SOAP communication styles, rpc or document.
<soap:operation ... >		Extends operation definition.

Extension and attributes		Explanation
	soapAction="uri" (0 or 1)	URN.
	style="rpc document" (0 or 1)	See binding level.
<soap:body ... >		Extends operation definition; specifies how message parts appear inside the SOAP body.
	parts="nmtokens"	Optional; allows externalizing message parts.
	use="encoded literal"	literal: messages reference concrete XSD (no WSDL type); encoded: messages reference abstract WSDL type elements; encodingStyle extension used.
	encodingStyle="uri-list"(0 or 1)	List of supported message encoding styles.
	namespace="uri" (0 or 1)	URN of the service.
<soap:fault ... >		Extends operation definition; contents of fault details element.
	name="nmtoken"	Relates soap:fault to wsd1:fault for operation.
	use, encodingStyle, namespace	See soap:body.
<soap:address ... >		Extends port definition.
	location="uri"	Network address of RPC router.
<soap:header ... >		Operation level; shaped after <soap:body ...>.
<soap:headerfault ... >		Operation level; shaped after <soap:body ...>.

1.8 Summary

We began by discussing service-oriented architectures and how Web services relate to SOAs. We continued by giving an overview of the major technologies that make Web services possible: XML, SOAP, WSDL, and UDDI. Then, we looked in detail at SOAP, which provides an XML text-based, platform-neutral and language-neutral message format. Finally, we explained how WSDL defines the application data to be conveyed in the SOAP message as well as the information required to access the service, such as the transport protocol used and the location of the service.

1.9 Support for Web services in CICS TS V3

Application programs running in CICS TS V3 can participate in a heterogeneous Web services environment as service requesters, service providers, or both, using either an HTTP transport or an WMQ transport.

1.9.1 What is provided in CICS TS V3.1

CICS TS V3.1 provides the following capabilities:

- ▶ It includes a Web services assistant utility.

The Web services assistant utility contains two programs: DFHWS2LS and DFHLS2WS:

- DFHWS2LS helps you map an existing WSDL document into a high level programming language data structure.
- DFHLS2WS creates a new WSDL document from an existing language structure.

The Web services assistant supports the following programming languages:

- COBOL
- PL/I
- C
- C++

- ▶ It supports two different approaches to deploying your CICS applications in a Web services environment.

- You can use the Web services assistant.

The Web services assistant helps you deploy an application with the least amount of programming effort. For example, if you want to expose an existing application as a Web service, you can start with a high-level language data structure and use DFHLS2WS to generate the Web services description. Alternatively, if you want to communicate with an existing Web service, you can start with its Web service description and use DFHWS2LS to generate a high level language structure that you can use in your program.

Both DFHLS2WS and DFHWS2LS also generate a file called the `wsbind` file. When your application runs, CICS uses the `wsbind` file to transform your application data into a SOAP message on output and to transform the SOAP message to application data on input.

- You can take complete control of the processing of your data.

You can write your own code to map between your application data and the message that flows between the service requester and provider. For example, if you want to use non-SOAP messages within the Web service infrastructure, you can write your own code to transform between the message format and the format used by your application.

- It reads a pipeline configuration file created by the CICS system programmer to determine which message handlers should be invoked in a pipeline.

Note: A pipeline can be configured as either a service requester pipeline or a service provider pipeline but not both.

Whether you use the Web services assistant or take complete control of the processing yourself, you can write your own message handlers to perform additional processing on your request and response messages. You can also use CICS-supplied message handlers.

- It supplies message handlers designed especially to help you process SOAP messages.

The pipelines that CICS uses to process Web service requests and responses are generic, in that there are few restrictions on what processing can be performed in each message handler. However, many Web service applications use SOAP messages, and any processing of those messages should comply with the SOAP specification. Therefore, CICS provides special SOAP message handler programs that can help you to configure your pipeline as a SOAP node.

- A service requester pipeline is the initial SOAP sender for the request, and the ultimate SOAP receiver for the response.
- A service provider pipeline is the ultimate SOAP receiver for the request, and the initial SOAP sender for the response.

You cannot configure a CICS pipeline to function as an intermediary node in a SOAP message path.

The CICS-provided SOAP message handlers can be configured to invoke one or more user-written header processing programs and to enforce the presence of particular headers in the SOAP message.

- It allows you to configure many different pipelines.

You can configure a pipeline to support SOAP 1.1 or SOAP 1.2. Within your CICS system, you can have some pipelines that support SOAP 1.1 and others that support SOAP 1.2.

- ▶ It provides the following resource definitions to help you configure your support for Web services:
 - PIPELINE
 - URIMAP
 - WEBSERVICE

If you used the SOAP for CICS feature, you might be able to use CICS resource definitions to replace the logic you provided in your pipeline programs to distinguish one application from another. For example, in a service provider, you might be able to replace code that distinguishes between applications based upon a URI, with a suitable set of URIMAP resources.
- ▶ It provides the following EXEC CICS application programming interface (API) commands:
 - SOAPFAULT ADD | CREATE | DELETE
 - INQUIRE WEBSERVICE
 - INVOKE WEBSERVICE
- ▶ It conforms to open standards, including:
 - SOAP 1.1 and 1.2
 - HTTP 1.1
 - WSDL 1.1
- ▶ It ensures maximum interoperability with other Web services implementations by conforming to the Web Services Interoperability Organization (WS-I) Basic Profile 1.0.
- ▶ It supports the WS-Atomic Transaction and WS-Security specifications.

1.9.2 What is new in CICS TS V3.2

CICS TS V3.2 provides the following new functions:

- ▶ It supports the WSDL 2.0 specification in addition to WSDL 1.1. CICS support for WSDL 2.0, however, is subject to some restrictions that are documented in the *CICS Web Services Guide*, SC34-6838.

The Web services assistant utilities, DFHWS2LS and DFHLS2WS, have been enhanced to enable creation of a Web service description that complies with WSDL 2.0. You can create both WSDL 1.1 and WSDL 2.0 documents during the same run of the assistant utility.

The batch jobs have new and modified parameters that provide you with more flexibility:

- You can specify an absolute URI for your Web service to DFHLS2WS.

- DFHWS2LS automatically determines the WSDL version of the Web service description that has been supplied as input.
- You can select a specific wsdl:Service element within the Web service description.
- You can specify a subset of wsdl:Operation elements that you want to invoke.

These enhancements are useful when you have a large WSDL file with many wsdl:Service and wsdl:Operation elements in it.

- ▶ In accordance with the WSDL 2.0 specification, CICS now supports four out of the eight Message Exchange Patterns (MEPs). These patterns describe typical interactions between requester and provider. The patterns are:
 - In-Only:
A request message is sent to the Web service provider, but the provider is not allowed to send any type of response to the Web service requester.
 - In-Out:
A request message is sent to the Web service provider, and a response message is returned. The response message can be a normal SOAP message or a SOAP fault.
 - In-Optional-Out:
A request message is sent to the Web service provider, and a response message is optionally returned to the requester. The response message can be a normal SOAP message, or a SOAP fault.
 - Robust In-Only:
A request message is sent to the Web service provider, and no response message is returned to the requester unless an error occurs. In this case, a SOAP fault message is sent to the requester.
- ▶ When CICS is acting as a service requester, that is, if your program issues an EXEC CICS INVOKE WEBSERVICE command, you can now define how long CICS should wait for a reply. The PIPELINE resource has a new RESPWAIT attribute that determines how many seconds CICS should wait. If you do not set a value for this attribute, either the default timeout for the transport protocol or the dispatcher timeout for the transaction is used.
The default timeout for HTTP is 10 seconds; the default timeout for WebSphere MQ is 60 seconds. If the value for the dispatcher timeout for the transaction is less than the default for either protocol, the timeout occurs with the dispatcher.
- ▶ New context containers have been provided in the Web service provider and requester pipelines to support WSDL 2.0.

- ▶ CICS TS V3.2 supports the SOAP Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specification commonly referred to as MTOM/XOP. These specifications define a method for optimizing the transmission of base64Binary data within SOAP messages.

CICS implements this support in both requester and provider pipelines when the transport protocol is HTTP or HTTPS. You can configure MTOM/XOP support by using additional options in the pipeline configuration file. With this support, base64Binary data is sent as a binary attachment and not directly in the SOAP message.

- ▶ CICS support for Web services has been extended to comply with the WSDL 1.1 Binding Extension for SOAP 1.2 specification. This specification defines the binding extensions that are required to indicate that Web service messages are bound to the SOAP 1.2 protocol. The goal is to provide functionality that is comparable with the binding for SOAP 1.1.
- ▶ The support that CICS provides for securing Web services has been enhanced to include an implementation of the Web Services Trust Language (WS-Trust) specification. CICS TS V3.2 can interoperate with a Security Token Service (STS), such as Tivoli Federated Identity Manager, to validate and issue security tokens in Web services.

This enables CICS to send and receive messages that contain a wide variety of security tokens, such as SAML assertions and Kerberos tokens, to interoperate securely with other Web services. CICS support for WS-Trust specification is subject to some restrictions, as shown in the *CICS Web Services Guide*, SC34-6838.

Web Services performance considerations

In this chapter, we look at how the following topics can impact the performance of a Web services solution:

- ▶ Transport considerations
- ▶ Payload size
- ▶ XML parsing
- ▶ Using MTOM/XOP for binary data
- ▶ Security choices
- ▶ System z10 server improvements

2.1 Transport considerations

The connector transport used to expose CICS applications in an SOA can have significant impact on the performance of the required solution.

2.1.1 CICS connector choices

CICS Web services is one of several CICS connector technologies that can be used to provide access to CICS applications. The other choices include the following three principal solutions:

- ▶ J2EE connector architecture, via CICS Transaction Gateway (CTG)
- ▶ HTTP, via CICS Web support
- ▶ Native messaging, via WebSphere MQ (WMQ)

The relative merits of each of these methods are discussed in detail in the IBM Redbooks publication, *Architecting Access to CICS within an SOA*, SG24-5466. In particular, a detailed comparison of the performance based on CICS TS V3.1 is outlined in Chapter 7, “Performance and scalability”. A recent performance study for CICS TS 3.2 analyzing CPU costs for these different connector solutions is shown in Figure 2-1. Figures represent basic transport costs for a simple 100 byte message, and do not include any additional cost for message parsing, security, transaction control or other qualities of service.

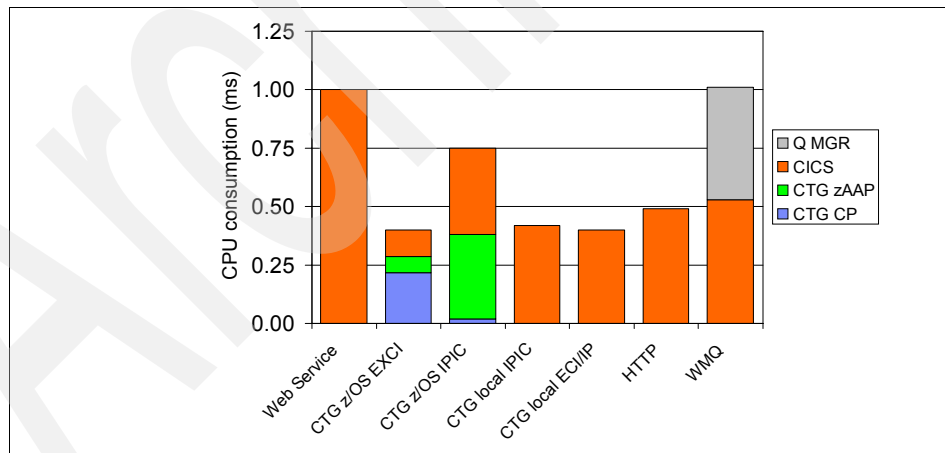


Figure 2-1 CICS connectors - CPU consumption

- ▶ “Web service” is a Web service call to invoke a CICS program.

- ▶ *CTG z/OS EXCI* is a configuration in which an application uses a remote connection to a CTG daemon, and then the CTG daemon uses the EXCI to invoke a CICS program.
- ▶ *CTG z/OS IPIC* is a configuration in which an application uses a remote connection to a CTG daemon, and then the CTG daemon uses the IPIC (IP connectivity) protocol to invoke a CICS program.
- ▶ *CTG local IPIC* is a configuration in which an application uses a direct CTG IPIC connection to invoke a CICS program.
- ▶ *CTG local ECI/IP* is a configuration in which an application uses an ECI over IP connection to invoke a CICS program.
- ▶ *HTTP* is a configuration in which an application uses the CICS Web support to send an HTTP request to a CICS program.
- ▶ *WMQ* is an application that uses an MQ message to invoke a CICS program.

These figures illustrate how the CPU usage per sub-system varies depending on the underlying connector technology. With CTG and HTTP options providing lower CPU usage, and CTG providing additional options for the off-load of CPU costs to the zAAP processors depending on the topology in use. Note that the Web service message used was a single element message and so did not include significant XML parsing costs. For further details on XML parsing refer to 2.2.2, “XML complexity” on page 41.

2.1.2 Web service transports

Web services requests into CICS can use SOAP over HTTP or MQ:

- ▶ When HTTP is used as the transport, persistent TCP/IP connections can be configured by setting the SOCKETCLOSE parameter in the relevant TCPIP SERVICE definition. Persistent connections outperform non-persistent connections, because the Web service requester is able to reuse the underlying socket connection on subsequent invocations of a service on the same CICS system. If the connection is not persistent, then the client will have to establish a new connection for every invocation of a service provider application. The use of persistent connections is particularly important when SSL/TLS is used to secure access to the Web service.
- ▶ When MQ is used as the transport, the performance of non-persistent messages is significantly better than that of persistent messages. Persistent messages are written to the queue manager log (for recovery purposes), whereas non-persistent messages are not written to the log. This also has an impact on the queue manager's restart time.

2.2 XML parsing and payload size

The resources, such as CPU and storage, that are consumed by CICS in processing Web service requests are principally affected by the efficiency of the XML parsing and the amount of data transmitted. In the following section we discuss the key considerations for when analyzing the performance of a Web services solution.

2.2.1 Payload size

The size of the SOAP messages is directly related to the amount of CPU and storage consumed by CICS in processing the Web service. Figure 2-2 shows how the size of both an inbound and outbound SOAP message affects the number of milliseconds of CPU consumed to process a Web service. The SOAP message consists of a single element to reduce any impact of XML parsing.

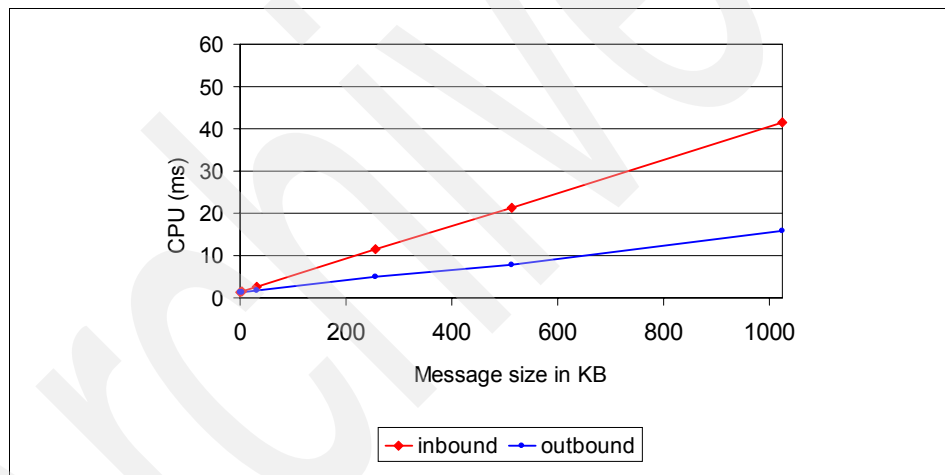


Figure 2-2 CPU usage based on message size

For a given SOAP message, the process of generating the outbound message consumes less CPU than the process of parsing an inbound message of the same size. Thus the inbound message size has a more significant impact on performance than the outbound message size. The two key ways in which message size can be reduced are as follows:

- ▶ Designing less complex XML schemas
- ▶ Using MTOM/XOP if large amounts of binary data are used

We discuss these topics in the following section.

2.2.2 XML complexity

XML parsing is a CPU intensive activity within the Web services pipeline. The XML associated with the SOAP envelope, SOAP headers, and SOAP body for inbound SOAP messages is parsed to extract the data elements that are to be passed to the application. Structuring the XML to reduce the complexity of the elements is likely to be one of the best methods to improve the scalability of Web service applications in CICS. Note that as with payload size, the complexity of the *inbound* SOAP message is of prime importance because XML parsing is only performed for the inbound message, and not on the outbound message.

The key factors affecting the XML parsing costs are:

1. The number of XML elements
2. The size of each element
3. The size of the element tags

If using a bottom-up approach by importing an existing copybook (such as with using the DFHLS2WS tooling), the first approach should be to perform an analysis of the XML structure. If it is possible to perform simple alteration on the existing copybook this may provide the performance gains required. The key items to investigate are:

- ▶ Removal of trailing spaces from data elements
- ▶ Reducing the size of the element tag names
- ▶ Reducing the general complexity of the language structure

The most important consideration, however, is the number of data elements. As an approximate guide, the CPU overhead within CICS to process multiple XML elements within a SOAP message is as follows:

- ▶ 0.5 ms of CPU per 100 inbound elements
- ▶ 0.1 ms of CPU per 100 outbound elements

Simply put, reducing the number of XML elements, especially for inbound messages, is key to improving performance. Carefully consider this issue when creating the Web services definition (WSDL).

Meet in the middle

If an existing copybook interface cannot be modified, then a *meet in the middle* approach using a wrapper program can provide a mechanism to simplify the XML structure and re-use existing CICS applications without modification. Using this approach, DFHWS2LS is used to create the wsbind file and a language structure for a wrapper program. The wrapper program acts as the service provider application and, based on the SOAP request message received, it creates a COMMAREA (or container) for the target business logic program.

In addition, usage of a wrapper means that the CICS wrapper program can be deployed in a different CICS region to the existing business logic programs. This makes it possible to service enable a CICS program running in a back level CICS region, and it is also consistent with the principal of configuring CICS regions with different roles within a CICSplex. For more details on workload management for CICS Web services, refer to the IBM Redbooks publication, *CICS Web Services Workload Management and Availability*, SG24-7144.

2.2.3 Using MTOM/XOP for binary data

It can often be necessary to transmit binary data, such as graphical images, along with the character data primarily used as the input to CICS applications. Before CICS TS V3.2, it was necessary to use the base64 encoding mechanism to transfer such binary data within SOAP messages. However, CICS TS V3.2 supports the SOAP Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specifications, commonly referred to as MTOM/XOP. These specifications define a method for optimizing the transmission of base64 encoded binary data within SOAP messages.

Benefits of MTOM/XOP

When using MTOM/XOP to transfer binary data, the overall payload size of data transmitted across the network is reduced. This occurs because MTOM/XOP allows binary data to be transmitted in its native binary form, rather than being encoded in the base64 encoding.

Base64 encoding uses a well known encoding mechanism to transform each 3 bytes of binary data to 4 bytes of encoded data, thus causing additional CPU usage due to both the overhead of encoding the data and the increase in payload size. However, when using MTOM/XOP, there is an additional payload overhead of the Multipurpose Internet Mail Extensions (MIME) headers, which also needs to be factored into the total message size.

The overhead negates some of the benefit of MTOM/XOP and means that the benefit will only be seen for messages over a given size. This is the reason that CICS will only use MTOM/XOP for messages over 1,500 bytes in size. However, in our tests that we ran using MTOM/XOP (see Chapter 9, “MTOM scenario”) we found that MTOM/XOP usage only demonstrated significant benefits for binary attachments over 5 KB.

The graph in Figure 2-3 shows the CPU consumed in processing a Web service containing a single binary element with CICS TS V3.1, CICS TS V3.2, and CICS TS V3.2 with MTOM/XOP.

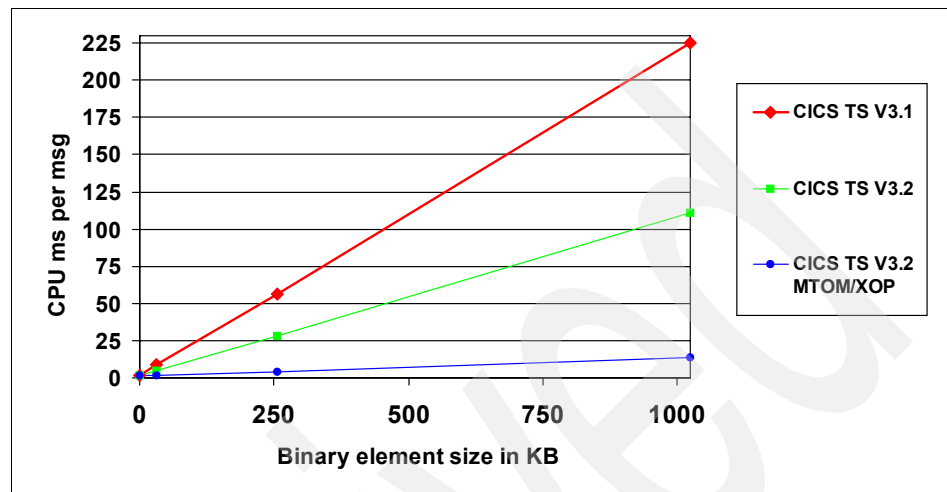


Figure 2-3 CPU usage of CICS Web services with binary data

The reduction in CPU usage when using MTOM/XOP is due to a combination of factors:

- ▶ There is no longer a requirement to convert binary data to and from base64.
- ▶ There is a reduction in the CPU consumed within the XML parser on inbound data, because the binary data is not parsed.
- ▶ There is an overall reduction in message size.

MTOM/XOP with WS-Security

When a component within the Web services pipeline does not support XOP, such as WS-Security, MTOM/XOP compatibility mode is used. For example, if both WS-Security and MTOM/XOP are configured within a CICS service provider pipeline, the inbound message must be converted from a MIME document format into standard XML format. This requires conversion of the binary data into base64 binary. Before the binary data is presented to the target application, it must be converted back from base64 binary to binary. Equivalent processing occurs for outbound processing. From a performance perspective, therefore, MTOM/XOP with WS-Security is not recommended.

MTOM/XOP best practice guidelines

To efficiently control usage of MTOM/XOP requests, you should create a separate pipeline configuration for the Web services using binary data. Within this file, the `<mtom_options>` element then specifies when to use MTOM for SOAP messages. Review the following guidance to enable efficient usage of MTOM/XOP. For details on the pipeline configuration file used in our tests, refer to Example 8-2 on page 183.

Service provider mode

Here we explain the various settings for service provider mode:

- ▶ If the binary data is only inbound to CICS, specify **no** for the **send_mtom** attribute of the `<mtom_options>` element, to indicate that MTOM is not to be used for outbound messages.
- ▶ Specify **yes** if the binary data is only outbound from CICS, to indicate that MTOM/XOP is to be used for all outbound messages.
- ▶ For inbound and outbound binary data, specify **same**, to indicate that MTOM is to be used for outbound messages only if the inbound message is in MTOM format.
- ▶ Specify **no** for the **send_when_no_xop** attribute of the `<mtom_options>` element, to indicate that MTOM is only to be used for the outbound message if binary attachments are present in the message.

Service requester mode

Here we explain the various settings for service requester mode:

- ▶ Specify **yes** for the **send_mtom** attribute of the `<mtom_options>` element.
- ▶ Specify **no** for the **send_when_no_xop** attribute of the `<mtom_options>` element.

2.3 Security and performance

Generally speaking, there exists a trade-off in terms of security and performance. The more secure a solution is, the more processing will be required to perform the desired security functions. The following topics summarize how SSL/TLS, WS-Security, and DataPower solutions provide authentication and confidentiality and the relative performance merits of each option. For further details about the implementation of each option, refer to the IBM Redbooks publication, *Securing CICS Web services*, SG24-7658.

2.3.1 Transport level security

Transport level security (TLS) or its predecessor Secure Sockets Layer (SSL) is a flexible security infrastructure that provides a transport based security solution between two TCP/IP endpoints. It provides both confidentiality and the option to authenticate the client and server identities. In terms of performance, TLS performs well because the connection can be re-used after the digital key exchange process has occurred, and the cryptographic operations for the key exchange and payload encryption can be off-loaded to the System z cryptographic hardware.

When using SSL/TLS, consider the following techniques for optimizing the performance of the solution:

- ▶ Utilizing cryptographic hardware.
- ▶ Increasing the value of the CICS system initialization table parameter SSLDELAY so the session IDs remain in the SSLCACHE longer, which will result in only partial SSL handshakes.
- ▶ Increasing the value of the CICS system initialization table parameter MAXSSLTCBS so there are more S8 TCBs in the SSL pool for the SSL handshake negotiation.
- ▶ Using the CICS SSLCACHE system initialization table parameter to implement SSL caching across a sysplex if Web service requests are being routed across a set of CICS regions. However, if you are using a single CICS region, then you should specify SSLCACHE(CICS) as opposed to SSLCACHE(SYSPLEX) in order to avoid the additional cost of making the SSL session ID shareable.
- ▶ Keeping the socket open by coding SOCKETCLOSE NO on the TCPIP SERVICE definition. This is the default for HTTP 1.1 persistent sessions and removes the need to perform an SSL handshake on the second or subsequent HTTP request.
- ▶ Only using client authentication by specifying SSL(CLIENTAUTH) on the TCPIP SERVICE definition when you really need your clients to identify themselves with a client certificate. Client authentication requires more network transmissions during the SSL handshake, and more processing by CICS to handle the received certificate.

SSL/TLS is a good choice if no intermediaries are used in the Web service environment or, if there are intermediaries, when you can guarantee that after the data is decrypted, it cannot be accessed by an untrusted node or process.

Note, however, that TLS does not provide end-to-end message level security, and so if non-trusted nodes exist between the requester and provider, WS-Security is a more appropriate solution.

2.3.2 WS-Security

Web service security (WS-Security) provides a set of SOAP message extensions for building secure Web services by defining new elements to be used in the SOAP header for message-level security. The specification allows you to protect the body of the message or any XML elements within the body or the header. You can give different levels of protection to different elements within the SOAP message. The advantage of using WS-Security over SSL/TLS is that it can provide end-to-end message-level security. This means that the message can be secured even if it passes through multiple services, called intermediaries.

CICS TS V3.2 provides in-built support for WS-Security, and provides a flexible set of options to provide the following security functions:

- ▶ Authentication, using X.509 or basic authentication options
- ▶ Signing of SOAP messages, either inbound or outbound
- ▶ Encrypting SOAP messages, either inbound or outbound

Because WS-Security support is a completely stateless protocol, the encrypted channel is not persisted between Web service requests. This means that expensive security functions, such as the XML digital signature validation, need to be repeated for each service request. The result of this is that security implementations using WS-Security are generally more CPU intensive than transport based security implementations.

2.3.3 WS-Security with DataPower

IBM WebSphere DataPower appliances are purpose-built, network devices that can be used to simplify and accelerate XML and Web services deployments. Using DataPower with WS-Security provides options to improve the performance of WS-Security with CICS by either verifying the X.509 digital signature or mapping the digital signature to a UsernameToken (such as a RACF user ID). To see results of our tests comparing these options, with WS-Security digital signature validation within CICS or within DataPower, refer to Chapter 10, “Security scenarios”.

2.3.4 Security summary

Figure 2-4 summarizes the CPU usage for three different WS-Security solutions. This information was obtained from the CICS performance team at IBM Hursley. The WS-Security scenarios used security for both inbound and outbound messages, the digital signature algorithm used was RSA-SHA1, and the encryption algorithm used was AES-256.

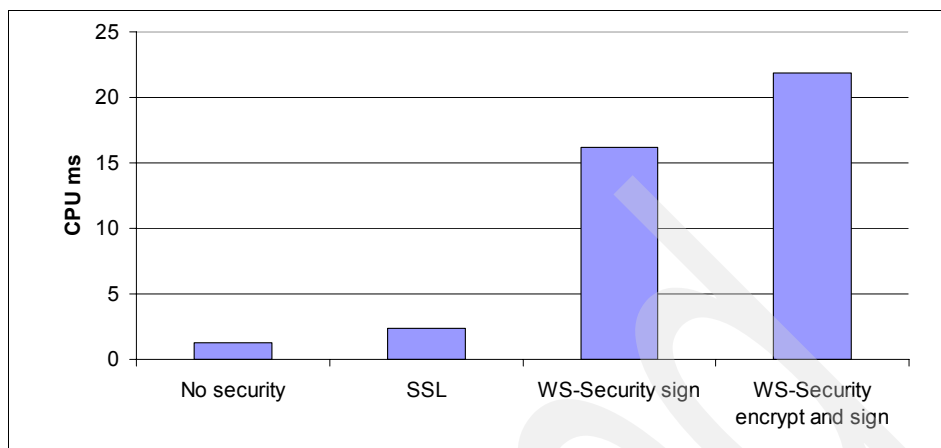


Figure 2-4 Relative performance of WS-Security solutions

The graph shows the total CPU consumed in processing an incoming Web service request in a CICS TS V3.2 provider system with the following options:

No security	Web service request without security
SSL	Web service request encrypted using SSL, (without client authentication).
WS-Security sign	Web service request with an XML digital signature (this is equivalent to our <i>CICS XML digital signature</i> scenario described in 8.5, “Security scenarios overview” on page 187.
WS-Security encrypt and sign	Web service request with an XML digital signature and XML encryption.

The data in Figure 2-5 on page 50 highlights the additional cost of WS-Security processing that is incurred within CICS for both XML signed and encrypted SOAP messages. Note that SSL also offers options for authenticating the client identity, and this will incur additional processing costs within CICS.

In summary, the three options each offer the following characteristics:

► **SSL/TLS:**

SSL/TLS is a mature technology that provides robust encryption and authentication solutions at the transport level. There are ways of optimizing performance, such as persistent TCP/IP connections and SSL session ID reuse, and these optimizations mean that expensive security functions, such as SSL handshaking, can be avoided for service requests following the initial handshake.

► **WS-Security:**

WS-Security works across multiple transports and nodes, and is independent of the underlying protocol. It offers a high level of security with a rich set of options. However, expensive security functions, such as XML digital signature validation, are repeated for each service request, and so the CPU costs will be highest when using WS-Security within CICS.

► **WS-Security with DataPower:**

A DataPower appliance can be used in conjunction with CICS Web services WS-Security support to off-load expensive operations by processing the XML digital signature and performing mappings to a predefined RACF user ID. This provides a more efficient WS-Security solution, but relies on additional hardware and the ability to place the DataPower appliance within a trusted network zone.

To see our results for the tests using WS-Security and DataPower scenarios, refer to Chapter 10, “Security scenarios”.

2.4 CICS TS V3.2 performance improvements

Next we show some examples of the performance improvements in CICS TS V3.2, compared to CICS TS V3.1, when using Web services requests. These figures were obtained from the performance team at IBM Hursley, using a dummy target application.

2.4.1 HTTP enhancements

CICS TS V3.2 offers a range of performance improvements in the HTTP transport infrastructure. The main benefits are as follows:

Optimized HTTP processing

The efficiency of HTTP message processing has been considerably improved for inbound and outbound messages.

The improvements in inbound HTTP processing are as follows:

- 12% less CPU for a 4 KB message
- 31% less CPU for a 32 KB message

The improvements in outbound HTTP processing are as follows:

- 20% less CPU for a 4 KB message
- 46% less CPU for a 32 KB message

These enhancements are automatic and apply to all HTTP communication using CICS Web support, including inbound and outbound Web service requests.

2.4.2 WS-Security improvements

The following WS-Security performance improvements are introduced in CICS TS V3.2.

UsernameToken basic authentication

The efficiency of basic authentication of a WS-Security UsernameToken has been significantly improved in CICS TS V3.2. This processing now avoids the overhead of initializing the DFHWSSE1 message handler. Tests have shown a greater than 80% reduction in CPU consumption when configuring the CICS WS-Security handler to authenticate a UsernameToken (user ID and password).

X.509 digital signature verification

The efficiency of WS-Security XML digital signature verification has been improved. Tests have demonstrated an 18% reduction in CPU consumption for inbound/outbound digital signature verification.

WS-Security encryption

The efficiency of WS-Security XML encryption has been improved. Tests have demonstrated a 17% reduction in CPU for bidirectional message encryption/decryption.

2.4.3 Code page conversion

Web service request and response messages are normally sent encoded with a UTF-8 encoding. To process UTF-8 data within CICS applications typically requires code page conversion to and from EBCDIC. Performance improvements for code page conversion have been made in CICS TS V3.2. The CPU used to perform code page conversion between UTF-8 and EBCDIC is reduced if the UTF-8 data consists of the 7-bit ASCII characters, with no national characters.

2.4.4 64-bit containers

CICS TS V3.2 introduced support for 64-bit containers, and in doing so, provided optimized container storage management. Performance improvements over CICS TS V3.1 in terms of reduced CPU are noticeable when using containers, and this is especially noticeable for larger messages with tests demonstrating around 30% less CPU required for managing a 1 MB container.

2.5 System z10 hardware

Running the same CICS Web services workload on an IBM System z10 machine as compared to an earlier System z machine can have a significant effect on the performance and throughput of the solution.

Figure 2-5 shows the results from running a CICS Web service provider application, on a System z9 and a System z10. The CPU usage is in CPU ms per Web service, and the SOAP message length is 100 KB.

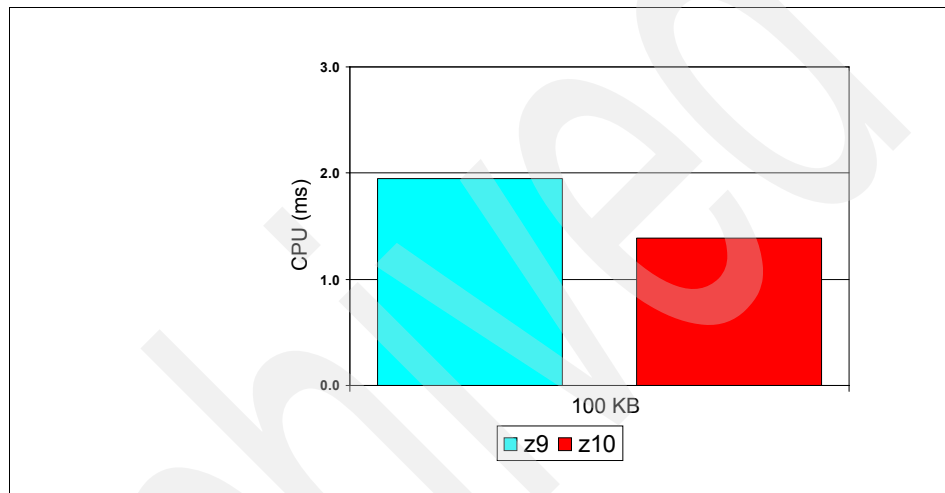


Figure 2-5 CPU usage for a Web services workload on System z9 and System z10

As can be seen, moving from a z9 to a z10 processor has reduced the CPU consumed per CICS Web service by approximately 30%.

In 9.2.3, “Effects of switching workload to System z10” on page 222 we present our own performance data, which illustrates the benefit of processing Web services requests that contain binary data on a System z10 compared to a System z9. Our figures illustrate a major improvement (approximately 45%) when using binary data without MTOM/XOP and a smaller benefit when using MTOM/XOP. The larger benefit when not using MTOM/XOP is attributed to the increased speed of the z10 that accelerates conversion between base64 and binary data.

A stylized graphic of a globe with a circular cutout showing the Americas, set against a background of concentric circles.

Part 2

CICS Tools overview

In this part of the book, we focus on the tools available to help you understand your system's performance. These tools include:

- ▶ RMF
- ▶ CICS Performance Analyzer
- ▶ IBM Tivoli Monitoring
- ▶ OMEGAMON XE for CICS on z/OS
- ▶ ITCAM for SOA

RMF

In this chapter, we describe how to measure CICS Web Services performance, using the z/OS Resource Measurement Facility (RMF).

Measuring CICS performance with other tools is discussed in the next few chapters. In this chapter we cover the following topics:

- ▶ RMF overview
- ▶ Measuring CICS Web Services performance using RMF

For detailed information about using RMF with zSeries®, refer to *Effective zSeries Performance Monitoring using Resource Management Facility*, SG24-6645.

3.1 RMF overview

RMF is an optional feature of z/OS, and is shipped with every release of z/OS at the current level of support. It is integration tested with z/OS and includes the enhancements available with every new release.

RMF is the IBM strategic product for z/OS performance measurement and management. It is the base product to collect performance and capacity planning data for z/OS and sysplex environments to monitor systems' performance behavior. RMF allows you to optimally tune and configure your system according to your business needs.

RMF collects system-wide data that describes the processor activity, I/O activity, main storage activity, and system resources manager (SRM) activity. Analysis of the data collected by RMF can then be made by running a variety of RMF reports, or observed dynamically using ISPF panels.

RMF measures the following activities:

- ▶ Processor usage
- ▶ Address space usage
- ▶ Channel activity
- ▶ Device activity
- ▶ Detailed system paging
- ▶ Detailed system workload
- ▶ Page and swap data set
- ▶ Enqueues
- ▶ Coupling Facility (CF) activity
- ▶ Cross Coupling Facility (XCF) activity

RMF allows the z/OS user to:

- ▶ Evaluate system responsiveness
- ▶ Check the effects of tuning
- ▶ Perform capacity planning evaluation

3.2 Using RMF

RMF is normally active in the system 24 hours a day. A report is generated at the time interval specified by the installation. An interval of 60 minutes is recommended for normal operation. When you are addressing a specific problem, reduce the time interval to 10 or 15 minutes.

The RMF records can be directed to the SMF data sets with the NOREPORT and RECORD options; the report overhead is not incurred and the SMF records can be formatted later. In terms of CPU costs, this is an inexpensive way to collect performance information.

3.3 Using RMF with CICS

The CICS monitoring facility (CMF) is a component of CICS that collects statistical and monitoring data. CMF can be used in conjunction with RMF to provide day-to-day monitoring of CICS transaction rates and response times data.

CICS usage of RMF transaction reporting

The objective of using the CMF with RMF is to enable transaction rates and internal response times to be monitored without incurring the overhead of running the full CICS monitoring facility and associated reporting.

This approach can be useful when only transaction statistics are required, rather than the very detailed information that CICS monitoring facility produces. An example of this is the monitoring of a production system where the minimum overhead is required.

CICS monitoring facility and the use of SYSEVENT

The CICS monitoring facility issues an MVS workload manager IWMRPT or IWMMNTFY macro that gives the following SYSEVENT information to the MVS workload manager (WLM):

- ▶ The time at which the user-task was attached
- ▶ Subsystem identification (MNSUBSYS SIT value if specified, otherwise derived from the first four characters of the CICS generic APPLID)
- ▶ Transaction identifier of the task
- ▶ User identifier
- ▶ APPLID of the CICS region

3.4 Defining reporting classes using WLM

Statistics in the RMF reports are grouped by reporting class. Therefore, to be able to extract meaningful information from RMF reports, it is necessary to define these reporting classes to the workload manager. This can be performed from ISPF panels.

Next, we show an example of how to do this.

1. First, in Figure 3-1, we invoke WLM from within ISPF (TSO WLM).

```
File  Help
-----
Command ==>

      W  W  L      M  M
      W  W  L     MM MM
      W W W  L     M M M
      WW WW  L     M  M
      W  W  LLLL  M  M

Licensed Materials - Property of IBM

5647-A01 (C) Copyright IBM Corp. 2006.
All rights reserved.

      ENTER to continue

F1=Help   F2=Split   F3=Exit   F9=Swap   F10=Menu Bar F12=Cancel
```

Figure 3-1 WLM main panel

2. Press Enter to continue. An example of the Choose Service Definition window appears.
3. Select Option 2 **Extract definition from WLM couple data sets** and press Enter.

4. Select Option 6 to manage **Classification Rules** (Figure 3-2).

```
File Utilities Notes Options Help
-----
Functionality LEVEL011          Definition Menu          WLM Appl LEVEL019
Command ==> _____

Definition data set . . . : none

Definition name . . . . . WPS (Required)
Description . . . . . WLM for WAS/Sysplex Perf monitor

Select one of the
following options. . . . . 6__ 1. Policies
                                2. Workloads
                                3. Resource Groups
                                4. Service Classes
                                5. Classification Groups
                                6. Classification Rules
                                7. Report Classes
                                8. Service Coefficients/Options
                                9. Application Environments
                                10. Scheduling Environments

F1=Help      F2=Split      F3=Exit      F9=Swap      F10=Menu Bar F12=Cancel
```

Figure 3-2 WLM main options panel

5. Select function required (for example, Option 3 to **Modify**) for the Subsystem Type (for example, CICS). See Figure 3-3.

Subsystem-Type View Notes Options Help					

Subsystem Type Selection List for Rules Row 1 to 12 of 15					
Command ==> _____					
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, /=Menu Bar					
-----Class-----					
Action	Type	Description	Service	Report	
—	ASCH	Use Modify to enter YOUR rules		OTHER	
—	CB	WebSphere App Server	WASDF	OTHER	
3_	CICS	CICS SERVER	CICSDFLT	OTHER	
—	DB2	Use Modify to enter YOUR rules	SCDB2STP		
—	DDF	DRDA Stored Procedures	SAPHIGH		
—	IMS	Use Modify to enter YOUR rules	IMS	IMS	
—	IWEB	Webserver Subsystem Type	WEBSRVC	OTHER	
—	JES	JES TYPE	VEL50	OTHER	
—	LSFM	Use Modify to enter YOUR rules		OTHER	
—	MQ	Workflow Request Classification	DEF_SC		
—	OMVS		VEL70	OTHER	
—	SAP	WLM definition for SAP R/3 4.5B	SAPAS	SAP	
F1=Help	F2=Split	F3=Exit	F4=Return	F7=Up	F8=Down
F9=Swap	F10=Menu Bar	F12=Cancel			

Figure 3-3 Classification Rules

6. The Service and Reporting classes required are defined on this panel (Figure 3-4).

Subsystem-Type	Xref	Notes	Options	Help

Modify Rules for the Subsystem Type			Row 4 to 12 of 12	
Command ==> _____			SCROLL ==> CSR	
Subsystem Type . : CICS Fold qualifier names? Y (Y or N)				
Description . . . CICS SERVER				
Action codes: A=After C=Copy M=Move I=Insert rule				
B=Before D=Delete row R=Repeat IS=Insert Sub-rule				
More ==>				
-----Qualifier-----				
Action	Type	Name	Start	Class
Service Report				
DEFAULTS: CICSDFLT C3C3				
___ 1	SI	A6POC3C3	___	CICSDFLT C3C3
___ 2	TN	CWXN	___	CICSDFLT C3C3CWXN
___ 2	TN	CPIH	___	CICSDFLT C3C3CPIH
___ 1	SI	A6POC3C2	___	CICSDFLT C3C2
___ 2	TN	CWXN	___	CICSDFLT C3C2CWXN
___ 2	TN	CPIH	___	CICSDFLT C3C2CPIH
___ 1	SI	A6POC3C1	___	CICSDFLT C3C1
___ 2	TN	CPIH	___	CICSDFLT C3C1CPIH
___ 2	TN	CWXN	___	CICSDFLT C3C1CWXN
***** BOTTOM OF DATA *****				
F1=Help F2=Split F3=Exit F4=Return F7=Up F8=Down F9=Swap				
F10=Left F11=Right F12=Cancel				

Figure 3-4 Define CICS APPLID and transaction id reporting classes

In this example, a Service Class of CICSDFLT with a Reporting Class of C3C3 has been defined for CICS region A6POC3C3. The Workload Activity Report, which we cover later, reports performance figures by Reporting Class. A6POC3C3 is the value specified on the **APPLID** parameter in the CICS SIT.

Using the **IS** Action code (Insert Sub-rule), transactions CWXN and CPIH have been placed in their own Reporting Classes. The Classification Rules also have to be updated for the CICS application as follows:

If CICS is started by submitting a batch job, the Classification Rules of the JES subsystem type needs to be updated.

If CICS is started by a started task, the Classification Rules of the STC subsystem type needs to be updated.

An example of this is shown in Figure 3-5 for the STC subsystem type.

Subsystem-Type Xref Notes Options Help

Modify Rules for the Subsystem Type Row 4 to 13 of 100

Command ==> _____ SCROLL ==> CSR

Subsystem Type . : STC Fold qualifier names? Y (Y or N)

Description . . . _____

Action codes: A=After C=Copy M=Move I=Insert rule

B=Before D=Delete row R=Repeat IS=Insert Sub-rule

More ==>

Action	-----Qualifier-----			-----Class-----	
	Type	Name	Start	Service	Report
_____	1	TN	A6POC3C3	_____	_____
_____	1	TN	I*CONN	_____	_____
_____	1	TN	RRS	_____	_____
_____	1	TN	D8%MSTR	_____	_____
_____	1	TN	D8%DBM1	_____	_____
_____	1	TN	*IRLM	_____	_____
_____	1	TN	WS6481	_____	_____
_____	1	TN	WS6482	_____	_____
_____	1	TN	WS6481A	_____	_____
_____	1	TN	WS6482A	_____	_____

DEFAULTS: STC_HIGH CICS_RGN A6POC3C3

STC_HIGH IMSCONN

SYSSTC RRS

STC_HIGH DB2_SYS

STC_HIGH DB2_SYS

SYSSTC DB2_SYS

_____ WS648

_____ WS648

STC WS648SRV

STC WS648SRV

F1=Help F2=Split F3=Exit F4=Return F7=Up F8=Down F9=Swap

F10=Left F11=Right F12=Cancel

Figure 3-5 Define CICS started task reporting class

In this example, a new entry is inserted for the CICS system A6POC3C3 with a Reporting Class of A6POC3C3.

A6POC3C3 is the name of the CICS *started task*.

Before activating these new definitions, they first need to be installed. This is achieved by returning to the *Definition Menu* panel and selecting **1. Install definition** from the *Utilities* drop-down menu.

When the definitions are installed, the service policy can be activated. This is achieved by selecting **3. Activate service policy** from the *Utilities* drop-down menu.

3.5 RMF monitors

RMF measures and reports system activity, mostly using a sampling technique to collect data.

In the following sections we describe the three monitors that RMF can use.

Monitor I

Monitor I measures and reports the use of system resources such as:

- ▶ Processors
- ▶ I/O devices
- ▶ Storage
- ▶ Datasets

Monitor I runs in the background and measures data over a period of time. Reports can be printed immediately after the end of the measurement interval, or the data can be stored in SMF records and printed later with the RMF postprocessor. PARMLIB member ERBRMFxx defines the options that are used on the RMF Monitor I background session. The default member name is ERBRMF00.

Monitor II

Monitor II, like Monitor I, measures and reports the use of system resources.

Monitor II runs in the background under TSO or on a console. It provides “snapshot” reports about resource usage, and also can have the data stored in SMF records. PARMLIB member ERBRMFxx defines the options that are used on the RMF Monitor II background session. The default member name is ERBRMF01.

Monitor III

Monitor III primarily measures the contention for system resources and the delay of jobs that such contention causes. It collects and reports the data in real time using ISPF panels, with optional printed copy backup of individual displays.

Monitor III must be used if XCF or CF reports are required. PARMLIB member ERBRMFxx defines the options that are used on the RMF Monitor III background session. The default member name is ERBRMF04 for Monitor III.

3.6 Starting RMF

By default, the Monitor I session is started at the time RMF itself is started.

RMF is started along with the Monitor I session by the command:

```
S RMF, , , (MEMBER(xx))
```

Where 'xx' indicates the ERBRMF member name suffix containing the Monitor I parameters.

The RMF Monitor II session is started by the command:

```
F RMF, START aa, MEMBER(xx)
```

Where 'aa' indicates any 2 alphabetic characters except ZZ and III, and 'xx' indicates the ERBRMF member name suffix, which contains the Monitor II parameters.

The RMF Monitor III session is started by the command:

```
F RMF, START III, MEMBER(xx)
```

Where 'xx' indicates the ERBRMF member name suffix, which contains the Monitor III parameters.

This starts the Monitor III procedure RMFGAT.

3.7 Sample RMF definitions

Example 3-1 shows an RMF startup procedure.

Example 3-1 RMF startup procedure

```
//IEFPROC EXEC PGM=ERBMFMFC,REGION=0M,DPRTY=(10,10),PERFORM=74
//IEFPARM DD DDNAME=IEFRDER
//IEFRDER DD DSN=USER.PARMLIB,DISP=SHR
//MFR00001 DD DSN=CICS.MVS2A.RMF1,DISP=SHR
//MFR00002 DD DSN=CICS.MVS2A.RMF2,DISP=SHR
//RMFAB001 DD DSN=CICS.MVS2A.MONII,DISP=SHR
```

An example of the Monitor I PARMLIB parameters in member ERBRMFxx is shown in Example 3-2 on page 63.

Note that in general all the PARMLIB member parameters for RMF are not specifically related to CICS.

Example 3-2 Sample monitor I PARMLIB member

```
CPU           /* COLLECT CPU STATISTICS      */
CHAN          /* COLLECT CHANNEL STATISTICS     */
CYCLE(200)     /* SAMPLE AT 5 TIMES / SECOND              */
DEVICE(NOCHDR) /* NO CHARACTER RDR DEV STATS              */
DEVICE(COMM)   /* COMM DEVICE STATS - MAY BE
GOOD DEBUG AID */
DEVICE(DASD)/* COLLECT DIRECT ACCESS DEVICE STATISTICS */
DEVICE(NOGRAPH)/* NO GRAPHICS DEVICE STATISTICS */
DEVICE(NOTAPE)/* NO TAPE DEVICE STATISTICS */
DEVICE(NOUNITR)/* NO UNIT RECORD DEVICE STATS */
DEVICE(NMBR(2A2A:2A2B))/* CTC CONNECTIONS */
NOENQ /* NO ENQ REPORTING */
NOEXITS/* DO NOT TAKE RMF EXITS */
INTERVAL(60M)/* REPORT AT 60 MIN INTERVALS */
NOIOQ /* I/O Q'ING FOR DEV IN LOG CU*KR */
NOOPTIONS/* OPERATOR MAY NOT CHG RMF OPTIONS */
PAGING/* COLLECT PAGING STATISTICS */
PAGESE/* COLLECT PAGE/SWAP DATASET STAT */
REPORT(REALTIME)/* PRINT REPORTS AT END OF SESSION */
RECORD/* RECORD INTO SMF DATASET */
STOP(600M)/* STOP AFTER 600 MINUTES **KR */
/* SET UP TO RUN 120 WITH */
/* INTERVAL STARTED BY SET ICS */
NOSYNC/* INTERVAL NOT SYNCED WITH HOUR */
SYSOUT(A)/* SYSOUT CLASS OF OUTPUT REPORT */
WKLD(PERIOD,SYSTEM)/* COLLECT WORKLOAD MANAGER STATISTICS
AND REPORT AT THE PERIOD LEVEL + TOTAL LINE */
```

A coding sample of the Monitor II PARMLIB parameters in member ERBRMFxx is shown in Figure 3-6.

```
ASD           /* COLLECT ADDRESS SPACE STATE DATA      */
NOUSER/* DO NOT COLLECT USER DATA */
NODELTA/* PRESENT DATA AS SESSION TOTALS */
SINTV (30S)/* SESSION INTERVAL = 30 SECONDS */
STOP (30M)/* STOP AFTER 30 MINUTES */
RECORD/* SMF RECORDING */
REPORT (DEFER)/* ALL REPORTS TO BE PRODUCED AFTER RMF ENDS
*/
OPTIONS/* OPERATOR MAY EXAMINE AND/OR CHANGE THE RMF OPTIONS */
SYSOUT(A)/* INTERVAL REPORTS TO CLASS A */
```

Figure 3-6 Sample Monitor II PARMLIB member

Example 3-3 shows the PARMLIB parameters for a Monitor III session.

Example 3-3 Sample Monitor III PARMLIB member

```
NOOPTIONS /* STARTS IMMEDIATELY */
          CYCLE(2000)/* CYCLE TIME 1000 MS */
          MINTIME(60)/* SECONDS PER REPORT */
          SYNC(0)/* SYNC(0) OKAY */
          SYSOUT(A)/* SYSOUT CLASS */
          RESOURCE(*JES2)/* WE ARE RUNNING JES2 */
          HFSNAME(ADD(WAS3.WAS302.HFS),
          ADD(OMVS.MVS25.ARG0.LIB2),
          ADD(OMVS.MVS25.ARG0.LIB1),
          ADD(OMVS.MVS25.ARG0),
          ADD(OMVS.MVS25.PERFTASK),
          NOSTOP/* DO NOT STOP */
```

3.8 Workload Activity Report

The Workload Activity Report is a typical RMF report used to report on CICS activity.

Example 3-4 shows JCL to create the Workload Activity Report using SMF dataset SYS1.MV2A.MANA as input.

Example 3-4 JCL to create Workload Activity Report

```
//RMF01 EXEC PGM=IFASMFDP
//DUMPIN DD DSN=SYS1.MV2A.MANA,DISP=SHR
//DUMPOUT DD DSN=&&SMF,UNIT=SYSDA,
// DISP=(NEW,PASS),SPACE=(CYL,(50,50))
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
          INDD(DUMPIN,OPTIONS(DUMP))
          OUTDD(DUMPOUT,TYPE(000:255))
//RMF02 EXEC PGM=ERBRMFPP,REGION=0M
//MFPINPUT DD DSN=&&SMF,DISP=(OLD,PASS)
//SYSUDUMP DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//MFPMMSGDS DD SYSOUT=A
//PPXSRPTS DD SYSOUT=A
//SYSIN DD *
          NOSUMMARY
          SYSOUT(A)
```

```
REPORTS (INTERVAL)
SYSRPTS (WLMGL (RCLASS, SCPER))
/*
//
```

3.9 Measuring CICS Web Services performance using RMF

The RMF reports produce statistics on a wide range of machine resources.

When measuring CICS Web Services performance, normally the metrics we are interested in are:

- ▶ CPU consumed to process a Web Service
- ▶ Web Service response time
- ▶ Web Service throughput

In general, the standard method for measuring CICS Web Services performance is as follows:

1. Create or capture a script that invokes a CICS Web Service.
2. Import the script into a workload simulator product, for example, Workload Simulator for z/OS or Rational® Performance Tester for z/OS.
3. Start the workload simulator to run the required script. Consideration needs to be given to the number of simulated clients and the “thinktime”. The thinktime determines how frequently each simulated client runs the script.
4. Ensure that no tasks other than those invoked by the workload simulator script are running on the CICS system.
5. Wait until all clients are connected to CICS and are running the script. It is important that a steady workload is running before collecting CICS CPU and transaction statistics.
6. Use RMF to capture the CPU and transaction statistics. This can be achieved by stopping and starting RMF to start a new reporting interval. Leave the workload running for a few minutes, then stop and start RMF again to end the reporting interval.
7. Run the RMF Workload Activity Report, and extract statistics for that reporting interval.
8. Adjust the number of clients and/or the thinktime, if necessary, so that CICS is being neither under nor over utilized in terms of CPU usage. Between 20% and 50% of each processor would be a reasonable aim. Also ensure that no CICS limits are being encountered, such as short-on-storage or maxtasks.

Sample Workload Activity Report output

Figure 3-7 shows some sample output from the Workload Activity Report.

REPORT BY: POLICY=CICSWORK				REPORT CLASS=A6POC3C3			
				DESCRIPTION =			
I/O--	---SERVICE----	--SERVICE TIMES--	---APPL %---	-----STORAGE-----			
2.9	IOC 5529	CPU 232.464	CP 92.12	AVG	127509.73		
0.8	CPU 6241K	SRB 0.297	AAPCP 0.00	TOTAL	127436.05		
0.6	MSO 0	RCT 0.000	IIPCP 0.00	SHARED	3.00		
0.0	SRB 7965	IIT 0.003					
0.2	TOT 6254K	HST 0.000	AAP N/A	--PAGE-IN RATES--			
0.0	/SEC 24751	AAP N/A	IIP N/A	SINGLE	0.0		
		IIP N/A		BLOCK	0.0		
	ABSRPTN 25K			SHARED	0.0		
	TRX SERV 25K	PROMOTED 0.000		HSP	0.0		

Figure 3-7 Workload Activity Report showing CICS CPU usage

The APPL% CP value of 92.12 for Reporting Class A6POC3C3 indicates that the CICS system A6POC3C3 was using an average of 92.12% CPU during this reporting interval.

Note that this figure might exceed 100% in a multi-processor environment. For example if the LPAR was running with 3 processors and CICS was consuming 70% of each of the processors, the APPL% CP value would be 210%.

Figure 3-8 is a Workload Activity Report showing CPIH transaction usage.

REPORT BY: POLICY=CICSWORK			REPORT CLASS=C3C3CPIH		
			DESCRIPTION =Reporting Class for C3C3 CPIH		
-TRANSACTIONS-			TRANS-TIME HHH.MM.SS.TTT		
AVG	0.00	ACTUAL	848		
MPL	0.00	EXECUTION	0		
ENDED	39730	QUEUED	0		
END/S	157.22	R/S AFFIN	0		
#SWAPS	0	INELIGIBLE	0		
EXCTD	0	CONVERSION	0		
AVG ENC	0.00	STD DEV	237		
REM ENC	0.00				
MS ENC	0.00				

Figure 3-8 Workload Activity report showing CPIH transaction usage

From our WLM definitions, Reporting Class C3C3CPIH is associated with CPIH transaction.

The END/S value of 157.22 indicates that during this reporting interval, on average, 157.22 CPIH tasks ended per second.

Figure 3-9 is a Workload Activity Report showing CWXN transaction usage.

REPORT BY: POLICY=CICSWORK		REPORT CLASS=C3C3CWXN	
		DESCRIPTION =Reporting Class for C3C3 CWXN	
-TRANSACTIONS-	TRANS-TIME	HHH.MM.SS.TTT	
AVG	0.00	ACTUAL	723
MPL	0.00	EXECUTION	0
ENDED	39750	QUEUED	0
END/S	157.23	R/S AFFIN	0
#SWAPS	0	INELIGIBLE	0
EXCTD	0	CONVERSION	0
AVG ENC	0.00	STD DEV	243
REM ENC	0.00		
MS ENC	0.00		

Figure 3-9 Workload activity report showing CWXN transaction usage

From our WLM definitions, Reporting Class C3C3CWXN is associated with CWXN transaction.

The END/S value of 157.23 indicates that during this five minute interval, on average, 157.23 CWXN tasks ended per second.

As each SOAP message is composed of a CWXN and CPIH task, we would expect the END/S value for the C3C3CPIH and C3C3CWXN Reporting Classes to be about the same.

Measuring the CPU consumed per Web Service

The following process enables us to calculate the CPU consumed per SOAP message.

The Workload Activity Report displays the number of tasks ended per second (END/S) for the reporting class we are interested in.

In a CICS requester system, there is a one-to-one correlation between tasks ending and SOAP messages processed within the time period being measured.

In a CICS provider system, there are normally two tasks (CWXN and CPIH when using HTTP transport) to process a Web Service request. However, for example, if context switching is used, a third task is used to run the back-end application. Note also that instead of using CPIH as the transaction ID, another ID can be substituted.

If you are unsure how many tasks run in a CICS provider system to process a single SOAP message, CICS statistics can be used to determine this, or even a CICS trace.

The APPL% CP value for the whole CICS region, obtained from the Workload Activity Report indicates the average percentage CPU consumed within the time period being measured.

This value is the average CPU consumed by all tasks running within that CICS system. Hence for a pure Web Services workload, to arrive at a measurement for milliseconds of CPU consumed per SOAP message, we do the following calculations.

If c is the APPL% CP value, the workload consumes $c/100$ seconds of CPU in a second. In milliseconds that would be $1000*c/100$, which resolves to $10*c$ milliseconds of CPU per second.

This is the amount of milliseconds of CPU consumed per second for all tasks that make up this workload. If t is the END/S value (the number of tasks that end per second), then $10*c/t$ gives us the average milliseconds of CPU per task. If n is the number of tasks per SOAP message (CWXN, CPIH, and so on), then the milliseconds of CPU consumed per SOAP message is:

$$n*10*c/t$$

So, for the previous example, the CPU consumed per Web Service calculation is:
 $2*10*92.12/(157.22+157.23) = 5.86$ milliseconds

For a CICS requester system, this milliseconds of CPU value includes:

- ▶ User task processing including issuing the EXEC CICS INVOKE WEBSERVICE
- ▶ Outbound and inbound SOAP pipeline request and response handling

For a CICS provider system, this milliseconds of CPU value includes:

- ▶ Inbound and outbound SOAP pipeline request and response handling
- ▶ Back-end business logic processing

When processing relatively large SOAP messages, the amount of processing performed by TCP/IP or WebSphere MQ outside of a CICS TCB can become significant. Therefore the APPL% CP value for the TCP/IP or WebSphere MQ reporting classes needs to be factored into the calculations.

Note that WebSphere MQ might itself be using TCP/IP in transmitting and receiving its messages.

Measuring the Web Service response time

The Workload Activity Report displays the internal response time in the “ACTUAL” field in the “TRANS.-TIME HHH.MM.SS.TTT” column for the reporting class we are interested in. Note that this time value is in milliseconds.

For a CICS requester system, this is the internal response time of the user task, and not just the response time within the SOAP pipeline.

For a CICS provider system, if all CICS tasks are in a single Reporting Class, this is the average internal response time for the tasks used to process the Web Service request.

To provide greater granularity for the CICS provider system response time, place tasks such as CWXN, CPIH and CPIQ in unique reporting classes.

When using HTTP, as the CWXN task terminates immediately after attaching the CPIH task, adding the CWXN and CPIH internal response times can give an approximate Web Service internal response time.

Note that response times can be affected by other work running on the machine, contending for machine resources.

Note also that response times measured by the client can be significantly higher than these internal response times due to network transmission overheads.

Measuring Web Services throughput

Web Services throughput in terms of the number of Web Services processed per second is obtained from the END/S field in the Workload Activity Report.

Note that throughput can be affected by a number of factors, including:

- ▶ Other work running on the machine where CICS is running
- ▶ Number of clients
- ▶ Client thinktime
- ▶ Network efficiency between clients and CICS

CICS PA

In this chapter, we introduce IBM CICS Performance Analyzer for z/OS (CICS PA) and describe how you can use it to analyze the performance of CICS Web Services. For scenarios that use CICS PA, see Chapter 10, “Security scenarios” on page 229.

4.1 Overview

CICS PA is a reporting tool that provides information on the performance of your CICS systems and applications to help you tune, manage, and plan your CICS systems effectively.

CICS PA is not an online monitoring tool. It produces reports and extracts using data collected by your system in MVS System Management Facility (SMF) data sets:

- ▶ CICS Monitoring Facility (CMF) performance class, exception class, and transaction resource class data in SMF 110 records
- ▶ CICS Transaction Server statistics data in SMF 110 records
- ▶ CICS Transaction Gateway statistics data in SMF 111 records
- ▶ System Logger data in SMF 88 records
- ▶ DB2 accounting data in SMF 101 records
- ▶ WebSphere MQ accounting data in SMF 116 records
- ▶ IBM Tivoli OMEGAMON XE for CICS on z/OS (OMEGAMON XE for CICS) data in SMF 112 records, containing transaction data for Adabas, CA-Datcom, CA-IDMS, and Supra database management systems

CICS PA can help:

- ▶ System Programmers to track overall CICS system performance and evaluate the results of their system tuning efforts
- ▶ Application Programmers to analyze the performance of their applications and the resources they use
- ▶ Database Administrators to analyze the usage and performance of database systems such as IMS and DB2
- ▶ MQ Administrators to analyze the usage and performance of their WebSphere MQ messaging systems
- ▶ Managers to ensure that transactions are meeting their required Service Levels and measure trends to help plan future requirements and strategies

CICS PA reports all aspects of CICS system activity and resource usage, including:

- ▶ Transaction response time
- ▶ CICS system resource usage
- ▶ Cross-system performance, including multi-region operation (MRO) and advanced program-to-program communication (APPC)
- ▶ CICS Business Transaction Services (BTS)
- ▶ CICS Web Support
- ▶ External subsystems, including DB2, IMS, and WebSphere MQ
- ▶ System Logger performance
- ▶ Exception events that cause performance degradation
- ▶ Transaction file and temporary storage usage

Rather than keeping large SMF data sets for reporting purposes, you can use CICS PA to load selected SMF records into a CICS PA historical database (HDB), optionally summarizing the records according to the time intervals that you require for reporting (such as hourly or daily). You can then use CICS PA to produce reports from the HDB instead of the SMF data sets. Loading selected and summarized SMF data into an HDB allows you to accumulate the performance data you want at the level of detail you need for reporting over long periods, without requiring large amounts of storage or processing time.

In addition to producing formatted reports from SMF data sets or HDBs, CICS PA can extract data to DB2 tables or comma-separated value (CSV) text files. You can then develop your own custom reports using DB2 SQL queries, or download CSV files to your PC, where you can view and manipulate the data using PC-based spreadsheet applications such as Microsoft Excel®.

CICS PA provides both an interactive ISPF dialog interface and a batch command interface. You can use either of these interfaces to request your reports and extracts. The ISPF dialog interface uses your interactive input to prepare JCL for the batch command interface. If you prefer to work directly with a command interface rather than an interactive interface, then you can use the ISPF dialog interface to prepare JCL that you can save and use as a starting point, and then edit the JCL later without using the ISPF dialog.

Figure 4-1 shows the SMF record types that CICS PA can read, and the output formats that it can produce:

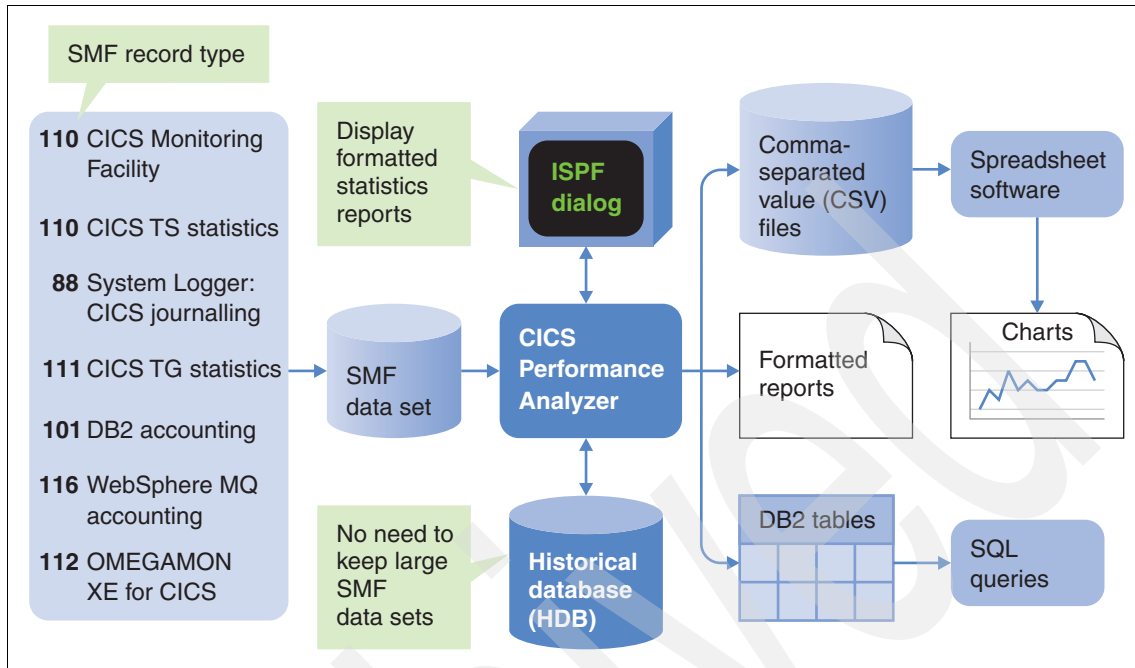


Figure 4-1 Overview of CICS PA inputs and outputs

Figure 4-2 shows the primary option menu of the CICS PA ISPF dialog.

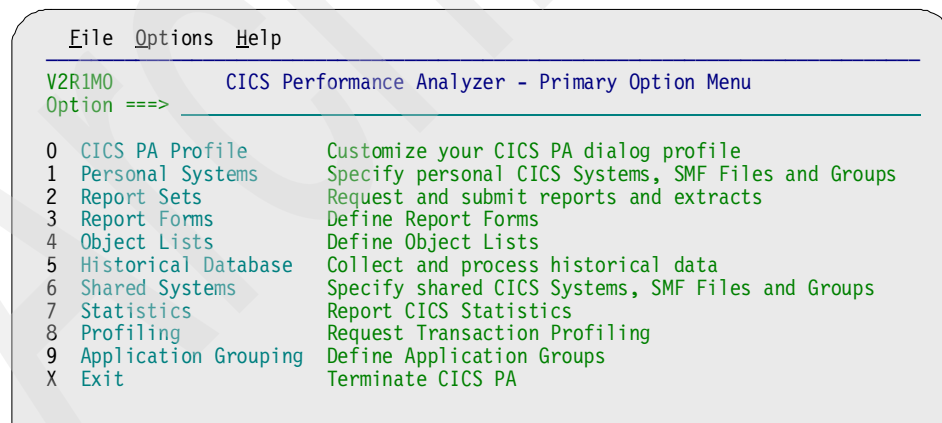


Figure 4-2 CICS PA primary option menu

Primary menu option 2, Report Sets, displays a list of the formatted reports that you can select, organized by category.

Figure 4-3 shows a sample report set. The Active column indicates the reports that this particular report set contains. You can generate and submit JCL to run reports individually in separate batch jobs, or you can submit an entire report set in a single batch job, performing a single pass over the input data.

```

File Systems Confirm Options Help
EDIT                                     Report Set - DEFAULT                               Row 1 of 38
Command ==>                               Scroll ==> CSR

Description . . . CICS PA Report Set

Enter "/" to select action.

--  ** Reports **                               Active
-  -- Options                                   Yes
-  --   Global                                   Yes
-  -- Selection Criteria                         No
-  --   Performance                             No
-  --   Exception                               No
-  -- Performance Reports                       Yes
-  --   List                                    Yes
-  --   List Extended                           No
-  --   Summary                                Yes
-  --   Totals                                 No
-  --   Wait Analysis                           No
-  --   Transaction Profiling                   No
-  --   Cross-System Work                       No
-  --   Transaction Group                       No
-  --   BTS                                    No
-  --   Workload Activity                       No
-  -- Exception Reports                         No
-  --   List                                    No
-  --   Summary                                No
-  -- Transaction Resource Usage Reports        No
-  --   File Usage Summary                     No
-  --   Temporary Storage Usage Summary        No
-  --   Transaction Resource Usage List        No
-  -- Subsystem Reports                        No
-  --   DB2                                    No
-  --   WebSphere MQ                           No
-  --   OMEGAMON                               No
-  -- System Reports                           No
-  --   System Logger                           No
-  -- Performance Graphs                       No
-  --   Transaction Rate                       No
-  --   Transaction Response Time              No
-  -- Extracts                                 Yes
-  --   Cross-System Work                       No
-  --   Export                                 Yes
-  --   Record Selection                       No
-  --   HDB Load                               Yes
-  --   System Logger                           Yes
-  -- ** End of Reports **

F1=Help    F3=Exit    F7=Backward  F8=Forward  F10=Actions  F12=Cancel

```

Figure 4-3 CICS PA report set

Some reports have their own specifically designed layout. Example 4-1 shows excerpts from a CICS PA Wait Analysis report. The data used in this and other examples in this chapter is taken from the “DataPower UsernameToken (200 clients)” scenario described in Chapter 10, “Security scenarios” on page 229.

Example 4-1 CICS PA Wait Analysis report

V2R1M0		CICS Performance Analyzer Wait Analysis Report			
WAIT0001 Printed at 15:00:30 10/20/2008		Data from 05:43:14 9/23/2008 to 05:54:00 9/23/2008			Page 1

Tran=CWXN					
Summary Data					
	----- Time -----		----- Count -----		----- Ratio -----
	Total Average		Total Average		
# Tasks			30469		
Response Time	39.5847	0.0013			
Dispatch Time	3.7077	0.0001	280555	9.2	9.4% of Response
CPU Time	3.3984	0.0001	280555	9.2	91.7% of Dispatch
Suspend Wait Time	35.8769	0.0012	280555	9.2	90.6% of Response
Dispatch Wait Time	2.7375	0.0001	250086	8.2	7.6% of Suspend
Resource Manager Interface (RMI) elapsed time	0.2292	0.0000	60938	2.0	0.6% of Response
Resource Manager Interface (RMI) suspend time	0.0091	0.0000	407	0.0	0.0% of Suspend

Suspend Detail					
	----- Suspend Time -----		----- Count -----		
	Total Average %age Graph		Total Average		
SOIOWTT Inbound Socket I/O wait time	17.3834	0.0006	48.5% *****	695	0.0
DSPDELAY First dispatch wait time	15.8406	0.0005	44.2% *****	30469	1.0
DSCHMDLY Redispatch wait time caused by change-TCB mode	2.4354	0.0001	6.8% *	243852	8.0
LMDELAY Lock Manager (LM) wait time	0.2175	0.0000	0.6%	5546	0.2
N/A Other Wait Time	0.0000	0.0000	0.0%	7	N/C

Tran=ORDR					
Summary Data					
	----- Time -----		----- Count -----		----- Ratio -----
	Total Average		Total Average		
# Tasks			30469		
Response Time	57.1524	0.0019			
Dispatch Time	25.6257	0.0008	1098942	36.1	44.8% of Response
CPU Time	23.0556	0.0008	1098942	36.1	90.0% of Dispatch
Suspend Wait Time	31.5266	0.0010	1098942	36.1	55.2% of Response
Dispatch Wait Time	14.3158	0.0005	1068473	35.1	45.4% of Suspend
Resource Manager Interface (RMI) elapsed time	0.2170	0.0000	60938	2.0	0.4% of Response
Resource Manager Interface (RMI) suspend time	0.0097	0.0000	375	0.0	0.0% of Suspend

Suspend Detail					
	----- Suspend Time -----		----- Count -----		
	Total Average %age Graph		Total Average		
N/A Other Wait Time	14.0310	0.0005	44.5% *****	60286	2.0
DSCHMDLY Redispatch wait time caused by change-TCB mode	10.8973	0.0004	34.6% *****	792194	26.0
DSPDELAY First dispatch wait time	3.7942	0.0001	12.0% **	30469	1.0
LMDELAY Lock Manager (LM) wait time	2.8042	0.0001	8.9% *	215993	7.1

Tran=ORDS					
Summary Data					
	----- Time -----		----- Count -----		----- Ratio -----
	Total Average		Total Average		
# Tasks			30469		
Response Time	18.0849	0.0006			
Dispatch Time	10.3556	0.0003	274513	9.0	57.3% of Response
CPU Time	9.9859	0.0003	274513	9.0	96.4% of Dispatch
Suspend Wait Time	7.7292	0.0003	274513	9.0	42.7% of Response
Dispatch Wait Time	3.7307	0.0001	244044	8.0	48.3% of Suspend
Resource Manager Interface (RMI) elapsed time	0.2473	0.0000	60938	2.0	1.4% of Response
Resource Manager Interface (RMI) suspend time	0.0263	0.0000	515	0.0	0.3% of Suspend

Suspend Detail					
	----- Suspend Time -----		----- Count -----		
	Total Average %age Graph		Total Average		
DSPDELAY First dispatch wait time	3.6047	0.0001	46.6% *****	30469	1.0
DSCHMDLY Redispatch wait time caused by change-TCB mode	2.2645	0.0001	29.3% *****	121876	4.0
LMDELAY Lock Manager (LM) wait time	1.1200	0.0000	14.5% **	80379	2.6
N/A Other Wait Time	0.7399	0.0000	9.6% *	41789	1.4

...

V2R1M0

CICS Performance Analyzer
Wait Analysis Recap Report

WAIT0001 Printed at 15:00:30 10/20/2008

Data from 05:43:14 9/23/2008 to 05:54:00 9/23/2008

Page 1

	Time		Ratio			
	Total	Average				
# Tasks	91624					
Response Time	115.4707	0.0013				
Dispatch Time	40.3321	0.0004	34.9% of Response			
CPU Time	36.5626	0.0004	90.7% of Dispatch			
Suspend Wait Time	75.1382	0.0008	65.1% of Response			
Dispatch Wait Time	20.7840	0.0002	27.7% of Suspend			
Resource Manager Interface (RMI) elapsed time	0.7001	0.0000	0.6% of Response			
Resource Manager Interface (RMI) suspend time	0.0450	0.0000	0.1% of Suspend			
	Suspend Time				Field Availability	
	Total	Average	%age	Graph	Present	Missing
DSPDELAY First dispatch wait time	23.2451	0.0003	30.9%	*****	91624	0
MXTDELAY > First dispatch MXT wait time	0.0000	N/C	0.0%		91624	0
TCLDELAY > First dispatch TCLSNAME wait time	0.0000	N/C	0.0%		91624	0
SOIOWTT Inbound Socket I/O wait time	17.3834	0.0002	23.1%	****	91624	0
DSCHMDLY Redispach wait time caused by change-TCB mode	15.5971	0.0002	20.8%	****	91624	0
N/A Other Wait Time	14.7710	0.0002	19.7%	***		
LMDELAY Lock Manager (LM) wait time	4.1416	0.0000	5.5%	*	91624	0
...						
Total (All Suspend Wait events)	75.1382	0.0008	100.0%	*****		

Other reports have a simple tabular layout (rows and columns of data). To produce these tabular reports, or to extract data to DB2 tables or CSV files, you use *report forms* to specify the performance data fields you want, any sorting and summarization options needed, and functions (such as average or total) that you want applied to the data. You can select and customize one of the many sample report forms supplied with CICS PA, or you can create your own report forms from scratch.

Chapter 10, “Security scenarios” on page 229 uses a custom report form with the **Performance Reports** → **Summary report** option. For details, see “Breakdown of results” on page 250.

Figure 4-4 shows a sample report form that summarizes, by transaction ID, the average values of various Web service-specific performance data fields.

```

File Edit Confirm Upgrade Profiling Options Help
EDIT SUMMARY Report Form - WBSV3SUM Row 1 of 13 More: >
Command ==> Scroll ==> PAGE
Description . . . CICS WEBSERVICE Usage (V3) Version (VRM): 640
Selection Criteria:
_ Performance * Page width . . 132

Field Sort
/ Name + K O Type Fn Description
---
TRAN K A Transaction identifier
TASKCNT Total Task count
WBIWBSCT CICS INVOKE WEBSERVICE requests
PGTOTCCT Total number of CONTAINER requests
PGBRWCCT BROWSE CONTAINER requests
PGGETCCT GET CONTAINER requests
PGGETCDL GET CHANNEL CONTAINER data length
PGMOVCCCT MOVE CONTAINER requests
PGPUTCCT PUT CONTAINER requests
PGPUTCDL PUT CHANNEL CONTAINER data length
PGCRECCT Number of Containers created
EOR ----- End of Report -----
EOX ----- End of Extract -----
***** Bottom of data *****

F1=Help F3=Exit F4=Prompt F5=Rfind F7=Backward F8=Forward
F10=Left F11=Right F12=Cancel

```

Figure 4-4 CICS PA: editing a report form

When creating your own report forms from scratch, you can select fields from all of the CICS performance groups (Figure 4-5).

Command ==>

Select Field Categories

CMF Groups:

DFHAPPL - Application naming

DFHBTS - BTS

DFHCHNL - CHANNEL option

DFHCICS - CICS task information

DFHDATA - Data processing

DFHDEST - Transient Data

DFHDOCH - Document Handler

DFHEJBS - EJB Server

DFHFEPI - Front End (FEPI)

DFHFILE - File Control

DFHJOUR - Journal

DFHMAPP - BMS Maps

DFHPROG - Program Control

DFHRMI - Resource Manager (RMI)

DFHSOCK - Secure Sockets

DFHSTOR - Storage Control

DFHSYNC - Syncpoint processing

DFHTASK - Task Control

DFHTEMP - Temporary Storage

DFHTERM - Terminal Control

DFHWEBB - Web Interface

Region Types:

AOR - Application-owning

FOR - File-owning

TOR - Terminal-owning

DB2 - AOR with DB2

User Fields:

DBCTL - IMS DBCTL

CROSSYS - Cross-System

OMCICS - OMEGAMON

Figure 4-5 CICS PA: selecting categories of fields to include in a report form

In addition to basic functions such as Minimum, Maximum, Average, and Total, report forms also support more advanced functions such as Range. The Range function allows you to analyze the distribution of your transaction performance within ranges of values.

For example, Figure 4-6 shows a custom report form that allows you to see how many (and what percentage) of your transactions fall within certain ranges of CPU time.

```

File Edit Confirm Upgrade Profiling Options Help
-----
EDIT SUMMARY Report Form - CPUDIST      Row 1 of 12 More: >
Command ==>                               Scroll ==> PAGE
Description . . . CPU distribution .0007-.0013      Version (VRM): 650
Selection Criteria:
  _ Performance *                               Page width . . 132

Field  Sort
/ Name + K O Type  Fn  From  To  Report
---
TRAN  K  A
TASKCNT
CPU   -  TIME  RNG  <= .0007  COUNT  Seconds
CPU   -  TIME  RNG  <= .0007  PERCENT  Seconds
CPU   -  TIME  RNG  <= .0010  COUNT  Seconds
CPU   -  TIME  RNG  <= .0010  PERCENT  Seconds
CPU   -  TIME  RNG  <= .0013  COUNT  Seconds
CPU   -  TIME  RNG  <= .0013  PERCENT  Seconds
CPU   -  TIME  RNG  > .0013  COUNT  Seconds
CPU   -  TIME  RNG  > .0013  PERCENT  Seconds
EOR
EOX
***** Bottom of data *****

```

Figure 4-6 CICS PA summary report form: CPU time distribution using Range function

This particular report form defines overlapping ranges: the range $\leq .0013$ includes transactions in the range $\leq .0010$, which includes transactions in the range $\leq .0007$. Alternatively, we could have defined discrete ranges with From and To values, where each range includes transactions within its own discrete band of values.

Example 4-2 shows the resulting report for a single transaction ID.

Example 4-2 CICS PA Performance Summary report showing CPU time distribution

Tran	#Tasks	<=.0007		<=.0007		<=.0010		<=.0010		<=.0013		<=.0013		>.0013		>.0013	
		User	CPU	User	CPU	User	CPU	User	CPU	User	CPU	User	CPU	User	CPU	User	CPU
		Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time	Time
ORDR	30469	11303	37.10	29834	97.92	30458	99.96	11	.04								

This report shows us that 99.96% of ORDR transactions used 0.0013 seconds, or less, of CPU time. However, 11 ORDR transactions used more than 0.0013 seconds of CPU time. Depending on our performance expectations, we might want to further analyze those particular transactions in more detail to find out why they used more CPU time than most other transactions.

4.2 Analyzing CICS Web Services performance

Before using CICS PA to analyze the performance of your CICS Web Services, it is useful to review how to configure CICS Web Services, because this affects the performance data that they generate, such as the CICS transaction IDs involved. Understanding this configuration helps you to know what performance data to measure, and what type of processing that data covers.

Figure 4-7 shows the configuration of a CICS Web Services provider that was created using the CICS Web Services assistant and that uses the HTTP transport (rather than WebSphere MQ).

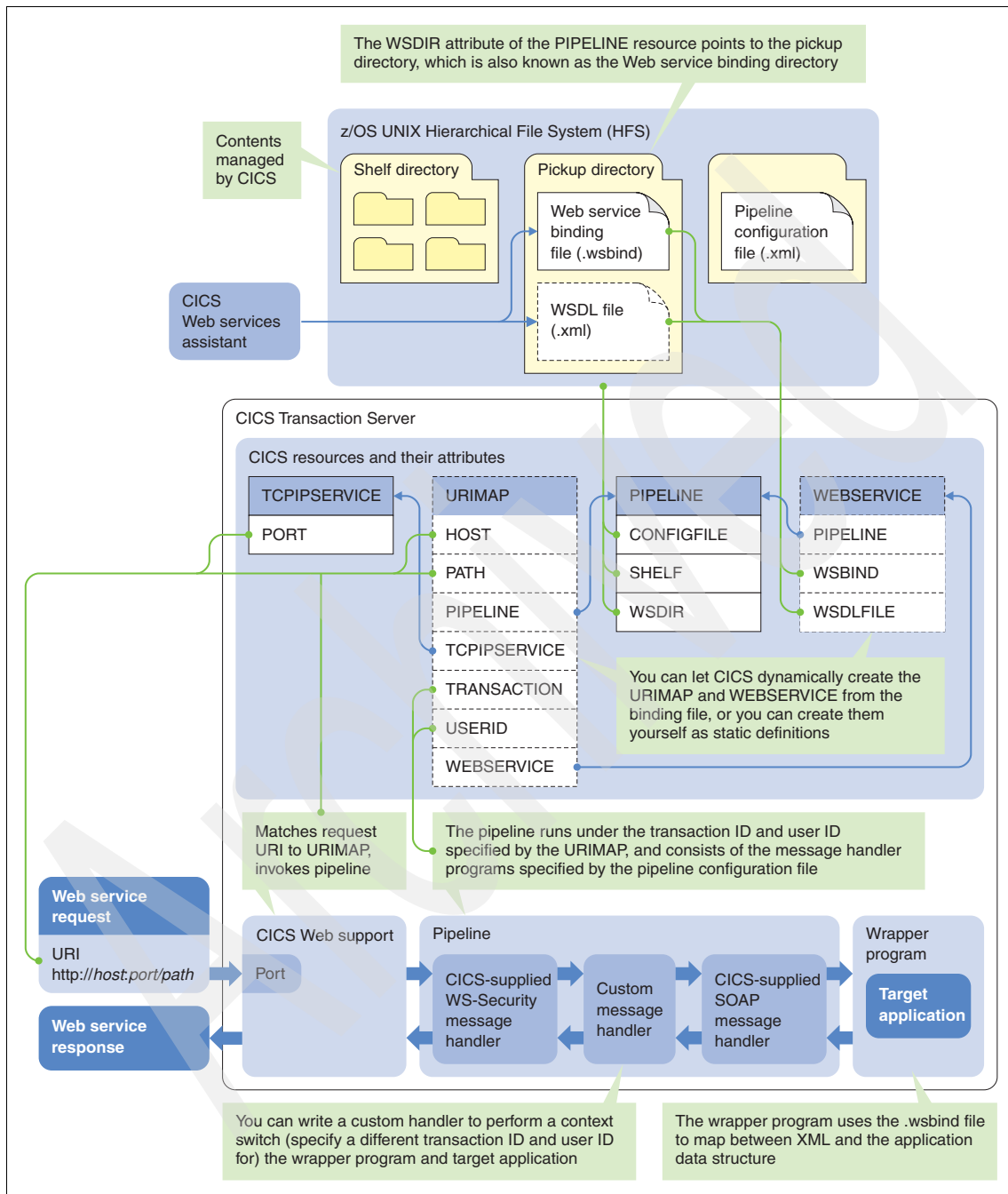


Figure 4-7 CICS Web Services provider configuration

With this configuration, processing a Web service request results in either two or three transaction instances, each of which generates a CMF performance record:

- ▶ When CICS Web support receives a Web service request, it matches the universal resource identifier of the request to a URIMAP resource, and then invokes the pipeline identified by the URIMAP. This CICS Web support processing runs under the transaction ID **CWXN**.
- ▶ The pipeline runs under the transaction ID specified by the URIMAP. If the URIMAP does not specify a transaction ID, the default is **CPIH**.
- ▶ The pipeline can contain a custom message handler program that causes the wrapper program and “target application” (the original CICS application) to run under a different transaction ID from the pipeline.

Figure 4-8 shows an example sequence of CICS transaction instances for processing a Web service request:

- ▶ **CWXN** handles the incoming request and passes it onto the appropriate pipeline.
- ▶ **ORDR**, the transaction code specified by the PIPELINE resource, covers the processing performed by the message handlers in the pipeline, in both directions: request processing and response processing. This includes sending the response to the client.
- ▶ **ORDS**, the transaction code specified by a custom message handler in the pipeline, covers the processing performed by the wrapper program (and target application).

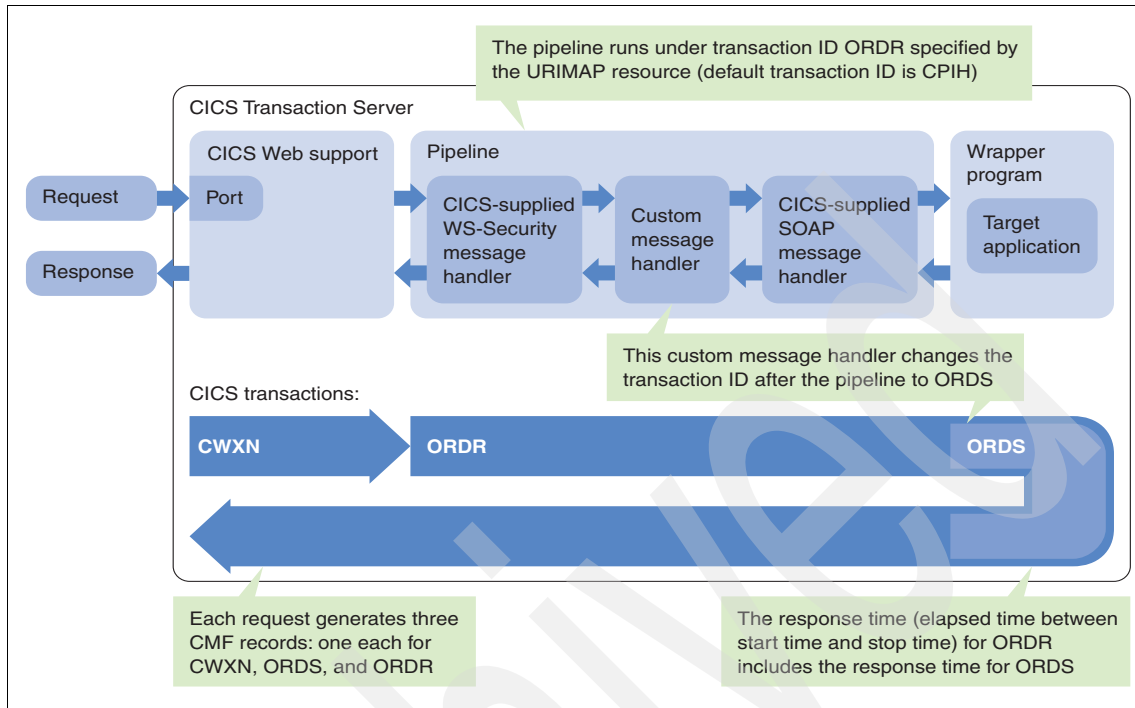


Figure 4-8 CICS Web Services provider: example sequence of CICS transactions per request

With the configuration shown in this example sequence, we expect three CMF performance records per Web service request: one for CWXN, one for ORDR, and one for ORDS.

4.3 Charting CICS Web Services performance

You can use CICS PA to create comma-separated value (.csv) files of CICS performance data that you can then transfer to your PC and chart using various PC-based applications. CICS PA SupportPac CP12 *Charting historical CICS performance data* includes a Timeline Chart add-in for Microsoft Excel for Windows® that makes it easy to chart and navigate around time-based performance data. See Figure 4-9.

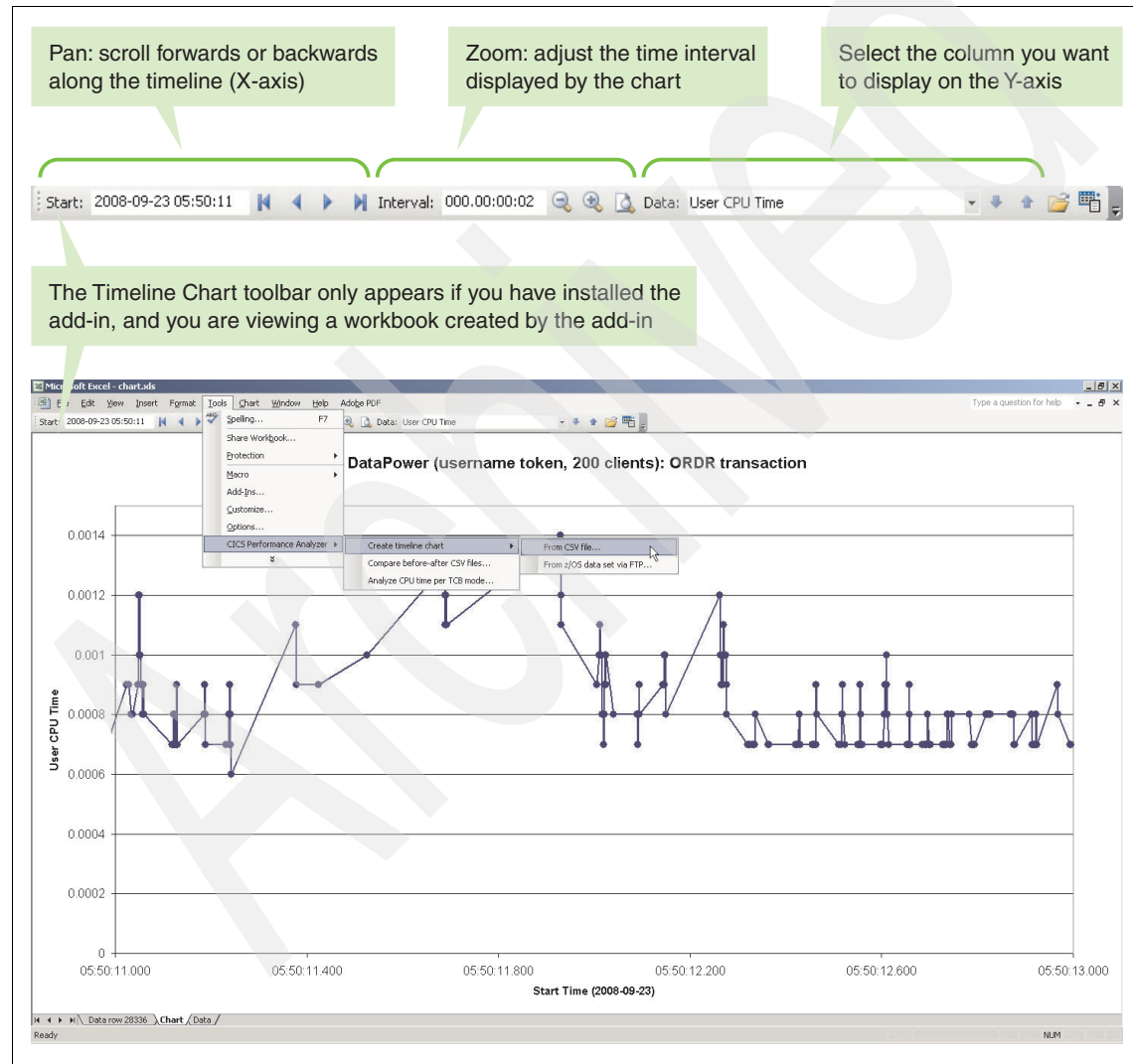


Figure 4-9 CICS PA SupportPac CP12: Timeline Chart add-in for Excel

You can download CICS PA SupportPac CP12 from the Web at no charge:

<http://www.ibm.com/support/docview.wss?uid=swg24011321>

Archived

IBM Tivoli Monitoring

In this chapter, we discuss IBM Tivoli Monitoring (ITM) solutions that provide a solid foundation for the development of management solutions addressing the complex needs of today's IT infrastructures. All Omegamon XE products and all ITCAM products use and require the Tivoli Management Services infrastructure (sometimes referred to as Tivoli Monitoring Services) included in ITM.

A fundamental understanding of ITM concepts and a basic knowledge of how to make use of them is required if you want to work with Omegamon XE for CICS or ITCAM for SOA.

We cover the following topics:

- ▶ Tivoli Management Services components
- ▶ Tivoli Management Services resources

5.1 Tivoli Management Services components

Tivoli Management Services components provide security, data transfer and storage, notification mechanisms, user interface presentation, and communication services for a number of products, including IBM Tivoli Monitoring (ITM), IBM Tivoli Composite Application Manager (ITCAM) and OMEGAMON XE monitoring agents, in an agent-server-client architecture.

Figure 5-1 gives a high-level overview of the ITM architecture.

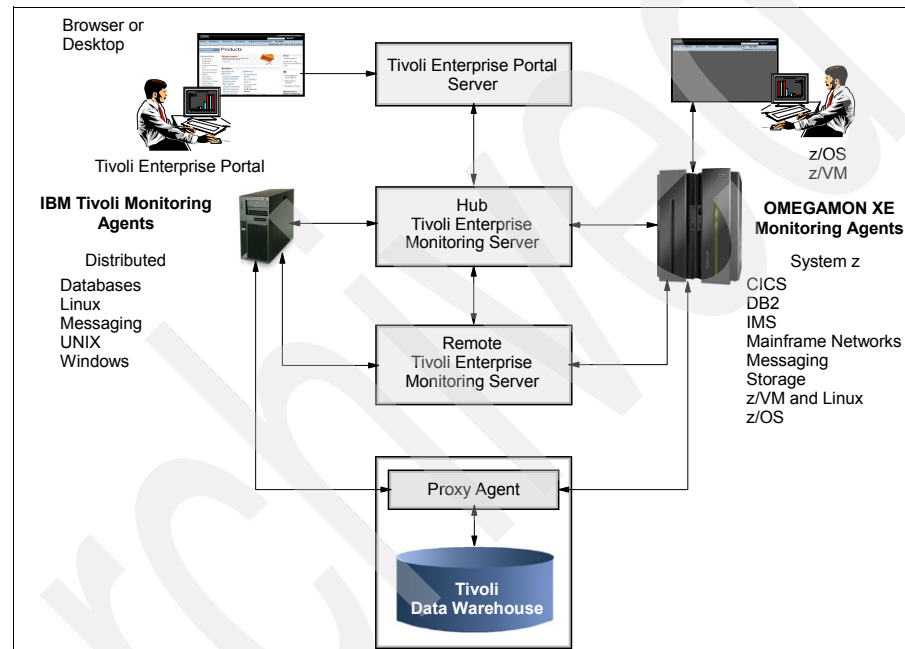


Figure 5-1 IBM Tivoli Monitoring architecture

5.1.1 Tivoli Enterprise Monitoring Server

The Tivoli Enterprise Monitoring Server (TEMS), referred to as the monitoring server, is the key component on which all other architectural components depend directly. The monitoring server acts as a collection and control point for alerts received from agents, and collects their performance and availability data.

The monitoring server stores, initiates, and tracks all situations and policies, and is the central repository for storing all active conditions on every Tivoli Enterprise Monitoring Agent. Additionally, it is responsible for initiating and tracking all generated actions that invoke a script or program on the Tivoli Enterprise

Monitoring Agent. The monitoring server also keeps track of all Tivoli Enterprise Monitoring Agents connected to it through a heartbeat mechanism.

The monitoring server storage repository is a proprietary database format (referred to as the Enterprise Information Base (EIB)) grouped as a collection of files located on the Tivoli Enterprise Monitoring Server.

The primary monitoring server is configured as a hub(*LOCAL). All IBM Tivoli Monitoring V6.2 installations require at least one monitoring server configured as a hub.

Optionally additional remote (*REMOTE) monitoring servers introduce a scalable hub-spoke configuration into the architecture. This hub/remote interconnection provides a hierarchical design that enables the remote monitoring server to control and collect its individual agent status and propagate the agent status up to the hub monitoring server. This mechanism enables the hub monitoring server to maintain infrastructure-wide visibility of the entire environment.

5.1.2 Tivoli Enterprise Portal Server

The Tivoli Enterprise Portal Server (TEPS) is a repository for all graphical presentation of monitoring data. The TEPS provides the core presentation layer, which allows for retrieval, manipulation, analysis, and reformatting of data. It manages this access through user workspace consoles. The TEPS:

- ▶ Receives requests from the Tivoli Enterprise Portal clients
- ▶ Communicates with a Hub Tivoli Enterprise Monitoring Server to send requests to and retrieve data from monitoring agents
- ▶ Is responsible for building and formatting the workspaces viewed in the Tivoli Enterprise Portal clients

The TEPS uses a working repository, the *TEPS database*, to store and retrieve Tivoli Management Services resources and relationship information. This is a IBM DB2 UDB or MS® SQL database and should reside on the machine where the TEPS is running. The TEPS uses ODBC on Windows and DB2 CLI on UNIX® or Linux to access the database. The TEPS database tables hold information such as:

- ▶ TEPS version information
- ▶ User ID's and permissions
- ▶ Login information: current and historical
- ▶ Workspace definitions
- ▶ Query definitions
- ▶ Custom navigator definitions
- ▶ Situation assignments to navigator items and event console

Any user-defined database accessible through ODBC can be defined as a datasource to TEPS and data can be retrieved with *Custom SQL* queries. Figure 5-2 shows the Tivoli Management Services components and their interactions.

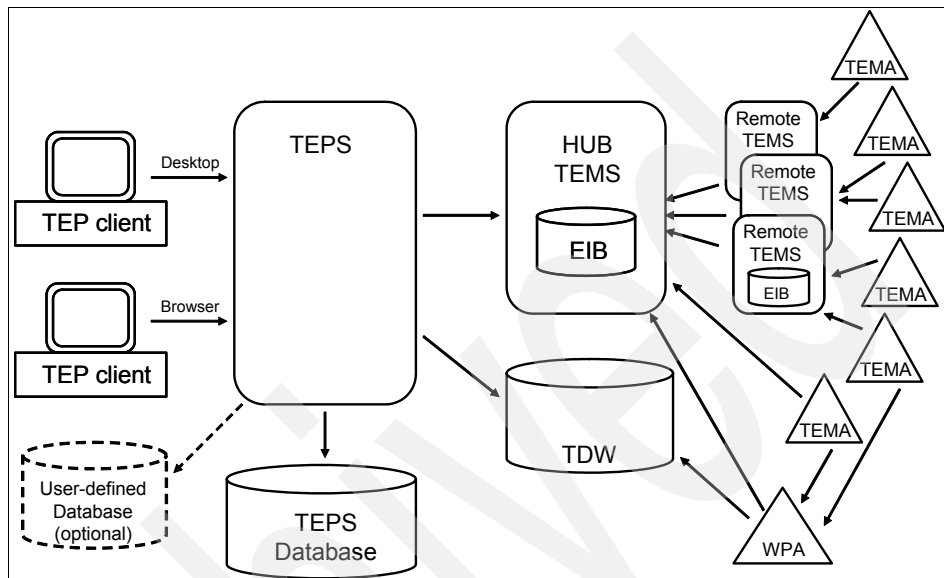


Figure 5-2 Tivoli Management Services components

5.1.3 Tivoli Enterprise Portal

The Tivoli Enterprise Portal (TEP) client, referred to as the portal or portal client or TEP, is a Java-based user interface that connects to the Tivoli Enterprise Portal Server. It is the user interaction component of the presentation layer. The client offers two, functionally identical modes of operation: a Java desktop client and an HTTP browser. The desktop client provides slightly better performance than the browser client.

The Tivoli Enterprise Portal can be launched from an Internet Explorer® browser, or can be installed as a client application on a workstation. If you're using ITM V6.2 with Fixpack 1 or higher, you can also use Firefox as your browser.

IBM Tivoli Monitoring V6.2 uses a Java Web Start capability for administering the desktop client. Java Web Start allows the portal desktop client to be deployed over the network, ensuring the most current version is used.

Tivoli Enterprise Portal:

- ▶ Displays the workspaces it receives from the TEPS. All monitoring data collections can be viewed through workspaces.
- ▶ Allows you to administer and customize most of the Tivoli Enterprise Portal resources through integrated interfaces such as Query Editor, Situation Editor and History Configurator. Proper authorization is required to use those tools.

5.1.4 Tivoli Enterprise Monitoring Agents

The Tivoli Enterprise Monitoring Agents (TEMA), also referred to as monitoring agents, are installed on the system or subsystem requiring data collection and monitoring. Agents exist for the purpose of monitoring operating systems and individual applications. OMEGAMON XE for CICS on z/OS is an example of an agent developed for monitoring CICS on z/OS.

The agents are responsible for the data gathering and distribution of attributes to the TEMS, including initiating the heartbeat status. These agents test attribute values against a threshold and report these results to the monitoring servers. The Tivoli Enterprise Portal displays an alert icon when a threshold is exceeded or a value is matched. The tests are called *situations*.

If historical data collection at the agent is requested, the agent collects the data at each historical collection interval and stores it in local history files. Requests for *short-term historical data* are usually processed directly from those files. On z/OS, the history files are called *Persistent Datastore* files and need to be allocated and formatted with the ICAT configuration tool. On Windows, Linux, and UNIX operating systems the local history is stored in binary files, which are automatically created when they are used for the first time.

5.1.5 Tivoli Data Warehouse

The Tivoli Data Warehouse (TDW) is a database created for the storage of *long-term historical data* collected by the monitoring agents. The Warehouse Proxy Agent retrieves the short-term history data and stores it into the warehouse.

The TEPS retrieves long-term historical data directly from the TDW via ODBC. This data does not pass through the TEMS. If you intend to use the TEPS to retrieve very large amounts of data from the TDW, then a second TEPS, used only for that purpose, can be defined.

5.1.6 Warehouse Proxy Agent

The Warehouse Proxy Agent (WPA) is a unique agent that performs the task of receiving and consolidating all historical data collections from the individual agents to store in the Tivoli Data Warehouse. You can also install multiple Warehouse Proxy Agents in your environment if they are on different computers.

5.1.7 Warehouse Summarization and Pruning Agent

Once per day, the Summarization and Pruning (S&P) Agent performs the aggregation and pruning functions for the historical raw data on the Tivoli Data Warehouse. It has advanced configuration options that enable exceptional customization of the historical data storage. One S&P is recommended to manage the historical data in the Tivoli Data Warehouse. Due to the large amounts of data processing requirements, we recommend that the S&P be always installed on the same physical system as the Tivoli Data Warehouse repository.

The S&P Agent creates additional tables in the TDW containing the summarized data. Data can be summarized Hourly, Daily, Weekly, Monthly, Quarterly, and Yearly. All detail records for the summarization interval are combined and Average, Minimum, and Maximum values are calculated. For data pruning it deletes old records from the detail and summarized tables according to the pruning criteria defined.

Different summarization and pruning criteria can be defined for different types of data with the History Configuration dialog of the TEP. If no values are entered, a global set of defaults, specified during the configuration of the S&P Agent are used.

5.1.8 Tivoli Management Services Engine

The Tivoli Management Services:Engine (TMS:Engine) provides common functions, such as communications, multi-threaded runtime services, diagnosis (dumps), and logging (RKLVLLOG), for Tivoli Enterprise Monitoring Server, and Tivoli Enterprise Monitoring Agents on z/OS. It also provides the VTAM® interfaces for OMEGAMON II products also called the CUA interface.

5.2 Tivoli Management Services resources

Working with a product based on Tivoli Management Services requires a basic understanding of the management resources involved. Using the portal client means using resources such as workspaces, queries, and so on.

5.2.1 Workspaces

After logging on to the TEP, you see a window displaying your initial workspace. All data retrieved from the monitoring agents or from the data warehouse to the TEP is displayed in workspaces. When you navigate through the TEP you move from one workspace to another. A workspace has a name and contains different views. One special view in every workspace is the navigator view, the primary means to move from one workspace to another. Figure 5-3 shows as an example the Omegamon XE for CICS Region Overview workspace.

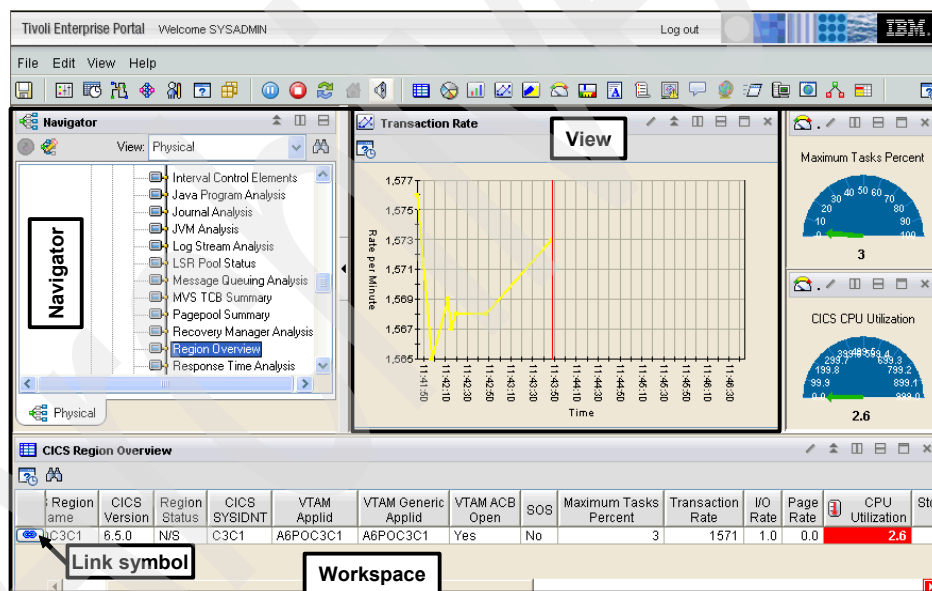


Figure 5-3 Example workspace with navigator and views

Most Omegamon and ITM products come with an extensive set of product-provided workspaces ready to use. If required, you can create and save your own workspaces. You can define:

- ▶ Number and size of views in a workspace
- ▶ What data is displayed in a view and how it is displayed
- ▶ Links to other workspaces, either static or dynamic

Note: Views and links are always stored as part of the workspace in which they are used, you cannot store individual views or links.

5.2.2 Navigator

Selecting an item on the navigator opens the default workspace for that item. If the item has multiple workspaces created for it, you can right-click the item, click **Workspaces** and select another workspace to open.

The following types of navigators are available:

- Physical navigator:** This navigator is automatically constructed from the monitored environment and groups the items in a hierarchy, Enterprise → Platform → Machine → Product → Monitored server or region → Details. Figure 5-4 on page 95 gives an example of a Physical navigator.
- Special navigator:** Some products use special navigators for specific purposes. ITCAM for SOA uses a special navigator to display the Operational Flows workspace and Omegamon XE for CICS has a CICS SLA navigator to define CICS workloads.
- Logical navigator:** This navigator is still supplied for compatibility with Candle Command Center installations. Nowadays only one navigator item is provided in most installations and you can use it as a starting point for defining a customized navigator.
- User-defined navigator:** The TEP provides a Navigator Tool that you can use to define your own navigator items and hierarchy. You can define specific navigators for HelpDesk, Management, Operating, and so on, thereby giving them easy access to just the information (workspaces) they need.

Figure 5-4 shows a physical navigator with z/OS, Linux, and Windows systems being monitored:

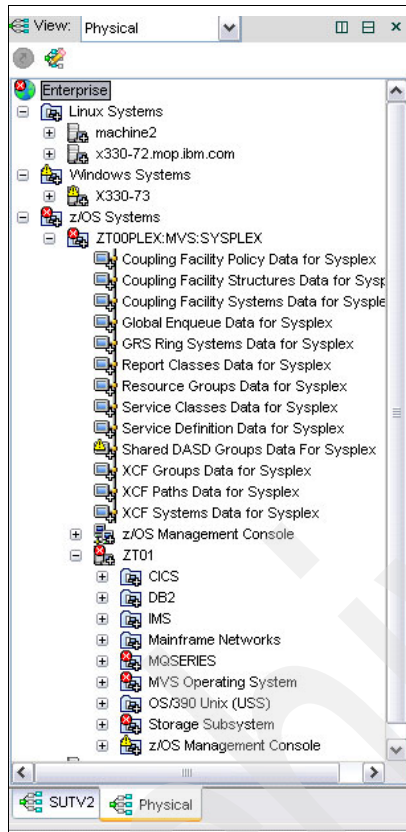


Figure 5-4 Physical navigator

Note: You need to be licensed for IBM Tivoli Omegamon DE on z/OS to utilize user-defined navigators with any Omegamon product.

5.2.3 Links

Another way to move from one workspace to another is through *Links*. There are two types of links:

- ▶ Static Links
- ▶ Dynamic Links

Links are normally selected through the Link symbol. Static links are just a quick way to move from one workspace to another without the need to select another navigator item. Dynamic links use information from the originating workspace to filter the data displayed in the target workspace.

5.2.4 Views

A view is a windowpane, or frame, in the workspace containing a chart or table showing data from one or more monitoring agents. Other types of views such as the topology view and graphic view can give a broader overview of the network. Specialized view such as the browser view and terminal view are also available. You can increase the number of views in a workspace by splitting a view into two separate views.

The data for a table, chart, or relational table-based topology view is chosen by the *query* it uses. The query specifies the attributes to include in the view. Although each view uses one query, you can add more views to the workspace, and each can use a different query.

5.2.5 Query

In views that display monitored data, data is retrieved by queries to the Tivoli Enterprise Monitoring Server. The queries use the attributes monitored by an agent to specify the data to be displayed in the views. Each product comes with a set of product-provided queries. You can use the *Query Editor* to edit the queries that retrieve data in predefined workspaces provided by your monitoring products, or create new queries to populate new views. In addition, you can retrieve data from any ODBC-compliant database to display in a chart or table by writing an SQL SELECT statement. The Query editor is provided as part of the TEP and can be accessed with Ctrl-Q or by selecting the Query Editor button from the tool bar.

Note: Proper authorization is required to use the query editor. Be extra careful if you change a query, because all workspaces using this query are affected.

5.2.6 Attributes

The information gathered from the various monitoring agents is stored for display in tables of attributes. Each *attribute* is a characteristic of an object. For example, Service Port Name is an attribute of a message that identifies the name of the service port where the message was intercepted and for which filtering is defined. Data that is stored in attribute tables is displayed in various table views in the Tivoli Enterprise Portal workspaces. Each table view corresponds to an *Attribute Group*. Columns in the table view correspond to the attributes in the group. Some of these attributes are used as parameters for the creation of situations, or for specifying Take Action commands.

Every product utilizing Tivoli Management Services has a fixed set of Attribute Groups and Attribute Items. They define the scope of data available from that product's agent. The only exception to that is the *Universal Agent*, which dynamically creates Attribute Groups from the Metafiles it receives. A complete documentation of all Attribute Groups and Attribute Items supplied with a product can be found in that product's Online Help and in the *Users Guide*. Figure 5-5 shows how attributes are selected when working with situations or queries.

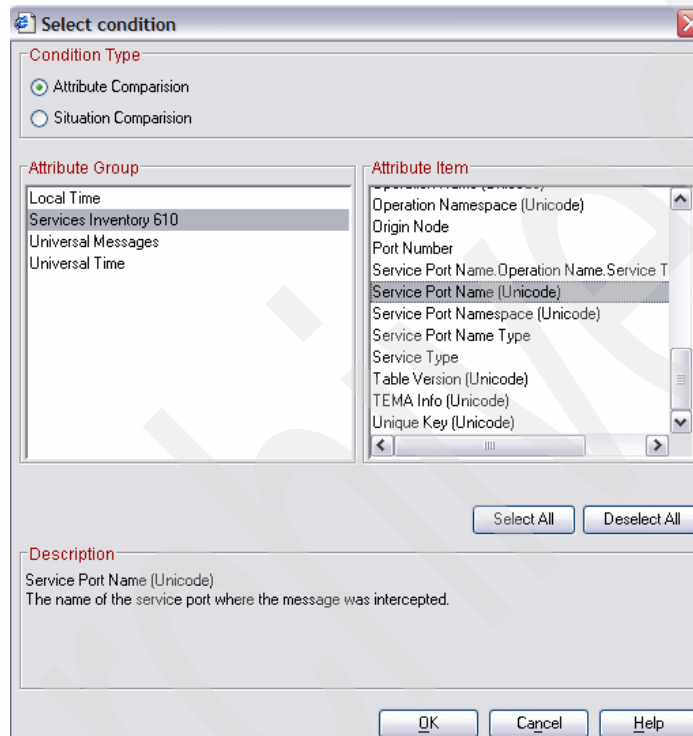


Figure 5-5 Attribute Group example

5.2.7 Situations

Situations and situation events are the key elements for pro-active monitoring. Every Tivoli monitoring agent ships with a rich set of *product-provided situations*, which are the starting point for creating your own customized situations and events, to monitor specific conditions in your enterprise. You create, edit, and manage situations using the Situation editor.

In the IBM Tivoli Monitoring V6 context, a situation is a condition in which a set of attributes are tested against a threshold within any filtering rules (if necessary). The situation evaluates these conditions at predefined intervals and invokes the necessary automated responses and notification methods when needed.

Situations are centrally maintained and stored at the Hub-TEMS, but some auxiliary data for each situation is stored in the TEPS database only. They have a distribution list which defines where this situation should run. When a *managed system* from that list connects, the situation is distributed to it.

Like a query, a situation is a data request against an attribute table. This request is automatically executed every situation interval by a component called SITMON. In a situation you can compare one or more Attribute Items from the same Attribute Group against thresholds you define. Each such comparison is called a *predicate*. Predicates can be logically combined with AND and OR clauses. In addition special *correlated situations* can be defined which query the status of multiple other situations. If qualifying data is found the situation is TRUE, otherwise it is FALSE.

When a situation is TRUE an action can be taken and an *event* can be raised. To raise an event when a situation becomes TRUE, this situation must be *associated* with at least one navigator item. This is done by right-clicking the navigator item to be associated and selecting **Situations**. This invokes the Situation Editor, and any situation selected or created is automatically associated with the navigator item. In Figure 5-6, this is the Web Services Analysis item.

Note: Not every situation needs to be associated with an event. In such a case you can access the situation editor by selecting the Situation Editor button from the tool bar or with Ctrl-E. You can see and edit all situations in your enterprise, but you cannot associate a situation and you cannot define an event state from there.

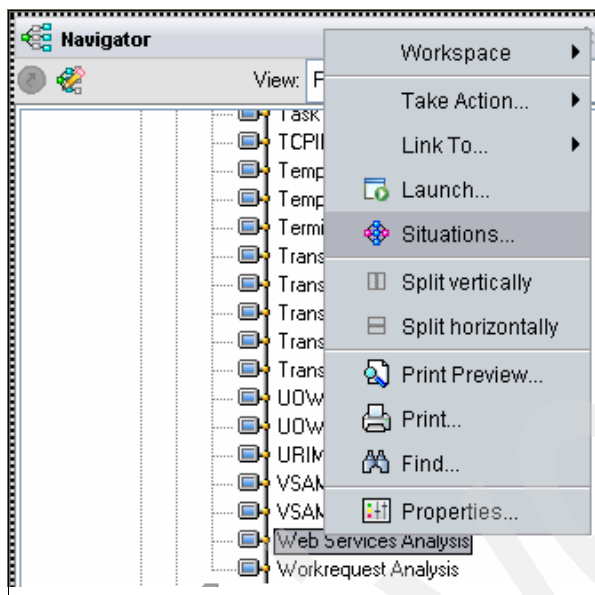


Figure 5-6 Situation Editor selection from navigator item

When an event is raised, the associated navigator item has a state indicator and the event is displayed when you remain with the cursor on the navigator items icon. Later in this book, Figure 11-15 on page 289 and Figure 11-16 on page 289 show an example of this situation.

5.2.8 Actions

The Tivoli Enterprise Portal *Take Action* feature lets you enter a command or stop or start a process on any system in your network where one or more monitoring agents are installed. Take Action commands can be issued either by typing a command into a text box or by selecting a predefined command from a drop-down menu. Take Action commands can also be added to situations to implement simple (reflex) automation.

Some applications provide predefined commands. You can customize those commands or create predefined commands of your own and invoke them as needed on the system you choose.

You can issue a Take Action by:

- ▶ Right-clicking:
 - A navigator item
 - A row in a table or event console view
 - A bar in a bar chart view
 - A slice in a bar chart view
 - A topology view object

Clicking **Take Action** → **Select**.

- ▶ Selecting or typing a command into a Take Action view included in a workspace.
- ▶ Defining an Action for a situation (Reflex Automation).
- ▶ Including an Take Action item in a workflow.

You can use *variable substitution* when defining an action. This powerful feature replaces the variable placeholder with the actual value of the attribute from the row selected or the attribute which triggered the situation, before the command is sent to the system where it is executed. Figure 5-7 gives an example of a CEKL PURGE command. CICS region name and task number are substituted from the row selected, but can also be manually edited if needed.

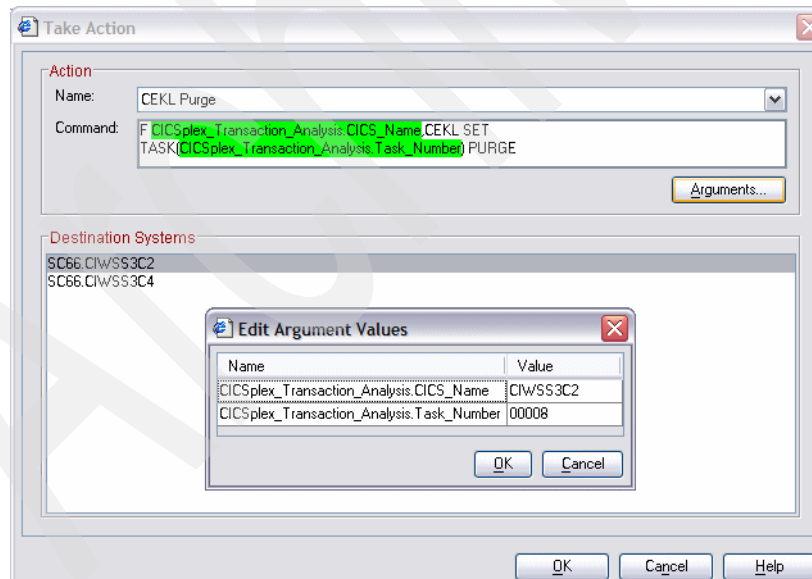


Figure 5-7 Take Action example

5.3 Additional information

The information provided in this chapter is intended to give you a basic understanding of IBM Tivoli Monitoring concepts and terms. For more comprehensive information, we refer you to the official product documentation:

- ▶ *IBM Tivoli Monitoring: Installation and Setup Guide*, GC32-9407
- ▶ *IBM Tivoli Monitoring: Administrators Guide*, GC32-9408
- ▶ *IBM Tivoli Monitoring: User's Guide*, GC32-9409

Also see the IBM Redbooks publication, *Getting Started with IBM Tivoli Monitoring 6.1 on Distributed Environments*, SG24-7143. In that book, Chapter 8, “Real-life scenarios”, contains valuable planning information and several usage examples.

ITCAM FOR SOA

IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) provides monitoring and management of services and mediations in a service oriented architecture (SOA) environment.

In this chapter, we give you a very brief overview of the supported environments and the general features and functions provided by ITCAM for SOA V7.1. We describe in detail how ITCAM for SOA monitors the CICS Web Services environments and explain what you need to do to enable it.

6.1 ITCAM for SOA overview

IBM Tivoli Composite Application Manager for SOA is the Tivoli product specifically designed to operate and to monitor the SOA layer in a composite application environment. To facilitate complete monitoring control of an SOA environment, including platform and middleware monitoring, other monitoring solutions, such as Omegamon XE for CICS, are used together with ITCAM for SOA. To address this need, IBM is offering *IBM Tivoli Composite Application Manager for SOA Platform*, which is a combination of the capabilities of the following products:

- ▶ IBM Tivoli Monitoring including Universal Agent, Agent Builder, and Tivoli Common Reporting
- ▶ ITM for Virtual Servers
- ▶ ITCAM for Web Resources
- ▶ Omegamon XE for Messaging
- ▶ ITCAM for SOA

However, this offering is currently not available for z/OS components, and we mention it here for completeness and clarification only.

6.1.1 Composite application management

A typical business application today has its components spread over several clustered application servers that are interconnected using several different mechanisms. These distributed interconnected applications are referred collectively as *composite applications*. Figure 6-1 shows a sample composite application.

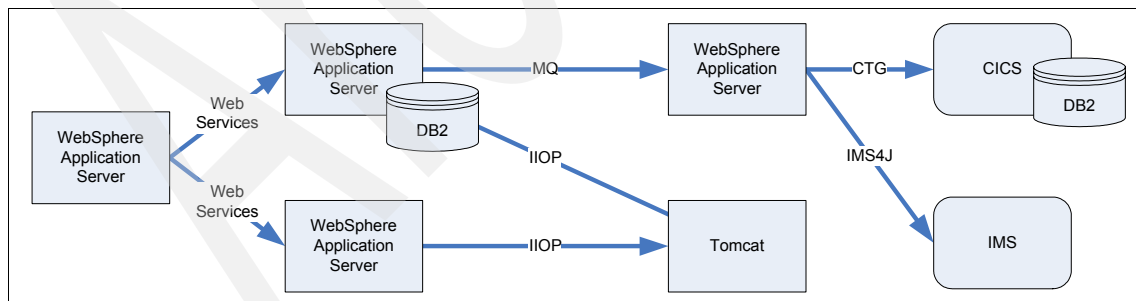


Figure 6-1 Composite application

Managing a composite application, as shown in Figure 6-1 on page 104, requires management of both the underlying resources and an understanding of how the components interact with each other. Understanding that the application is performing poorly from a user view does not necessarily mean that the user-interaction application server has a problem; a back-end server might be suffering from a lack of resources.

Composite application management aims to be able to understand these relationships and present the *root cause* of the application problem. This includes decomposing the application and understanding the individual component resource needs in order to be able to pinpoint resource problems on an application context.

The IBM Tivoli Composite Application Manager family of products addresses the composite application management. These products address different components and decompose transactions to get to the root cause of the problem.

The composite application management product suite consists of the following products:

- ▶ IBM Tivoli Composite Application Manager for WebSphere
- ▶ IBM Tivoli Composite Application Manager for J2EE
- ▶ IBM Tivoli Composite Application Manager for Web Resources
- ▶ IBM Tivoli Composite Application Manager for Response Time Tracking
- ▶ IBM Tivoli Composite Application Manager for Response Time
- ▶ IBM Tivoli Composite Application Manager for Transactions
- ▶ IBM Tivoli Composite Application Manager for Internet Service Monitoring
- ▶ IBM Tivoli Composite Application Manager for SOA
- ▶ IBM Tivoli Composite Application Manager for CICS Transactions
- ▶ IBM Tivoli Composite Application Manager for IMS Transactions
- ▶ OMEGAMON XE for Messaging

You can find a detailed introduction to all ITCAM products in the IBM Redbooks publication, *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151.

6.1.2 Basic architecture

ITCAM for SOA is installed and operates within the management infrastructure of the IBM Tivoli Monitoring environment. ITCAM for SOA manages services, including Web services, SCA components, and services flowing through Datapower and Message broker.

Web services can be viewed as a remote processing facility that is defined through the use of Web Services Description Language (WSDL). Typical access uses SOAP over HTTP. ITCAM for SOA utilizes a data collector (DC) for each of the supported environments:

- ▶ For J2EE Web services, ITCAM for SOA collects data as a JAX-RPC handler.
- ▶ For .NET-based Web services, ITCAM for SOA installs a service extension.
- ▶ For SCA, ITCAM for SOA acts as an SCA handler.
- ▶ For WebSphere Message Broker, ITCAM for SOA uses a message broker exit.
- ▶ For CICS Web Services, ITCAM for SOA uses a message handler in the CICS PIPELINE.

The DC intercepts the SOAP messages and stores information such as metric and operations data in local log files. The log files are periodically monitored by the ITCAM for SOA monitoring agent which makes the data available as attribute groups to TMS. The data is eventually propagated through TEMS and TEPS up to the TEP for viewing.

A set of product provided workspaces displays the collected data utilizing the powerful presentation and filtering capabilities of the TEP and provide immediate productive use of ITCAM for SOA. In addition to that ITCAM for SOA provides several ways to visualize the Web Services interactions and the flow of SOAP messages:

- ▶ Dynamic Service-to-Service topology views in the TEP derived entirely from the observed SOA traffic (new in V7.1)
- ▶ Static Service-to-Service topology views in the TEP based on data from the ITCAM for SOA Discovery Library Adapter (DLA) or from the WebSphere Service Registry and Repository (WSRR) DLA
- ▶ Web Services Navigator, an Eclipse based tool to provide a deep understanding of the service flow, patterns, and relationships for developers and architects

Figure 6-2 gives an example of a dynamic Service-to-Service topology view.

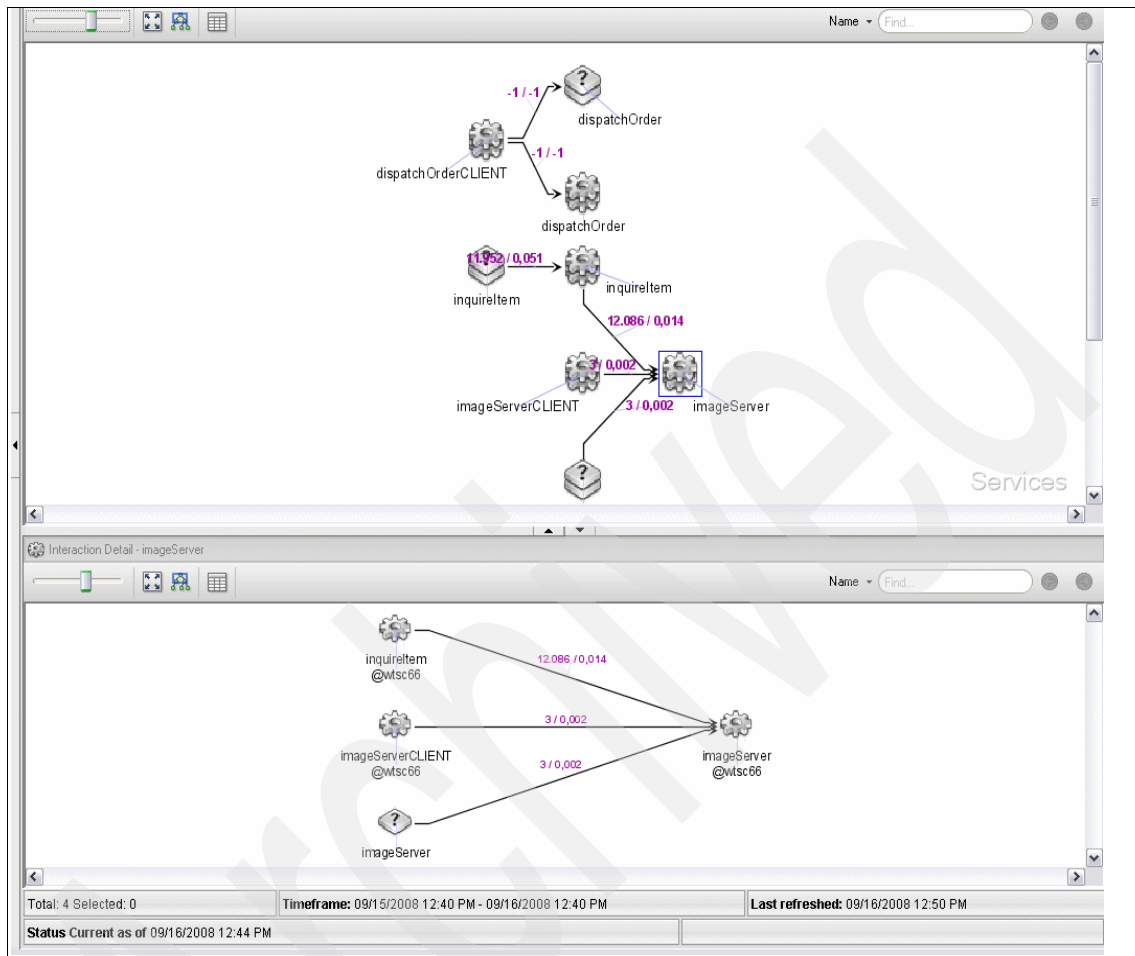


Figure 6-2 Service-to-Service topology view

The DC can be configured to reject messages based on a service, operation, and IP address.

ITCAM for SOA also contains a component for mediation service management based on Service Component Architecture (SCA). There is a special type of SCA component called a mediation. In a service-oriented architecture (SOA), where services are loosely coupled rather than being connected directly to each other, mediations can be inserted between the services, where they can intercept and process messages that are being passed between the services. Mediations can process these messages and take appropriate actions, such as reroute, log, or transform a message, or create a notification or an event.

ITCAM for SOA provides the ability to dynamically enable and disable the deployed mediation functions. This facility is available for applications in the WebSphere Enterprise Service Bus or WebSphere Process Server runtime environment.

6.1.3 Supported application environments

ITCAM for SOA uses the term *services* to describe both Web services and other services with well defined interfaces. Because the definition of a service from an SOA perspective is fairly broad, ITCAM for SOA defines its scope through support for the application server runtime environments that host the services. ITCAM for SOA V7.1 works with a wide variety of application environments:

- ▶ IBM WebSphere Application Server V5.1.0.5 with PQ89492, V6.0, and V6.1
- ▶ IBM WebSphere Business Integration V5.1.1.1
- ▶ IBM WebSphere Process Server V6.0.1 and V6.0.2
- ▶ IBM WebSphere Enterprise Service Bus V6.0.1 and V6.0.2
- ▶ IBM CICS Transaction Server V3.1 and later
- ▶ BEA WebLogic Server V8.1.4, V8.1.5, V9.1, V9.2
- ▶ Microsoft .NET V1.1 with Service Pack 1, V2.0, and V3.0
- ▶ JBoss V4.03
- ▶ WebSphere Community Edition V1.0 and its service packs
- ▶ SAP® NetWeaver V6.40 with Service Pack 9 or later service packs
- ▶ IBM WebSphere DataPower SOA Appliance Firmware V3.6.1 or later
- ▶ WebSphere Message Broker V6 with fixpack 5 or later
- ▶ Apache Axis SOAP engine V1.2, V1.3, V1.4

For the most current list of supported application environments see the official product documentation *Composite Application Manager for SOA V7.1.0 Release Notes*, GI11-4096.

6.2 Product components

This book is focused on CICS Web Services and their performance. For the rest of this chapter we concentrate on those components and features of ITCAM for SOA which are required to manage a CICS Web Services environment. Real life environments are usually much broader in scope, and you might want to deploy additional data collectors from the supported list and use the ITCAM for SOA interface to the WSRR to retrieve information about your Web services from there. We do not cover those topics here, and for more details, we refer you to the official product documentation and to the previously mentioned book, *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151.

6.2.1 Data collector

Each supported application environment has its specific Web services data collector (DC) which usually is installed locally on the runtime environment to be monitored and it collects the services data appropriate to the environment it is collecting from. The message handler component of the DC intercepts Web services calls to collect statistical information and writes it to a log file. The handler is given control when either of the following events occurs:

- ▶ A client application invokes a Web service, which is referred to as a *client-side interception*.
- ▶ The Web service request is received by the hosting application server, which is referred to as a *server-side interception*.

Figure 6-3 shows the interception points 0, 1, 2, and 3, at which data is collected and written to the metrics log file.

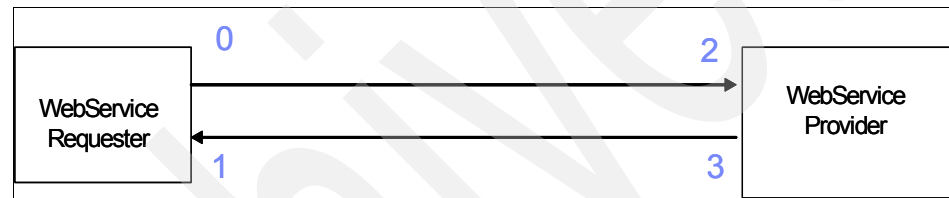


Figure 6-3 Web service interception points

Data collectors typically have access to the following types of information about the message traffic that they monitor:

- ▶ Source and destination (machine name, application server name, service port name, and operation name)
- ▶ Whether the message is a request or a response
- ▶ Interaction type (synchronous or asynchronous)
- ▶ The association of a request to its response during an asynchronous interaction
- ▶ The response time for the message
- ▶ Whether or not the message generated a fault
- ▶ Whether the message is a one-way or a two-way message
- ▶ Optionally, the actual header and body content of the message itself

In addition to the metrics log file and depending on the DCs configuration, other log files might be used by the DC as well. For instance, the header and content of a message are stored in the content log. The DC configuration is stored in the config file, which is regularly scanned by the DC for changes. Figure 6-4 illustrates the usage of logs by an SOA DC.

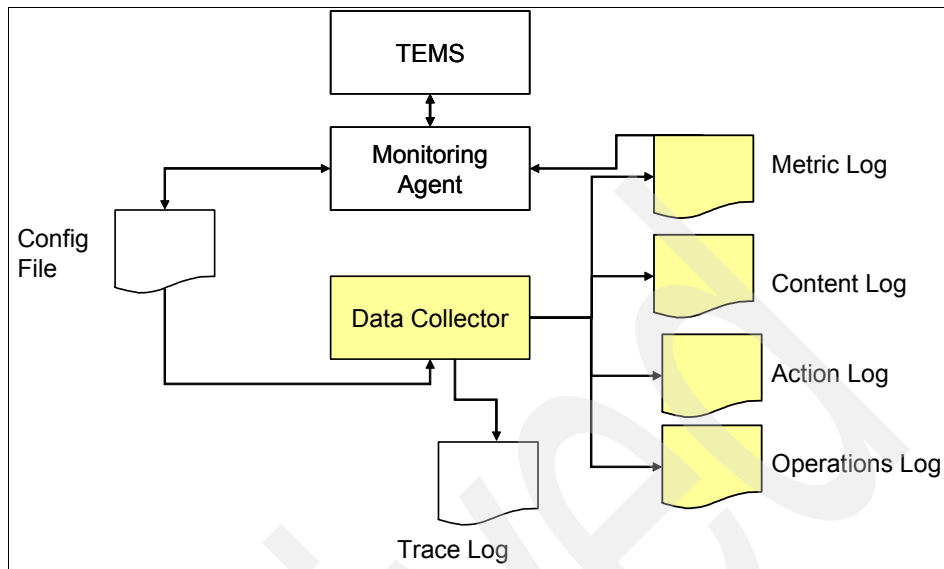


Figure 6-4 SOA data collector file usage

On z/OS, all those files are implemented as z/OS UNIX files.

The CICS DC

The ITCAM for SOA data collector for CICS runs completely inside each monitored CICS region. It interfaces with the ITCAM for SOA monitoring agent in the same way as other data collectors, by reading configuration information from the configuration file and creating metric log files that describe the Web services activity that is being monitored. This data collector only supports CICS TS version 3.1 or later.

The CICS specific processing of the CICS DC is summarized in Figure 6-5:

- ▶ CICS receives and sends SOAP messages through pipelines.
- ▶ To be able to monitor all SOA traffic, the ITCAM for SOA program KD4HAND needs to be added as a Message Handler to all CICS PIPELINE definitions used by the Web services to be monitored.
- ▶ KD4HAND sees all SOAP messages, and writes information about each to a queue within CICS: the Output Storage Queue (OSQ).
- ▶ KD4HAND can also reject messages if the message matches defined filtering criteria for rejection.
- ▶ The File Output Task (KD4O) checks the OSQ in a fixed interval. For each entry on OSQ, it outputs metric, action, and content information to the relevant logs (USS files).

- The Config Reader Task (KD4C) checks the Data Collector Control Configuration File (KD4.dc.properties) regularly. Configuration information is stored in two CICS Temporary Storage Queues (TSQ): one for the General parameters (KD4TSPL), and one for the File Output parameters (KD4FO_PL).

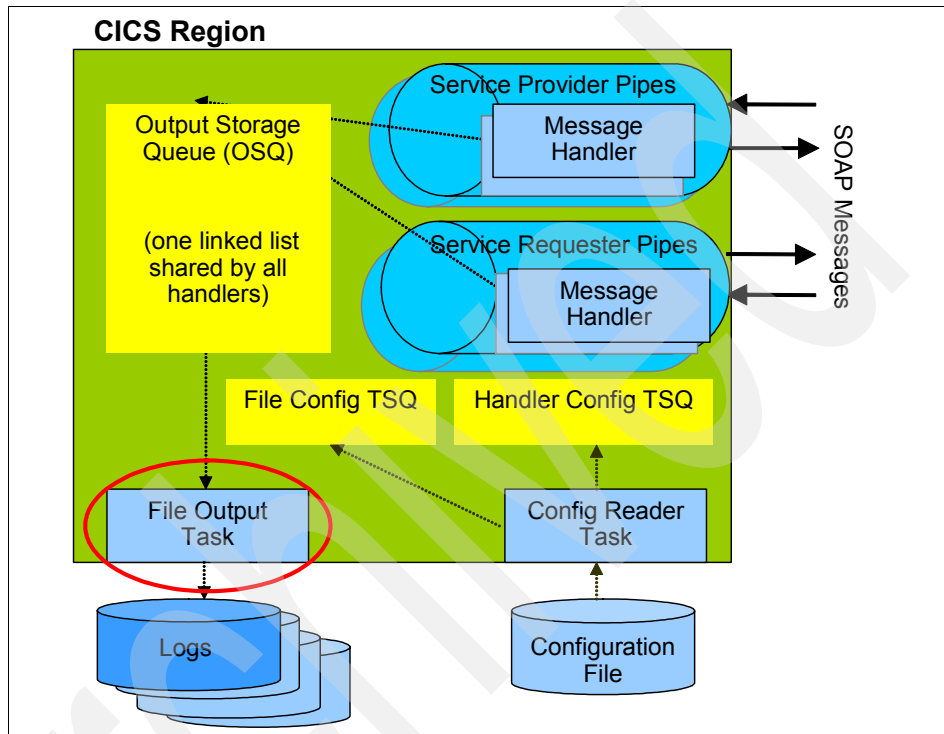


Figure 6-5 CICS DC operation

Because of CICS limitations, the CICS DC cannot determine the following attributes:

- Web Services Description Language (WSDL) Port Namespace
- WSDL Port Name, the monitoring agent uses the CICS WEBSERVICE name
- Operation namespace
- Operation name on incoming messages when acting as a service provider unless a similar message has already been received, and Correlation is active
- WSDL Port Name, Port Namespace, Operation Name, Operation Namespace for any applications that do not use the standard CICS URIMAP and WEBSERVICE resources, or CICS provided terminal handlers

- Remote hostname or Remote IP address for SOAP messages when acting as a service provider

The CICS DC does not output an Operations log; messages are output to the CICS syslog on z/OS.

6.2.2 Tivoli Enterprise Monitoring Agent

The Tivoli Enterprise Monitoring Agent collects information from the DC and forwards it to the Tivoli Enterprise Monitoring Server. It does so by reading the metrics log file and the other files populated by the DC and makes their content available to the TMS infrastructure. The data is displayed in Tivoli Enterprise Portal and is sent to the data warehouse proxy, or to the SOA Domain Management Server in response to a query. One monitoring agent services all SOA DCs on the image that it is running on, regardless of the application environment that the DC is monitoring.

Figure 6-6 shows the interactions of the monitoring agent.

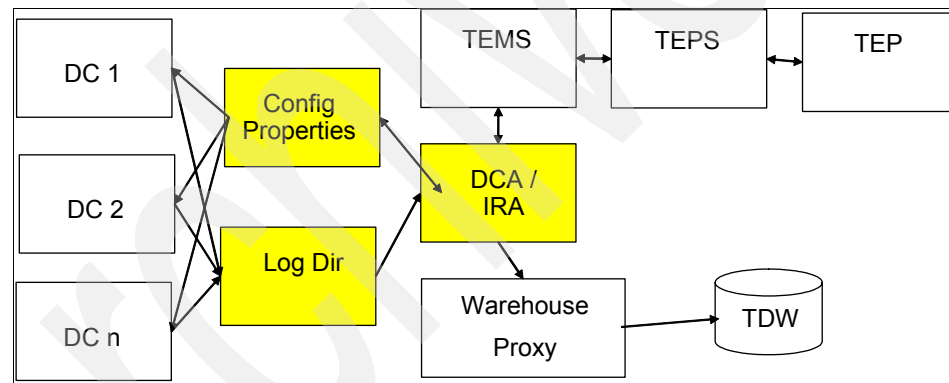


Figure 6-6 SOA monitoring agent overview

These are some additional tasks of the monitoring agent:

- Perform housekeeping of the log files: Deletes processed metric log files based on age and size limits.
- Transform Take Action commands received from the TEP into changes to the config file of a DC.
- Evaluate the Situations received from the TEMS and report any positive evaluation back to the TEMS.
- Collect historical data into history files and forward this data to the Tivoli Data Warehouse (TDW) upon request from the Warehouse Proxy.

HFS subdirectories used by the SOA monitoring agent

A significant portion of the monitoring agent runs in UNIX System Services (USS) on the z/OS operating system, and uses the UNIX Hierarchical File System (HFS) files for configuration and logging. The path to the HFS files used by the ITCAM for SOA monitoring agent and by the DCs is defined during the configuration and is referred to as the <CandleHome> directory.

These are the subdirectories used for configuration and log files:

- ▶ KD4 config subdirectory:
 - Default location: <CandleHome>/KD4/config
 - Configuration properties file is kept in this subdir
 - Configuration properties filename: KD4.dc.properties
- ▶ KD4 log subdirectory:
 - Default location: <CandleHome>/KD4/logs
 - DC Metrics logs and Content logs are created in this subdir
- ▶ KD4 DCA Cache subdirectory:
 - Location: <KD4 log subdirectory>/KD4.DCA.CACHE
 - DC Metrics logs are moved to this subdir as they are processed.
- ▶ KD4 DCA Archive subdirectory:
 - Location: < KD4 DCA Cache subdirectory >/archive
 - DC Metrics logs are moved to this subdir after they have been history collected or after they have expired.

On z/OS the config file is in EBCDIC. All output logs are in UTF-8 and require an ASCII browser to see the contents.

All DCs on a machine write their metrics log files to the same directory. The monitoring agent does not use the filename to determine the originating Application Server. This information is in the metric log header at the beginning of each file.

6.2.3 ITCAM for SOA topology support

IBM Tivoli Composite Application Manager for SOA version 7.1.0 provides additional function in support of service-to-service topology.

The service-to-service topology views supplied with ITCAM for SOA require an additional component, the *SOA Domain management Server (SDMS)*. The SDMS queries the Tivoli Enterprise Monitoring Agent tables to retrieve data on relationships, service ports and operations, deployment environments, and request and response metrics. All of this data is correlated and stored in the SDMS database and used to derive the nodes and links that are displayed in the service-to-service topology views. Dynamic service-to-service topology views use data from the SDMS database to provide a near real-time view of the Web services flow.

If static service-to-service views from WSRR or from a supported DLA are required another database needs to be defined, the *Tivoli Common Object Repository database(TCORE)*. An overview of the WSRR integration with ITCAM for SOA can be found in the *IBM SOA Sandbox information center* at:

http://publib.boulder.ibm.com/infocenter/soasandbox/v1r0m0/index.jsp?topic=/com.ibm.soln.ContentSecAndMgmt.doc_1.0.0/topics/content_planningWSRR.html

Figure 6-7 shows where the topology support is positioned within the ITCAM for SOA architecture.

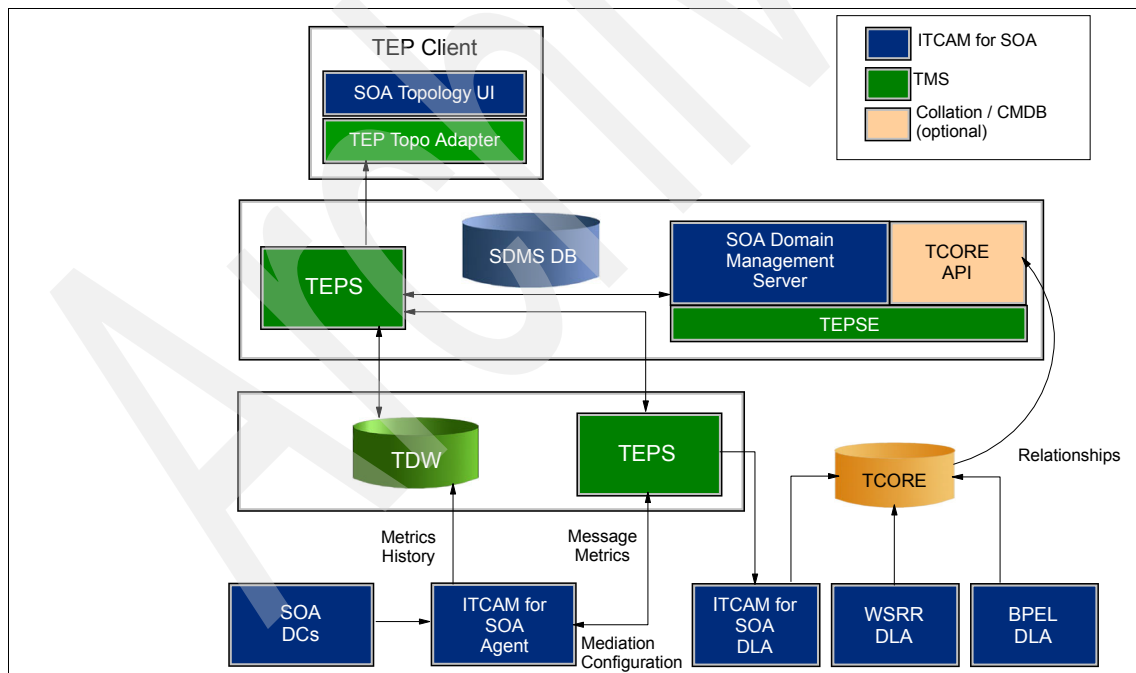


Figure 6-7 ITCAM for SOA topology support

- ▶ SDMS and optionally TCORE are created and configured by running the *SOA Domain Management Server Configuration Utility (ConfigDMS)*. The TEPSE needs to be re-configured and restarted after running ConfigDMS to pick up the changes.
- ▶ Historical relationship data for the past 24 hours is stored in the SDMS database, long-term historical relationship data is retrieved from the TDW.
- ▶ The SDMS requires *Tivoli Enterprise Portal Extensions (TEPSE)* to be installed. TEPSE is automatically installed as part of the base product installation of ITM V6.2, but it needs to be installed separately for ITM V6.1.
- ▶ The Tivoli Enterprise Portal Server must be installed on one of the operating systems that supports SOA Domain Management Server and Tivoli Common Object Repository. See *Composite Application Manager for SOA V7.1.0 Release Notes*, GI11-4096 for details.
- ▶ The operational flow workspace views displayed in Tivoli Enterprise Portal show aggregate and instance operational flows, but not service transaction flows. The numbers above the lines in Figure 6-8 show the number of Web services and the average response time *observed on that particular connection* over a defined interval. It does not necessarily mean that every call to queryItem was followed by an invocation of imageServer. You can use the *Web Services Navigator* to investigate individual transaction flows from the historical data it is analyzing.

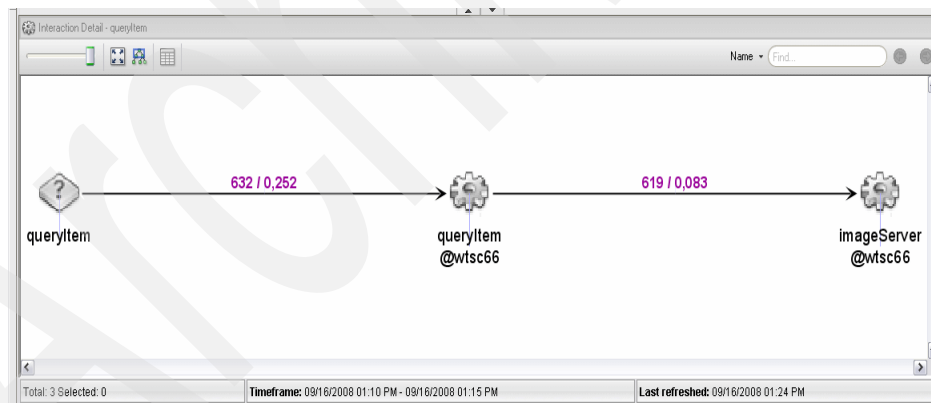


Figure 6-8 SOA topology interaction detail view

6.2.4 Product features

ITCAM for SOA manages service-oriented architecture (SOA). It can monitor, manage, and control the Web services layer of IT architectures.

Next we introduce the product features by describing the TMS management resources provided by ITCAM for SOA:

- ▶ Workspaces
- ▶ Attributes
- ▶ Situations
- ▶ Take Actions

Workspaces

Workspaces provide access to the collected data for your monitored Web services. ITCAM for SOA delivers a set of predefined workspaces, which you can select from the Tivoli Enterprise Portal navigator view. Each workspace has its own set of views that display Web services data and metrics in various levels of detail. Figure 6-9 shows the workspace navigator area.

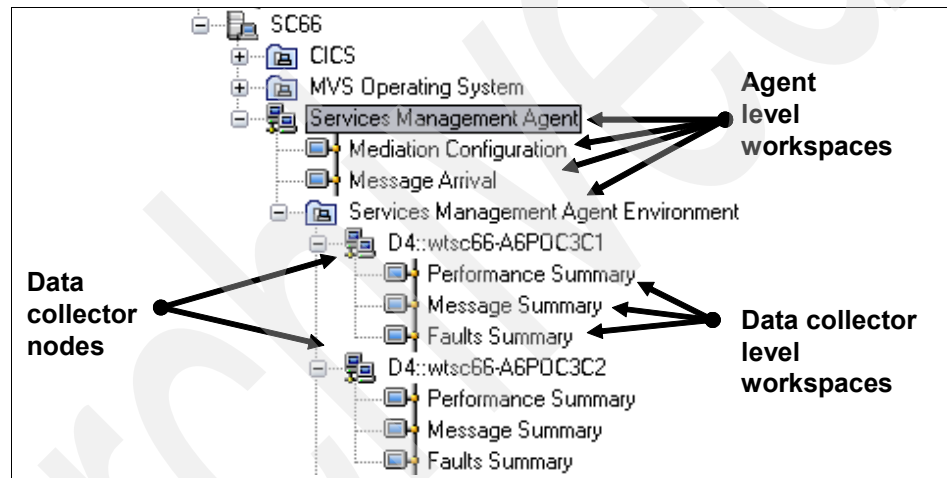


Figure 6-9 ITCAM for SOA physical navigator

The following workspaces are available:

- ▶ The Service Management Agent workspace displays the current configuration details for the monitoring agent data collectors that are configured in different application server instances. This workspace contains the following views:
 - Data Collector Global Configuration
 - Data Collector Monitor Control Configuration
 - Data Collector Filter Control Configuration
- ▶ The Mediation Configuration workspace that contains the SCA mediation configuration. This workspace can be used to launch the SCA mediation actions.

- ▶ The Message Arrival Summary workspace provides a summary of the number of messages that arrive from all the data collectors for each combination of service name, operation name, and remote IP address that has been configured as a situation. This workspace contains the following views:
 - Message Arrival Details
 - Message Arrival by Service
 - Message Arrival by Operation
- ▶ The Services Management Agent Environment workspace provides a set of views that summarize the performance, message activity, and fault occurrences associated with the Web services traffic from all DCs connected to this monitoring agent. This workspace contains the following views:
 - Average Response Time by Operation
 - Number of Messages by Operation
 - Average Message Size by Operation
- ▶ The Performance Summary workspace provides the inventory of currently active and monitored services, as well as the response time of the services. This workspace contains the following views:
 - Average Response Time by Operation
 - Services Inventory

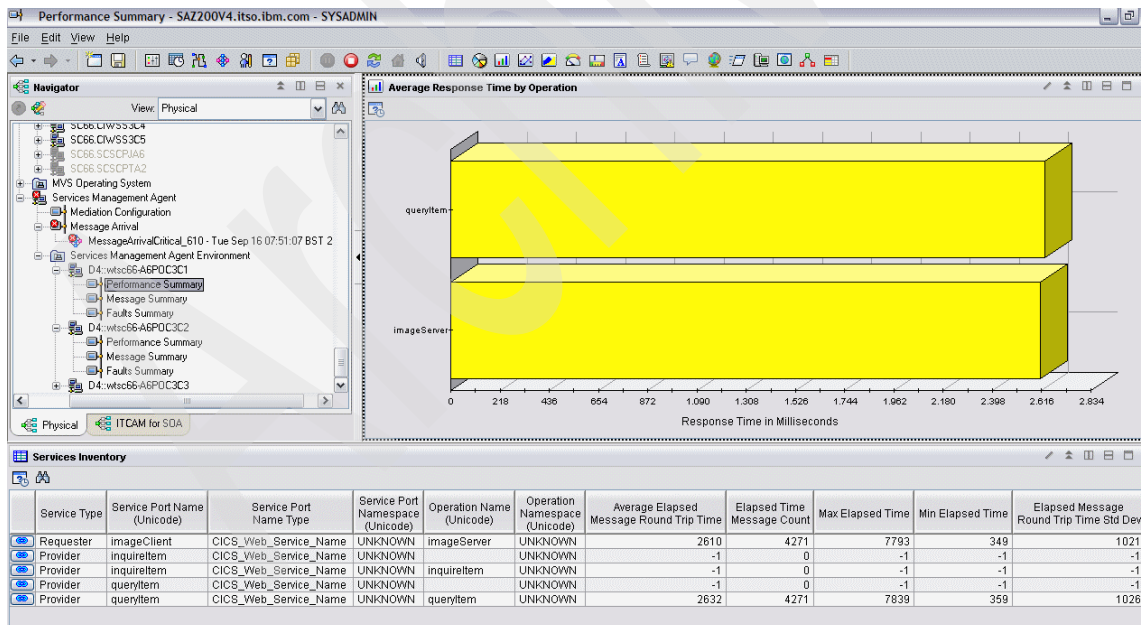


Figure 6-10 Performance Summary workspace

- ▶ The Messages Summary workspace provides details about the number and size of messages received for services and service/operation combinations. This workspace contains the following views:
 - Number of Messages by Service - Operation - Type
 - Average size of Messages by Service - Operation - Type
- ▶ The Faults Summary workspace provides a general faults summary. This workspace contains the following views:
 - Faults Summary by Operation
 - Fault Details
- ▶ Service-to-service topology information is mainly displayed in a workspace called *Operational Flows*. It is accessed in different ways:
 - ITCAM for SOA navigator view, a custom navigator that needs to be assigned to a user or user group through the TEP Administer Users dialog
 - Secondary workspace from the data collector nodes in the Physical navigator
 - *Operational Flow for Operation* link from the Services Inventory view of the Performance Summary workspace

Service-to-service topology is composed of three elements:

Structure	The nodes and their relationships that make up the service flows
Status	The situation status shown at both the aggregate-level and instance-level
Metrics	The computed numbers summarizing each call-path relationship

Call relationship or Node *Tooltips* are displayed when you remain with the cursor on a node or connecting line. Figure 6-11 shows an example of a Call relationship Tooltip.

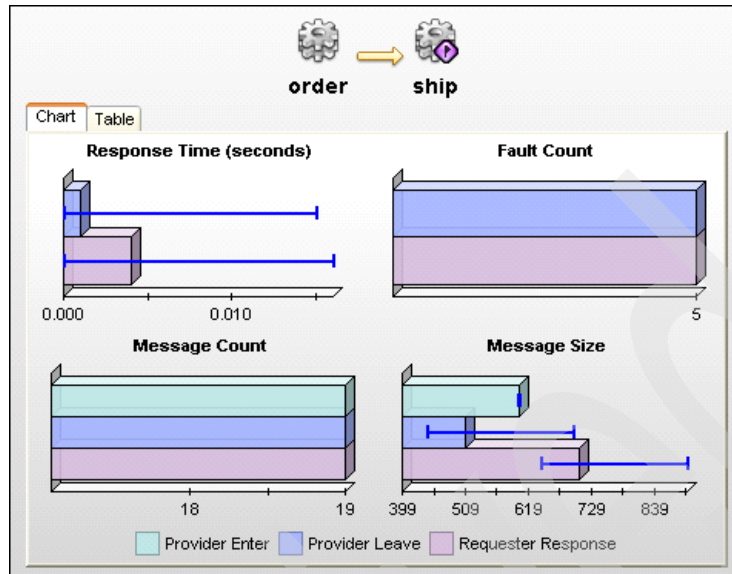


Figure 6-11 Call relationship Tooltip

Service topology views are a powerful tool to quickly analyze a Web services environment. It is important to know that the data presented is an *approximation* designed to meet this specific need. A more detailed and accurate analysis is supported by the Performance Summary workspace and the Web Services navigator.

Attributes

ITCAM for SOA stores specific measurements or attributes relevant to its needs and function. Related attributes are gathered in attribute groups and stored in tables.

Refer to the *Tivoli Composite Application Manager for SOA Installation Guide*, GC23-8803, where an appendix lists the attribute groups or tables provided with the ITCAM for SOA product. The tables that are available for long-term historical data collection are indicated in the description of the table, and show the historical reference information that identifies each attribute within each table. Each table identifies the column name, attribute name, and a description of the data provided. These attributes are used in ITCAM for SOA situations and workspaces retrieve the attributes through queries.

Many of the attribute groups provided by ITCAM for SOA V7.1 are normally only used by the product internally to control the operation of the DCs or to synthesize the service topology views.

The attribute groups used by most situations and workspaces are:

Services Message Metric_610	Provides metric information about messages that have flowed through the interception point
Services Inventory_610	Provides data on current service inventory and contains aggregate metric data
Fault Log_610	Provides information about simple object access protocol (SOAP) faults received by the data collector

Note: The suffix _610 means that normally this table is valid for ITCAM for SOA V6.1 and V7.1 agents.

Situations

ITCAM for SOA provides a set of predefined situations that are designed to help monitor critical activities and serve as templates for creating customized situations for your own use. Most predefined situations are started automatically when the product is installed. After they have been configured, the situation alerts provided with ITCAM for SOA trigger event notification.

Table 6-1 lists the predefined situations that are provided with ITCAM for SOA V7.1.

Table 6-1 ITCAM for SOA situations

Situation name	Description
Fault_610	Monitors the messages in the Web services flow to determine whether a Web services fault has occurred.
Message Arrival Critical_610	Alerts you to excessive amounts of Web services traffic. (The number of messages received from one or more remote clients exceeds a threshold that you specify.)
Message Arrival Clearing_610	Clears a previously triggered message arrival critical situation. This situation can also be used to alert that a message has fallen below a specified threshold (lack of activity alert).
MaxMessage Size_610	Monitors the length, in bytes, of each message in the Web services flow during the most recently completed monitoring interval. If the maximum length of any message is more than the threshold value, this situation is triggered.
Message Size_610	Monitors the average length, in bytes, of the messages in the Web services flow during the most recently completed monitoring interval. If the length of the message is more than the threshold value, this situation is triggered.

Situation name	Description
Response Time Critical_610	Monitors average elapsed round-trip response time, in milliseconds, for the completion of a Web services request.
Response Time Warning_610	Monitors average elapsed round-trip response time, in milliseconds, for the completion of a Web services request.
MaxResponse Time Critical_610	Monitors elapsed round-trip response time, in milliseconds, for the completion of a Web services request. If the round-trip response time for any Web services request exceeds the threshold value, this situation is triggered.
MaxResponse Time Warning_610	Monitors elapsed round-trip response time, in milliseconds, for the completion of a Web services request. If the round-trip response time for any Web services request exceeds the threshold value, this situation is triggered.

Note: Each implementation provides its own set of thresholds, because the product's default can *not* be expected to fit your environment. You must tune your monitoring thresholds.

Take Actions

In ITCAM for SOA predefined Take Actions are available to change the configuration file for the agent and control the operation of the data collector.

The available take action methods for ITCAM for SOA are:

AddFiltrCntl_610	Creates new filter control settings to reject messages.
AddMntrCntl_610	Creates new monitor control settings. These monitor settings affect the data logging for use with IBM Web Service Navigator.
DelFiltrCntl_610	Deletes existing filter control settings.
DelMntrCntl_610	Deletes existing monitor control settings.
UpdMntrCntl_610	Updates existing message logging levels for monitor control.
EnableReqIDMntr_610	Enables data collection for Web service requestor ids.
DisableReqIDMntr_610	Disables data collection for Web service requestor ids.
AddRequesterIdentity_610	Adds Web service identity to be monitored.

DeleteRequesterIdentity_610	Deletes Web service identity from the list to be monitored.
SetReqIDTypeHostIP	Uses host name or IP address as requester identity.
SetReqIDTypeUserInfo	Uses available user information as requester identity.
ConfigureMediation_610	Enables or disables SCA mediation primitives.
DeletePrimitiveProperty_610	Deletes the configuration of a managed SCA mediation primitive.
DisableDC_610	Disables data collection and the ability to reject messages.
EnableDC_610	Enables data collection and the ability to reject messages.
DeleteSubnode	Removes information for all operation instances and relationships associated with the data collector subnode.
updateLogging_610	Defines the level of logging information.
updateTracing_610	Enables or disables tracing.

These actions can be invoked manually or triggered by a situation. Actions can also be triggered by workflows, which are predefined automations that you can build on the IBM Tivoli Monitoring platform.

6.3 ITCAM for SOA installation on z/OS

In many cases, ITCAM for SOA is added to an already existing IBM Tivoli Monitoring infrastructure that has been set up for other products such as Omegamon or ITM distributed monitoring. In this section we give you an overview of the required implementation steps to install and configure the ITCAM for SOA monitoring agent on a z/OS system and the ITCAM for SOA data collector for CICS. Refer to *Tivoli Composite Application Manager for SOA Installation Guide*, GC23-8803 and *Configuring IBM Tivoli Composite Application Manager for SOA on z/OS*, SC32-9493 for more detailed information.

This is the overall implementation procedure for ITCAM for SOA:

1. Plan for the configuration. It is important to have a good understanding of the managed environment and the capability of the product. ITCAM for SOA has some unique requirements that might not be fulfilled in your existing environment.

You should review:

- Supported platforms for the TEPS
 - Which TEMS the monitoring agent connects to
 - SDMS and TCORE database
 - TEP User Administration specifics for SOA
 - Tivoli Data Warehouse capacity
2. ITCAM for SOA is installed and operates within the management infrastructure of the Tivoli Enterprise Monitoring Server services platform. The installation of IBM Tivoli Monitoring V6.1 or V6.2 must be performed before any other ITCAM for SOA component. For Tivoli Monitoring installation information, see the *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407.
 3. Install the application support component for ITCAM for SOA on your Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal systems.
 4. Install and configure the SDMS and optionally the TCORE component of ITCAM for SOA (see 6.2.3, “ITCAM for SOA topology support” on page 113) and reconfigure the TEPS after that.
 5. Install and configure the monitoring agents of ITCAM for SOA. For z/OS, we cover this in 6.3.1, “ITCAM for SOA management agent for z/OS” on page 123. If you need ITCAM for SOA monitoring agent on other platforms, refer to the *Tivoli Composite Application Manager for SOA Installation Guide*, GC23-8803.
 6. Enable and configure the required ITCAM for SOA data collectors. Because the DCs run on the monitored application environment, you most likely need to contact the responsible administrator to assist you with the necessary tasks to enable the DC. For CICS, we cover the required steps in 6.3.2, “Enabling the CICS data collector” on page 134. Refer to *Configuring IBM Tivoli Composite Application Manager for SOA on z/OS*, SC32-9493 for any other DCs on z/OS.
 7. Configure the DC properties with the supplied Take Action commands to set monitoring controls, filtering criteria, and level of detail of the collected data. (see “Take Actions” on page 121).

6.3.1 ITCAM for SOA management agent for z/OS

ITCAM for SOA supplies a complete SMP/E distribution for installing on z/OS systems. The ITCAM for SOA product tape includes all of the z/OS components required for installation, which are:

- ▶ Configuration Tool
- ▶ Tivoli Enterprise Monitoring Server on z/OS
- ▶ ITCAM for SOA Monitoring Agent

Here is an outline of the installation procedure for ITCAM for SOA on z/OS:

1. We perform the standard SMP/E installation. We run this by using batch jobs or by using dialog boxes under the Interactive System Productivity Activity/Program Development Facility (ISPF/PDF). For more information about SMP/E processing, see *IBM Tivoli Composite Application Manager for SOA V7.1.0 Program Directory*, GI11-8685. The installed FMIDs are:
 - HKCI310: Configuration Tool V3.1
 - HKDS610: Tivoli Enterprise Monitoring Server Version 6.1.0
 - HKD4710: ITCAM for SOA Agent V7.1.0
 - HKLV610: CT Engine
2. When ITCAM for SOA has been installed in the ACCEPT stage, we can start the installation configuration tool. The configuration tool is executed from a copy of the KCIINST target library that we call INSTLIB. The high-level qualifier that we use is ITCAMSOA. Note that this configuration tool can be executed by only one TSO user at a time.

We describe the configuration process for installing an ITCAM for SOA agent and connecting it to a Tivoli Enterprise Monitoring Server running on a distributed workstation. We do not configure the Tivoli Enterprise Monitoring Server on z/OS, but it is fully supported if your planning decisions advise or require a TEMS on z/OS, especially if you have it already defined for other products. We also create a full runtime environment (RTE) instead of splitting the environment for a sysplex environment. See *IBM Tivoli OMEGAMON XE V3.1.0 Deep Dive on z/OS*, SG24-7155, for more information about RTE.

Figure 6-12 shows the overall configuration process for a new installation.

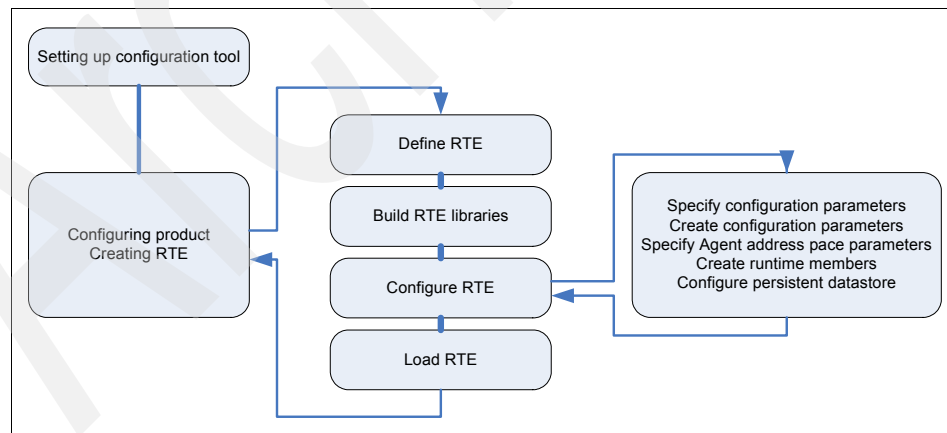


Figure 6-12 Configuration tool processing

3. On the Configuration Tool initial page, shown in Figure 6-13, select option **1** to set up your Configuration Tool.

```
----- MAIN MENU -----
OPTION ==>

Enter the number to select an option:

  1  Set up work environment

  2  Install products

  3  Configure products

  I  Installation information          <=== Revised
  S  Services and utilities

      Installation and Configuration Assistance Tool Version 310.12
      (C) Copyright IBM Corp. 1992-2008
      Licensed Material - Program Property of IBM

F1=Help  F3=Back
```

Figure 6-13 Configuration tool: main menu

4. The setup involves defining the JOB card and high-level qualifiers and allocates work libraries.

Select option **3** to configure a product.

ITCAM for SOA should be available, as shown in Figure 6-14.

```
----- PRODUCT SELECTION MENU -----
COMMAND ==>

Actions: S Select product

S  IBM Tivoli Composite Application Manager for SOA V7.1.0
F1=Help  F3=Back  F5=Refresh  F7=Up  F8=Down
```

Figure 6-14 Configuration tool: product selection

5. When we configure the product, we define the RTE. We only define a full RTE, instead of split SHARING and BASE RTEs. The split RTE is useful, especially if you are running on multiple systems in a sysplex environment. Figure 6-15 shows the definition.

```
----- RUNTIME ENVIRONMENTS (RTEs) -----
COMMAND ==>

Actions: A Add RTE, B Build libraries, C Configure,
        L Load all product libraries after SMP/E,
        D Delete, U Update, V View values, Z Utilities

Action Name      Type      Sharing Description
a      SOA      FULL              ITCAM for SOA full RTE
F1=Help F3=Back (No RTEs defined, use Action A to Add)
```

Figure 6-15 Configuration tool: creating a full RTE

6. As shown in the process overview in Figure 6-16, we need to add (see Figure 6-14 on page 125), build, configure, and load the RTE. We do not create a TEMS, because we use the Tivoli Enterprise Monitoring Server on our distributed server.

```
----- ADD RUNTIME ENVIRONMENT (1 of 2) -----
COMMAND ==>

RTE: SOA      Type: FULL      Desc: ITCAM for SOA full RTE

Libraries      High-level Qualifier      Volser Unit      Storclas Mgmtclas PDSE
Non-VSAM      ITCAMSOA      TAROM2 3390      N
VSAM          ITCAMSOA      TAROM2

Mid-level qualifier ==> SOA

JCL suffix      ==> SOA      Remote RTE for transport ==> N (Y, N)
STC prefix      ==> CANS      Runtime members analysis ==> Y (Y, N)
SYSOUT class    ==> X      Diagnostic SYSOUT class ==> X
Load optimization ==> N      (Y, N)

Will this RTE have a Tivoli Enterprise Management Server ==> N (Y, N)
If Y, TEMS name ==> SOA:CMS      (Case sensitive)

Copy configuration values from RTE ==>      (Optional)

Enter=Next F1=Help F3=Back
```

Figure 6-16 Configuration tool: defining a RTE

7. Press Enter to go to the second RTE definition window, as shown in Figure 6-17.

```
----- ADD RUNTIME ENVIRONMENT (2 of 2) -----
COMMAND ==>

Use z/OS system variables? ==> N (Y, N) Use VTAM model applids? ==> N (Y, N)
  RTE name specification      ==> &SYSNAME
  RTE base alias specification ==>                                     n/a
  Applid prefix specification ==> K&SYSCLONE.

Security system      ==> NONE (RACF, ACF2, TSS, NAM, None)
ACF2 macro library   ==>
Fold password to upper case ==> Y (Y, N)

If you require VTAM communications for this RTE, complete these values:
  Applid prefix      ==> CTD      Network ID      ==> USIBMSC
  Logmode table      ==> KDSMTAB1 LU6.2 logmode    ==> CANCTDCS

If you require TCP/IP communications for this RTE, complete these values:
  Hostname           ==> WTSC58.ITS0.IBM.COM

  Started task       ==> TCPIP (Recommended default = *)
  Port number        ==> 1918
Enter=Next F1=Help F3=Back
```

Figure 6-17 Configuration tool: defining a RTE (second window)

8. In the configuration window for the RTE (Figure 6-18), we configure only the ITCAM for SOA agent. We do not configure the Tivoli Enterprise Monitoring Server because we intend to connect to Tivoli Enterprise Monitoring Server on a distributed system.

```
---CONFIGURE IBM TIVOLI COMPOSITE APPLICATION MANAGER FOR SOA RTE: SOA -----
OPTION ==>

Perform the appropriate configuration steps in order:      Last selected
                                                           Date      Time

If you have defined a TEMS in this RTE that this Agent
will communicate with, select option 1.
  1 Register with local TEMS

  2 Specify configuration parameters
  3 Create configuration parameters

  4 Specify Agent address space parameters
  5 Create runtime members
  6 Configure persistent datastore

  7 Create HFS directories and copy files on USS
  8 Complete the configuration

F1=Help  F3=Back
```

Figure 6-18 Configuration tool: ITCAM for SOA steps

9. The configuration parameters require a UNIX System Services path. We use the path /usr/lpp/Candle, as shown in Figure 6-19.

```
----- SPECIFY CONFIGURATION PARAMETERS -----  
OPTION ==>  
  
HFS CandleHome directory (case sensitive):  
/usr/lpp/Candle_____  
_____  
_____  
_____  
_____  
  
USS CLIST library (Required) ==> SYS1.SBPXEXEC  
HFS migration directory (case sensitive):  
  
Enter=Next F1=Help F3=Back
```

Figure 6-19 Configuration tool: configuration parameters

10. The ITCAM for SOA agent starts in its own address space. Figure 6-20 shows the address space configuration.

```
----- SPECIFY AGENT ADDRESS SPACE PARAMETERS -----
COMMAND ==>

The following information is needed to define the Agent address space:

Agent started task          ==> CANSD4
Connect to TEMS in this RTE ==> N      (Y, N)
Name of Primary TEMS       ==> BEIJING

Specify communication protocols in priority sequence:
IP.PIPE ==> 1 (Non-secure NCS RPC)
IP.UDP  ==> 2 (Non-secure NCS RPC)
SNA.PIPE ==> (Non-secure NCS RPC)
IP6.PIPE ==> (IP.PIPE for IPV6)
IP6.UDP  ==> (IP.UDP for IPV6)
IP.SPIPE ==> (Secure IP.PIPE)
IP6.SPIPE ==> (Secure IP.PIPE for IPV6)

Note: Enable only protocol(s) in use by the Primary TEMS.

Enter=Next F1=Help F3=Back F5=Advanced F10=CMS List
```

Figure 6-20 Configuration tool: agent address space

11. We configure the connection to the Tivoli Enterprise Monitoring Server, as shown in Figure 6-21. This panel is accessed by pressing F10 from the previous panel, which shows the CMS List panel and then F5=Advanced from there.

```
----- SPECIFY AGENT PRIMARY TEMS VALUES -----
COMMAND ==>

TEMS name (case sensitive)      ==> BEIJING

Complete this section if the primary CMS requires SNA support:
LU6.2 logmode                   ==>
Logmode table name              ==>
Local location broker applid    ==>
Network ID                      ==>

Complete this section if the primary TEMS requires TCP support:
* Hostname ==> BEIJING.ITSC.AUSTIN.IBM.COM
Primary TEMS port number based on protocol in use:
IP.PIPE port number ==> 1918 (Non-secure NCS RPC)
IP6.PIPE port number ==> (IP.PIPE for IPV6)
IP.SPIPE port number ==> (Secure IP.PIPE)
IP6.SPIPE port number ==> (Secure IP.PIPE for IPV6)
IP.UDP port number ==> (Non-secure NCS RPC)
IP6.UDP port number ==> (IP.UDP for IPV6)
* Note: See F1=Help for TSO HOMETEST command instructions.

Enter=Next  F1=Help  F3=Back
```

Figure 6-21 Configuration tool: CMS values

Figure 6-22 shows the communication parameter for the ITCAM for SOA agent.

```
----- SPECIFY AGENT IP.PIPE CONFIGURATION VALUES -----
COMMAND ==>

Specify the IP.PIPE communication values for this Agent.

* Hostname      ==> BEIJING.ITSC.AUSTIN.IBM.COM
  Started task  ==> *          (Recommended default = *)
  Network interface list:      (If applicable)
    ==>

Specify Agent IP.PIPE configuration.

  Address translation      ==> N          (Y, N)
    Partition name        ==>

* Note: See F1=Help for TSO HOMETEST command instructions.

Enter=Next  F1=Help  F3=Back
```

Figure 6-22 Configuration tool: IP.PIPE configuration

12. The persistent data store is used to store short-term historical data for the ITCAM for SOA agent. We generally use the default values for the configuration menu.
13. Load and populate the runtime environment. You have the option to populate the UNIX System Services directory with this step or defer it to be executed manually from RKANSAM(KD4USSJB). See Figure 6-23.

```

----- LOAD JOB - INCLUDE USS STEPS -----
COMMAND ==>

The IBM Tivoli Composite Application Manager for SOA component is
configured in this RTE. In order to include information for this RTE on
z/OS UNIX System Services (USS), you must meet the following conditions:
- the job to create directories and copy files to HFS must be submitted
  on a machine that has access to the USS directories.
- the job to create directories and copy files to HFS must be submitted
  by a TSO userid that has write access to the HFS directories specified
  in the Specify configuration parameters panel from the IBM Tivoli
  Composite Application Manager for SOA menu.

If the above conditions are met, then IBM recommends that you specify Y below
to ensure that all maintenance is synchronized. If not, then specify N and
the configuration job will create a KD4USSJB job for later submission. Submit
the hilev.RKANSAMU(KD4USSJB) job from an appropriate machine with a TSO
userid that satisfies these conditions.

Do you want to include the USS steps in the configuration job? ==> Y (Y, N) Y

Enter=Next F1=Help

```

Figure 6-23 Configuration tool: load job

14. Copy startup JCLs to the system procedure library, such as SYS1.PROCLIB. These JCLs reside in the RKANSAM data set:

CANS4	ITCAM for SOA agent
KDSPROC1	Persistent data store maintenance
KDSPROC2	Persistent data store backup and initialization

KDSPROC1 and KDSPROC2 only exist if you configured the Persistent Data Store (PDS).

15. Authorize the RKANMOD, RKANMODU, and RKANMODL from your RTE. You can use the SETPROG console command to perform this. Our setup uses the following commands:

```

SETPROG APF,ADD,DSN=ITCAMSOA.SOA.RKANMOD,VOL=TAROM2
SETPROG APF,ADD,DSN=ITCAMSOA.SOA.RKANMODU,VOL=TAROM2
SETPROG APF,ADD,DSN=ITCAMSOA.SOA.RKANMODL,VOL=TAROM2

```

The ITCAM for SOA monitoring agent should only be started after “Add Application support” was done and the TEMS has been restarted.

6.3.2 Enabling the CICS data collector

There are several configuration steps that must be performed:

1. The CICS Started Task user ID must have read/write access to the agent directory in UNIX System Services. Our setup is /usr/lpp/Candle.
2. The CICS started task needs access to the TKANMOD data set in the DFHRPL concatenation.
3. Data collector programs and transactions must be defined using the CICS System Definition program. A sample JCL is provided in TKANSAM (KD4CSD).

Note: The KD4CSD sample job we had specifies no DATALOCATION for the DEFINE PROGRAM commands. The default is still BELOW. You should modify the job and add DATALOCATION (ANY) to avoid running into storage problems in the UDSA.

4. Review the EDSA definitions for the CICS region, the data collector consumes up to 10 MB of EDSA, and if necessary, increase the amount of storage.
5. Update the CICS program load table (PLT) to include THE KD4INIT module to be loaded and unloaded with startup and shutdown of CICS.
6. The Web services through CICS are handled using the CICS pipelines. These pipelines are configured using an XML definition. The pipeline that you want to monitor must be handled by the KD4HAND program. This program is basically the Web services interceptor. KD4HAND can be defined similarly for the requestor pipeline and provider pipeline.

7. Example 6-1 shows the handler definition excerpt. The name of the configuration file can be found using CICS transaction CEMT I PI.

Note: The KD4HAND program should be the first handler defined in a provider pipeline, and the last handler defined in a requester pipeline. If you are using Web Service Security, define the KD4HAND handler program after the Web Services handler program in a provider pipeline and before the Web Services handler program in a requester pipeline. This allows IBM Tivoli Composite Application Manager for SOA to search through the SOAP message text before encryption or after decryption.

Example 6-1 CICS handler definition

```
<service_handler_list>
  <handler>
    <program>KD4HAND</program>
    <handler_parameter_list><soap_1.2/></handler_parameter_list>
  </handler>
</service_handler_list>
```

8. You do not need to run the KD4ConfigDC.bat script for the CICS DC.

Archived

OMEGAMON XE for CICS on z/OS

OMEGAMON XE for CICS on z/OS is the component of IBM Tivoli Monitoring that provides the capability to monitor CICS Transaction Server (TS) environments. It is installed as a monitoring agent in the IBM Tivoli Monitoring (ITM) framework. Its workspaces and situations alert you when components of the CICS TS environment are not meeting the expected performance parameters.

As a component of ITM, OMEGAMON XE for CICS also provides the capability to combine CICS monitoring with that of other monitored components of the enterprise. It uses Dynamic Workspace Linking (DWL) to allow a user to switch between OMEGAMON XE for CICS and other OMEGAMONs, such as OMEGAMON XE for DB2 on z/OS. This provides users the capability to follow transactions to identify the route of a problem.

The 3270 interface of OMEGAMON XE for CICS on z/OS allows you to perform in depth diagnosis of a CICS TS system.

In this chapter, we explain the major components of OMEGAMON XE for CICS and describe some of the major features. This is not meant to be a comprehensive description, but it can be used to gain an understanding of the product.

7.1 OMEGAMON XE for CICS components

In order for OMEGAMON XE for CICS to be able to provide information relating to CICS Transaction Server, the following components must be configured on the z/OS image where the monitored CICS regions reside:

- ▶ An OMEGAMON XE for CICS Monitoring Agent
- ▶ An OMEGAMON II for CICS Menu System
- ▶ An OMEGAMON II for CICS CUA interface (optional)
- ▶ OMEGAMON for CICS component in monitored CICS regions

Next we discuss the function and specific requirements of each of these components in more detail. For the purposes of this chapter, we assume that the ITM framework has been installed and configured.

7.1.1 OMEGAMON XE for CICS Monitoring Agent

The OMEGAMON XE for CICS monitoring agent is the component that is responsible for responding to queries from the ITM framework for data relating to the CICS TS systems that are currently monitored. The agent obtains information by interfacing with OMEGAMON code that runs in the CICS address space. It also communicates directly with the Menu System started task (KOCCI) to query data that it maintains. The agent has a Workload Manager (WLM) component that is used to generate summaries of transaction performance against specified goals.

The monitoring agent provides the data required to populate the CICS workspaces on the TEP. Figure 7-1 on page 139 shows an example of a workspace that is provided with the OMEGAMON XE for CICS on z/OS product.

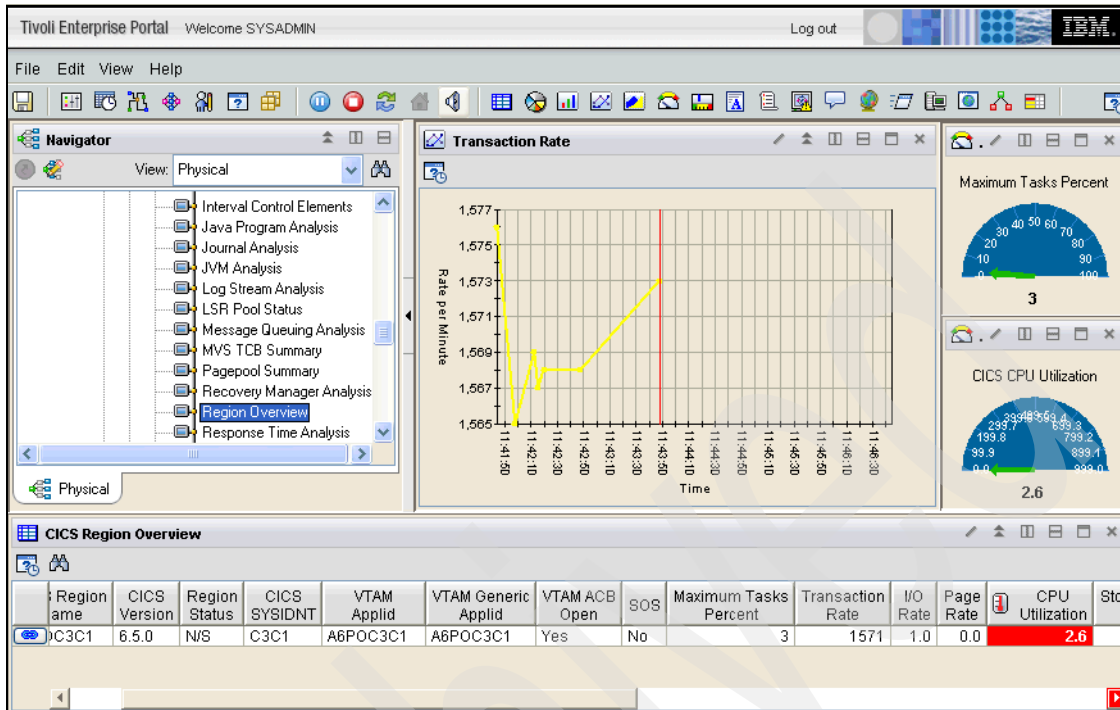


Figure 7-1 OMEGAMON XE for CICS Region Overview

An agent must be deployed on each LPAR where CICS TS regions run. It is recommended that the OMEGAMON CICS agent run in its own address space.

The OMEGAMON XE for CICS agent is configured using the Installation and Configuration Assistance Tool provided with ITM.

Figure 7-2 shows the configuration page for the OMEGAMON XE for CICS agent.

```

----- CONFIGURE IBM TIVOLI OMEGAMON XE FOR CICS ON Z/OS / RTE: SC66 -----
OPTION ==>

Perform the appropriate configuration steps in order:

I Configuration information (What's New)

1 Register with local TEMS (required if the Agent
   will connect to the TEMS in this RTE.)      07/09/24  09:40

2 Specify configuration parameters             08/08/28  16:56

Agent address space configuration:
3 Specify Agent address space parameters      08/08/28  16:56
4 Create runtime members                      08/08/28  16:58
5 Configure persistent datastore (in Agent)   08/08/28  16:58

6 Complete the configuration                   07/08/22  15:22

Note: This Agent is running in its own Agent address space.

F1=Help  F3=Back  F5=Advanced

```

Figure 7-2 Configuration page for OMEGAMON XE for CICS on z/OS

1. Option 1, *Register with local TEMS*, adds application support files to the TEMS. Here is a summary of when you are required to select this option:
 - If the agent you are configuring will report to a TEMS configured in the same (that is, local) RTE, you must select this option.
 - If your hub TEMS is running on z/OS but the agent reports to a remote TEMS, then you need to register the monitoring agent with both the remote TEMS it directly reports to and the hub TEMS.
 - If the agent reports to the hub TEMS directly, then you only need to register it once. If the hub TEMS is configured in a separate RTE from the agent, then you need to select that RTE for configuration of the OMEGAMON XE agent, but you only select option 1, **Register with local TEMS**. You would not need to complete any of the other configuration items unless you were also going to define an OMEGAMON XE agent in the same RTE as the Hub TEMS.
 - If the agent reports directly to the Hub TEMS but this TEMS is configured in a different RTE from the agent, then you need to select that RTE for configuration and then only select option 1, **Register with local TEMS** for this configuration. Do not perform any of the other selected agent configuration items that follow.

In our configuration, where we want to report directly to a remote TEMS in the same RTE as the monitoring agent and our hub TEMS is installed on another operating system, we need to register our monitoring agent just once with the remote TEMS.

2. In option 2, *Specify configuration parameters*, we specify the configuration options that are unique to OMEGAMON XE for CICS on z/OS (Figure 7-3).

```
----- SPECIFY CONFIGURATION PARAMETERS -----
OPTION ==>

Complete the items on this panel.

WLM block allocation    ==> 236                (10-524287 blocks)
WLM collection interval ==> TEP                (TEP or 1 minute)

Enter=Next  F1=Help  F3=Back  F5=Advanced
```

Figure 7-3 OMEGAMON XE for CICS on z/OS product specific parameters

The *WLM block allocation* specifies the number of 4096 byte blocks that will be allocated to the WLM dataspace. This dataspace is used to store information while summaries are created. While the usage of this storage is stable for a given workload, it is not easy to predict. It is recommended that the number start at 236. If the message KCP0244 is produced by the agent, this indicates that the value should be increased.

The *WLM collection interval* controls the timespan of Service Level Analysis displays at the TEP. OMEGAMON WLM accumulates data once a minute in two sets of records, the “5minute records” and the “interval records.” The “5minute records” are displayed at the TEP. The “interval records” are used for history (Persistent Data Store).

Calculations for values such as average response time are performed every 5 minutes and at the end of the history interval.

Specifying a value of 1 transforms the “5minute records” to “1minute records.” Starting WLM with a frequency interval value of 1 will result in the TEP displaying Service Level data collected at the last “1minute” interval.

Specifying a value of TEP will result in the TEP display of the “5minute records” using the values specified via TEP.

The acceptable values for the INTERVAL keyword are 1 or TEP; values other than 1 can be specified using the SLA CICS view of the TEP.

After setting the values for WLM, pressing PF3 takes you back to the panel displayed in Figure 7-2 on page 140. Options 3, 4, and 5 are common to all of the OMEGAMON XE monitoring products. These steps are required to be completed if you are configuring the agent to run in its own address space.

3. Option 3, *Specify Agent address space parameters*, is where you specify the communication values for reporting to the hub, as well the “self-describing” values for configuring up your agent address space, such as security, locale, and so on.
4. In option 4, *Create runtime members*, a batch job is generated that will update the user parameter libraries with the values you have specified as part of configuration options 2 and 3. After submitting this job, check that the return code has produced no errors.
5. Option 5, *Configure persistent data store (in Agent)*, can be used to allocate the persistent data store files and start up parameters as part of the agent address space. During the persistent data store configuration, you are asked to enter JCL jobcard information; it should be noted that this jobcard information is shared by all persistent data store runtime JCL generated for this RTE.

If you have a TEMS defined in this RTE and would like to run the agent in the local TEMS address space, then you need to select F5 and follow the advanced configuration options. This is not the recommended agent configuration, so it is not discussed in this book.

6. Option 6, *Complete the configuration*, guides you through some manual configuration steps that must be performed outside of the Configuration Tool, such as copying the agent started task to your PROCLIB and ensuring libraries are APF authorized and configuring CICS TS regions to be monitored. These steps are required to be completed prior to testing your configuration.

7.1.2 OMEGAMON II for CICS Menu System

The OMEGAMON II for CICS Menu system consists of a started task known as the common interface. This is also referred to as the KOCCL. It provides a 3270 interface to allow monitoring of the CICS regions on the same LPAR, and it provides an anchor to the OMEGAMON for CICS components that run in a CICS region. It is responsible for loading the Global Data areas that control which features of OMEGAMON II are to be active. It also provides an environment for the subtasks required to provide Bottleneck Analysis, Response Time Analysis and Transaction History collection.

The 3270 interface provided by the KOCCI gives users a highly efficient mechanism to view specific information relating to a CICS region. The Menu system allows you to see resource detail information as well as detailed information regarding the memory usage and DASD performance of the devices currently be used by the CICS region. The Menu system also allows authorized users to view and modify the storage in the CICS region.

Figure 7-4 shows the primary panel provided by the Menu system.

```

      ZMENU   VTM   A6POC3C1 V560./C SC66 09/21/08 12:08:24
> PF1 Help/News/Index   PF3 Exit   PF18 Color   PA2 REGION STATUS
=====
>
>          OMEGAMON II FOR CICS PERFORMANCE MONITOR SYSTEM
>          Enter a selection letter on the top line.
> W REGIONS ..... List CICS regions, switch monitoring
> V OVERVIEW ..... Performance overview
> R RESPONSE TIME ..... CICS and end-to-end response time monitoring
> E EXCEPTIONS ..... Current system problems and problems this session
> T TASKS ..... Task analysis
> H HISTORY ..... Historical, traced transaction viewing and selection
> B BOTTLENECKS ..... Resource contention (bottlenecks, impacts, enqueues)
> S STORAGE ..... Storage summary, violations, DSA, EDSA, PAM, subpools
> F FILES ..... CICS datasets, VSAM, LSR, string and buffer waits
> D DATABASES ..... DB2, DLI and MQ
> C CICS ..... CICS tables and control blocks
> M MVS ..... Operating system control blocks for CICS
> I I/O ..... DASD performance
> U UTILITIES ..... Utility functions and displays
> O CONTROL ..... Control options (response time, history, contention,
>                  database collectors, shutdown, SMF, trace)
>
> G GROUPS ..... Group definitions
> P PROFILE ..... Profile options and maintenance
=====

```

Figure 7-4 OMEGAMON II menu system interface

The Menu system can only monitor CICS regions that are running on the same LPAR. For that reason, a KOCCI must be configured on each LPAR where CICS regions that are to be monitored can run.

OMEGAMON II for CICS is configured using the Installation and Configuration Assistance Tool provided with ITM.

Figure 7-5 shows the configuration page for OMEGAMON II for CICS.

```

----- CONFIGURE OMEGAMON II FOR CICS / RTE: SC66 -----
OPTION ==>

Perform these configuration steps in order:

I Configuration information (What's New)

1 Specify configuration values          08/09/18  13:35
2 Allocate additional runtime datasets  07/08/22  15:19
3 Create runtime members                08/08/28  16:55
4 Complete the configuration           08/09/11  22:08

Optional:

5 Allocate/initialize task history datasets  08/09/11  21:35
6 Manage CICS global data area(s)          08/09/16  15:08
7 Modify menu system command security      08/08/28  16:56
8 Install OMEGAMON Subsystem              07/08/22  15:20

F1=Help  F3=Back

```

Figure 7-5 OMEGAMON II for CICS configuration panel

You follow these steps for the configuration:

1. Option 1, *Specify configuration values*. This step walks you through the product specific values for OMEGAMON II for CICS.

Figure 7-6 shows the first panel presented for option 1.

```

----- OMEGAMON II FOR CICS CONFIGURATION VALUES / RTE: SC66 -----
OPTION ==>

VTAM information:
  Maximum number of CUA users          ==> 99      (10-256)
  Enable ACF/VTAM authorized path      ==> N        (Y, N)

CUA security options:
  Specify security                     ==> RACF      (RACF, ACF2, TSS,
                                                NAM, None)
  Function level security resource class ==> $OMEG    (ACF2 max is 3 char)

Started task:
  End-to-End (ETE)                    ==> started_task_name

Advanced options:
  Maximum number of KOCCI users (UMAX) ==> 99      (1-99)
  Copies of OMEGAMON II address spaces ==> 1        (1-16)
  Fold CUA output to upper case        ==> N        (Y, N)
  Enable CUA simplified signon          ==> Y        (Y, N)
  Enable CUA WTO messages               ==> N        (Y, N)

Enter=Next  F1=Help  F3=Back

```

Figure 7-6 OMEGAMON II configuration values

Enter the name of the started task for the End-to-End component. Make sure that the name of the started task for End-to-End (ETE) is correct and is unique for this RTE. You can accept the defaults for the remainder of the values.

When the values are complete, press Enter to reveal the following panel (Figure 7-7).

```

----- OMEGAMON II FOR CICS CONFIGURATION VALUES / RTE: SC66 Row 1 from 48
COMMAND ==>

Modify the parameters below to suit your site's requirements.

ID: 00  Menu STC:  CANSOCO_  Menu applid: &SYSNAME.OCO_____
        CUA STC:   CANSO20_  CUA applid:  &SYSNAME.C20_____
                                CUA operator: &SYSNAME.C200_____
                                VTERM prefix: &SYSNAME.CO_____
                                Default CICS rgn: *_____
        Major node: &SYSNAME.C20N_____
Enter=Next  F1=Help  F3=Back  F7=Up   F8=Down

```

Figure 7-7 OMEGAMON II configuration values continued

Here the started task names and APPLIDs are entered. In this case we are using the z/OS system variables to prefix the APPLIDs. This makes it easier to configure environments for multiple LPARs.

2. Option 2, *Allocate additional runtime datasets*. This step can be ignored at this point, because OMEGAMON II for CICS has no additional datasets allocated in this step.
3. Option 3, *Create runtime members*. This step generates a job that can be submitted to create the required configuration members. This job should run with a return code zero.
4. Option 4, *Complete the configuration*. This step provides a checklist of the steps that need to be completed. This describes such tasks as copying the started task JCL to the system PROCLIB libraries, authorizing the required libraries, moving the VTAM definitions to the correct libraries, and installing the components in the CICS regions.

This completes the required steps to configure OMEGAMON II for CICS. The following optional steps might be necessary depending upon the options that you would like to use and the other products installed in the system.

5. Option 5, *Allocate/initialize task history datasets*. This step is required if you want task history data to be maintained after a CICS region is shut down. It is covered more in 7.2.4, "Transaction history" on page 162.

6. Option 6, *Manage CICS global data area(s)*. In this step, you can create new global data areas or edit existing ones. This global step specifies:
 - a. The features that are to be active for a CICS region.
 - b. The group definitions for response time analysis and bottleneck analysis.
 - c. The rules that are active for the resource limiting feature.
 - d. The location of the task history data.

When editing a global data area, there is a comprehensive help system available that describes each option in detail. This help is available by pressing PF1. The help is context sensitive and it displays information based upon the location of the cursor. Upon exiting from edit the global data area is validated and any errors are displayed.

When the changes have been made to the global data areas, they need to be copied to the RKANPARU datasets. Selecting the option to copy global definitions generates JCL to move the members. This JCL specifies DISP=OLD for the RKANPARU dataset. If you do not want to shut down all the tasks which use this library you should change this to DISP=SHR. The job does not run until you exit the configuration tool completely to free up the datasets that are allocated to your TSO ID.

7. Option 7, *Modify menu system command security*, should be used to specify the type of security that is to be active for the menu system commands. It can be used to set passwords for authorized commands if an external security manager is not to be used. It can also be used to specify the level of security required for specific commands.
8. Option 8, *Install OMEGAMON subsystem*, is only required if this has not been done for another OMEGAMON product on this LPAR.

7.1.3 OMEGAMON II for CICS CUA interface: Optional

The OMEGAMON II for CICS CUA interface (optional) provides a means of viewing the data provided by the Menu system that conforms to the Common User Access (CUA) standard.

OMEGAMON II for CICS requires that a Program and a Transaction be defined as shown in Figure 7-9.

```
DEFINE PROGRAM(KOCOME00) GROUP(OMEGAMON) CONCURRENCY(THREADSAFE)
      EXECKEY(CICS)

DEFINE TRANSACTION(OMEG) PROGRAM(KOCOME00) GROUP(OMEGAMON)
      TASKDATAKEY(CICS)
```

Figure 7-9 RDO definitions for OMEGAMON CICS

CICS PLT changes

For OMEGAMON CICS to be configured to start and stop automatically the CICS PLTs must be updated. This statement shows the PLT additions for installation:

```
DFHPLT TYPE=ENTRY,PROGRAM=KOCOME00
```

Specify the PLT additions, as shown, in the following tables:

- ▶ PLTPI (initialization) *immediately after* the DFHDELIM entry for CICS Transaction Server systems.
- ▶ PLTSD (shutdown) *before* the DFHDELIM entry.

CICS JCL changes

Add a DD statement for RKANMOD and concatenate the Tivoli OMEGAMON II load library, RKANMOD, to DFHRPL:

```
//RKANMOD DD DISP=SHR,DSN=rhilev.RKANMOD
//DFHRPL DD DISP=SHR,DSN=.....
// DD DISP=SHR,DSN=rhilev.RKANMOD
```

In addition to the foregoing changes, there are some optional JCL changes that you can make:

```
//KOCGLBnn DD DUMMY
```

This instructs OMEGAMON CICS that the Global Data area with the suffix *nn* is to be used for this CICS region:

```
//OCCIREQ DD DUMMY
```

This instructs OMEGAMON to check for the presence of the KOCCI address space. If it is not found, message OC0806 is issued asking the operator if this CICS region is to continue in this case.

It is possible to run more than one copy of a KOCCI or OMEGAMON CICS agent on an LPAR. If this is the case, you must indicate which address space is to monitor this CICS region. This is achieved by placing corresponding DD cards in the CICS region JCL and the KOCCI address space JCL or agent JCL.

If more than one KOCCI is required, place the following card in both the CICS region and the KOCCI JCL:

```
//RKC2XMnn DD DUMMY
```

Where *nn* is a number between 00 and 15. If no card is specified, it is the same as using the number 00.

Only one KOCCI can be active on an LPAR for a given release with any one number.

If more than one agent is required, place the following card in both the CICS region and the AGENT JCL:

```
//RKCPXMnn DD DUMMY
```

Where *nn* is a number between 00 and 15. If no card is specified, it is the same as using the number 00.

Only one agent can be active on an LPAR with any one number.

7.2 OMEGAMON XE for CICS features

OMEGAMON XE for CICS provides a number of distinct features to allow users to monitor their CICS regions. While much of the data provided is displaying the current state of resources or the usage of such resources, some of the features use the collected data to provide real time or near real time summaries of the data that is collected. This provides users with extra insight into the performance of their CICS regions without having to run batch jobs to process large amounts of data.

7.2.1 Resource monitoring

The majority of the workspaces provided by OMEGAMON XE for CICS fall into the category of resource monitoring. That is, they provide information relating to the current state and usage of the resources that are available to the CICS region. This includes resources that are implicit, such as storage, and those that are defined, such as files, programs, and so on.

Resource monitoring primarily allows for the creation of situations, which allow the user to be notified when resources are unavailable, resource consumption has exceeded a certain threshold, or available resources are below safe values. Careful tuning of the situations on a system can help ensure that support staff are notified before the system availability is threatened.

OMEGAMON XE for CICS currently provides information relating to the following resources in a CICS region:

- ▶ Automatic Initiate Descriptors (AIDs)
- ▶ Business Transaction Services process type details
- ▶ Business Transaction Services process details
- ▶ Business Transaction Services containers
- ▶ MRO connections
- ▶ ISC connections
- ▶ IP connections
- ▶ DB2 subsystem
- ▶ DB2 transactions
- ▶ DB2 thread activity
- ▶ Database control for IMS
- ▶ Dispatcher Summary
- ▶ Dispatcher TCB pools
- ▶ Dispatcher TCB modes
- ▶ Dump details
- ▶ Enqueue contention
- ▶ Enterprise Java Corba servers
- ▶ Enterprise Java request models
- ▶ Enterprise Java deployed java programs (DJARs)
- ▶ Exit programs
- ▶ File control datasets
- ▶ CICS region datasets
- ▶ Interval control elements
- ▶ Java programs
- ▶ CICS Journals
- ▶ CICS JVMs
- ▶ JVM™ profiles
- ▶ JVM classcaches
- ▶ Log streams
- ▶ LSR pools
- ▶ MQ status
- ▶ MVS TCBs
- ▶ Recovery manager Unit of work links
- ▶ Storage usage
- ▶ Storage DSA usage
- ▶ Storage Subpool usage
- ▶ System initialization values
- ▶ Transaction classes
- ▶ TCPIP activity
- ▶ TCPIP services
- ▶ Temporary storage
- ▶ Temporary storage queues

- ▶ Terminal storage violations
- ▶ Transactions
- ▶ Transient data
- ▶ Transient data queues
- ▶ Transaction manager
- ▶ UOW disposition
- ▶ UOW enqueues
- ▶ VSAM files
- ▶ VSAM RLS conflicts
- ▶ Web services
- ▶ Web services virtual hosts
- ▶ Web services pipelines
- ▶ Workrequests

In addition to this list, the Region Overview query and workspace provide information from several sources, allowing many critical metrics to be retrieved simultaneously. Region overview provides information like the transaction rate, the percent of maximum tasks, the i/O rate, CPU rate, and other details.

7.2.2 Service level analysis

Service level analysis is a great way to determine if transactions are meeting their performance objectives. The objectives can be either based upon the average response time or by percent of transactions meeting their goal response time.

Service level analysis produces summaries at the LPAR level. The report can be obtained by selecting the CICS group node for an LPAR on the physical tree.

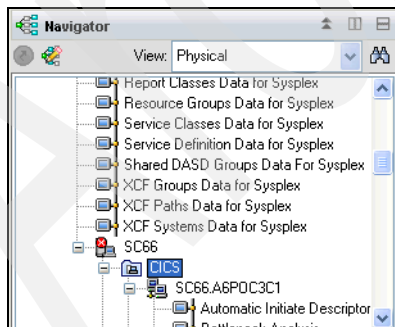
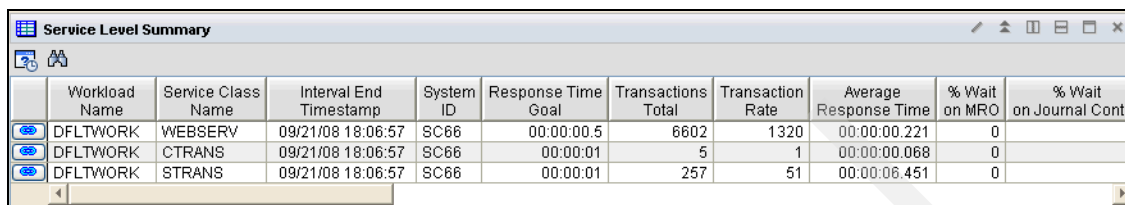


Figure 7-10 CICS Group node

The Service Level summary report appears on the bottom pane of the workspace (Figure 7-11).

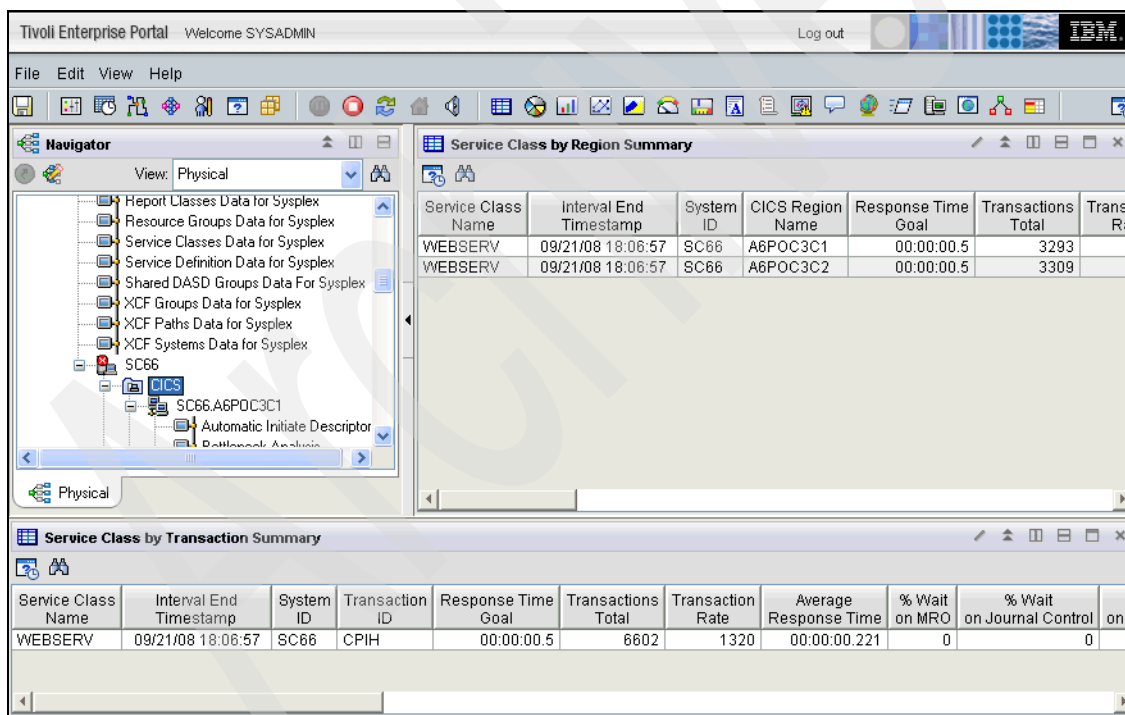


	Workload Name	Service Class Name	Interval End Timestamp	System ID	Response Time Goal	Transactions Total	Transaction Rate	Average Response Time	% Wait on MRO	% Wait on Journal Control
	DFLTWORK	WEBSERV	09/21/08 18:06:57	SC66	00:00:00.5	6602	1320	00:00:00.221	0	
	DFLTWORK	CTTRANS	09/21/08 18:06:57	SC66	00:00:01	5	1	00:00:00.068	0	
	DFLTWORK	STRANS	09/21/08 18:06:57	SC66	00:00:01	257	51	00:00:06.451	0	

Figure 7-11 Service Level Summary report

This report shows the summary for all the transactions in all the CICS regions for an LPAR that match the service Level analysis definition.

Selecting the link to Service Class Summary produces the following workspace (Figure 7-12).



The workspace displays the Tivoli Enterprise Portal interface. On the left is a Navigator pane showing a tree structure of data sources, including 'CICS' and 'SC66.A6POC3C1'. The main area contains two summary reports:

Service Class by Region Summary

Service Class Name	Interval End Timestamp	System ID	CICS Region Name	Response Time Goal	Transactions Total	Transaction Rate
WEBSERV	09/21/08 18:06:57	SC66	A6POC3C1	00:00:00.5	3293	
WEBSERV	09/21/08 18:06:57	SC66	A6POC3C2	00:00:00.5	3309	

Service Class by Transaction Summary

Service Class Name	Interval End Timestamp	System ID	Transaction ID	Response Time Goal	Transactions Total	Transaction Rate	Average Response Time	% Wait on MRO	% Wait on Journal Control	on P
WEBSERV	09/21/08 18:06:57	SC66	CPIH	00:00:00.5	6602	1320	00:00:00.221	0		0

Figure 7-12 Service Class Summary workspace

The Service Class Summary workspace shows two reports. To the right of the panel is the Service Class by Region Summary. This provides a summary of all the CICS regions where any tasks that have been classified in the service class have run.

The lower portion of the panel shows Service Class by Transaction. This shows all the transaction IDs that have run that have been classified in this service class. In this case there is only one, CPIH, but there is no restriction upon the number of transactions that can be part of a service class.

The classification rules used by OMEGAMON XE for CICS to produce the summaries can either be those defined to z/OS Workload Manager or defined in OMEGAMON XE for CICS.

Configuring Service Level Analysis

To configure OMEGAMON XE for CICS, you must first enable the CICS SLA view. This is achieved by adding the CICS SLA view to the assigned views for your user ID:

1. First click in the Administer Users icon in the top left hand corner of the TEP display (Figure 7-13).

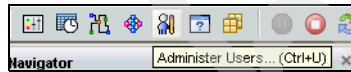


Figure 7-13 Administer Users icon

2. Then select your current USERID and the Navigator Views tab as shown in Figure 7-14 on page 154.
3. Ensure that CICS SLA is in the Assigned View box. If it is not, click **CICS SLA** in the Available Views box and then click the left arrow.

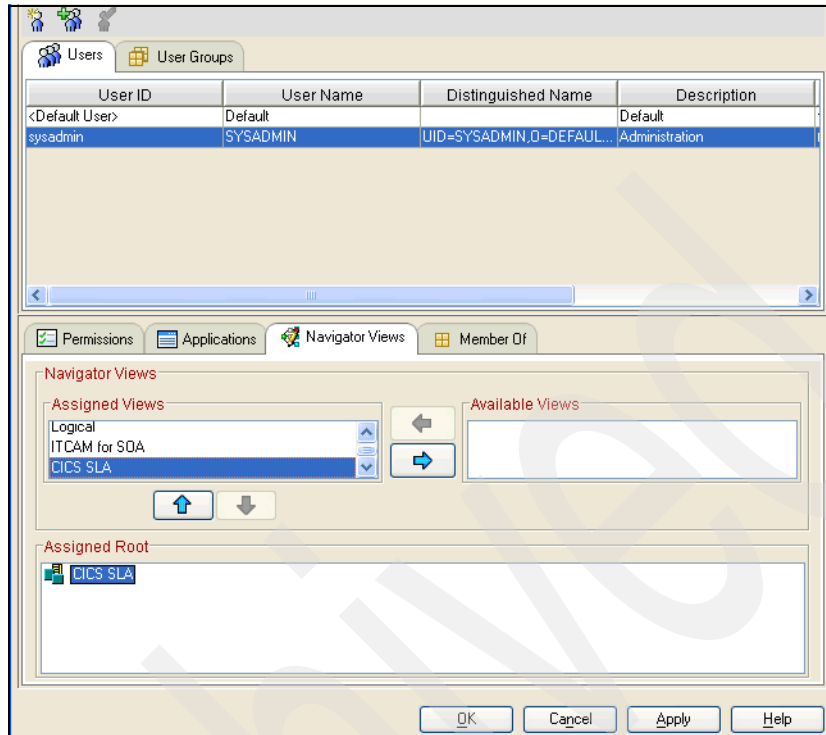


Figure 7-14 Administer Users Navigator Views

- After the CICS SLA view has been added to the user, it should be selectable from the navigator pane in the TEP (Figure 7-15).

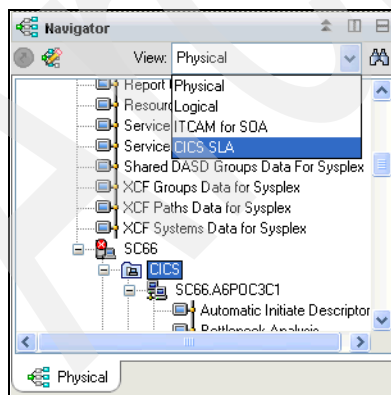


Figure 7-15 Selecting CICS SLA view

The CICS SLA view is shown here (Figure 7-16).

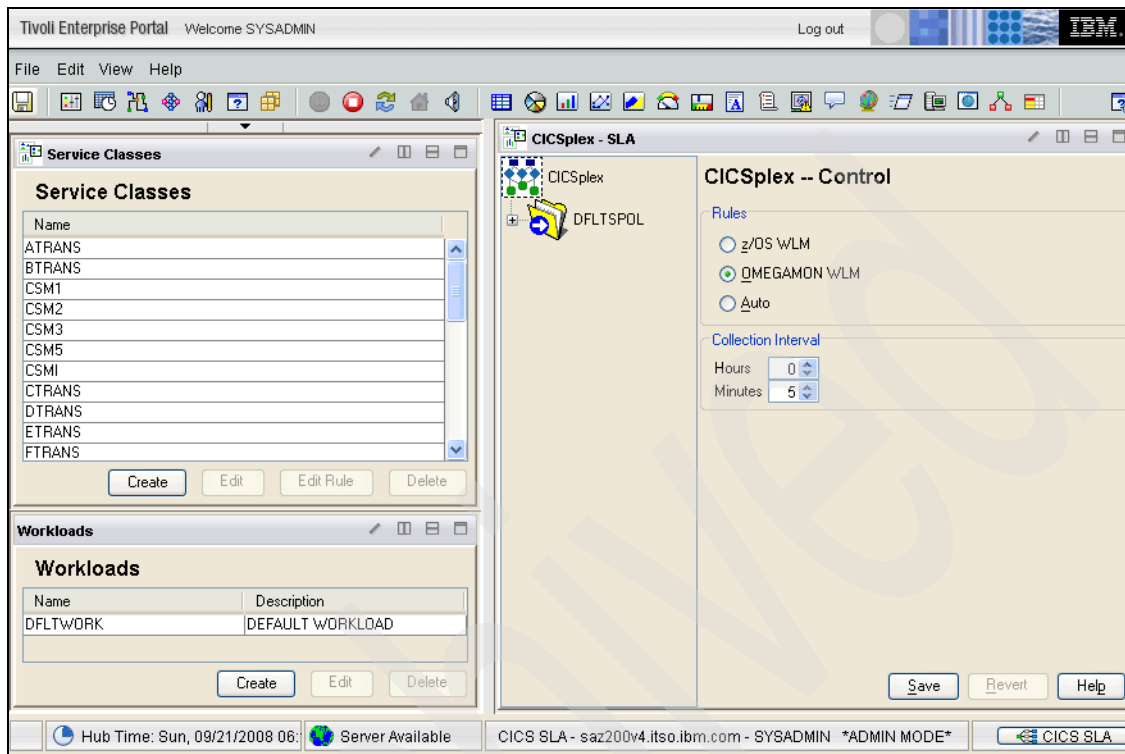


Figure 7-16 CICS SLA view

To the right of the view, we see the CICSplex Control information and the default Service Policy of DFLTSPOL.

In the CICSplex Control, we configure how the classification is to take place.

- ▶ *z/OS WLM* indicates that the WLM assigned service class name will be used to classify the transactions.
- ▶ *OMEGAMON WLM* indicates that the rules defined using this interface will determine the service class names that the transactions are classified in.
- ▶ *Auto* indicates that the z/OS WLM classification will be used if available. If the service class cannot be determined this way, then the OMEGAMON rules will classify the task.

The collection interval determines how often response data for service classes, CICS regions, and individual transactions is summarized and reported via the Service Class Analysis workspace.

A service policy applies to all service classes defined for your enterprise. Service-class goals vary by service policy: Service policies let you override a service class's response-time goals as dictated by your site's varying requirements. For example, you could have one service policy for prime-shift operation, another for nighttime operation, and a third for weekend operation.

To the left of the view you can see service classes and workload groups. A workload groups one or more service classes that you want to monitor as a unit; with it, you can monitor related service classes, highlighting the worst-performing class within the group.

A service class identifies a block of related transactions that share common response-time goals for a single workload; the transactions must be related by transaction name, the user ID that invoked them, the VTAM terminal ID (LU name) that submitted them, the CICS region running them, or any combination thereof.

You can create a service class by following these steps:

1. To define a new Service Class, click the **Create** button in the Service Class pane (Figure 7-17).

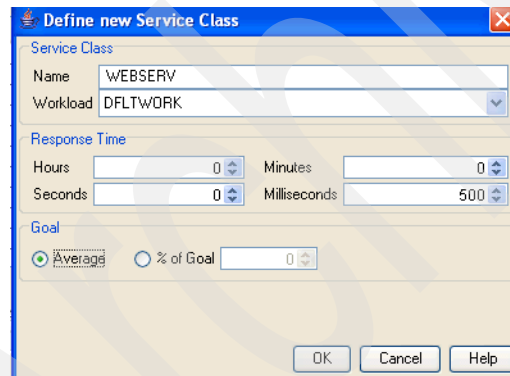


Figure 7-17 Define new Service Class window

2. Within the service-class editor, define a name for your new service class. The name you supply is converted to uppercase. Pull down the list of available workloads, and select the existing workload that you want to associate this service class with.
3. Within the Response Time pane, specify the response time you expect for the CICS transactions associated with this service class.

4. Within the Goal pane, select one of these options:
 - Activating the Average radio button means that the average response time for all matching transactions must at least meet the Response Time specified % of Goal.
 - Activating the % of Goal radio button and specifying a percentage mean that the given percentage of matching transactions must at least meet the Response Time specified.Click **OK** to accept the values.
5. Next you need to create rules that control which transactions are classified as a part of this service class. Highlight the service class and click the **Edit Rule** button (Figure 7-18).

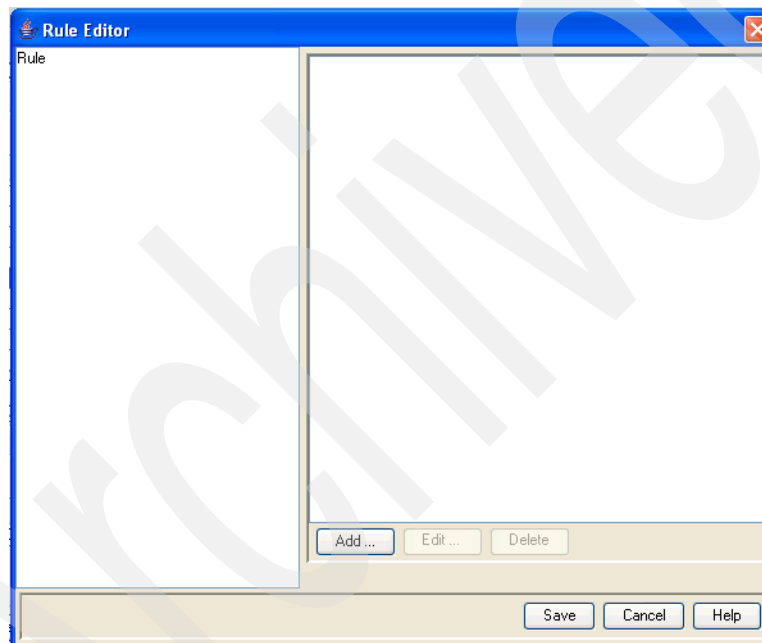


Figure 7-18 Rule definition window

6. Next right-click **Rule** in the left pane (Figure 7-19).

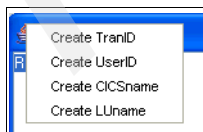


Figure 7-19 Rule popup menu

7. Select one of the following choices:
 - *Create TranID*, to classify based on Transaction ID.
 - *Create UserID*, to classify based on the user ID that is associated with the transaction.
 - *Create CICSname*, to classify based on the JOB name of the CICS region.
 - *Create LUname*, to classify based on the VTAM LU name for the terminal where the transaction originated.
8. A pop-up is then displayed where you can enter a value (Figure 7-20).

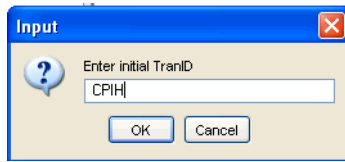


Figure 7-20 pop-up window to enter rule value

9. You can repeat this for other values such as these (Figure 7-21).

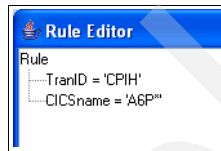


Figure 7-21 completed rules

10. All transactions with transaction ID *CPIH*, running in CICS regions whose application IDs start with the characters *A6P*, will be classified in this service class.

7.2.3 Bottleneck Analysis

Bottleneck Analysis is a useful tool for people concerned with improving the performance of the applications that run on a CICS region. Getting the most performance out of a CICS application can be compared to tuning almost anything, including a car! Extracting the maximum performance involves maximizing the time spent on productive work and minimizing that which puts a strain on the system.

In computer software terms, this means minimizing the time that an application has to wait for something. If an application spends the bulk of its time waiting for a resource such as storage, or LSR buffers, then the application will not be providing the maximum performance. In addition to elongated response times, the transactions will hold resources themselves for longer and therefore potentially add to the contention occurring in the system.

Bottleneck Analysis helps users identify the wait reasons their applications are experiencing. Every task in the system will have a *wait reason* identified. That wait reason might be Running, in which case the task is doing productive work. Bottleneck Analysis has a list of wait reasons that it understands. Bottleneck Analysis runs as a subtask of the KOCCL. It periodically scans all the tasks in the CICS region and accumulates counts for each of the wait reasons. Wait reasons that it knows nothing about are accumulated in an UNKNOWN bucket. If a task is in a wait reason that is turned off in the Bottleneck Analysis table, then it is not counted.

By carefully selecting which wait reasons are active for a given system, it is possible to see those wait reasons that are truly impacting your applications and therefore what resources your applications are waiting for (Figure 7-22).

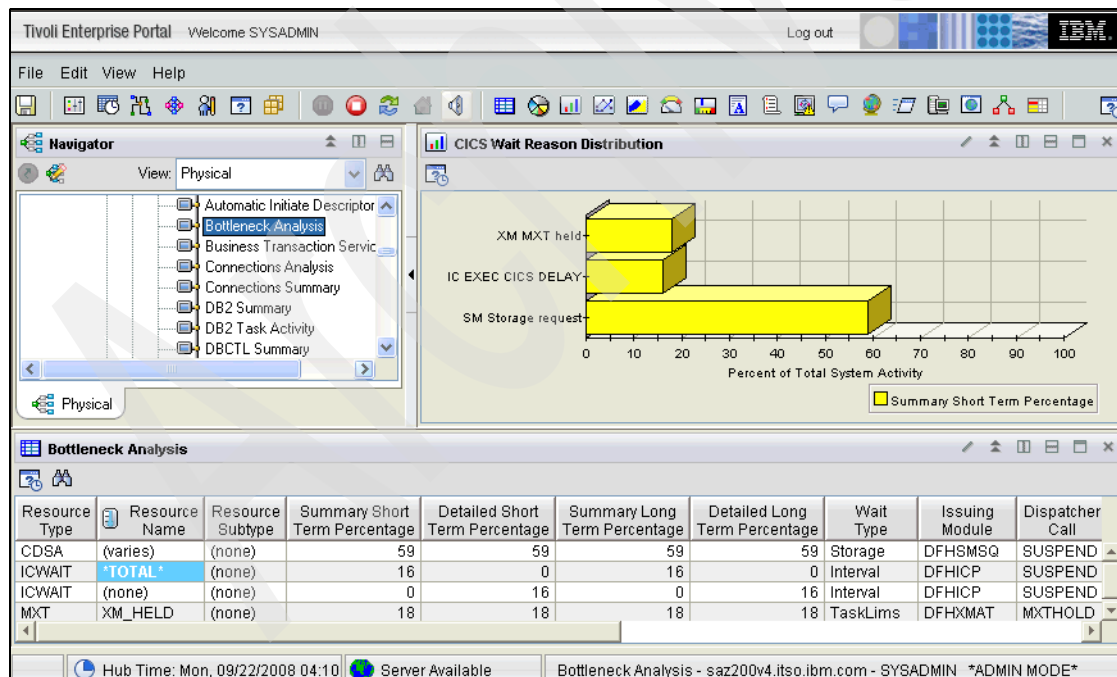


Figure 7-22 Bottleneck Analysis example

In the example in Figure 7-22 on page 159, we can see that there are three wait reasons that have been identified:

- ▶ *IC EXEC CICS DELAY* is an application issue because it indicates that the task is explicitly waiting for a given period of time. There is not much from a performance standpoint that can be done to improve this situation.
- ▶ *XM MXT HELD* is indicating that 18 percent of the recorded wait reasons were for tasks that were delayed because of MAXTASKS. This might be because the value is too low.
- ▶ *SM STORAGE REQUEST* is indicating that nearly 60 percent of the samples showed transactions waiting for storage requests to be honored. This being the largest wait reason, you should examine it first. Possibly the DSALIMIT is too low or the application simply does not fit in one CICS region and some of the workload needs to be spread across other regions.

The same information can be displayed in the menu system as seen in Figure 7-23.

```

ZBPDEX      VTM      A6POC3C1 V560./C SC66 09/22/08 16:15:53
> PF1 Help   PF3 Back   PF4 Main Menu   PF7 Up   PF8 Down   PF11 Zoom

> A-Bottleneck Graph   B-Bottlenecks   C-Group Graph   D-Group Bottlenecks
> E-Impact Analysis     F-Impact Profile   G-Impact Detail   H-Enqueues
=====
>
      BOTTLENECKS

PDEX
+
+ Resource      Resource
+ Type          Name
+ -----
+
+              % 0 _____ 50 _____ 100
+ CDSA          (varies) 59 |-----> . . . .| 59 |-----> . . . .|
+ ICWAIT        *TOTAL* 16 |--> . . . .| 16 |--> . . . .|
+              (none) (16) |--> . . . .| (16) |--> . . . .|
+ MXT           XM_HELD 18 |--> . . . .| 18 |--> . . . .|
+
+
+              Samples . . . : 43318      Samples . . . : 43318
+              Elapsed . . . : 5:59 MN      Elapsed . . . : 5:59 MN
+              Interval . . . : 10:00 MN      Interval . . . : 30:00 MN
=====

```

Figure 7-23 Bottleneck Analysis in the Menu system

Control of Bottleneck Analysis

Bottleneck Analysis is controlled via the global data area. The enablement of the feature is controlled in the <STARTUP_CONTROL> section as follows:

```
*
<STARTUP_CONTROL>
*
  BOTTLENECK_ANALYSIS=AUTO
```

The BOTTLENECK_OPTIONS section specifies the parameters for the subtask:

```
*
<BOTTLENECK_OPTIONS>
*
  CLEAR_INTERVAL_LONG=30
  CLEAR_INTERVAL_SHORT=10
  SAMPLE_INTERVAL=20
  VARIABLE_BUCKETS=1000
  EXCLUDED_TRANS=(CSSY,CSJC,CVST,CSSX,CSGX,CSNC,DSNC,CFQ*,KD4*)
```

The *CLEAR_INTERVAL_LONG* and *CLEAR_INTERVAL_SHORT* control the timespan for the long and short term accumulations. By providing long and short term displays it makes it possible to determine if a particular wait reason is a sudden change in the profile of an application or a longer term trend.

The *SAMPLE_INTERVAL* value specifies in tenths of a second how often the subtask is to sample the active CICS region. Bottleneck Analysis can be a heavy user of CPU. The interval by default is set to 2 seconds. However, in a busy system, this might be too high. You should try to increase this value while still attempting to provide a statistically valid sample. It would serve no purpose to make the interval such that only a few samples were accumulated for each wait reason.

The *VARIABLE_BUCKETS* value specifies how many slots are reserved to wait reasons that have a variable portion to their name. Bottleneck Analysis does not expand the slots available, so if a value of 1000 is specified and more variable wait reasons than this are encountered, some information will be lost. The default here is normally sufficient.

The *EXCLUDED_TRANS* keyword specifies those transactions that are to be ignored by Bottleneck Analysis. It serves no purpose to collect information on system tasks and background server transactions that are permanently active, unless that is the workload you are interested in tuning, of course.

The last item of control is the list of wait reasons. This can be seen by using the menu system option O.J. (Figure 7-24).

ZCDLST VTM CIWSS3C2 V560./C SC66 09/22/08 17:03:39						
> PF1 Help	PF3 Back	PF4 Main Menu	PF7 Up	PF8 Down	PF11 Zoom	
> A-RTA On	B-RTA Off	C-RTA Status	D-RTA Intervals	E-RTA Scaling		
> F-ONDV On	G-ONDV Off	H-ONDV Status	I-Bottleneck Ctl	J-Wait Reasons		
> K-INTR Ctl	L-IANL On	M-IANL Off	N-IANL Settings	O-IANL Groups		
> P-Collection	Q-Shutdown	R-RLIM On	S-RLIM Off	T-RLIM Status		
> U-SMF Status	V-ATF Filters	W-ATF Status				
=====						
>	CONTROL BOTTLENECK ANALYSIS WAIT REASON BUCKETS					
BLST						
+ BLST	On/	Resource	Resource	Issuing	Wait Reason	Wait
+ ID	Off	Type	Name	Module	Description	Type
+ ---	---	---	---	---	---	---
: SY1W	OFF	(none)	(none)	DFHDUIO	DU: Dump dataset I/O	Systasks
: SY2W	OFF	(none)	(none)	DFHTISR	TI: Timer service rq	Systasks
: RMSL	ON	(none)	(none)	DFHRMSL7	RM: Keypoint process	Systasks
: ZNAC	ON	(none)	(none)	DFHZNAC	ZC: Terminal error	Systasks
: DLCN	ON	(none)	DLCNTRL	DFHDBCT	DBCTL: Work element	DBCnt1

Figure 7-24 Menu system wait reason control

Certain wait reasons are turned off for Bottleneck Analysis monitoring by default. If you determine that a wait reason is active and that you would like to disable it, you can achieve this temporarily by changing its On/Off value dynamically. To disable it on a more permanent basis, change the BOTTELENECK_ANALYSIS section in the global as follows:

```
*
<BOTTELENECK_ANALYSIS>
*
DSDF=NO
SODM=NO
SMRE=NO
```

7.2.4 Transaction history

It is often very useful to look back at recent transactions to see in detail the response time for a particular transaction, or possibly look at resource consumption for an individual task. While CICS SMF data provides the detailed information for a transaction, it can be a cumbersome task to gain access to recent SMF data and run an ad-hoc report.

OMEGAMON XE for CICS provides a feature called Transaction History, also known as Online Data Viewing (ONDV). This feature creates a datastore for each CICS region and manages the space to hold as much transaction data as possible. Comprehensive information about a transaction together with detailed file and database requests are collected and stored by Transaction History.

Transaction History can be selected by using the Online Data Viewing workspace under Transactions Analysis for a CICS region in OMEGAMON XE for CICS (Figure 7-25).

System ID	CICS Region Name	CICS Version	End Time	Transaction ID	Task Number	Terminal ID	Transaction Type	User ID	Program ID	CPU Time
SC66	CIWSS3C2	6.5.0	09/22/08 18:11:39	CSOL	00004	n/a	TRM	CIWS3D	DFHSOL	00:00:00
SC66	CIWSS3C2	6.5.0	09/22/08 17:40:11	CSOL	00004	n/a	TRM	CIWS3D	DFHSOL	00:00:00
SC66	CIWSS3C2	6.5.0	09/22/08 17:37:20	CWBG	79601	n/a	SYS	CIWS3D	DFHWBGB	00:00:00
SC66	CIWSS3C2	6.5.0	09/22/08 17:08:44	CSOL	00004	n/a	TRM	CIWS3D	DFHSOL	00:00:00.35
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDR	78882	n/a	TRM	PRIVAT01	DFHPIDSH	00:00:00.01
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDR	78877	n/a	TRM	PRIVAT01	DFHPIDSH	00:00:00.01
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDR	78881	n/a	TRM	PRIVAT01	DFHPIDSH	00:00:00.01
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDS	79289	n/a	TRM	USERWS01	DFHPIAP	00:00:00
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDS	79288	n/a	TRM	USERWS01	DFHPIAP	00:00:00
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDR	78873	n/a	TRM	PRIVAT01	DFHPIDSH	00:00:00.01
SC66	CIWSS3C2	6.5.0	09/22/08 16:45:12	ORDS	79287	n/a	TRM	USERWS01	DFHPIAP	00:00:00

Figure 7-25 Transaction history display in TEP

The Menu system provides an equivalent display (Figure 7-26).

```
> PF1 Help      PF3 Back      PF4 Main Menu    PF7 Up      PF8 Down      PF11 Zoom

>      A-History Record View      B-History Record Selection
>      C-Trace Record View        D-Trace Filters Management

=====
>      HISTORICAL TRANSACTION OVERVIEW

ONDV
+      Transaction Overview: 09/22/08
+
+      End Time      Tran ID      Task Num      Term ID      Type      CPU Time      Resp Time      Storage      File      Term      Abend
+      Time          ID          Num          ID          Type      Time          Time          HWM          Reqs      I/O      Code
+
+      16:03:09      CPIH      15713      n/a      TRM      .2 2.046357      33335K      3      0
+      16:03:09      CPIH      15706      n/a      TRM      .2 1.980787      33335K      3      0
+      16:03:09      CPIH      15711      n/a      TRM      .2 1.828366      33335K      3      0
+      16:03:09      CPIH      15721      n/a      TRM      .2 1.544157      33335K      3      0
+      16:03:09      CPIH      15720      n/a      TRM      .2 1.513379      33335K      3      0
+      16:03:08      CPIH      15704      n/a      TRM      .2 1.558967      33335K      3      0
+      16:03:08      CPIH      15699      n/a      TRM      .2 1.588833      33335K      3      0
+      16:03:08      CPIH      15696      n/a      TRM      .2 2.098515      33335K      3      0
+      16:03:08      CPIH      15695      n/a      TRM      .2 2.027101      33335K      3      0
+      16:03:08      CPIH      15694      n/a      TRM      .2 2.195676      33335K      3      0
+      16:03:08      CPIH      15679      n/a      TRM      .2 2.673689      33335K      3      0
+      16:03:08      CPIH      15689      n/a      TRM      .2 2.423562      33335K      3      0
+      16:03:08      CPIH      15690      n/a      TRM      .2 2.392998      33335K      3      0
+      16:03:08      CPIH      15684      n/a      TRM      .2 2.577388      33335K      3      0
+      16:03:08      CPIH      15686      n/a      TRM      .2 2.372017      33335K      3      0
```

Figure 7-26 Transaction History in the Menu system

The Menu system also allows for the display of comprehensive detail information relating to a transaction. Selecting one of the tasks in the previous display via the zoom key (PF11) provides the following display (Figure 7-27 and Figure 7-28).

ZZONDV VTM A6POC3C1 V560./C SC66 09/22/08 18:43:48			
> PF1 Help	PF3 Back	PF4 Main Menu	PF7 Up PF8 Down
=====			
> HISTORICAL TRANSACTION DETAIL			
ONDV 08			
+ Task Detail Information			
+ General Information			
+ Transaction ID	CPIH	Task number	15696
+ Userid	CICSUSER	Luname	None
+ Facility ID (local) . . .	n/a	Facility type (local) . .	Term
+ Real transaction ID . . .	CPIH	Umbrella transaction ID .	None
+ Program ID - first . . .	DFHPIDSH	Umbrella program	None
+ Abend code	None		
+ Time Statistics			
+ CPU time	0.173600	Overall elapsed time . .	2.098515
+ Dispatch time	0.267648	Total wait time	1.830848
+ Re-dispatch wait time . .	0.132016	Exception wait time . . .	0.000000
+ TS VSAM I/O wait time . .	0.000000	TD VSAM I/O wait time . .	0.000000
+ File I/O wait time . . .	0.000000	JC I/O wait time	0.000000
+ TC I/O wait time	0.000000	MRO wait time	0.000000
+ 1st dispatch delay time .	0.000112	Transaction class delay .	0.000000
+ Max tasks delay	0.000000	Local ENQ delay	0.000000
+ LU61 wait time	0.000000	LU62 wait time	0.000000
+ FEPI wait time	0.000000	RMI elapsed time	0.000016
+ RMI suspend time	0.000000	RLS file I/O wait time .	0.000000
+ Syncpoint elapsed time .	0.000656	Lock manager delay . . .	1.288544
+ WAIT EXTERNAL wait time .	0.000000	WAITCICS and WAIT EVENT .	0.000000
+ Interval control wait . .	0.000000	Dispatchable wait time .	0.000000
+ Shared TS I/O wait time .	0.000000	RLS CPU time	0.000000
+ IMS wait time	0.000000	DB2 Readyq wait time . .	0.000000
+ DB2 Connection wait time .	0.000000	DB2 wait time	0.000000
+ SOCKET I/O wait time . .	0.505376	Global ENQ delay	0.000000
+ RRMS/MVS wait time . . .	0.000000	MAXOPENTCBS delay time .	0.000000
+ JVM elapsed time	0.000000	JVM suspend time	0.000000
+ QR TCB wait-for-dispatch .	0.012592	QR TCB elapsed time . . .	0.017008
+ QR TCB CPU time	0.000720	Other TCBs elapsed time .	0.006112
+ Other TCBs CPU time . . .	0.005984	JVM(J8) TCB CPU time . .	0.000000
+ LE(L8) TCB CPU time . . .	0.166880	SS(S8) TCB CPU time . .	0.000000
+ Program fetches wait . . .	0.000000	Wait for a JVM TCB . . .	0.000000
+ JVM initialisation time .	0.000000	JVM reset time	0.000000
+ Key 8 TCB elapsed time . .	0.244512	Key 8 TCB CPU time . . .	0.166880
+ Key 9 TCB elapsed time . .	0.000000	Key 9 TCB CPU time . . .	0.000000
+ J9 TCB CPU time	0.000000	Wait for H8 TCB time . .	0.000000
+ RO TCB elapsed time . . .	0.000000	RO TCB CPU time	0.000000
+ TCB mismatch time	0.000000	TCB change mode delay . .	0.036784
+ TCB create delay	0.000000	3270 Partner wait time .	0.000000
+ General Statistics			
+ Primary term input msgs .	0	Primary term output msgs:	0
+ Primary term input chars:	0	Primary term output chars:	0
+ Sec LU61 input msgs . . .	0	Sec LU61 output msgs . .	0
+ Sec LU61 input chars . . .	0	Sec LU61 output chars . .	0

Figure 7-27 Menu system Task History detail

+	Sec LU62 input msgs . . .	0	Sec LU62 output msgs . .	0
+	Sec LU62 input chars . .	0	Sec LU62 output chars . .	0
+	TD gets	0	TD puts	0
+	TD purges	0	TS gets	0
+	TS puts to aux	0	TS puts to main	0
+	Shared TS wait count . .	0	PC links	11
+	PC loads	0	PC xctls	0
+	JC writes	0	IC starts	0
+	Syncpoint requests . . .	1	BMS requests	0
+	BMS map requests	0	BMS in requests	0
+	BMS out requests	0	CICS logger writes . . .	0
+	PC link URMs	0	IC requests	0
+	3270 bridge tran ID . . .	n/a	TS total requests	0
+	DPL requests	0	IMS/DBCTL requests . . .	0
+	DB2 requests	0	OO class requests	0
+	SSL bytes encrypted . . .	0	SSL bytes decrypted . . .	0
+	CICS TCBs attached . . .	0	TCB Mode Switches	110
+	WEB Receive requests . .	1	WEB Characters received .	0
+	WEB Send requests	0	WEB Characters sent . . .	0
+	WEB total request count .	12	WEB Repository READs . .	0
+	WEB Repository WRITES . .	0	Local container bytes . .	0
+	Remote IC channel starts:	0	Remote IC channel data .	0
+	Total channel requests .	172	Browse channel requests .	0
+	Get container requests . .	80	Put container requests . .	88
+	Move container requests .	4	Total bytes for GETs . . .	36710198
+	Total bytes for PUTs . . .	31465325	DPL CHANNEL data bytes .	0
+	DPL RETURN CHNL bytes . .	0	LINK with CHANNEL	2
+	XCTL with CHANNEL	0	DPL with CHANNEL	0
+	RETURN with CHANNEL . . .	0	RETURN CHNL bytes	0
+	Client IP address	10.1.100.43		
+	Tran Group ID (char)USIBMSC.A6POC3C1C....zc..		
+	Tran Group ID (hex) . . .	11EECCDEC4CFDDCFCC0126AA800		
+		904292423B1676333139DD8F9300		
+	TRGID: X'1910E4E2C9C2D4E2C34BC1F6D7D6C3F3C3F1C3091D2D68AFA9830000'			
+				
+		Storage Statistics		
+	Getmains <16M	0	Getmains >16M	56
+	HWM <16M	0	HWM >16M	32553K
+	Occupancy <16M	OK	Occupancy >16M	7570M
+	HWM of total pgm storage:	70K		
+	HWM of pgm storage <16M .	0	HWM of pgm storage >16M .	70K
+	HWM pgm storage cdsa . . .	0	HWM pgm storage ecdsa . .	0
+	HWM pgm storage rdsa . . .	0	HWM pgm storage erdsa . .	40K
+	HWM pgm storage sdsa . . .	0	HWM pgm storage esdsa . .	30K
+				
+		Application Trace Active		
+		File Control Statistics		
+	Local Browsers	0	Local Gets	3
+	Local Adds	0	Local Puts	0
+	Local Deletes	0	Total Local Requests . . .	3
+	Local VSAM calls	3	Total Remote Requests . .	0
+	Total Requests	3		
+				
+		Umbrella Data		
+	User work area (char) . . .	n/a		
+		Unit-of-work Information		
+	Netname	USIBMSC.A6POC3C1....		
+	CICS token info (char)n....		
+	CICS token info (hex) . . .	012B9100		
+		9DD35001		

Figure 7-28 Menu system Task History detail (continued)

As can be seen from the foregoing figures, a large amount of data is available for each task. In addition, when the File Control Statistics heading is highlighted, it indicates that it is possible to zoom for more details (Figure 7-29).

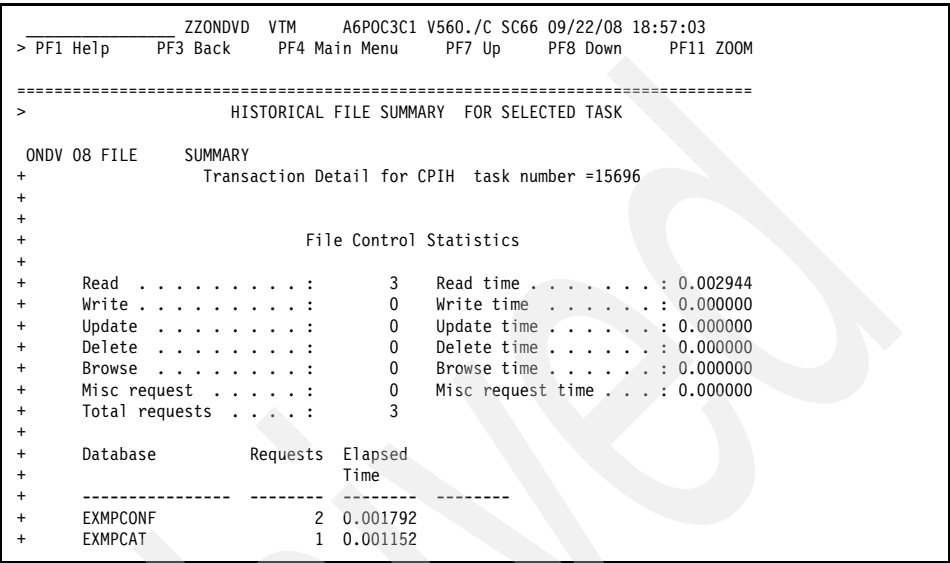


Figure 7-29 File summary for a task.

This display shows that the transaction accessed two VSAM files. Again, more details can be obtained for each file as follows (Figure 7-30).

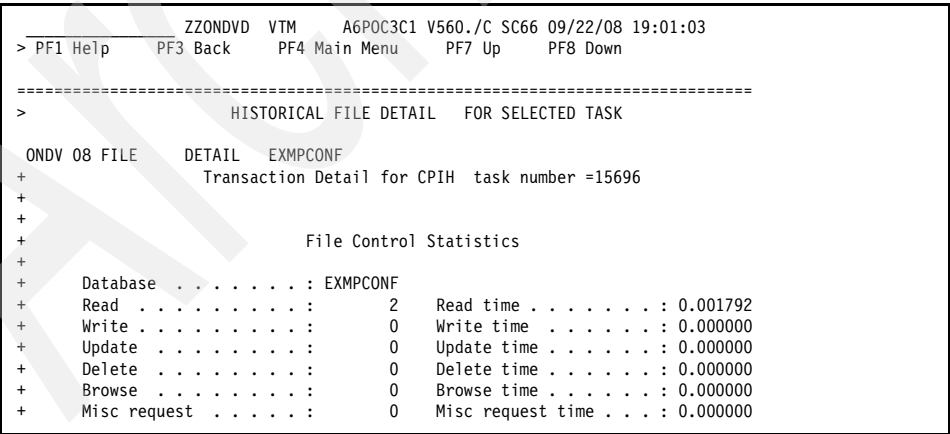


Figure 7-30 File detail for a task

This level of detail shows, for each file accessed, the type of request and the response time to the application for those requests.

Control of Transaction History

Transaction History is controlled via the global data area. The enablement of the feature is controlled in the <STARTUP_CONTROL> section as follows:

```
*
  <STARTUP_CONTROL>
*
  ONLINE_DATA_VIEWING=AUTO
```

The ONLINE_VIEWER section specifies the parameters for the subtask possible configuration values are:

```
*
  <ONLINE_VIEWER>
  DATA_STORE_TYPE=FILEOCMP
  DATA_STORE_FILE_NAME=OMEGAXE.SC66.*.RKC2HIST
  EXCLUDED_TRANS=(KD40,KD4C,KD4,KD4D,CSSY)
```

```
*
  or
*
  <ONLINE_VIEWER>
  DATA_STORE_TYPE=DSPACE
  DATA_STORE_SIZE=956
  RESERVED_SIZE=25
  EXCLUDED_TRANS=(KD40,KD4C,KD4,KD4D,CSSY)
```

The *DATA_STORE_TYPE* keyword specifies how the data is to be stored. There are two options:

- ▶ **FILEOCMP** is where the data is stored to a VSAM Linear dataset. This option allows the data to persist if the CICS region or the KOCCL is shut down.
- ▶ **DSPACE** is where a dataspace is allocated to the KOCCL address space to hold the data. This option does not persist the values.

The *DATA_STORE_FILE_NAME* is only applicable to a type of FILEOCMP. If the name provided contains an asterisk as in the example above, the asterisk is replaced with the JOB name of the CICS region.

DATA_STORE_SIZE and *RESERVED_SIZE* are only applicable to DSPACE. The size refers to the size of the dataspace in kilobytes, and the reserved size is the percentage of the space that is reserved for file details and application trace information.

The *EXCLUDED_TRANS* keyword is applicable to both types and specifies transactions that are not to be recorded to history.

Application Trace

In addition to file and database detail statistics OMEGAMON offers the capability to trace the application calls made by a transaction. OMEGAMON traces calls made via the EXEC interface, the resource manager interface, and from third party database providers such as ADABAS, SUPRA, DATACOM, and IDMS.

The Application Trace workspace can be linked to, in OMEGAMON XE for CICS workspaces that show transaction History data. These are the Online Data Viewing, Units of Work, and the Web Services Transactions workspaces. The Application Trace workspace is shown in Figure 7-31.

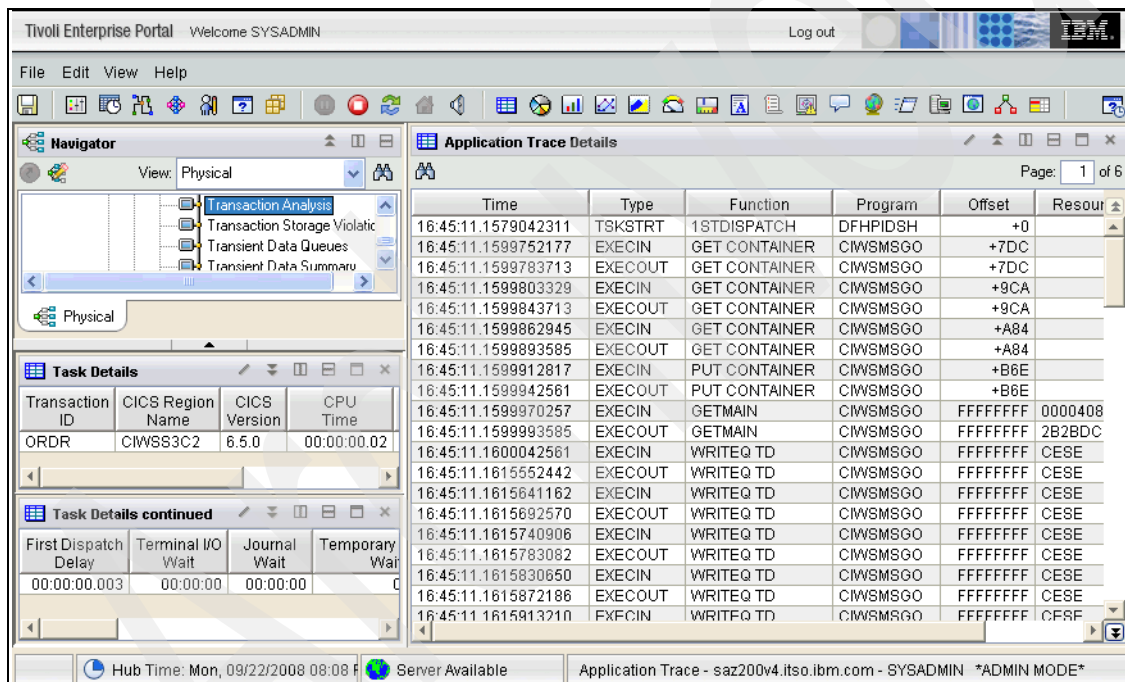


Figure 7-31 Application Trace workspace

This workspace displays the trace data for a task to the right of the panel. To the left are details from the transaction history record.

Similar information is displayed in the Menu system (Figure 7-32). The following items can be displayed by pressing the zoom key (PF11) on the Application Trace Active heading in the Transaction Detail display. Zooming on the task from the Transaction History Trace record view also displays Application Trace (H.C).

ZZONDA VTM CIWSS3C2 V560./C SC66 09/22/08 20:12:29							
> PF1 Help PF3 Back PF4 Main Menu PF7 Up PF8 Down							
=====							
> TRACED TRANSACTION SUMMARY							
ONDV 05 TRACE SUMMARY							
Transaction Detail for ORDR task number =78877							
+							
+							
+							
Application Trace Facility							
+ Lines 1 to 243 of 55							
+ Trace Program Offset Function Resource Response Elapsed							
+ Type							
+-----+-----+-----+-----+-----+-----+-----							
+	TSKSTR	DFHPIDSH	0	1ST DISPATCH			
+	EXECIN	CIWSMSG0	7DC	GET CONTAINER			0.00006
+	EXECOUT	CIWSMSG0	7DC	GET CONTAINER		NORMAL	0.00000
+	EXECIN	CIWSMSG0	9CA	GET CONTAINER			0.00000
+	EXECOUT	CIWSMSG0	9CA	GET CONTAINER		NORMAL	0.00000
+	EXECIN	CIWSMSG0	A84	GET CONTAINER			0.00000
+	EXECOUT	CIWSMSG0	A84	GET CONTAINER		NORMAL	0.00000
+	EXECIN	CIWSMSG0	B6E	PUT CONTAINER			0.00000
+	EXECOUT	CIWSMSG0	B6E	PUT CONTAINER		NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	GETMAIN	00004080		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	GETMAIN	2824DC58	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	FFFFFF	WRITEQ TD	CESE		0.00000
+	EXECOUT	CIWSMSG0	FFFFFF	WRITEQ TD	CESE	NORMAL	0.00000
+	EXECIN	CIWSMSG0	C52	GET CONTAINER			0.00000
+	EXECOUT	CIWSMSG0	C52	GET CONTAINER		NORMAL	0.00000

Figure 7-32 Application Trace Menu System display

Though the OMEGAMON CICS Application Trace feature provides valuable information, it incurs high overhead. We recommend that it be turned on dynamically when required. Application Trace can be controlled via the Control section of the Menu System. Specifically, menu option O.W can be used to activate and de-activate the trace. Menu option O.V can be used to specify trace filters. These allow for the trace to be active only for certain transactions.

7.2.5 CICS Web Services monitoring

OMEGAMON XE for CICS provides CICS Web Services monitoring as part of 7.2.1, “Resource monitoring” on page 149. The primary workspace for monitoring is the Web Services branch under a CICS region. An example of the Web Services workspace is shown in Figure 7-33.

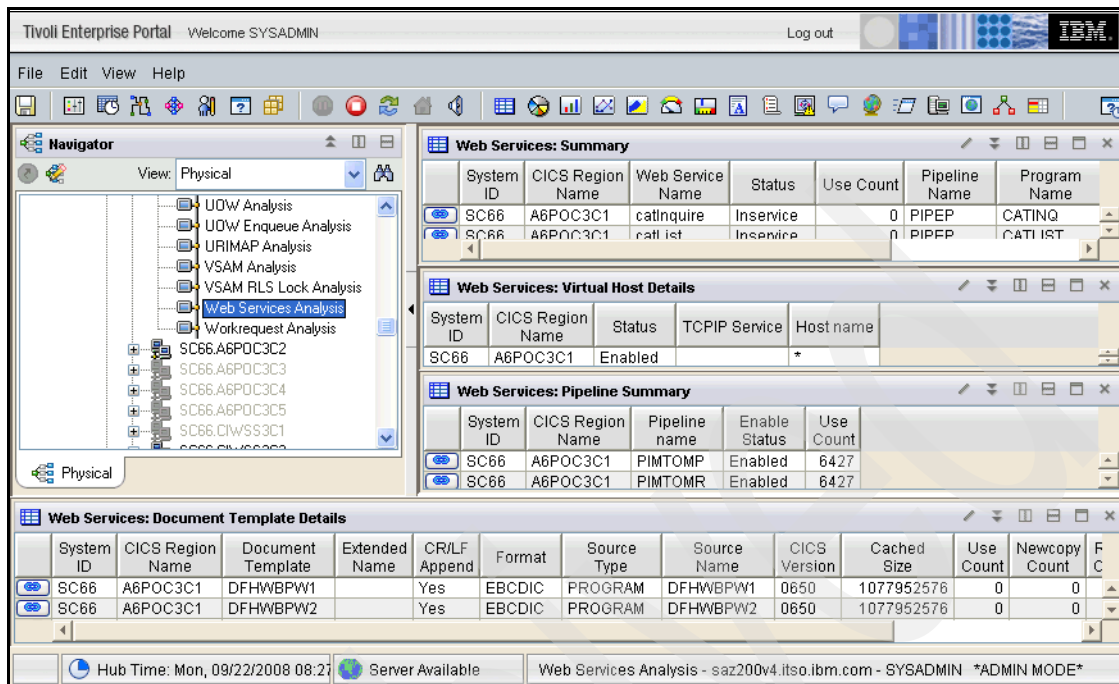


Figure 7-33 Web Services workspace

From this workspace you can see summary information relating to the Web Services activity in a particular CICS region. This workspace shows the following resources defined in CICS:

- ▶ Web Services
- ▶ Virtual Hosts
- ▶ Pipelines
- ▶ Document Templates

Most of these resources contain links to more details:

- ▶ For Document Templates, the link displays more details about the resource where the source is located.
- ▶ For Pipelines, the link shows more details about the Pipeline including more static information such as the location of the Web Services configuration file and WS binding directory.
- ▶ For Web Services, there are a number of options. Details of the Pipeline, the URIMAP and the Web Service. Also, there is a link to Web Service Transactions and another to all Web Service Transactions. This shows transaction details about recent transactions that were dispatched to process a service provider Web Service.

Figure 7-34 shows an example of the Web Services: Transactions workspace.

The screenshot shows the 'Web Services: Transactions' workspace. The Navigator pane on the left lists various analysis options, with 'Web Services Analysis' selected. The main pane displays a table of transaction data. The 'Web Service Details' pane at the bottom provides a detailed view of the selected transaction.

Web Service Name	Web Service Operation	Response Time	CPU Time	End Time	Transacti ID
imageServ...	imageServ...	00:00:01.129	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.177	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.125	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.046	00:00:00.06	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:00.998	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.072	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.011	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.011	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:02.039	00:00:00.05	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.03	00:00:00.07	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.952	00:00:00.06	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.951	00:00:00.05	09/29/08 10:51:04	CPIH

CICS Region Name	Web Service Name	System ID	Use Count	Pipeline Name	Program Name	URIMAP Name	Container Name	Last Modified	Program Interface	Status	Validatic Indicator
A6POC3C2	imageServ...	SC66	19295	PIMTOMP	IMGSRV	\$042380	IMGSRV-DATA	09/04/08 16:42:38	Channel	Inservice	No

Figure 7-34 Web Services: Transactions Workspace

OMEGAMON XE for CICS is currently unable to determine automatically if a transaction is running as a Web Service provider. For this workspace to be of value, OMEGAMON must be informed of the Web Service name and Operation at some point in the Web Services transaction. OMEGAMON provides an interface for providing extra information about a transaction to OMEGAMON. This is known as Umbrella services and is implemented via the KOCRMCLL and KOCGLCLL callable programs. Details of using this interface can be located in the KOCAPITX member of the TKANSAM library.

To use Umbrella services to inform OMEGAMON of the Web service details, make an Umbrella services request with request type OC@API_SOA_WRITE that passes an area mapped by the KOCSOA macro provided in the TKANMAC library.

The KOCSOA macro defines the Web Service name as a 32-character field. This should contain the first 32 characters of the Web Service name padded with blanks.

The Operation name is a 64 character field and should contain the first 64 characters of the Web Service operation, again padded with blanks.

The sample code in Figure 7-35 can be used to report these values to OMEGAMON. This example shows how to request the information about the Web Services name and operation from CICS. Because this code is intended for use where the CICS API is available, the KOCRMCLL interface is used. This does require that DATAREG on the DFHEIENT macro is either set to register 13 or allowed to default to that value.

```

***-----***
**/* CICS STORAGE
***-----***
DFHEISTG DSECT
FUNCTION DS CL16
LENGTH DS F
REQUEST DS F
KOCCALL CALL ,(0,0),MF=L
          KOCSOA ,
          EJECT
***-----***
**/* PROGRAM START
***-----***
OMEGPROG DFHEIENT
***-----***
**/* GET THE FIRST 32 BYTES OF WSNAME
***-----***
          MVC MN#WSNAME,SPACES
          MVC LENGTH,=A(L'MN#WSNAME)
          EXEC CICS GET CONTAINER('DFHWS-WEBSERVICE') *
                INTO(MN#WSNAME) FLENGTH(LENGTH) *
          NOHANDLE
          CLC EIBRESP,DFHRESP(NORMAL)
          BE GETOPER
          CLC EIBRESP,DFHRESP(LENGERR)
          BNE EXIT
***-----***
**/* AND THE FIRST 64 OF THE OPERATION
***-----***
GETOPER DS OH
          MVC MN#OPERATION,SPACES
          MVC LENGTH,=A(L'MN#OPERATION)
          EXEC CICS GET CONTAINER('DFHWS-OPERATION') *
                INTO(MN#OPERATION) FLENGTH(LENGTH) *
          NOHANDLE
          CLC EIBRESP,DFHRESP(NORMAL)
          BE TELLOMEG
          CLC EIBRESP,DFHRESP(LENGERR)
          BNE EXIT
***-----***
**/* NOW REPORT TO OMEGAMON
***-----***
TELLMEG DS OH
          MVC REQUEST,=A(OC@API_SOA_WRITE)
          CALL KOCRMCLL,(REQUEST,MN#SOA),VL,MF=(E,KOCCALL)
          EXIT DFHEIRET
***-----***
**/* STATIC DATA
***-----***
SPACES DC CL64' '

```

Figure 7-35 Sample code to report SOA data to OMEGAMON

OMEGAMON also provides a sample pipeline handler program KOCSOAP that can be used to report the SOA data without having to change any application code. It does require that the pipeline configuration file be modified to specify the pipeline handler program. The following pipeline definition file shows how to ensure the OMEGAMON provided handler is invoked for the Web Services provided by this pipeline (Figure 7-36).

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline"
..... xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
..... xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline provider
  <service>
    <service_handler_list>
      <handler>
        <program>KOCSOAP</program>
        <handler_parameter_list> </handler_parameter_list>
      </handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

Figure 7-36 Pipeline Configuration file with OMEGAMON handler program.

7.2.6 Resource Limiting

Resource Limiting (RLIM) is a special feature provided by OMEGAMON CICS. This feature is designed to automatically protect the environment from rogue CICS transactions that might be looping in a way that cannot be detected by the CICS runaway task protection, or transactions that use extraordinary amounts of certain resources.

Resource limiting examines the values for certain resources or the number of certain types of requests. If the thresholds specified have been exceeded OMEGAMON either issues a message or causes the transaction to abnormally terminate (ABEND).

The enablement of the feature is controlled in the <STARTUP_CONTROL> section as follows:

```
*
  <STARTUP_CONTROL>
*
  RESOURCE_LIMITING=AUTO
  RESOURCE_LIMITING_MSG_DEST=TDQ
  RESOURCE_LIMITING_SYSTEM_TASKS=NO
  RESOURCE_LIMITING_ABEND_CANCEL=YES
```

The option RESOURCE_LIMITING_MSG_DEST specifies if the message is to be written to the transient data queue CSSL (TDQ) or to the system console (LOG).

RESOURCE_LIMITING_SYSTEM_TASKS allows you to specify if resource limiting is to issue warning messages for CICS system transactions. OMEGAMON will not cause a system task to ABEND.

RESOURCE_LIMITING_ABEND_CANCEL allows you to specify whether any application abend handling exits are to be honored when RLIM elects to ABEND a transaction. Specifying NO indicates that ABEND exits will not be cancelled and therefore will remain in effect.

The <RESOURCE_LIMITING> section of the global specifies the limits that are to be in effect for a given transaction ID:

```
*
  <RESOURCE_LIMITING>
*
  <<CPU>>
  INCLUDED_TRANS=(TRLN,DE*)
  KILL_LIMIT=10
  WARN_LIMIT=5
*
  <<VSAM>>
  INCLUDED_TRANS=(STRS,F?L?,TRLN)
  KILL_LIMIT=100
  WARN_LIMIT=50
*
```

In this example transaction, TRLN and transactions whose ID starts with DE will have the message 0C8902 issued if the transaction makes more than 50 EXEC CICS requests against a FILE or if the CPU used exceeds 5 seconds. If the transaction exceeds 10 seconds of CPU or 100 EXEC CICS FILE requests, the message 0C8903 is issued and the transaction is abended.

Specifying a question mark (?) in the transaction ID indicates that the rule will match for a transaction ID that has any character in that location. So a transaction ID of FALZ will have WARN and KILL limits of 50 and 100 for VSAM. Transaction ID FLAZ will not.

An asterisk (*) can be specified at the end of a transaction ID to indicate that any trailing characters will match.



Part 3

Performance scenarios

In this part of the book, we cover usage scenarios, showing the performance obtained.

We decided that for each scenario, we would use a different tool, thereby showing the different tools that can be used for performance management. We describe the following scenarios and tools:

- ▶ Security, using IBM CICS Performance Analyzer for z/OS
- ▶ MTOM, using z/OS Resource Measurement Facility (RMF)
- ▶ Omegamon scenario, Using IBM Tivoli Monitoring Tools

Environment overview

In this chapter, we provide an overview of the scenarios documented in the subsequent chapters.

We also describe the basic configuration tasks required to run the scenarios, including these topics:

- ▶ High level description of the application used for testing
- ▶ Benefits of using MTOM support for large messages
- ▶ Customizing CICS system initialization parameters
- ▶ Creating basic RACF® definitions
- ▶ Modifying the wsbind files
- ▶ Testing the environment

8.1 The example application

The scenarios we that describe use the CICS supplied Catalog Manager example Application. This is a working COBOL application that is designed to illustrate best practice when connecting CICS applications to external clients and servers. The example is constructed around a simple sales catalog and order processing application, in which the end user can perform these functions:

- ▶ List the contents of a stored catalog: Inquire catalog
- ▶ Select an item from the list: Inquire single
- ▶ Enter a quantity to order: Place order

The catalog is implemented as a VSAM file. After an order, the application updates this file to reflect the new stock levels. The base application has a 3270 user interface, but the modular structure, with well-defined interfaces between the components, makes it possible to add further components. In particular, the application comes with Web service support, which is designed to illustrate how you can extend an existing application into the Web services environment.

8.2 Installing and setting up the application

Before we run the application, we define and populate two KSDS VSAM datasets, and define two transactions.

Creating and defining the VSAM datasets

To create the datasets, two sets of JCL are provided by CICS. These are located in CICSTS32.CICS.SDFHINST and are named:

DFH\$ECFN	JCL to generate the configuration dataset
DFHECAT	JCL to generate catalog dataset

Defining the 3270 interface

The example application is supplied with a 3270 user interface to run the application and to customize it. The user interface consists of two transactions, EGUI and ECFG. We use the CEDA commands shown in Example 8-1 to define and install the transactions.

Example 8-1 Defining the Catalog application transactions

```
CEDA DEFINE TRANSACTION(EGUI) PROGRAM(DFH0XGUI) GROUP(S3C1)
CEDA DEFINE TRANSACTION(ECFG) PROGRAM(DFH0XCFG) GROUP(S3C1)
CEDA INSTALL GROUP(S3C1)
```

Note: Because we use program autoinstall, we do not define the programs and BMS maps used by the application. For a full list of resources used, see the CICS TS V3.2 Information Center, at:

<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>

8.3 MTOM scenario overview

In this section we discuss various considerations for the use of MTOM.

8.3.1 Why we use MTOM/XOP

A SOAP message must contain only printable characters. To handle binary data, the data must first be encoded in base64Binary format, which significantly increases its size because 6 bits of the binary object become 8 bits for transmission. When included in the message body, processing is required to perform base64Binary encoding by the sender of the message, and decoding by the receiver. Naturally this can be very CPU intensive.

The Message Transmission Optimization Mechanism (MTOM) specification defines a method for optimizing SOAP messages by using MIME Multipart/Related packages to move binary data elements into separate MIME attachments separated by MINIME boundary delimiters, in a similar manner to email attachments. No encoding of the attachments is necessary.

The XML-binary Optimized Packaging (XOP) specification defines an implementation for optimizing XML messages. It converts the XML in the SOAP message to XOP format by replacing the base64Binary data with a special <xop:Include> element containing a unique content-ID that references the relevant MIME attachment.

The use of MTOM/XOP allows the binary objects to remain in binary form and does not require them to be included within the SOAP body of the message. This can mean great savings in CPU usage, and so leads to a higher throughput.

MTOM is cheaper for the following reasons:

- ▶ It avoids the base64 conversion process.
- ▶ The data length is shorter.
- ▶ The data is not parsed by the SOAP parser.

8.3.2 Modifications to the catalog application

To gain a noticeable benefit from using MTOP/XOP support, we have to transfer larger amounts of data. This requires a CICS application communicating via Channels and Containers rather than a COMMAREA because of the 32K size limit on COMMAREAs.

The original CICS catalog example application is designed to demonstrate how to extend a 3270 CICS COBOL application to exploit Web service support. It has a modular structure with well defined interfaces that can be called via EXEC CICS LINK commands using a COMMAREA. The Web service support extension for the base application provides a Web client front end and a Web service endpoint. This allows either the Web client front end, or the 3270 interface of the catalog manager application to drive the business logic of the example application.

For the purposes of this scenario, the CICS catalog application has been extended to allow the Web front end to display an image of items in the catalog. Because images can be relatively large, they do not fit within the 32K limit of a COMMAREA. Channels and containers are now used instead of COMMAREAs, in order to allow the transfer of these images. The modifications we made are discussed in detail in Appendix A, “The modified catalog manager application” on page 303.

8.4 MTOM scenario preparation

In this section we describe the preparation of our MTOM scenarios.

8.4.1 System properties

Here are some recommendations regarding system properties:

- ▶ Increase CICS storage (both DSALIMIT and EDSALIMIT)
- ▶ Ensure that programs are defined as threadsafe.
- ▶ Increase MAXSOCKETS.
- ▶ Set RESPWAIT to 30.

8.4.2 Definition checklist

We use two different CICS regions to receive different figures for requester and provider. The following details, as shown in Table 8-1, apply for both regions.

Table 8-1 Definition checklist

Value	CICS region 1	CICS region 2
IP name	9.12.4.75	9.12.4.75
Primary Role	Service Requester	Service Provider
TCP/IP port	14304	14305
Job name	A6POC3C1	A6POC3C2
APPLID	A6POC3C1	A6POC3C2
TCPIP SERVICE	C3C1	C3C2

The configuration we used is shown in Figure 8-1.

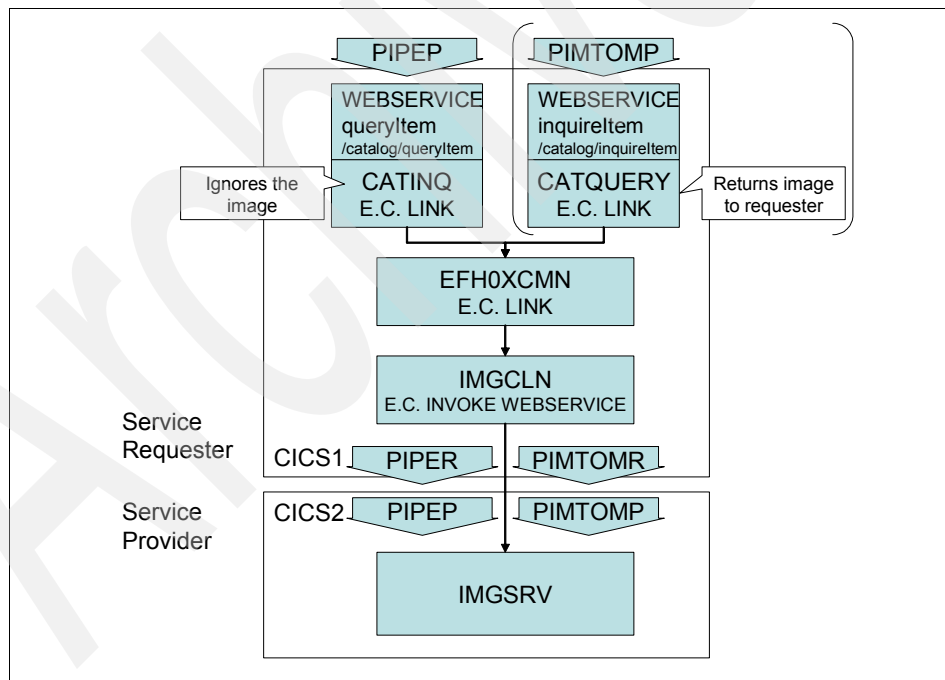


Figure 8-1 CICS as a service provider

To enable our testing scenarios, we use four pipelines as shown in Table 8-2.

Table 8-2 MTOM pipelines

MTOM status	Provider	Requester
No MTOM	PIPEP	PIPER
MTOM	PIMTOMP	PIMTOMR

Our first region, which hosts the catalog manager application, needs requester and provider pipelines; whereas the second region, hosting the image server, only needs provider pipelines. PIPEP and PIPER are the standard (non-MTOM) pipelines. Their definitions are in Appendix A, “The modified catalog manager application” on page 303. The definitions of the MTOM pipelines are shown in Figure 8-2 and Figure 8-3.

```
CEDA View Pipeline( PIMTOMP )
Pipeline      : PIMTOMP
Group         : C3C1PERF
Description   :
Status        : Enabled          Enabled ! Disabled
Respwait      : Deft             Default ! 0-9999
Configfile    : /CIWS/C3C1/config/ProviderMTOM.xml
(Mixed Case)  :
              :
Sshelf        : /CIWS/C3C1/shelf
(Mixed Case)  :
              :
Wsdire        : /CIWS/C3C1/PIMTOMP/wsbire
(Mixed Case)  :
```

Figure 8-2 Definition of the PIMTOMP provider pipeline


```

CEDA View Pipeline( PIMTOMR )
Pipeline      : PIMTOMR
Group        : C3C1PERF
Description   :
Status       : Enabled          Enabled ! Disabled
Respwait     : 0030             Default ! 0-9999
Configfile   : /CIWS/C3C1/config/RequesterMTOM.xml
(Mixed Case) :
:
SHelf        : /CIWS/C3C1/shelf
(Mixed Case) :
:
Wsdire       : /CIWS/C3C1/PIMTOMR/wsbind
(Mixed Case) :

```

Figure 8-3 Definition of the PIMTOMR requester pipeline

We altered our config file to enable support for MTOM. The modified configuration for the provider pipeline PIMTOMP is shown in Example 8-2, and for the requester pipeline PIMTOMR it is shown in Example 8-3.

Example 8-2 Pipeline configuration for PIMTOMP

```

<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline
xmlns="http://www.ibm.com/software/http/cics/pipeline"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline
provider.xsd">
  <cics_mtomp_handler>
    <dfhmtomp_configuration version="1">
      <mtomp_options send_mtomp="yes" send_when_no_xop="yes"/>
    </dfhmtomp_configuration>
  </cics_mtomp_handler>
  <transport>
  </transport>
  <service>
    <terminal_handler>
      <cics_soap_1.2_handler>
      </cics_soap_1.2_handler>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>

```

Example 8-3 Pipeline configuration PIMTOMR requester

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<requester_pipeline
  xmlns="http://www.ibm.com/software/http/cics/pipeline"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline requester.xsd">
  <service>
    <service_handler_list>
      <cics_soap_1.2_handler/>
    </service_handler_list>
  </service>
  <cics_mtom_handler>
    <dfhmtom_configuration version="1"/>
  </cics_mtom_handler>
</requester_pipeline>
```

If you compare the received response (see Example 8-4 here) to the result from our previous request including the image (see Figure A-15 on page 333), you can see that the binary data has been moved to a MIME attachment.

Example 8-4 MTOM/XOP response from inquireItem provider

```
--MimeBoundary-50b70e70-57838215-d14eb84f-64974f55
Content-Type: application/xop+xml; charset=UTF-8; type="text/xml"
Content-Transfer-Encoding: 8bit
Content-ID: <mime-root@cicsts>
<SOAP-ENV:Envelope
  xmlns:q0="http://www.exampleApp.inquireItemRequest.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <inquireItemResponse
      xmlns="http://www.exampleApp.inquireItemResponse.com">
      <returnCode>0</returnCode>
      <responseMessage>RETURNED ITEM: REF =0010</responseMessage>
      <singleItem>
        <itemReferenceNumber>10</itemReferenceNumber>
        <itemDescription>Ball Pens Black 24pk</itemDescription>
        <department>10</department>
        <unitCost>002.90</unitCost>
        <inStock>5948</inStock>
        <onOrder>2</onOrder>
        <imageData>
```

```

        <xop:Include
            xmlns:xop="http://www.w3.org/2004/08/xop/include"
            href="cid:8247dc80-857350e5-a3be6abf-b6679da5%40cicsts"/>
        </imageData>
    </singleItem>
</inquireItemResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--MimeBoundary-50b70e70-57838215-d14eb84f-64974f55
Content-Type: application/octet-stream
Content-ID: <8247dc80-857350e5-a3be6abf-b6679da5@cicsts>
Content-Transfer-Encoding: binary
<<<<<large-Binary-Content>>>>>
--MimeBoundary-50b70e70-57838215-d14eb84f-64974f55--

```

We want to see the improvement that MTOM/XOP delivers. Therefore we only want to measure the difference between switching it on or switching it off while everything else stays constant.

In our first CICS region we want to measure the requester, therefore we toggle the imageClient service to be picked up either by PIPER without or by PIMTOMR with MTOM.

We choose the queryItem, rather than the inquireItem Web service, so we do not send back the image to the original requester. This avoids influences from outbound message generation depending on the image there. The Web service is kept stable at the non MTOM pipeline PIPEP.

In our second CICS region we want to measure the image server provider, therefore we toggle the imageServer service to be either picked up by PIPEP without or PIMTOMP with MTOM.

8.4.3 Generate workload with varying binary data size

We use WebSphere Studio Workload Simulator to generate workload for the queryItem service. Therefore we send a Web service request as shown in Example 8-5.

Example 8-5 SOAP request used to generate workload for queryItem

```

<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
    xmlns:q0="http://www.exampleApp.queryItemRequest.com"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"

```

```

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
      <q0:queryItemRequest>
        <q0:itemRequiredReference>9001</q0:itemRequiredReference>
      </q0:queryItemRequest>
    </soapenv:Body>
  </soapenv:Envelope>

```

To observe the influence of the binary data size on performance, we created images with different sizes, as displayed in Table 8-3.

Table 8-3 Image numbers and sizes

item number	size in bytes
9001	1024
9002	2048
9003	2560
9004	5120
9005	10240
9011	20480
9012	51200
9013	76800
9006	102400
9007	512000
9008	1048576
9009	5242880
9010	10485760

8.5 Security scenarios overview

In this section we describe the security scenarios to be tested in Chapter 10, “Security scenarios” on page 229. These scenarios include:

1. CICS XML digital signature:

CICS acts as a Web service provider, with a WebSphere Application Server (WAS) client sending a digitally signed request to CICS. CICS provides a digitally signed response. Both CICS and WebSphere Application Server are configured to use Web Services Security (WS-Security) to provide integrity with an XML digital signature. For results, see 10.4, “Results” on page 259.

2. DataPower with X.509 certificate:

DataPower acts as an intermediary server, verifying the signature on behalf of CICS and forwarding the signing X.509 certificate to CICS as an asserted identity. For results, see 10.4, “Results” on page 259.

3. DataPower with UsernameToken:

DataPower maps the X.509 certificate to a UsernameToken, and forwards the UsernameToken to CICS as an asserted identity.

This allows us to compare the performance of processing X.509 tokens (BinarySecurityTokens) in CICS with the performance of UsernameTokens. For results, see 10.4, “Results” on page 259.

CICS XML digital signature

In this scenario the Web service requester uses an XML digital signature to sign the SOAP message.

This scenario is shown in Figure 8-4.

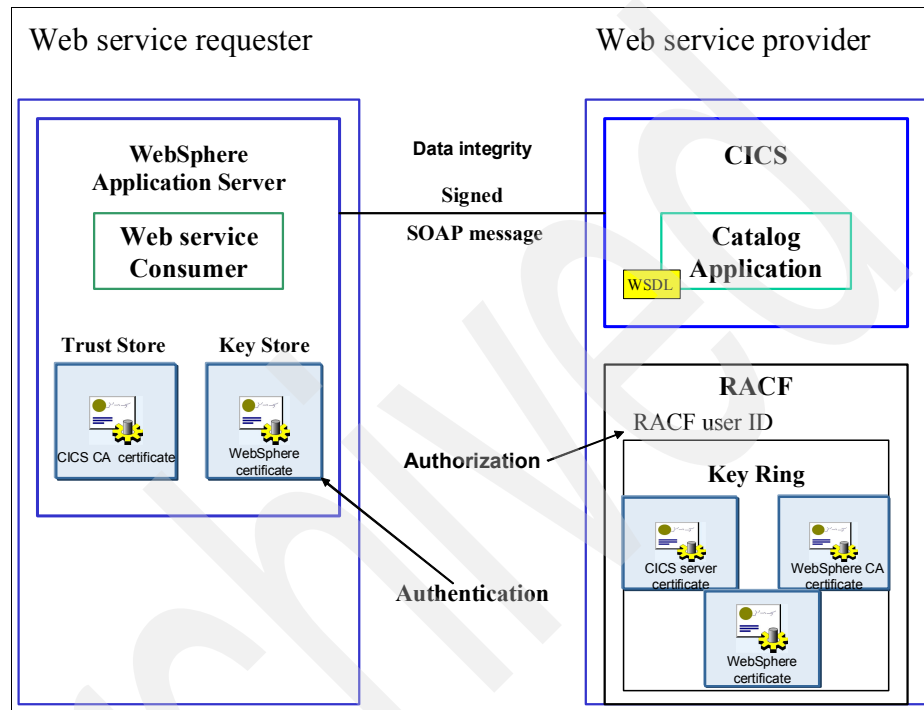


Figure 8-4 CICS XML digital signature

The following processing takes place:

1. The Web service requester signs the SOAP message, using an XML digital signature.
2. The Web service provider validates the signature.
3. The Web service provider runs using an identity mapped from the certificate it received.
4. The Web service provider sends a response.

We test this scenario in Chapter 10, “Security scenarios” on page 229.

DataPower with X.509 certificate

In this scenario we use WebSphere DataPower to validate a digital signature sent by the Web service requester, and send an X509 certificate to CICS. Then CICS uses RACF to map the certificate to a valid user ID under which to run the Web services transaction.

This scenario is shown in Figure 8-5.

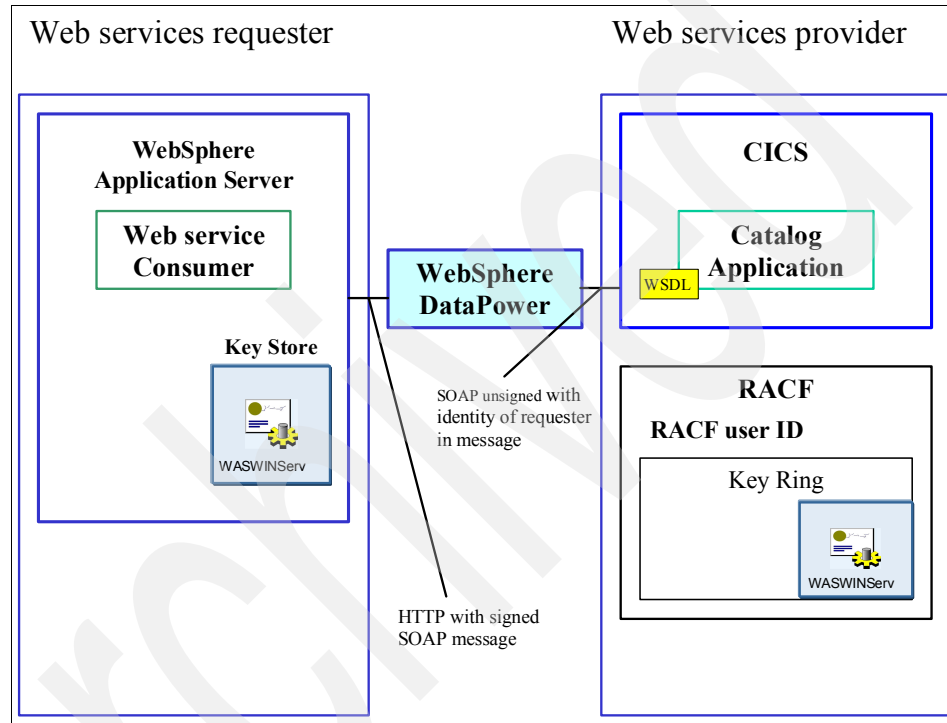


Figure 8-5 DataPower with X.509 certificate

The following processing takes place:

1. WebSphere Application Server sends a Digital Signature to WebSphere DataPower.
2. DataPower validates and strips the signature.
3. DataPower sends an X509 certificate to CICS.
4. The CICS PIPELINE maps the certificate into a RACF user ID and runs the Web services transaction under this RACF user ID.

In order to configure CICS and DataPower for this scenario, refer to *Securing CICS Web Services*, SG24-7658-00.

DataPower with UsernameToken

In this scenario, we use WebSphere DataPower to do the certificate validation, map it to a RACF ID, and pass that ID directly to CICS. This scenario offers savings in CPU processing on System z:

1. WebSphere Application Server sends a Digital Signature to WebSphere DataPower.
2. DataPower validates and strips the signature.
3. DataPower maps the certificate into a RACF user ID
4. DataPower forwards the RACF ID to CICS, and CICS runs the Web services transaction under this RACF user ID.

We now describe the steps required to configure this scenario. This is done by adding an additional DataPower transform to the configuration in the scenario above.

Assuming that you already have the configuration in place to use WebSphere DataPower with certificate authentication, you can modify it using the following steps to send a UsernameToken to CICS instead:

1. We begin by uploading an XML file to DataPower. This XML file contains the mapping from the identity associated with the certificate, to a RACF identity. Start by opening the DataPower console in a Web browser.
2. From the left-hand panel, click **Administration**, then select **File Management**. You should see a directory listing as shown in Figure 8-6.
3. Click **Actions** for the *local* directory, then select **Upload Files**. A new page opens allowing you to browse to the XML file on your workstation and upload it to DataPower.

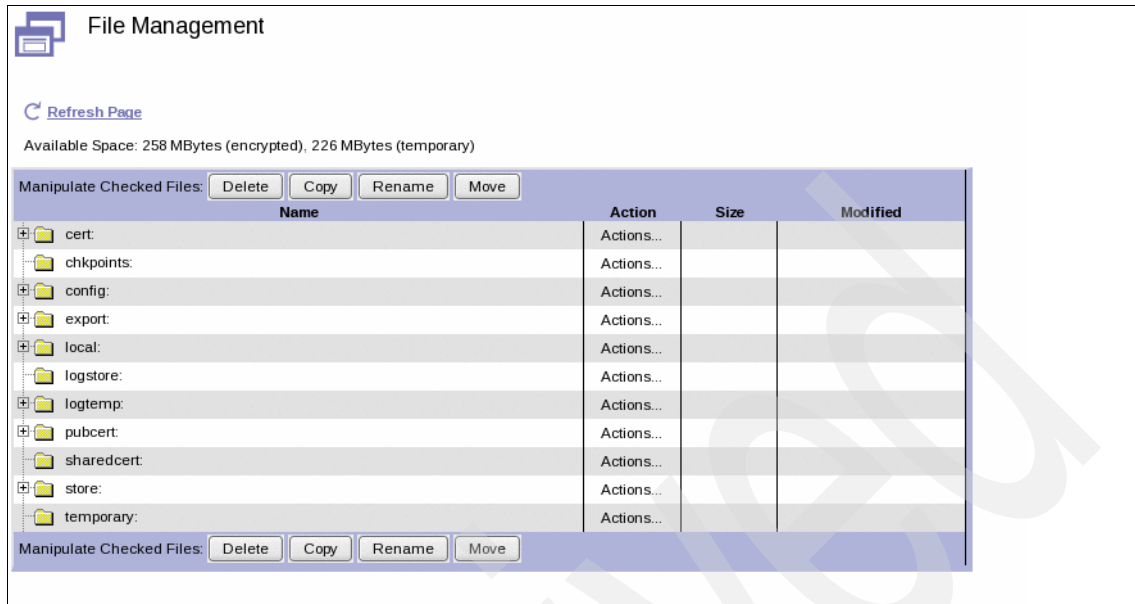


Figure 8-6 The File Management dialog

- We need to add a new transform to the DataPower rule currently in place. Starting from the DataPower console, select the **Web Service Proxy** icon to load the *Configure Web Service Proxy* panel. Clicking the name of the proxy (in this case, *CatalogWSProxy*) allows you to alter its configuration.

-
- Configure Web Service Proxy**
- WSDLs | SLM | Services | **Policy** | Proxy Settings | Advanced Proxy Settings | Headers/Params | Ws
- Web Service Proxy Name [up]
CatalogWSPProxy
- Apply Cancel Delete Refresh
- [View Log](#) | [View Status](#) | [View Operations](#) | [Show Probe](#) | [Validate Conformance](#) | [Help](#)
- Web Service Proxy Policy**
- Open tree to: Proxy | WSDLs | Services | Ports | Operations
- Tree view of the Web Service Proxy Policy configuration:
- Root: CatalogWSPProxy
 - request-rule (request-rule)
 - response-rule (response-rule)
 - Add Rule
 - wsdl: placeOrder.wsdl
 - WS-Policy: (default)
 - WS-I Conformance: (none)
 - Priority: Normal
 - Add Rule
 - service: DFHOXCMNService3
 - WS-Policy: (default)
 - WS-I Conformance: (none)
 - Priority: Normal
 - Add Rule
 - port: DFHOXCMNPort
 - WS-Policy: (default)
 - WS-I Conformance: (none)
 - Priority: Normal
 - Add Rule
 - port-operation: DFHOXCMN
 - WS-Policy: (default)
 - WS-I Conformance: (none)
 - Priority: Normal
 - CatalogWSPProxy_rule_0 (request-rule)
 - Add Rule

- Click the name of the rule (*CatalogWSPProxy_rule_0*) to edit it. The rule is displayed in Figure 8-8.

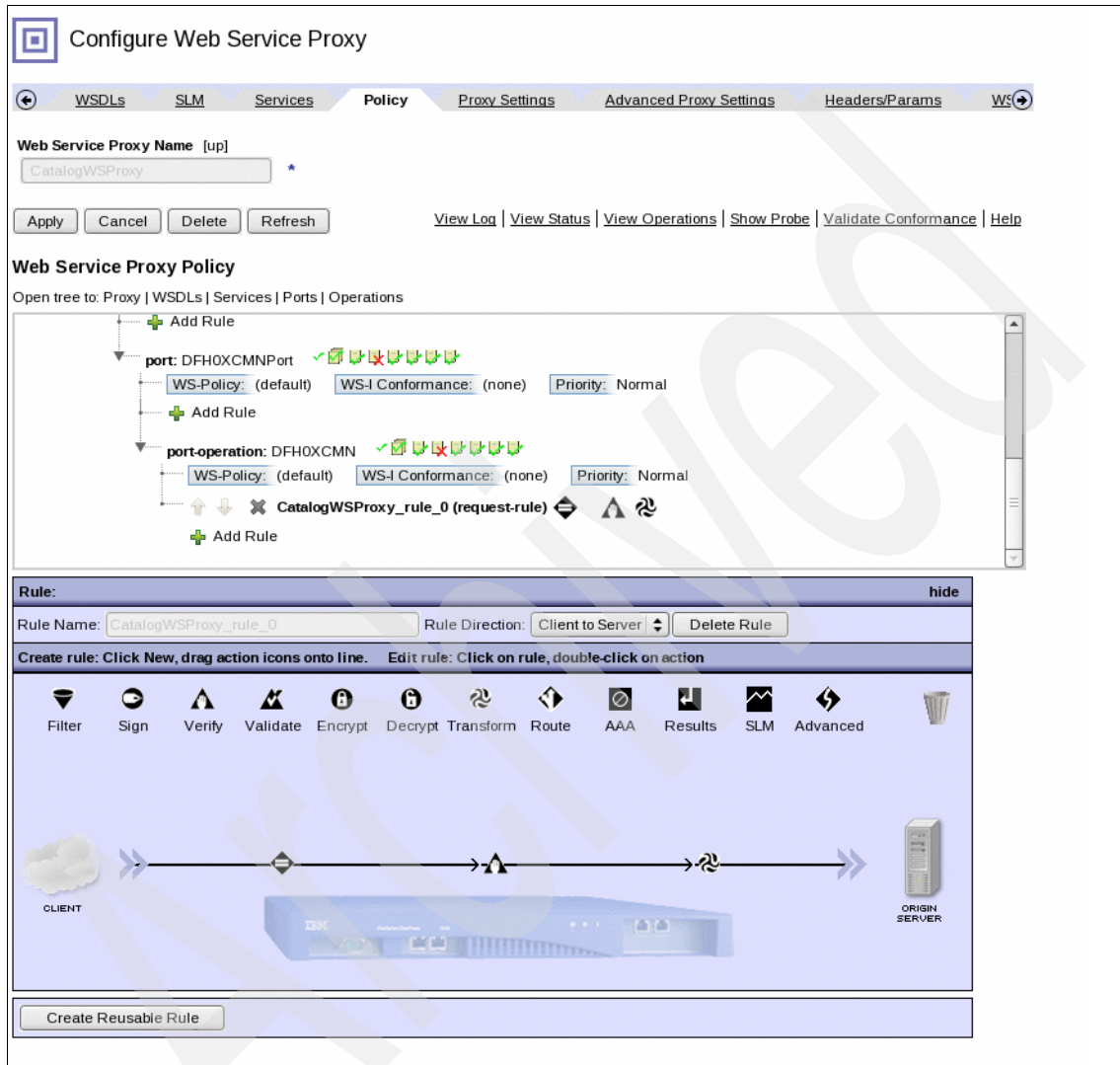


Figure 8-8 Editing the policy rule

Now we need to insert our new transform:

1. We drag and drop a new transform icon (see legend) somewhere along the line between the existing validation and transform rules.
2. Now we specify what this transform will do:
 - a. We create an XSL file, which uses the XML file that we already uploaded to describe the transform.
 - b. To upload this XSL file, double-click the new transform (highlighted with a yellow square). Select **Upload...**, and navigate to where you have stored your xsl file on your workstation.
 - c. When it is uploaded, select **local://**, and **adduserNameToken.xsl** from the two drop-down menus. Click **Done** to complete.
3. Next, click the other transform (*ExtractCertificateFromSignature.xsl*), which is already in place.
 - a. Select **store://** instead of local://
 - b. Then, in the second drop-down list, select **strip-wssec-signature.xsl**. This is an inbuilt transform to strip the wssecurity signature from a message, which is all we need to do now.
 - c. Click **Done** to complete.
4. You should now have both transforms in place, as shown in Figure 8-9. Click **Apply** to complete the modification.



Figure 8-9 Completed rule for UsernameToken scenario

The system is now set up to add the UsernameToken, and then strip the ws-security certificate from the messages, before handing them on to CICS.

8.6 Security scenario preparation

In this section we describe the basic CICS and RACF security configuration tasks required before starting to configure the specific security scenarios. CICS acts as the service provider in the scenarios described. The configuration required for the scenarios is described in the following chapters.

8.6.1 Software checklist

The software we use is listed in Table 8-4.

Table 8-4 Software used in the security scenarios.

z/OS	Windows
zOS V1.9	Windows XP SP3
CICS Transaction Server V3.2	WebSphere Application Server V 6.1.0.17

8.6.2 Definition checklist

The definitions we use are listed in Table 8-5.

Table 8-5 Settings used in the security scenarios

Value	CICS TS
IP name	wtsc66.itso.ibm.com
TCP/IP port	1430n
Jobname	CIWSS3Cn
APPLID	A6POS3Cn
TCPIPSERVICE	S3C1
Provider PIPELINE	EXPIPE01
Configuration files	ITSO_7658_basicsoap12provider.xml

The Web services we use in the scenarios are listed in Figure 8-6.

Table 8-6 Web services used in the scenarios

Web service	CICS transaction ID	User ID
inquireSingle	INQS	PUBLIC01
inquireCatalog	INQC	PUBLIC01
placeOrder	ORDR ORDS	PRIVAT01 Web services requester identity

The message handler program we use in the scenarios is listed in Table 8-7.

Table 8-7 CICS programs

value	CICS TS
Program	CIWSMSGO , message handler program, we use to switch the transaction ID from ORDR to ORDS, whenever the name of the invoked Web service is placeOrder.

The CICS user IDs we use in our configuration are listed in Table 8-8.

Table 8-8 CICS User IDs

Value	CICS TS
CICS region user ID	CIWS3D
CICS default user ID	CICSUSER
User IDs for browse operations	PUBLIC01
User IDs for ordering operations	PRIVAT01 USERWS01 USERWS02 USERWS03 USERWS05

The certificates we use in our configuration are listed in Table 8-9.

Table 8-9 Certificates

Value	Format	File
CICS server certificate	PKCS12DER	CIWSS3C1.P12 CIWSS3C2.P12
CICS CA certificate	CERTDER	CIWSROOT.CER

Value	Format	File
WebSphere CA certificate	CERTDER	WASWINROOT.CER
WebSphere server certificate	PKCS12DER	WWSERV01.P12 WWSERV02.P12
WebSphere z/OS CA certificate	CERTDER	WZOSROOT.CER
WebSphere z/OS server certificate	PKCS12DER	

8.6.3 Basic CICS security configuration

First we discuss our basic security configuration, taking a CICS region with no security and configuring it to enable transaction security (we do not implement other types of CICS security such as resource security and command security). We document the system initialization table parameters necessary to set up basic CICS security and then we test our basic security configuration.

For detailed information about CICS security, see *CICS Transaction Server for z/OS RACF Security Guide*, SC34-6454.

Because the INQC and INQS transactions are browse-only transactions, we chose not to secure them. We do not want to start any Web services transactions under the default transaction ID, CPIH or the default user ID CICSUSER, so we decided to start the placeOrder Web service with the ID of ORDR, and protect it from unauthorized use. The ORDS transaction updates the database, so we decided to protect it from unauthorized use too.

Setting up basic security configuration

We configured our CICS region with security prefixing, transaction security, and surrogate user security active using the following SIT parameters:

- ▶ SEC=YES
- ▶ SECPRFX=YES
- ▶ XTRAN=YES
- ▶ XUSER=YES

SEC=YES was specified to indicate that we wanted RACF services to control access to CICS resources.

We used security prefixing (SECPRFX=YES) in our CICS region, which prevents our RACF security profiles from affecting other CICS regions. This is useful in a production environment because it means that all security profiles are unique to an individual region; however, it can mean more work for the security administrator because more profiles must be defined.

XTRAN=YES was specified so CICS would control who could start transactions and XUSER=YES specifies that CICS is to perform surrogate user checking.

Defining the CICS resources in RACF

Next we define the example application transaction IDs and user IDs in RACF.

The commands we use are listed in Example 8-6.

Example 8-6 RACF commands to define transactions and transaction groups

```
RDEFINE TCICSTRN CIWS3D.INQS
RDEFINE TCICSTRN CIWS3D.INQC
RDEFINE TCICSTRN CIWS3D.ORDR
RDEFINE TCICSTRN CIWS3D.ORDS
```

To activate the definitions, we issue the following command:

```
SETROPTS RACLIST(TCICSTRN) REFRESH
```

Adding the users

Next we add the user IDs that will be permitted to run the Web services transactions:

- ▶ PUBLIC01
- ▶ PRIVAT01
- ▶ WSUSER01
- ▶ WSUSER02
- ▶ WSUSER03
- ▶ WSUSER05

The command we use to create the users is shown in Example 8-7.

Example 8-7 ADDUSER command

```
ADDUSER PUBLIC01 NOPASSWORD
```

8.6.4 CICS Web Services configuration

In this section we describe how we create the certificates used to secure our CICS Web Services environment in the scenarios in the following chapters. The certificates we use are listed in Table 8-9 on page 197.

Building the key ring

In CICS, the required server certificate and related information about certificate authorities are held in a key ring in the RACF database. The key ring contains your system's private and public key pair, together with your server certificate and the certificates for all the certificate authorities that might have signed the certificates you receive from your clients.

We used the command listed in Example 8-8 to build the key ring.

Example 8-8 Building a keyring

```
EXEC 'CICSTS32.CICS.SDFHSAMP(DFH$RING)' +  
'CIWS CIWSS3C1 wtsc66.itso.ibm.com FORUSER(CIWS3D)'
```

Modifying the wsbind file using CICS SupportPac CS04

By default, CICS runs all Web services under the same transaction ID, CPIH, and the CICS default user ID, CICSUSER. Depending on the nature of the Web service, we want to assign different security constraints to the transactions under which the Web service runs. The transaction and user IDs we want the different Web services to run under are listed in Table 8-6 on page 197.

There are different ways of achieving that. We are using the CICS supplied example application and have no need to regenerate the wsbind files. Therefore we use the CICS SupportPacCS04 - CICS TS for z/OS: WSBIND File Display and Change Utility to change the existing wsbind files. The utility can be used to list and change an existing wsbind file.

The following content of the wsbind file can be changed:

TRANSACTION=	Used to add or change a transaction name.
USERID=	Used to add or change a user ID.

Creating the HFS directories

In order to modify the CICS supplied wsbind files we create a set of new HFS directories we can use as output files for the utility. The directories we created are listed in Example 8-9. We later use these directories when we configure the PIPELINE in “Configuring the PIPELINE definition” on page 206.

Example 8-9 HFS directories used in the PIPELINE definition

```
/CIWS/S3C1/config  
/CIWS/S3C1/shelf  
/CIWS/S3C1/wsbind/provider
```

Modifying the WSBIND files

We modify the following wsbind files using SupportPac CS04:

- ▶ We specify /CIWS/S3C1/wsbind/provider as the output directory.
- ▶ As the input directory, we use the CICS supplied directory /usr/lpp/cicsts/cicsts31/samples/webservices/wsbind/provider/ and modify the following files:
 - inquireSingle.wsbind
 - inquireCatalog.wsbind
 - placeOrder.wsbind

The job we use is shown in Example 8-10.

Example 8-10 Job used to modify wsbind files

```
// SET QT='''
//WSBINDUT EXEC CS04PROC,
// TMPFILE=&QT.&SYSUID.&QT.
//INPUT.SYSUT1 DD *
#-> Add a Tranid and Userid to a WSBind file
INPUT=/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/*
inquireCatalog
OUTPUT=/CIWS/S3C1/wsbind/provider/inquireCatalog
TRANSACTION=INQC
USERID=PUBLIC01
#-> Add a Tranid to a WSBind file and overwrite
INPUT=/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/*
inquireSingle
OUTPUT=/CIWS/S3C1/wsbind/provider/inquireSingle
TRANSACTION=INQS
USERID=PUBLIC01
#-> Add a Tranid to a WSBind file and overwrite
INPUT=/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/*
placeOrder
OUTPUT=/CIWS/S3C1/wsbind/provider/placeOrder
TRANSACTION=ORDR
USERID=PRIVAT01
/*
```

A sample output from the utility is shown in Example 8-11.

Example 8-11 Sample output from SupportPac CS04

```
=> WSBind File Display and Change Utility starting (V1.0.1)
=> Collecting statements. Statements will be scanned twice. If there are any
detected problems on the first pass,
```

```

    no actions will be taken.
==> -> stmt 1, comment => #-> Add a Tranid to a WSBind file and overwrite
==> -> stmt 2, =====>
INPUT=/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/placeOrder
==> -> stmt 3, =====> OUTPUT=/CIWS/S3C6/wsbind/provider/placeOrder
==> -> stmt 4, =====> TRANSACTION=ORDR
==> -> stmt 5, =====> USERID=PRIVAT01

==> ==> Starting pass 1 (a syntax check).

==> ==> Starting pass 2 (taking actions).

==> -> statement 2 =>
INPUT=/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/placeOrder
==> Input WSBind file name set to
/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/placeOrder.wsbind
Details for
/usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider/placeOrder.wsbind
- Creation Timestamp: 20050914 08:23
- XML Conversion: Interpretive XML Conversion
- Mapping Level: 1.0
- Minimum Runtime Level: 1.0
- Provider
- Target program name: DFHOXCMN
- Program interface: COMMAREA
- Transaction ID: -none specified-
- User ID: -none specified-
- URI: /exampleApp/placeOrder
- WSDL File Name: /u/chrisb/WasV6/wsd1/placeOrder.wsd1

- Operation: DFHOXCMNOperation
- WSDL Binding: DFHOXCMNHTTPSoapBinding

==> -> statement 3 => OUTPUT=/CIWS/S3C6/wsbind/provider/placeOrder
==> Output WSBind file name set to /CIWS/S3C6/wsbind/provider/placeOrder.wsbind
==> WSBind File /CIWS/S3C6/wsbind/provider/placeOrder.wsbind already exists,
    but will be overwritten...
==> WSBind File was written to /CIWS/S3C6/wsbind/provider/placeOrder.wsbind

==> -> statement 4 => TRANSACTION=ORDR
==> ==> The old value for Transaction Id was "null"
==> ==> The new value for Transaction Id is "ORDR"

==> -> statement 5 => USERID=PRIVAT01

```

```
==> => The old value for User Id was "null"
==> => The new value for User Id is "PRIVAT01"
```

```
==> Output WSBind file name set to placeOrder.wsbind
==> WSBind File placeOrder.wsbind already exists, but will be overwritten...
==> WSBind File was written to placeOrder.wsbind

==> WSBind File Display and Change Utility ending
```

Creating the CICS Web Services resource definitions

In this section we create the necessary resource definitions.

Configuring the TCPIPService definition

We create a TCPIPService resource definition, as shown in Figure 8-10, using CEDA.

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0650
CEDA DEFINE TCpipservice( S3C1      )
TCpipservice   : S3C1
GRoup         : S3C1
DEscription   ==> TCPIPService DEFINITION FOR CATALOG APPLICATION
URm           ==> NONE
PORTnumber    ==> 14301          1-65535
STatus        ==> Open          Open | Closed
PROtocol      ==> Http          Iiop | Http | Eci | User
TRAnsaction   ==> CWXN
Backlog       ==> 00001         0-32767
TSqprefix     ==>
Ipaddress     ==>
SOcketclose   ==> No           No | 0-240000 (HHMMSS)
Maxdatalen    ==> 000032       3-524288
SECURITY
SSl           ==> No           Yes | No | Clientauth
CErtificate   ==>
(Mixed Case)

SYSID=S3C1 APPLID=A6POS3C1
```

Figure 8-10 CEDA DEFINE TCPIPService

We set the PORTNUMBER to 14301, the PROTOCOL to HTTP, and the URM to NONE. We allow the other attributes to default, and we install the S3C1 group.

Customizing the pipeline configuration file

We use SupportPac CS04 to change the transaction ID and user ID under which the service requests get initiated and run. The placeOrder Web service will not be allowed to run under one single user ID. We therefore write a message handler program **CIWSMSG0** to replace the transaction ID in the DFHWS-TRANID container with the transaction ID **ORDS**, when the service request (which can be retrieved from the DFHWS-WEBSERVICE container) is placeOrder. Now, the user who wants to run the placeOrder Web service must be permitted to run the ORDS transaction. We use the RACF command shown in Example 8-12 to give the permission.

Example 8-12 RACF command to permit a user ID to a transaction

```
PERMIT CIWS3D.ORDS CLASS(TCICSTRN) ID(TOMMYJ) ACCESS(READ)
```

To activate the message handler program, we need to make changes to the PIPELINE configuration file. We copy the file, basicsoap12provider.xml to the /CIWS/S3C1/config directory shown in Example 8-9 on page 200. The change to the configuration file is shown in Example 8-13.

Example 8-13 ITSO_7658_basicsoap12provider.xml, pipeline configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<provider_pipeline
xmlns="http://www.ibm.com/software/http/cics/pipeline"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline
provider.xsd ">
  <transport>
    <default_transport_handler_list>
    </default_transport_handler_list>
  </transport>
  <service>
    <service_handler_list>
      <handler>
        <program>CIWSMSG0</program>
        <handler_parameter_list/>
      </handler>
    </service_handler_list>
    <terminal_handler>
      <cics_soap_1.2_handler/>
    </terminal_handler>
  </service>
  <apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

The message handler program: CIWSMSGO

Example 8-14 shows the flow of control of the message handler program. The program only executes for Web service requests.

Example 8-14 CIWSMSGO - evaluate function

```
IF WS-DFHFUNCTION equal 'RECEIVE-REQUEST'
    PERFORM VALIDATE-REQUEST THRU END-VAL-REQUEST
    PERFORM CHANGE-TRANID    THRU END-CHANGE-TRANID
    EXEC CICS
        DELETE CONTAINER('DFHRESPONSE')
    END-EXEC
END-IF
EXEC CICS RETURN END-EXEC.
GOBACK.
```

Example 8-15 shows the code to get the Web service request from the DFHWS-WEBSERVICE container.

Example 8-15 CIWSMSGO - get WEBSERVICE container

```
GET-SOAP-WEBSERVICE.
EXEC CICS
    GET CONTAINER('DFHWS-WEBSERVICE')
    SET(ADDRESS OF CONTAINER-DATA)
    FLENGTH(CONTAINER-LEN)
END-EXEC.
```

Example 8-16 shows the code that determines the new transaction ID, replacing the default transaction ID in the DFHWS-TRANID container with ORDS if the Web service request is placeOrder.

Example 8-16 CIWSMSGO - determine new transaction ID

```
CHANGE-TRANID.
EXEC CICS GET CONTAINER('DFHWS-TRANID')
    SET(ADDRESS OF CONTAINER-DATA)
    FLENGTH(CONTAINER-LEN)
END-EXEC.
IF WS-WEBSERVICES = 'placeOrder'
    MOVE 'ORDS' TO CA-TRANID
    PERFORM CHANGE-CONTAINER THRU END-CHANGE-CONTAINER
END-IF.
END-CHANGE-TRANID. EXIT.
```

Example 8-17 shows how the program changes the transaction ID in the DFHWS-TRANID container, and performs an EXEC CICS PUT CONTAINER.

Example 8-17 CIWSMSGO - change transaction ID

```
CHANGE-CONTAINER.  
    MOVE CA-TRANID TO CONTAINER-DATA(1:4)  
    EXEC CICS PUT CONTAINER('DFHWS-TRANID')  
        FROM(CONTAINER-DATA)  
        FLLENGTH(CONTAINER-LEN)  
END-EXEC.
```

Configuring the PIPELINE definition

Next we define the PIPELINE for the CICS service provider using the following CICS command:

```
CEDA DEFINE PIPELINE(EXPIPE01) GROUP(S3C1)
```

We define the EXPIPE01 pipeline as shown in Figure 8-11.

```
OVERTYPE TO MODIFY                                     CICS RELEASE = 0650  
CEDA DEFine PIpeline(EXPIPE01 )  
  PIpeline      : EXPIPE01  
  Group        : S3C1  
  Description   ==>  
  SStatus      ==> Enabled          Enabled | Disabled  
  Configfile    ==> /CIWS/S3C1/config/ITS0_7658_basicsoap12provider.xml  
  (Mixed Case) ==>  
               ==>  
               ==>  
               ==>  
  SHeIf        ==> /CIWS/S3C1/shelf  
  (Mixed Case) ==>  
               ==>  
               ==>  
               ==>  
  Wsdir        : /CIWS/S3C1/wsbind/provider/  
  (Mixed Case) :  
               :
```

Figure 8-11 CEDA DEFINE PIPELINE

As shown in Figure 8-11 on page 206, we establish the following settings:

- ▶ We set CONFIGFILE to the name of our pipeline configuration file:
/CIWS/S3C1/config/ITS0_7658_basicsoap12provider.xml
- ▶ We set SHELF to the name of the shelf directory:
/CIWS/S3C1/shelf
- ▶ We set WSDIR to the Web service binding directory that contains the wsbind files for the sample application:
/CIWS/S3C1/wsbind/provider/

Installing the PIPELINE definition

Next we use CEDA to install the PIPELINE definition. When the PIPELINE is installed CICS scans the wsdir directory, and dynamically creates WEBSERVICE and URIMAP definitions for the wsbind files found.

Figure 8-12 shows a CEMT INQUIRE PIPELINE for EXPIPE01.

```
INQUIRE PIPELINE
RESULT - OVERTYPE TO MODIFY
Pipeline(EXPIPE01)
Enablestatus( Enabled )
Configfile(/CIWS/S3C1/config/ITS0_7658_basicsoap12provider.xml)
Shelf(/CIWS/S3C1/shelf/)
Wsdir(/CIWS/S3C1/wsbind/provider/)

SYSID=S3C1 APPLID=A6POS3C1
```

Figure 8-12 CEMT INQUIRE PIPELINE - EXPIPE01

Defining the transactions

The default pipeline alias transaction ID used for inbound HTTP Web service requests is CPIH.

Because we want to run the pipeline under a different transaction ID, we need to make copies of the CPIH transaction definition as listed in Example 8-18.

Example 8-18 CICS definitions - TRANSACTION

```
CEDA COPY TRANSACTION(CPIH) GROUP(DFHPIPE) TO(S3C1) AS(INQS)
CEDA COPY TRANSACTION(CPIH) GROUP(DFHPIPE) TO(S3C1) AS(INQC)
CEDA COPY TRANSACTION(CPIH) GROUP(DFHPIPE) TO(S3C1) AS(ORDR)
CEDA COPY TRANSACTION(CPIH) GROUP(DFHPIPE) TO(S3C1) AS(ORDS)
```

MTOM scenario

In this chapter, we focus on performance benefits that can be achieved by taking advantage of the MTOM/XOP standard to transmit large binary data objects. The single inquiry function of the CICS catalog manager application has been modified to return an image of the requested item in addition to its details. This function is exposed through a Web service. We now enable MTOM/XOP to improve performance.

9.1 Introduction to MTOM/XOP

In standard SOAP messages, binary objects are base64 encoded and included in the message body. This significantly increases their size, and for very large binary objects, it can impact transmission time. Implementing MTOM/XOP provides a solution to this problem.

The SOAP Message Transmission Optimization Mechanism (MTOM) and XML-binary Optimized Packaging (XOP) specifications, often referred to as MTOM/XOP, define a method for optimizing the transmission of large base64binary data objects within SOAP messages.

The MTOM specification conceptually defines a method for optimizing SOAP messages by separating out binary data that would otherwise be base64 encoded, and sending it in separate binary attachments using a MIME Multipart/Related message. This type of MIME message is called an MTOM message. Sending the data in binary format significantly reduces its size, thus optimizing the transmission of the SOAP message.

The XOP specification defines an implementation for optimizing XML messages using binary attachments in a packaging format that includes but is not limited to MIME messages.

The size of the base64binary data is significantly reduced because the attachments are encoded in binary format. The XML in the SOAP message is then converted to XOP format by replacing the base64binary data with a special `<xop:Include>` element that references the relevant MIME attachment using a URI.

MTOM is cheaper for the following reasons:

- ▶ It avoids the base64 conversion process.
- ▶ The data length is shorter.
- ▶ The data is not parsed by the SOAP parser.

CICS implements support for MTOM/XOP in both requester and provider pipelines. As an alternative to including the base64binary data directly in the SOAP message, CICS applications that are deployed as Web service providers or requesters can use this support to send and receive MTOM messages with binary attachments. You can configure this support by using additional options in the pipeline configuration file.

9.1.1 XOP processing considerations

There are certain scenarios where CICS cannot support the XOP document format in MTOM messages directly. For example, the Web Services security functionality and Web services validation cannot parse the <xop:Include> elements in the XOP document. Therefore, two modes of support are provided in the pipeline to handle XOP documents and any associated binary attachments.

If the application handler program is capable of supporting XOP documents, such as the standard handlers that are provided when you deploy a Web service using the Web services assistant, then CICS performs XOP processing in direct mode. If you are using a different application handler in the pipeline that is not capable of handling XOP documents, all XOP processing is performed in compatibility mode.

Compatibility mode is used if you are using the Web Services Security functionality (where the original message is needed to do the security checking) or are testing with validation switched on. In such cases, all XOP processing is performed in compatibility mode even if you have specified direct mode in the pipeline configuration file. The switch to compatibility mode is done automatically by CICS,

9.1.2 Catalog application changes for MTOM/XOP

Previously, we described the changes that had to be made to the base application in the environment (see Chapter 8, “Environment overview” on page 177).

We now have an application available that can be triggered using a single Web service request and that provides both:

- ▶ A Web service provider sending large binary data
- ▶ A Web service requester receiving large binary data

Depending on which single inquiry program we use, we might or might not return the received image back to the requester:

- ▶ CATQUERY returns no image.
- ▶ CATINQ returns the image.

To compare performance figures, we have to:

- ▶ Switch MTOM on or off.
- ▶ Generate workload with varying binary data size.

The configuration we used is shown in Figure 9-1. Depending on which pipeline a Web service requester or provider is associated to, MTOM is enabled or disabled. To vary the binary data size, we request different items with different corresponding image sizes.

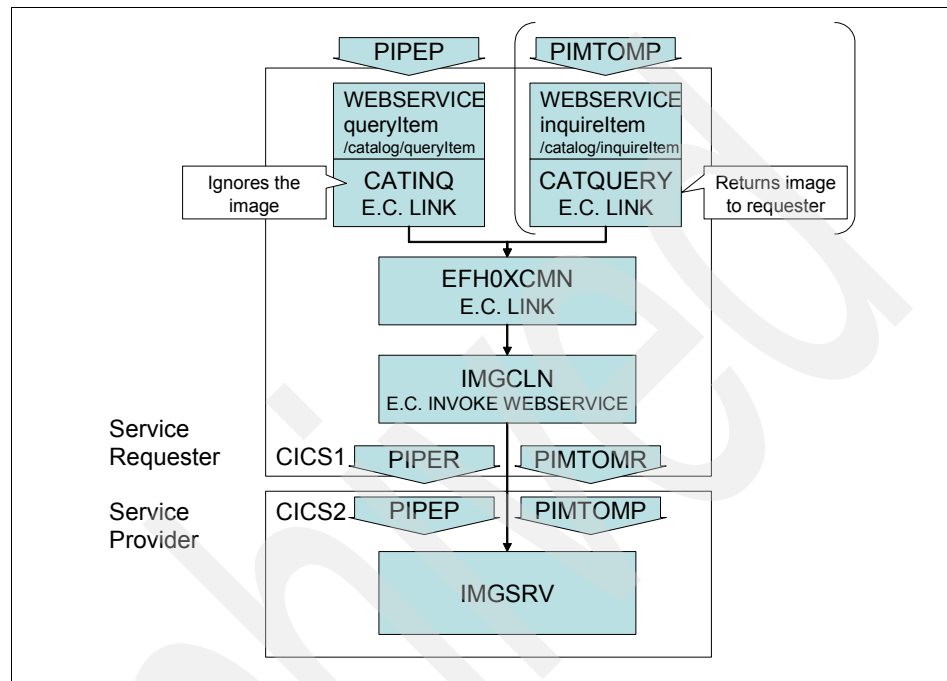


Figure 9-1 CICS as a service provider

WebSphere Studio Workload simulator is configured to send a request for a single item. The requests are sent continuously for 10 minutes by 75 clients with a startup delay of 500 ms. The element delay is 100.

9.2 Results

In this section we review the results of our measurements.

9.2.1 MTOM provider results

The figures depending on alternating file sizes are shown in Figure 9-2 for MTOM switched off and Figure 9-3 for MTOM switched on. We measured the consumed CPU, the number of ended CWXN and CPIH transactions per second, and CPU consumed per SOAP message in ms.

MTOM			Throughput				Response
Request #	File size(KB)	CP%	END/S CPIO	ACTUAL CPIO	END/S CWOXN	ACTUAL CWOXN	Total ACTUAL
9001	1	7.28	32.51	8	32.53	1	9
9002	2	7.41	32.50	8	32.51	1	9
9003	2.5	7.49	32.50	8	32.50	1	9
9004	5	7.54	32.51	8	32.50	1	9
9005	10	7.81	32.48	8	32.49	1	9
9011	20	7.94	32.51	10	32.49	1	11
9006	100	10.26	32.50	13	32.48	1	14
9007	500	22.00	32.10	30	32.10	1	31
9008	1024	38.75	31.75	55	31.74	1	56

Figure 9-2 Provider no MTOM

MTOM			Throughput				Response
Request #	File size(KB)	CP%	END/S CPIO	ACTUAL CPIO	END/S CWOXN	ACTUAL CWOXN	Total ACTUAL
9001	1	7.28	32.51	8	32.53	1	9
9002	2	7.41	32.50	8	32.51	1	9
9003	2.5	7.49	32.50	8	32.50	1	9
9004	5	7.54	32.51	8	32.50	1	9
9005	10	7.81	32.48	8	32.49	1	9
9011	20	7.94	32.51	10	32.49	1	11
9006	100	10.26	32.50	13	32.48	1	14
9007	500	22.00	32.10	30	32.10	1	31
9008	1024	38.75	31.75	55	31.74	1	56

Figure 9-3 Provider MTOM

Figure 9-4 displays a comparison depending on file size. For smaller file sizes, we cannot observe any relevant differences. Starting from a cutoff point of about 10 KB, there is an increasing significant advantage that makes it reasonable to enable MTOM.

File size (KB)	END/S CPH	END/s CWXN	CPU
	MTOM/ NOMTOM	MTOM/ NOMTOM	NOMTOM/ MTOM
1	1.08	1.08	0.95
2	1.29	1.28	0.95
2.5	1.00	1.00	0.98
5	1.00	1.00	1.00
10	1.12	1.12	0.98
20	1.00	1.00	1.05
100	1.07	1.07	1.43
500	1.03	1.03	2.18
1024	1.41	1.41	2.43
5120	3.27	3.27	2.45

Figure 9-4 Provider comparison no MTOM versus MTOM

The following charts illustrate the foregoing figures. As expected, there is a linear relationship between CPU consumption per SOAP message and the file size, and for file sizes over about 5 KB in size, using MTOM/XOP consumes less CPU than not using MTOM/XOP.

Figure 9-5 displays the CPU consumption per SOAP message in ms depending on the image size.

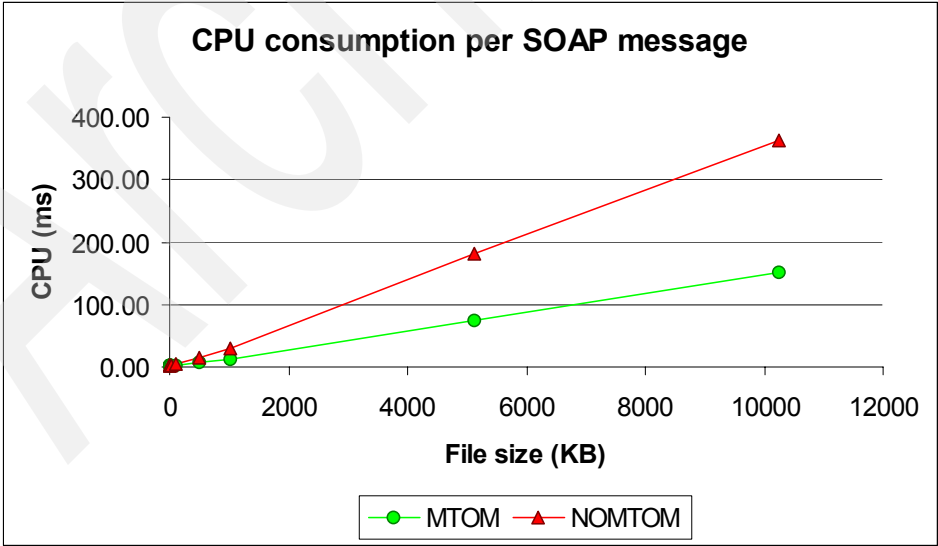


Figure 9-5 Provider CPU chart full

Figure 9-6 shows an excerpt from Figure 9-13 on page 219 for smaller image sizes. We observed a cutoff point at about 5 KB message size.

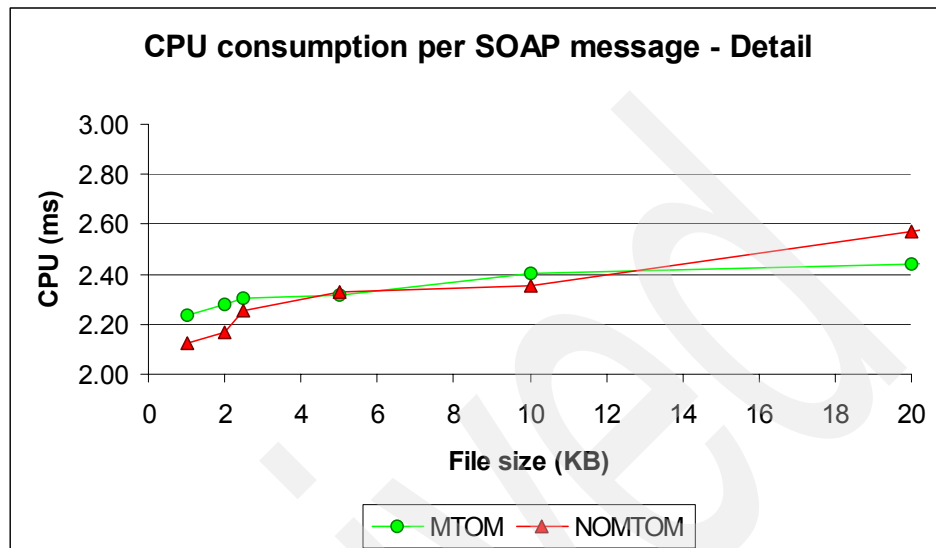


Figure 9-6 Provider CPU chart detail

In Figure 9-7 you can see how the number of CPIH transactions per seconds declines with increasing message size.

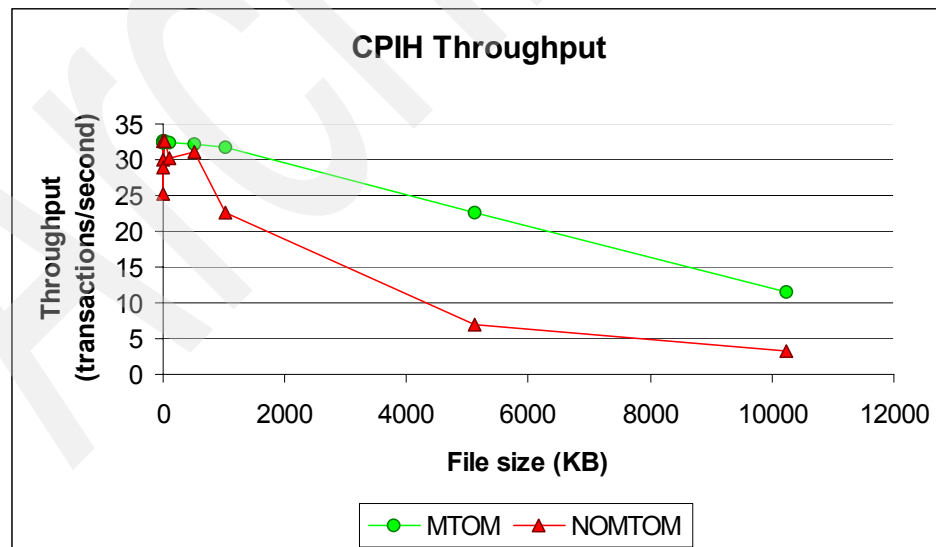


Figure 9-7 Provider CPIH throughput chart

A similar decline of transactions per second can be seen in Figure 9-8 for CWXN.

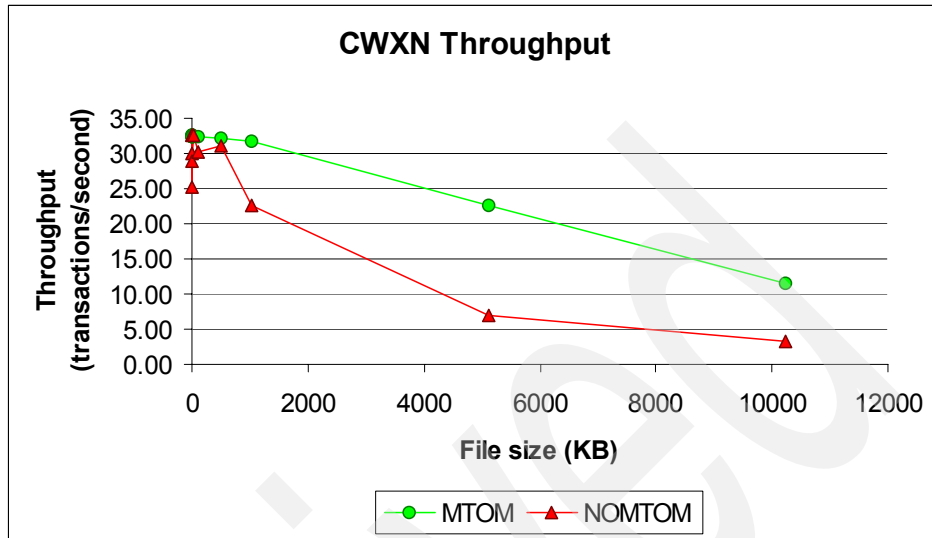


Figure 9-8 Provider CWXN throughput chart

The total transaction response time for CPIH and CWXN increases with rising message size (see Figure 9-9). The usage of MTOM results in a more gentle incline.

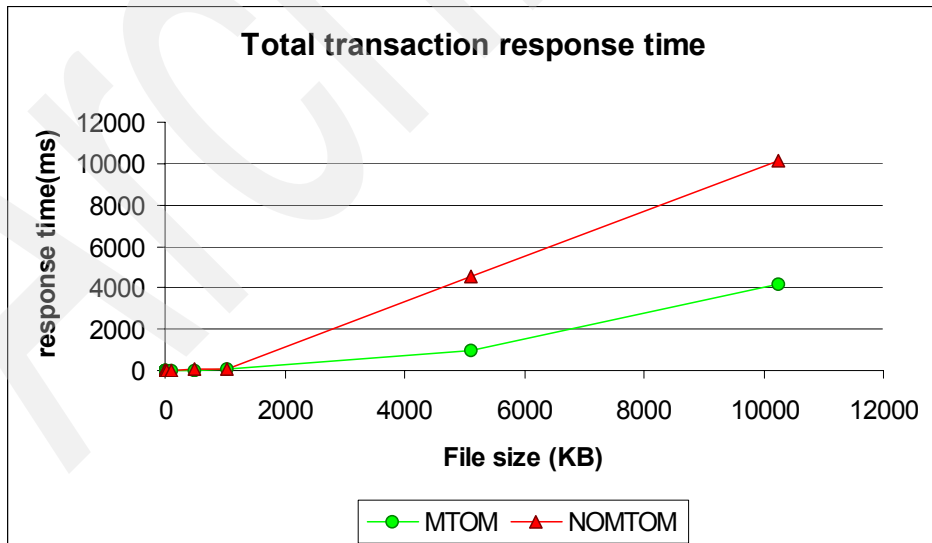


Figure 9-9 Provider total response time for CPIH and CWXN

9.2.2 MTOM requester results

The figures depending on alternating file sizes are shown in Figure 9-10 for MTOM switched off and Figure 9-11 for MTOM switched on. We measured the consumed CPU, the number of ended CWXN and CPIH transactions per second, and CPU consumed per SOAP message in ms.

MTOM			Throughput				Response
Request #	File size(KB)	CP%	END/S CPIH	ACTUAL CPIH	END/S CWXN	ACTUAL CWXN	Total ACTUAL
9001	1	11.33	32.51	28	32.53	245	273
9002	2	11.29	32.50	29	32.50	246	275
9003	2.5	11.31	32.52	31	32.49	245	276
9004	5	11.25	32.52	28	32.48	246	274
9005	10	11.48	32.49	31	32.48	246	277
9011	20	11.68	32.51	30	32.47	246	276
9006	100	12.87	32.50	33	32.48	243	276
9007	500	19.40	32.11	69	32.12	240	309
9008	1024	28.11	31.75	78	31.75	252	330

Figure 9-10 Requester no MTOM

MTOM			Throughput				Response
Request #	File size(KB)	CP%	END/S CPIH	ACTUAL CPIH	END/S CWXN	ACTUAL CWXN	Total ACTUAL
9001	1	11.33	32.51	28	32.53	245	273
9002	2	11.29	32.50	29	32.50	246	275
9003	2.5	11.31	32.52	31	32.49	245	276
9004	5	11.25	32.52	28	32.48	246	274
9005	10	11.48	32.49	31	32.48	246	277
9011	20	11.68	32.51	30	32.47	246	276
9006	100	12.87	32.50	33	32.48	243	276
9007	500	19.40	32.11	69	32.12	240	309
9008	1024	28.11	31.75	78	31.75	252	330

Figure 9-11 Requester MTOM

Figure 9-12 displays a comparison depending on file size. For smaller file sizes, we cannot observe any relevant differences. Starting from a cutoff point of about 5 KB, there is an increasing significant advantage that makes it reasonable to

enable MTOM. Compared to the provider, the savings delivered by the usage of MTOM are even higher.

File size (KB)	END/S CPIH	END/s CWXN	CPU
	MTOM/ NOMTOM	MTOM/ NOMTOM	NOMTOM/ MTOM
1	1.08	1.08	0.92
2	1.29	1.28	0.95
2.5	1.00	1.00	0.98
5	1.00	1.00	1.02
10	1.12	1.12	1.05
20	1.00	1.00	1.20
100	1.07	1.07	2.21
500	1.03	1.03	5.24
1024	1.41	1.41	7.10
5120	3.26	3.27	8.92

Figure 9-12 Requester comparison no MTOM versus MTOM

In the following charts we illustrate these figures. Generally a large impact of file size on CPU consumption and transaction throughput can be observed, although with MTOM enabled, the impact seems less significant.

Figure 9-13 displays the CPU consumption per SOAP message in ms depending on the image size.

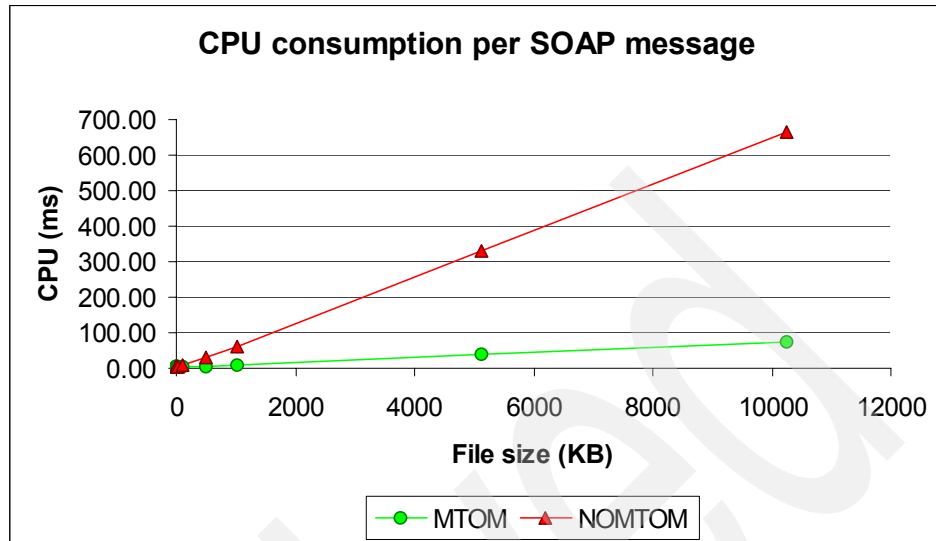


Figure 9-13 Requester CPU chart full

Figure 9-14 shows an excerpt from Figure 9-13 for smaller image sizes. We observed a cutoff point at about 5 KB message size.

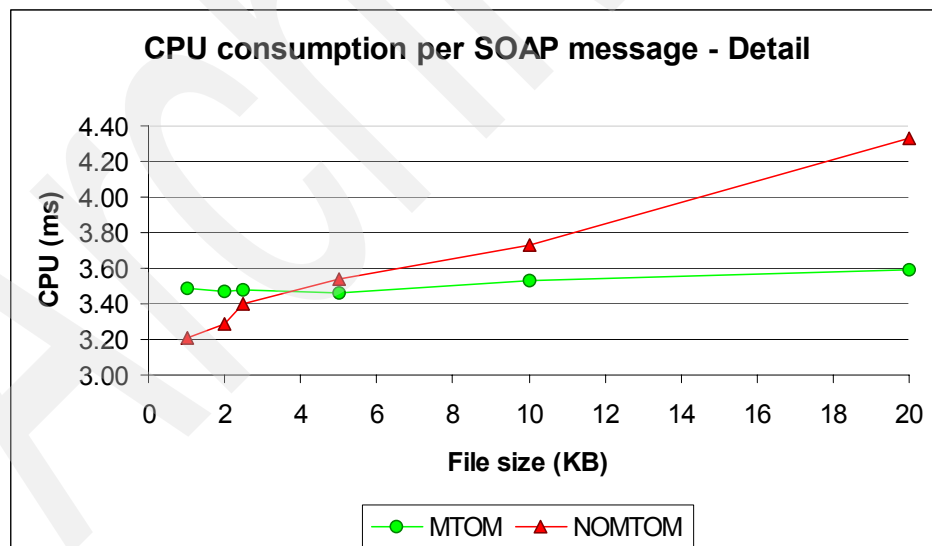


Figure 9-14 Requester CPU chart detail

In Figure 9-15 you can see how the number of CPIH transactions per second declines with increasing message size.

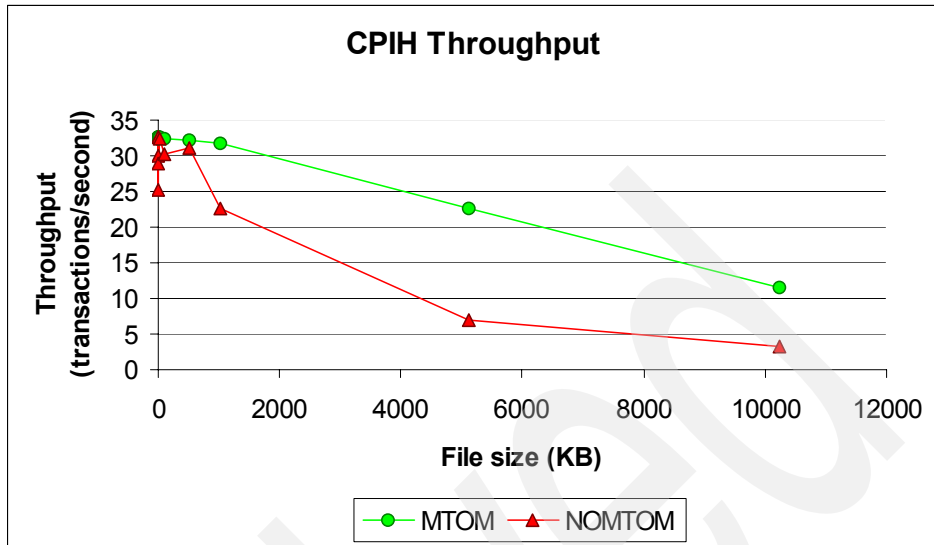


Figure 9-15 Requester CPIH throughput chart

A similar decline of transactions per second can be seen in Figure 9-16 for CWXN.

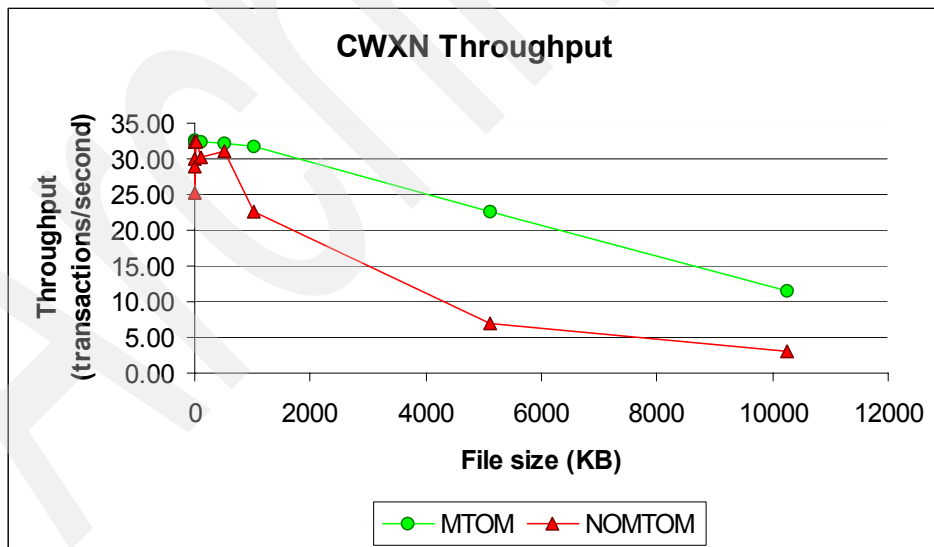


Figure 9-16 Requester CWXN throughput chart

The total transaction response time for CPIH and CWXN increases with rising message size (see Figure 9-17). The usage of MTOM results in a more gentle incline.

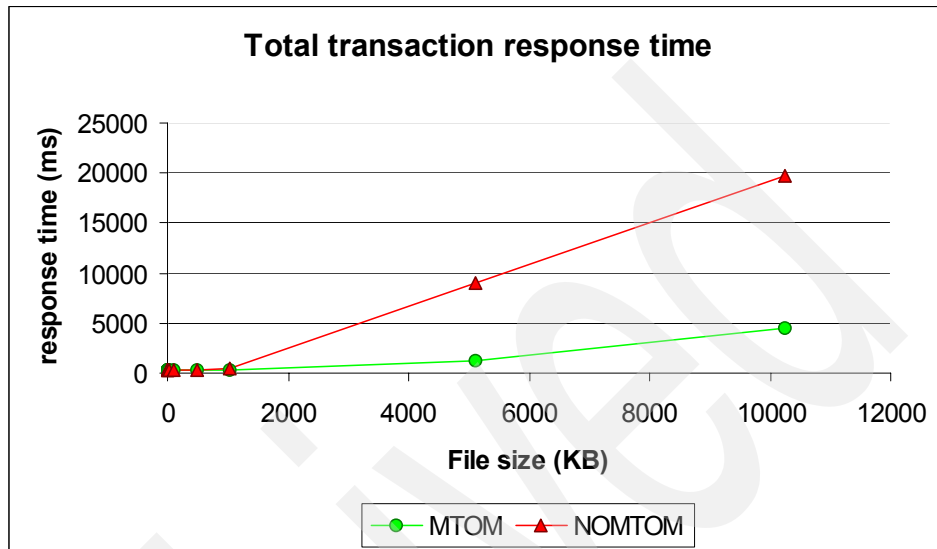


Figure 9-17 Requester total response time for CPIH and CWXN

9.2.3 Effects of switching workload to System z10

During our tests we had the opportunity to run our workload on a System z10™. The following tables and charts show the results for three exemplary file sizes (0.1 MB, 1 MB, and 10 MB) that we used to rerun the tests on the System z10.

The table in Figure 9-18 compares the provider scenario with MTOM switched on or off, on System z9 or System z10.

			Throughput				Response
Request #	File size (KB)	CP%	END/S CPIH	ACTUAL CPIH	END/S CWXN	ACTUAL CWXN	Total ACTUAL
	MTOM z10						
9006	100	5.98	32.58	5	32.58	0	5
9008	1024	28.11	32.39	31	32.39	0	31
9010	10240	180.91	14.07	2909	14.08	72	2981
	NO MTOM z10						
9006	100	8.13	32.59	8	32.58	0	8
9008	1024	54.34	31.24	59	31.24	0	59
9010	10240	227.02	8.87	3213	8.9	101	3314
	MTOM z9						
9006	100	10.26	32.50	13	32.48	1	14
9008	1024	38.75	31.75	55	31.74	1	56
9010	10240	173.56	11.45	4023	11.45	115	4138
	NO MTOM z9						
9006	100	13.72	30.32	20	30.31	1	21
9008	1024	66.96	22.55	103	22.54	3	106
9010	10240	116.77	3.25	9860	3.19	265	10125

Figure 9-18 Provider comparison z9® to z10

The graph in Figure 9-19 visualizes our results and gives reason to believe that the new hardware delivers a significant performance improvement for MTOM switched off. If MTOM is enabled, we do observe improvements.

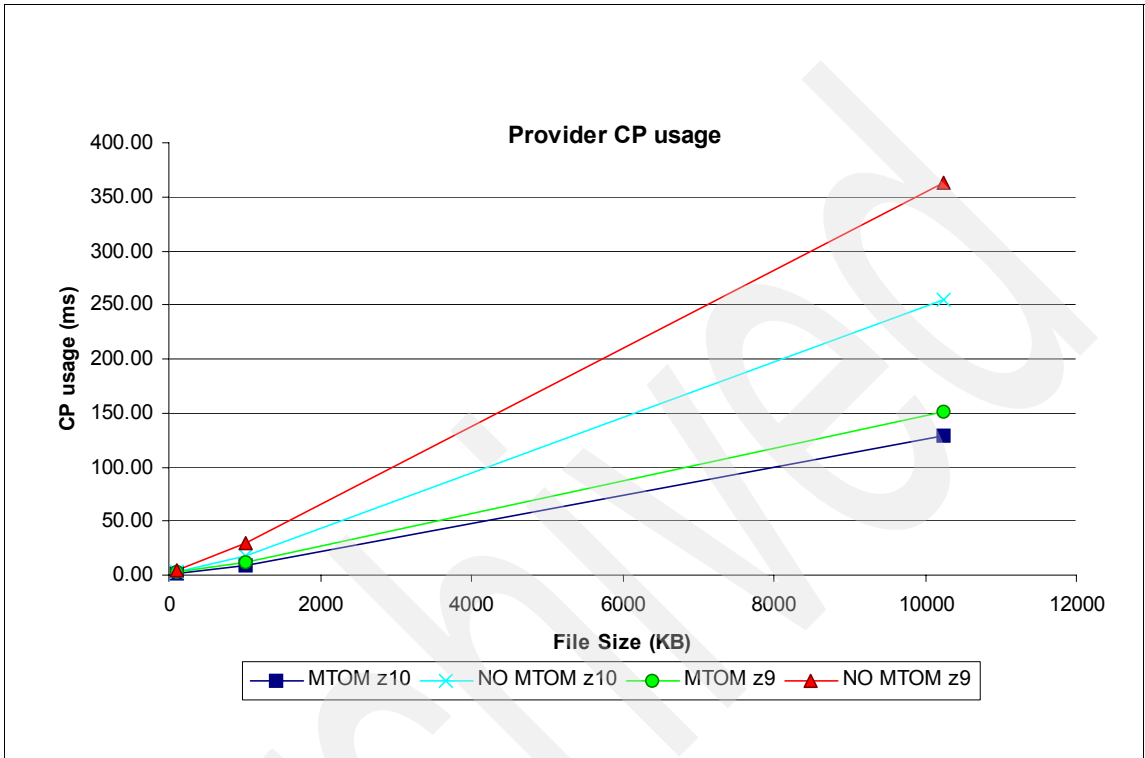


Figure 9-19 Provider CPU comparison chart

The results for the requester on z10 in comparison to z9 can be viewed in Figure 9-20.

			Throughput	Response time		Response	
Request #	File size(KB)	CP%	END/S CPIH	ACTUAL CPIH	CWX N	ACTUAL CWXN	Total ACTUAL
	MTOM z10						
9006	100	6.99	32.58	12	32.58	257	269
9008	1024	17.55	32.39	41	32.39	241	282
9010	10240	76.59	14.07	3039	14.07	258	3297
	NO MTOM z10						
9006	100	13.09	32.59	17	32.58	253	270
9008	1024	87.14	31.25	113	31.24	255	368
9010	10240	287.27	8.89	6006	8.89	688	6694
	MTOM z9						
9006	100	12.87	32.50	33	32.48	243	276
9008	1024	28.11	31.75	78	31.75	252	330
9010	10240	85.99	11.44	4265	11.43	260	4525
	NO MTOM z9						
9006	100	26.50	30.32	50	30.30	244	294
9008	1024	141.79	22.56	257	22.55	253	510
9010	10240	213.29	3.26	19175	3.15	609	19784

Figure 9-20 Requester comparison z9 to z10

The graph in Figure 9-21 illustrating our table also shows significant improvement when MTOM is switched off.

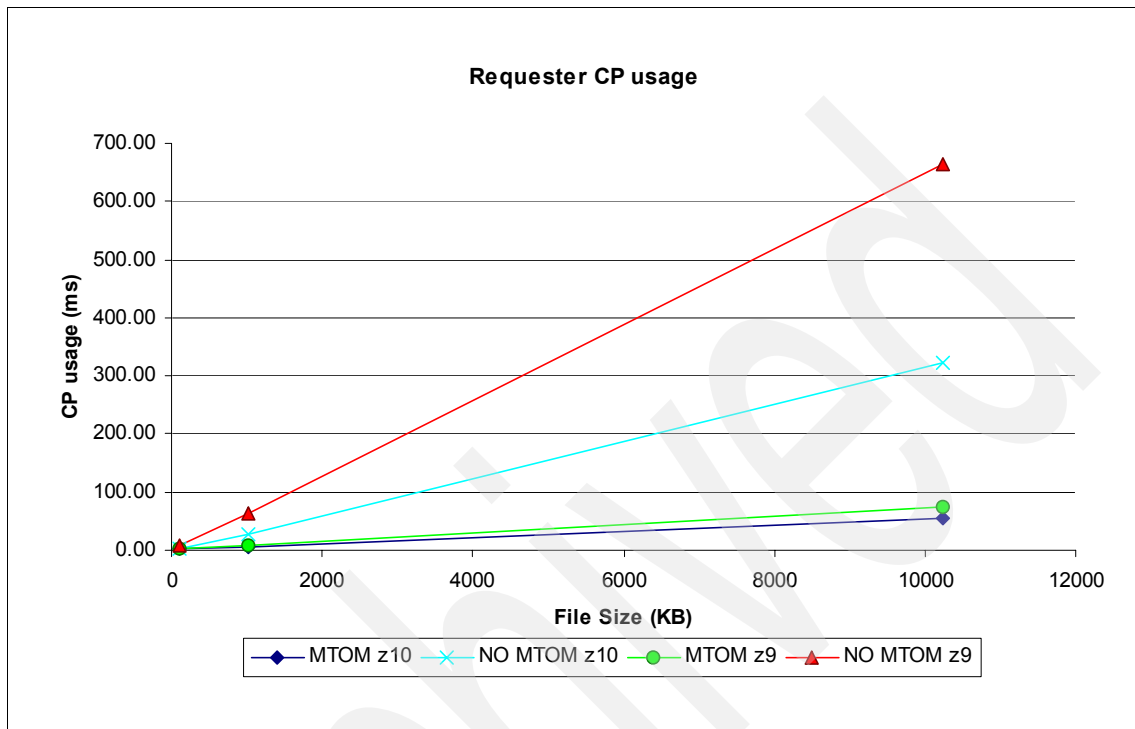


Figure 9-21 Requester CPU comparison chart

The major improvements observed for MTOM switched off can be traced back to increased computing power of the new hardware that accelerates conversion between base64 and binary.

9.3 Conclusions

In general, for our tests, we can observe savings in CPU consumption by enabling MTOM for file sizes from 5 KB. The influences on inbound SOAP messages (the receiver in our case) are more significant than on outbound SOAP messages (the provider in our case) as shown in Figure 9-22 on page 228 where we compare the no MTOM to MTOM CPU consumption ration of requester and provider.

When using MTOM/XOP, the overall payload size of data transmitted across the network is reduced. This is because binary data is converted to base64Binary when not using MTOM/XOP, but is transmitted in binary data form when using MTOM/XOP.

Note that 3 bytes of binary data converts to 4 bytes of base64Binary data. So there is a 25% reduction in the size of the binary component of the message when using MTOM/XOP.

However, there is an additional payload overhead of the Multipurpose Internet Mail Extensions (MIME) headers when using MTOM/XOP, which needs to be factored into the total message size.

The benefits of payload size reduction can best be seen by two examples:

Example 1: Sending a 4 KB binary element

► MTOM/XOP

A 4 KB binary element will be sent in a payload size of 5219 bytes.

This includes a 4096 bytes binary attachment, HTTP, SOAP, and MIME headers.

► Non-MTOM/XOP

A 4 KB binary element will be sent in a payload size of 5970 bytes.

This includes 5464 bytes of base64Binary data, HTTP and SOAP headers.

This is a saving of 245 bytes.

Example 2: Sending a 1 MB binary element

► MTOM/XOP

A 1 MB binary element will be sent in a payload size of 1,049,699 bytes.

This includes a 1,048,576 bytes binary attachment, HTTP, SOAP, and MIME headers.

► Non-MTOM/XOP

A 1 MB binary element will be sent in a payload size of 1,398,610 bytes.

This includes 1,398,104 bytes of base64Binary data, HTTP, and SOAP headers.

This is a saving of 348,911 bytes.

So significant payload data size reductions can be made with MTOM/XOP, especially for large binary attachments.

Note: For attachments of less than 1500 bytes, CICS does not use MTOM/XOP even if requested in the pipeline configuration file.

CPU benefits

For both CICS requester and provider regions, there is a CPU reduction using MTOM/XOP when the binary data size is 10 KB or more. Below 10 KB, the CPU overhead of using MTOM/XOP is minimal.

With MTOM/XOP, extra CPU is consumed in handling the MIME headers, but this overhead is minimal compared to the more significant CPU reduction due to the following factors:

- ▶ There is no longer a requirement to convert binary data to and from base64Binary.
- ▶ There is a reduction in the CPU consumed within the XML parser on inbound data, because the binary data is extracted from the SOAP XML and placed in a separate MIME header.
- ▶ The message size is smaller for large binary data.

Comparison NOMTOM / MTOM CPU consumption ratio

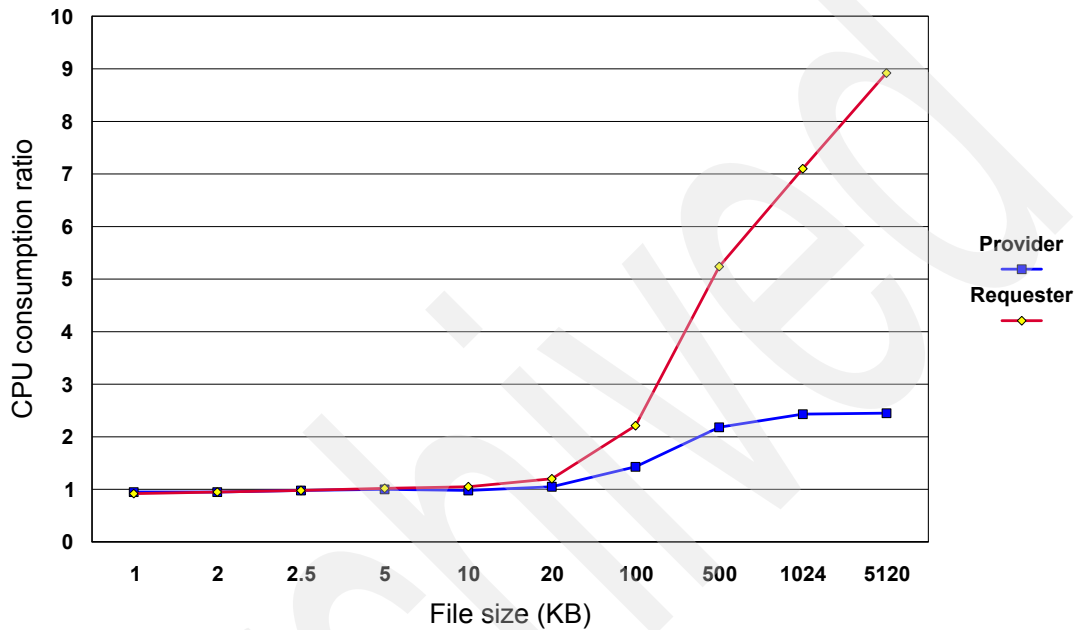


Figure 9-22 NOMTOM / MTOM CPU consumption ratio

Security scenarios

There are a number of ways to secure CICS Web services. In this chapter, we look specifically at XML digital signature processing, in which integrity is applied to the XML message to ensure that the message is not illegally modified in transit. An X.509 certificate that is used for signing a message can also be used for authentication. In service provider mode, CICS maps the certificate to a RACF user ID and places the user ID in the container DFHWS-USERID so that it is used for running the target service.

Signature processing is expensive and in some circumstances it makes sense to use an appliance such as WebSphere DataPower in conjunction with CICS Web services to help secure the services and to offload expensive operations (such as an XML digital signature) by processing the complex part of XML messages at wirespeed.

In this chapter, we compare the performance and relative merits of using XML digital signatures with and without DataPower. We also show how to obtain performance metrics using CICS Performance Analyzer (CICS PA), and how to drive a workload using a workload simulator.

10.1 Scenario overview

All the scenarios presented in this chapter are based on the IBM Redbooks publication, *Securing CICS Web Services*, SG24-7658. We take the CICS XML digital signature and DataPower security scenarios and extend them to generate a workload simulation for each. For more information on setting up the scenarios, see Chapter 8, “Environment overview” on page 177, and Chapter 7 of *Securing CICS Web Services*, SG24-7658.

We test the following three scenarios:

1. CICS XML digital signature:

CICS acts as a Web service provider, with a WebSphere Application Server (WAS) client sending a digitally signed request to CICS. CICS provides a digitally signed response. Both CICS and WebSphere Application Server are configured to use Web Services Security (WS-Security) to provide integrity with an XML digital signature.

2. DataPower with X.509 certificate:

DataPower acts as an intermediary server, verifying the signature on behalf of CICS and forwarding the signing X.509 certificate to CICS as an asserted identity.

3. DataPower with UsernameToken:

DataPower maps the X.509 certificate to a UsernameToken, and forwards the UsernameToken to CICS as an asserted identity.

This allows us to compare the performance of processing X.509 tokens (BinarySecurityTokens) in CICS with the performance of UsernameTokens.

10.1.1 Our environment

Before running any performance tests, it is worth noting a few considerations about our environment. Although we did not have a dedicated performance system, it was a lightly used system and we ran during off-peak hours.

Here are the highlights of our system:

- ▶ 6 dedicated z10 processors
- ▶ 1 LPAR
- ▶ 8 GB storage (memory)
- ▶ CICS with a WLM service class of SYSSTC
- ▶ A shared RACF installation
- ▶ Cryptographic hardware of 1 - Crypto Express2 Accelerator (CEX2A) and 7 - Crypto Express2 Coprocessors (CEX2C)
- ▶ All DASD shared
- ▶ Network - 100Mbps Ethernet

Note:

All System z models come equipped with PR/SM, the type 1 hypervisor that enables logical partitions to share system resources. PR/SM provides the ability to divide physical system resources (dedicated or shared) into isolated logical partitions.

Each logical partition operates like an independent system running its own operating environment. On the latest System z models, you can create up to 60 logical partitions running z/VM®, z/OS®, z/OS.e, Linux®, Transaction Processing Facility (TPF), or z/VSE™ on a single system. PR/SM enables each logical partition to have dedicated or shared processors and I/O, and dedicated memory (which you can dynamically reconfigure as needed).

Logical partitions with dedicated resources own the resources to which they are assigned. Logical partitions with shared resources appear to own the resources to which they are assigned, but the resources are actually shared by many logical partitions. In other words, PR/SM transforms physical resources into virtual resources so that many logical partitions can share the same physical resources.

10.1.2 Workload simulation

The scenarios documented in *Securing CICS Web Services*, SG24-7658 use a WebSphere Application Server client to drive a single secure request. This single request is of limited value when measuring performance. Factors within CICS such as program load, resource load, and caching policies will influence the figures. To average out these effects and drive a larger workload, we use a workload simulator.

There are a number of tools available that can capture and simulate workload. One such tool is WebSphere Studio Workload Simulator (WSWS).

Our primary requirement for a workload simulator is the ability to capture the existing SOAP message generated by the Web service-enabled CICS catalog manager example application and to scale it up into numerous concurrent requests.

By varying the number of simulated clients, the delay between client requests, and the “start-up” delay, we are able to drive workload in a fashion more representative of the real world. Increasing the numbers of clients and reducing delays allows us to explore the boundaries of our particular system.

10.1.3 Performance metrics

To compare the performance of our scenarios, we measure CPU consumption, response time, and throughput during each workload simulation:

- ▶ *CPU* is perhaps the most important measure. This value determines the fixed “cost” of a particular technology. It is a good indication of path-length, and it can indicate the absolute performance when all other factors are equal.
- ▶ *Response time* is the time taken from the moment the client sends the request to the moment it receives a response. A degradation in response times as client numbers are increased can help predict the system limits and give an indication of a system’s ability to scale.
- ▶ *Throughput*, measured in pages per second or transactions per second, is a measure of the concurrency of the system. This measurement, when used in conjunction with response time, can allow us to determine the point at which our system is most efficient. Typically, throughput will approach a maximum; driving workload beyond this point results in larger response times with no further gain in throughput.

Throughout this book, our aim is not to give definitive performance figures, or to tune our systems in search of ever better figures. Instead we provide performance figures from a typical system set-up and investigate the relative merits of each technology.

To present the performance figures shown in this chapter, we used CICS Performance Analyzer (CICS PA) to summarize the CICS monitoring facility (CMF) performance records generated by each scenario for each workload. For information on using CICS PA, see Chapter 4, “CICS PA” on page 71.

10.2 Preparing the scenarios

In this section we take a look at the preparation for each of the three scenarios.

10.2.1 Preparing the CICS XML digital signature scenario

In this scenario, we use the WS-Security capabilities of CICS to provide integrity at the message level with an XML digital signature. The advantage of using WS-Security over SSL is that it can provide *end-to-end* message-level security, meaning that the message can be protected even if it passes through multiple servers, called intermediaries. SSL security is considered *point-to-point*, and it is possible for the data to be decrypted prior to reaching the intended recipient.

Note: You might need to increase the value of the EDSALIM system initialization parameter. The three DLLs that must be loaded require approximately 15 MB of EDSA storage.

We have based this scenario on Chapter 7 of *Securing CICS Web Services*, SG24-7658, which takes the CICS catalog manager example application and applies security to it.

For more information on setting up the scenarios, see Chapter 8, “Environment overview” on page 177, and Chapter 7 of *Securing CICS Web Services*, SG24-7658.

We provide a summary of the software used in the scenario and the main definitions used for the CICS and WebSphere servers.

Software checklist

The software that we used in our configuration is listed in Table 10-1.

Table 10-1 Software used in the CICS XML digital signature scenario

Windows	z/OS
Internet Explorer V7.0	z/OS V1.9
Windows XP Professional Version 2002 SP3	CICS Transaction Server V3.2
IBM WebSphere Application Server V6.1.0.17	
IBM WebSphere Application Server Toolkit V6.1.1.6	

Windows	z/OS
<p>Our J2EE applications:</p> <ul style="list-style-type: none"> ► ExampleAppClientV6.ear This is the Web client for the CICS catalog example application supplied with CICS TS. 	<p>Our user-written CICS message handler programs:</p> <ul style="list-style-type: none"> ► CIWSMSG0 This program changes the transaction ID for placeOrder requests to ORDS.

Definition checklist

The definitions that we used in our configuration are listed in Table 10-2.

Table 10-2 Settings used in the CICS XML digital signature scenario

Value	CICS TS	WebSphere Application Server
IP name	wtsc66.itso.ibm.com	saz200v1.itso.ibm.com
TCP/IP port	14312	9080
Jobname	CIWSS3C2	
APPLID	A6POS3C2	
TCPIP SERVICE	S3C2	
Provider PIPELINES	EXPIPE01 for inquireSingle and inquireCatalog services EXSECURE for placeOrder service	
Configuration files	ITSO_7658_basicsoap12provider.xml for inquireSingle and inquireCatalog services ITSO_7658_wssec_signature_provider.xml for placeOrder service	

The certificates that we used in our configuration are listed in Table 10-3.

Table 10-3 Certificates used in the CICS XML digital signature scenario

Value	Format	File
CICS server certificate	PKCS12DER	CIWSS3C2.P12
CICS CA certificate	CERTDER	CIWSROOT.CER
WebSphere CA certificate	CERTDER	WASWINROOT.CER
WebSphere server certificate	PKCS12DER	WWSERV01.P12

The user IDs that we used in our configuration are listed in Table 10-4.

Table 10-4 User IDs used in the CICS XML digital signature scenario

Value	CICS TS
CICS region user ID	CIWS3D
User ID for which we want to permit access to inquireSingle and inquireCatalog services	PUBLIC01
User ID for which we want to permit access to the ORDR transaction of the placeOrder service	PRIVAT01
User ID for which we want to permit access to the ORDS transaction of the placeOrder service	USERWS01

Workload simulation

To ensure that all our tests are consistent we configure the workload simulator with the parameters shown below. While these parameters are specific to WebSphere Studio Workload Simulator (WSWS), the concepts behind them are common to any simulator:

```
clients: varied (10-200)
element_delay: 150
startup_delay: 50
repeat: 0
time_limit: 300
max_clients: 500
max_io_timeout: 500
```

We used the following values:

- ▶ The *clients* value was varied to give an appreciation of the spread of real-world loading.
- ▶ The *element_delay* is a percentage of the “thinktime” captured in the original script. It signifies the delay between successive messages sent from a single client.
- ▶ *Startup_delay* is the interval between one client and the next as they enter the system for the first time.
- ▶ *Repeat*, set to 0, signifies that we want to send infinite numbers of requests until our time-limit is hit.
- ▶ The *time-limit* is how long we want to run our workload for (in seconds).
- ▶ The *max_clients* value is the maximum number of clients WebSphere Studio Workload simulator will drive.
- ▶ Finally, *max_io_timeout* is the number of seconds we want the client to wait for a response, before timing out.

Using the foregoing values as our basic configuration, we can select and apply them to any script. This ensures we are consistent in our settings across the scenarios.

An example of a WSWS script is shown in Example 10-1. This script is essentially the HTTP message that we want to send, embedded within some control information. For readability, the embedded XML is shown here indented and with line breaks.

Example 10-1 WebSphere Studio Workload Simulator script

```
//
string CRLF = "\r\n";
startpage(1);
thinktime(1000);
soap("9.12.4.75:14312", "/exampleApp/placeOrder", 1, close, 0, start, "",
    "<soapenv:Envelope
xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
xmlns:soapenc=\"http://schemas.xmlsoap.org/soap/encoding/\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">
    <soapenv:Header>
        <wsse:Security soapenv:mustUnderstand=\"1\"
xmlns:wsse=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity
-secect-1.0.xsd\">
            <wsse:BinarySecurityToken
EncodingType=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-mes
sage-security-1.0#Base64Binary\">
```

```

ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3\" wsu:Id=\"x509bst_7\"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd\">MIIDGjCCAoOgAwIBAgIBATANBgqhkiG9w0BAQUFADCBjDEPMAoGA1UEBhMGR
nJhbmNlMRAwDgYDVQQIEwdIZXJhdWx0MRQwEgYDVQQHEwtNb250cGVsbG11c2EMMAoGA1UEChMDSUJN
MQQwCwYDVQLEwRJVFNPMR8wHQYDVQMEZXQVNXSU5ST09ULWN1cnRpZm1jYXR1MRMwEQYDVQQDEwp
XQVNXSU5ST09UMB4XDTA4MDcwMjA0MDAwMFOxDTExMDEwMTAzNTk1OVowgZExDzANBgNVBAYTBkZyYW
5jZTEQMA4GA1UECBMHSGVYyYVsdDEUMBIGAIUEBxMLTW9udHB1bGxpZXIxDDAKBgNVBAoTA0lCTTENM
AsGA1UECzMESVRTTzEiMCAgA1UEDBMQ1dBU1dJTkN1cnQwMS1jZXJ0aWZpY2F0ZTEVMBMGA1UEAxMM
VOFTV01OQ2VydDAXMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCvnuouR9PSUIJ7eBDZDt1sAZX
I4uguonNHPW1gtDyuil2eeU6Cb0RaJ7rH9F+pMFoBQoW/19P0Df1ArJR+9ke9Qa0kxfoSr6297zjhI0m
JE02/hmQ1Hdkd00LELXxRmu9mRpf2IUmX2bVGjWYXXJ9EwH3SWU0fuYTKXtdcIYQIDAQAB04GEM
IGBMD8GCWCGSAGG+EIBDQyEzBHZW51cmFOZWMQYnkGdGh1IFN1Y3VyaXR5IFN1cnZ1ciBmb3Igei9P
UyAoUkFDRikwHQYDVRO0BBYEFpd/peXs1FLhiG5BtB0vqavgVJiUMB8GA1UdIwQYMBaAFEQJQNs75jW
4hP47c2Zvb+iY880kMAOGCSqGSIb3DQEBAQUAA4GBAIBsModLosVGutRy99W6hbWlIuktnbGyhW121PW
MjeY661z6mn0QcpCCKXTuhzZdQnrNMIaHKXz9BFe9Am2+gIGYF+Vj+dZ7VLhISzbijcdc733MVD0U9
TJH23QvC/FhWHyIZ64HNIq2BcCc4HetPAftV2Tkd6zOU5Geqbs01jmG</wsse:BinarySecurityTok
en>

```

```

<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod
      Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      <ec:InclusiveNamespaces PrefixList="xsi xsd soapenv
soapenc wsse ds \" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </ds:CanonicalizationMethod>
    <ds:SignatureMethod
      Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="#wssecurity_signature_id_6">
      <ds:Transforms>
        <ds:Transform
          Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <ec:InclusiveNamespaces PrefixList="xsi xsd soapenv
soapenc wsu p635 \" xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>kWLeKibL8pzv4uMayc6I02t9W/0=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
E89JAR1E7mD32cc/a1tKPTudiMdq6r2eUs1XtDw0QGyz1u7pG00hiXRRP8bTaXccg9910pL
aRwfdWrtWqWL7kmByMc3me3D+uKtIthkSZ1nB2BaoMD1B3uCNGbnzZR1c/G/1NJwuTQIUzc
BQUxjN5HJOi5stpWEikXThyi6dN3U=</ds:SignatureValue>

```

```

        <ds:KeyInfo>
            <wsse:SecurityTokenReference>
                <wsse:Reference URI="\#x509bst_7\"
ValueTypes="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3\"/>
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>
</soapenv:Header>
<soapenv:Body wsu:Id=\"wssecurity_signature_id_6\"
xmlns:wsu=\"http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd\">
    <p635:DFHOXCMNOperation
xmlns:p635=\"http://www.DFHOXCMN.DFHOXCP5.Request.com\">
        <p635:ca_request_id>010RDR</p635:ca_request_id>
        <p635:ca_return_code>0</p635:ca_return_code>
        <p635:ca_response_message>[C@1beelbee</p635:ca_response_message>
        <p635:ca_order_request>
            <p635:ca_userid>Bob</p635:ca_userid>
            <p635:ca_charge_dept>Mydep</p635:ca_charge_dept>
            <p635:ca_item_ref_number>10</p635:ca_item_ref_number>
            <p635:ca_quantity_req>1</p635:ca_quantity_req>
            <p635:filler1 xsi:nil=\"true\"/>
        </p635:ca_order_request>
    </p635:DFHOXCMNOperation>
</soapenv:Body>
</soapenv:Envelope>" + CRLF,
    "Host: 9.12.4.75",
    "Content-Type: text/xml; charset=utf-8",
    "Content-Length: 634",
    "Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2",
    "User-Agent: Java/1.5.0",
    "Cache-Control: no-cache",
    "Pragma: no-cache",
    "Connection: keep-alive",
    "SOAPAction: \"\"");

endpage;

```

The SOAP request message contains a security header that provides the signature along with the X.509 certificate that was used to sign the body of the message. The X.509 certificate is transported within a BinarySecurityToken element. the CICS WS-Security handler (DFHWSSE1) verifies the signature and uses the certificate to determine the user ID under which the CICS application transaction is to be run (ORDS, as opposed to ORDR).

The diagram in Figure 10-1 shows the flow of this message.

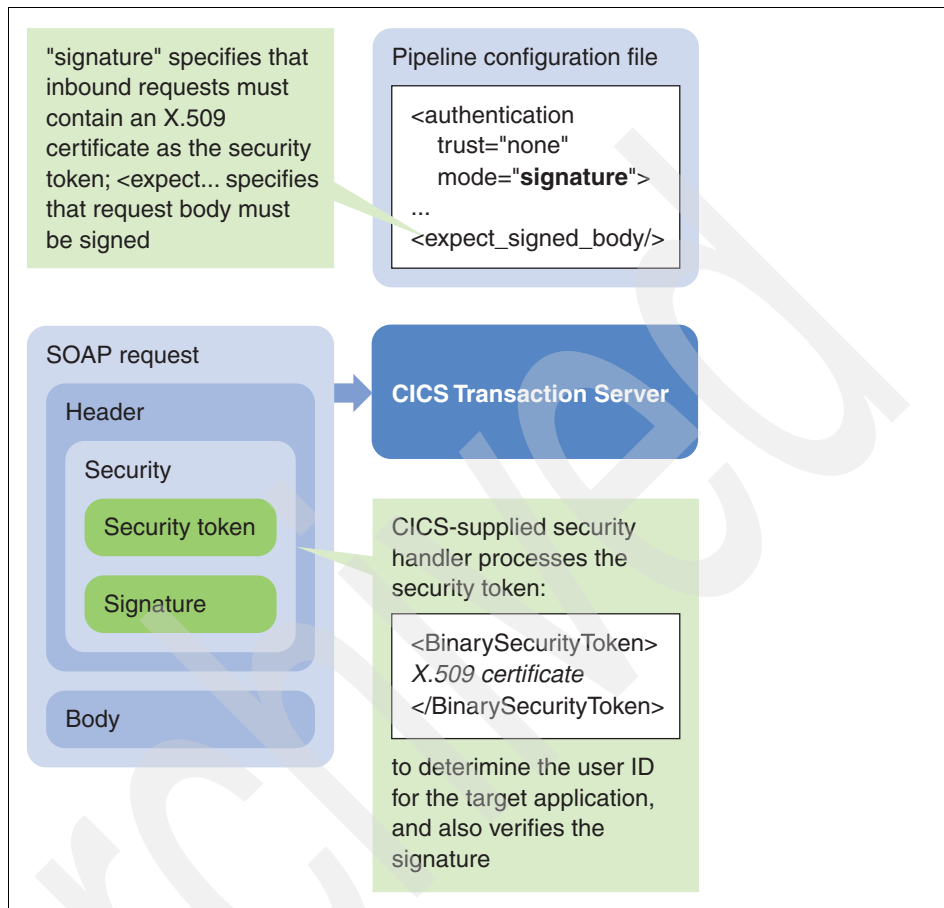


Figure 10-1 CICS XML digital signature scenario

10.2.2 Preparing the DataPower X.509 certificate scenario

DataPower can be used to offload XML processing. WS-Security is an XML-based solution and is therefore subject to offload through DataPower. Because CICS does not need to verify/generate the signature, we benefit from a reduction in CPU time.

Note: You might need to increase the value of the EDSALIM system initialization parameter. The three DLLs that must be loaded require approximately 15 MB of EDSA storage.

In this scenario we send a signed request to DataPower, which acts as an intermediary between WebSphere Application Server and CICS. DataPower processes and verifies the signature, and then forwards the request to CICS (having removed the signature from the message). The X.509 certificate that was used to sign the message has been placed in the security header of the SOAP message. CICS maps this certificate to a RACF user ID. DataPower and CICS are connected by a secured network; CICS “trusts” DataPower to verify the signature, and this trust is implied in CICS by selecting the “blind-signature” mode of pipeline operation.

Figure 10-2 illustrates this security processing scenario.

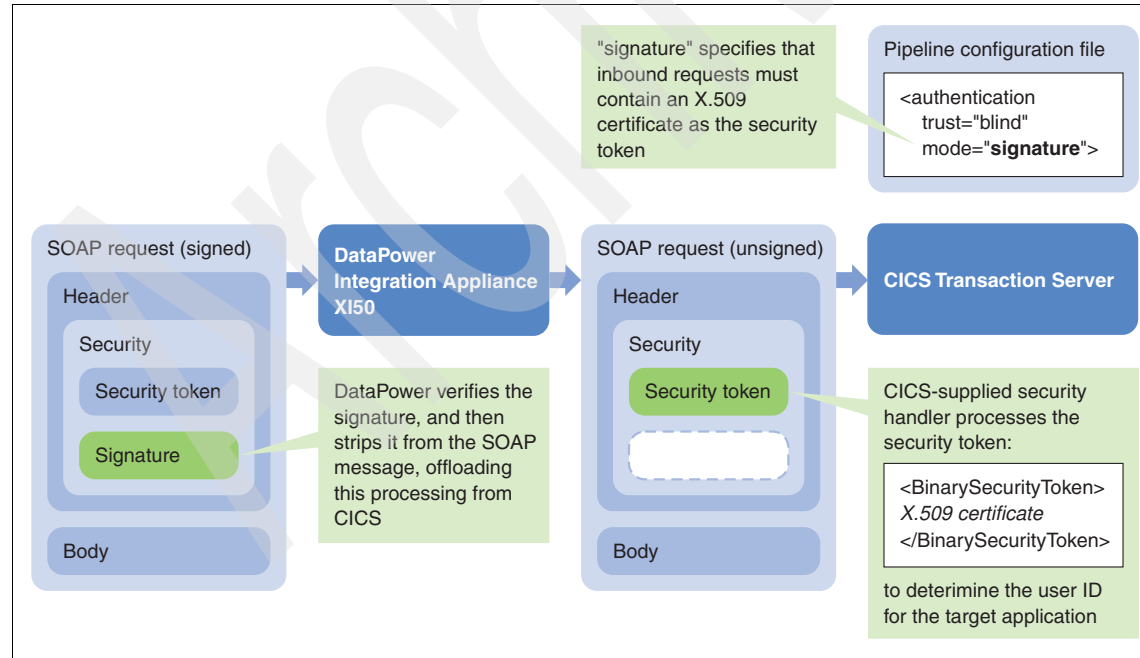


Figure 10-2 DataPower with X.509 certificate scenario

CICS then sends back an *unsigned* response to DataPower, which then signs the response back to the client.

The benefits from this scenario are two-fold. DataPower is doing the cryptographic work and verifying the signature, saving CPU cycles in CICS. In addition, because DataPower is on the same secure network, we do not need CICS to sign the response to DataPower. To the client, DataPower is the endpoint, so signing the response becomes the responsibility of DataPower.

Software checklist

The software that we used in our configuration is listed in Table 10-5.

Table 10-5 Software used in the security scenarios

Windows	z/OS
Microsoft Internet Explorer Version 6.0.2.2900.2180.xpsp.051011-1528 Update Versions: SP2	zOS V1.9
Operating System Windows XP SP4	CICS Transaction Server V3.2
IBM WebSphere Application Server - ND V6.1.0.17	RACF V1.9
Our J2EE applications: ► ExampleAppClientV6WithDsig.ear Catalog manager service requester application with digital signature enabled	

We used the WebSphere DataPower Integration Appliance XI50 (9003 type, version 2).

The firmware version installed is XI50.3.6.1.5 / Build:156262.

The configuration of the DataPower box in our second scenario is the same as that from Chapter 9 of *Securing CICS Web Services*, SG24-7658, and as such, is not duplicated here.

Note: The WSDL file can be found in the **ExampleAppClientV6WithDsig.ear** file.

Definition checklist

The definitions that we used in our configuration are listed in Table 10-6.

Table 10-6 Settings used in the security scenarios

Value	CICS TS	WebSphere Application Server	DataPower
IP name	wtsc66.itso.ibm.com	saz200v1.itso.ibm.com	datapower.itso.ral.ibm.com
IP address	9.12.4.75	9.12.5.46	9.42.170.230
TCP/IP port	14314	9080	9080

The user IDs that we used in our configuration are listed in Table 10-7.

Table 10-7 User IDs

Value	CICS TS
CICS region user ID	CIWS3D
CICS default user ID	CICSUSER

The Certificates that we used in our configuration are listed in Table 10-8.

Table 10-8 Certificates

Value	CICS TS
Signer Certificate the personal certificate used to sign the request	WASWINCert01

We loaded the certificate of the WebSphere Application Server on Windows CA in DataPower to validate the trust of the signer. See Table 10-9 for more details about this certificate.

Table 10-9 Certificate details

Field	Value
Version	3
SerialNumber	0
SignatureAlgorithm	sha1WithRSAEncryption
Issuer	C=France, ST=Herault, L=Montpellier, O=IBM, OU=ITSO, title=WASWINROOT-certificate, CN=WASWINROOT
NotBefore	2008-06-27T04:00:00Z
NotAfter	2011-01-01T03:59:59Z
Subject	C=France, ST=Herault, L=Montpellier, O=IBM, OU=ITSO, title=WASWINROOT-certificate, CN=WASWINROOT

DataPower setup

In this scenario we send exactly the same signed request as in scenario 1 (CICS XML digital signature scenario), however, the request is sent to DataPower. DataPower is configured to verify the signature and transform the message into a state where CICS can simply extract the signing certificate and trust that DataPower has verified the signature.

Figure 10-3 shows our deployed environment. The WebSphere Application Server where we ran the example CICS catalog manager application Web service client was on a Windows desktop PC located in Poughkeepsie, the WSWS engine that we used to play back the captured Web service request was located in Poughkeepsie, our DataPower box was located in Raleigh, and our CICS regions were also located in Poughkeepsie.

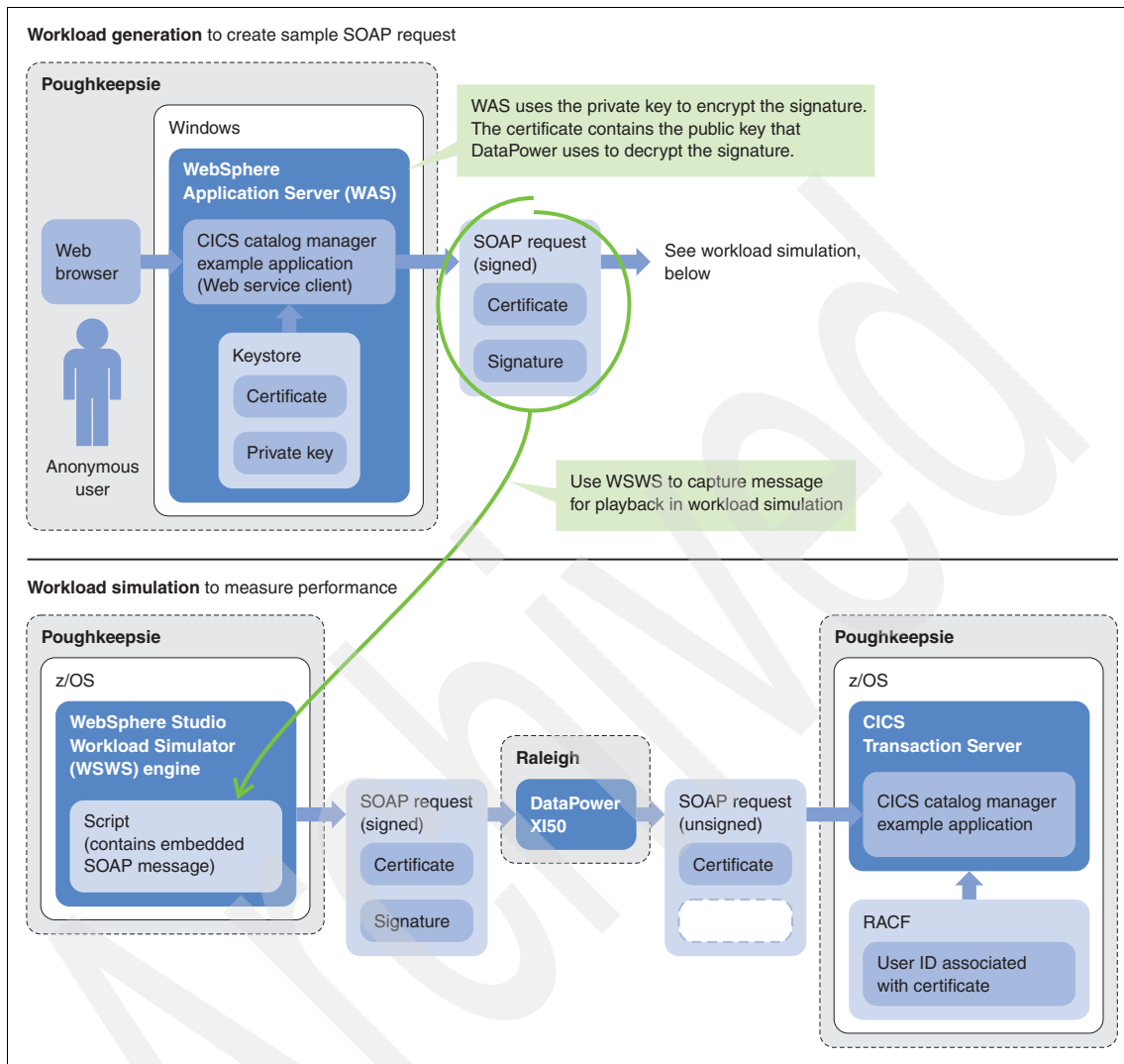


Figure 10-3 Deployed environment

10.2.3 Preparing the DataPower UsernameToken scenario

Despite the savings offered by DataPower (when processing and stripping the XML digital signature on behalf of CICS), the identity that is passed to CICS is still in the form of a BinarySecurityToken (X.509 certificate). To process this BinarySecurityToken still requires the CICS Security Handler, and this causes some CPU intensive DLL loads.

In CICS TS V3.2, support was added to optimize the security handler in cases where no real cryptographic work was needed. Instead of a full DOM parsing approach, an internal SOAP header parsing and identity checking routine is used.

Processing a UsernameToken (for example, a RACF user ID) is one such area that can take advantage of the new function. By using a UsernameToken rather than a BinarySecurityToken (assuming that there is no other cryptographic element in the message), we can process the message faster, avoid loading the large DLLs, and save another portion of CPU time. We take advantage of that feature in this scenario.

Figure 10-4 illustrates this security processing scenario. DataPower and CICS are connected by a secured network; CICS “trusts” DataPower to verify the signature, and this trust is implied in CICS by selecting the “blind-basic” mode of pipeline operation.

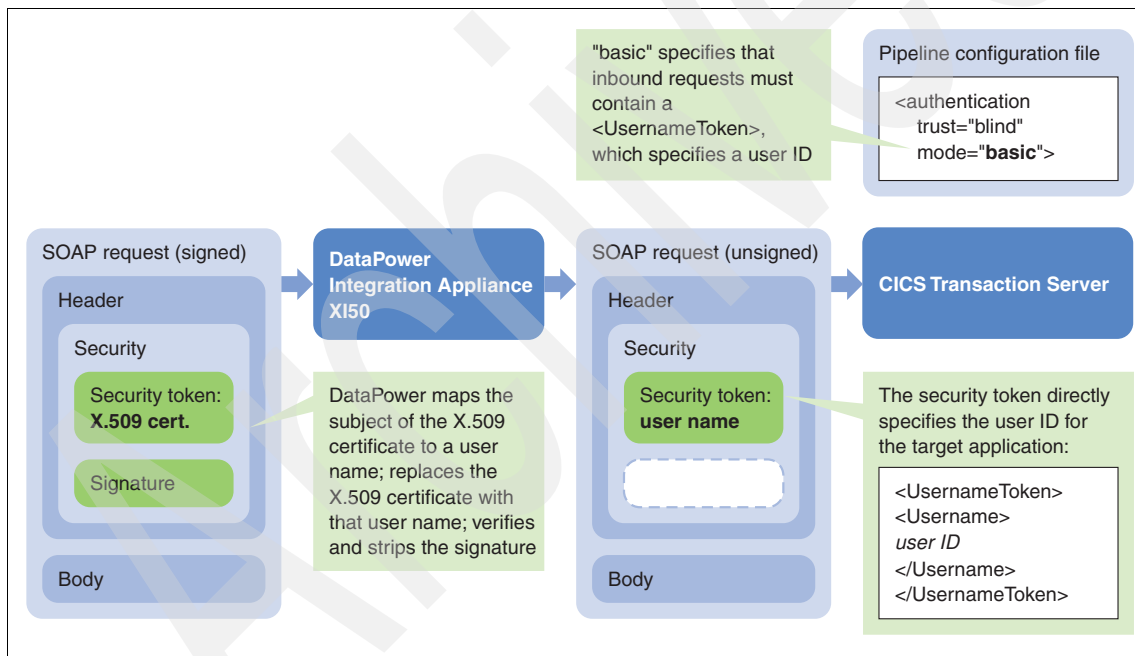


Figure 10-4 DataPower with UsernameToken scenario

Software checklist

The software that we used is listed in Figure 10-10.

Table 10-10 Software used in the security scenarios

Windows	z/OS
Microsoft Internet Explorer Version 6.0.2.2900.2180.xpsp.051011-1528 Update Versions: SP2	zOS V1.9
Operating System Windows XP SP4	CICS Transaction Server V3.2
IBM WebSphere Application Server - ND V6.1.0.17	RACF V1.9
Our J2EE applications: ► ExampleAppClientV6WithDsig.ear Catalog manager service requester application with digital signature enabled	

We used the WebSphere DataPower Integration Appliance XI50, which is a 9003 type, version 2.

The firmware version installed is XI50.3.6.1.5 / Build:156262.

The configuration of the DataPower box is very similar to that in the previous scenario. See Chapter 8, “Environment overview” on page 177 for more details.

Definition checklist

Note: The WSDL file can be found in the **ExampleAppClientV6WithDsig.ear** file.

The definitions that we used in our configuration are listed in Table 10-11.

Table 10-11 Settings used in the security scenarios

Value	CICS TS	WebSphere Application Server	DataPower
IP name	wtsc66.itso.ibm.com	saz200v1.itso.ibm.com	datapower.itso.ral.ibm.com
IP address	9.12.4.75	9.12.5.46	9.42.170.230
TCP/IP port	14314	9080	19980

The user IDs that we used in our configuration are listed in Table 10-12.

Table 10-12 User IDs

Value	CICS TS
CICS region user ID	CIWS3D
CICS default user ID	CICSUSER

The Certificates that we used in our configuration are listed in Table 10-13.

Table 10-13 Certificates

Value	CICS TS
Signer Certificate the personal certificate used to sign the request	WASWINCert01

We are going to load a certificate in DataPower to validate the trust of the signer. See Table 10-14 for more details about this certificate.

Table 10-14 Certificate details

Field	Value
Version	3
SerialNumber	0
SignatureAlgorithm	sha1WithRSAEncryption
Issuer	C=France, ST=Herault, L=Montpellier, O=IBM, OU=ITSO, title=WASWINROOT-certificate, CN=WASWINROOT
NotBefore	2008-06-27T04:00:00Z
NotAfter	2011-01-01T03:59:59Z
Subject	C=France, ST=Herault, L=Montpellier, O=IBM, OU=ITSO, title=WASWINROOT-certificate, CN=WASWINROOT

Pipeline updates

To configure CICS to expect a UsernameToken in the request message (rather than an X.509 certificate), we update its pipeline configuration file. In particular, we change the trust/mode attribute of the security handler element from *trust="blind" mode="signature"* to *trust="blind" mode="basic"*. The new pipeline configuration file is shown in Example 10-2.

Example 10-2 Blind basic pipeline configuration XML

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.ibm.com/software/http/cics/pipeline provider.xsd ">
  <service>
    <service_handler_list>
      <handler>
        <program>CIWSMSG0</program>
        <handler_parameter_list/>
      </handler>
      <wsse_handler>
        <dfhwsse_configuration version="1">
          <authentication trust="blind" mode="basic">
            </authentication>
          </dfhwsse_configuration>
        </wsse_handler>
      </service_handler_list>
      <terminal_handler>
        <cics_soap_1.2_handler/>
      </terminal_handler>
    </service>
    <apphandler>DFHPITP</apphandler>
  </provider_pipeline>
```

10.3 Running the scenarios

In this section we take a look at how we ran the scenarios.

10.3.1 Starting the simulation

We want to capture the performance of our system with minimal measurement impact. It is important to ensure that extra cycles are not being spent on unnecessary functions, so there is one final checklist that you might find useful:

1. Make sure that the CICS trace is off both internal and auxiliary.

2. Turn OFF any network sniffers or monitors that you might have used.
3. Ensure that nobody is using the system by checking that there are no unexpected active tasks.
4. Dump the SMF records just before beginning, to ensure the capture of only the intended workload.

WebSphere Studio Workload Simulator provides a dynamic monitoring panel allowing you to see *response time, throughput, number of active clients, number of SOAP requests*, and so on. These metrics allow the monitoring of health and performance characteristics during the simulation. See Figure 10-5.

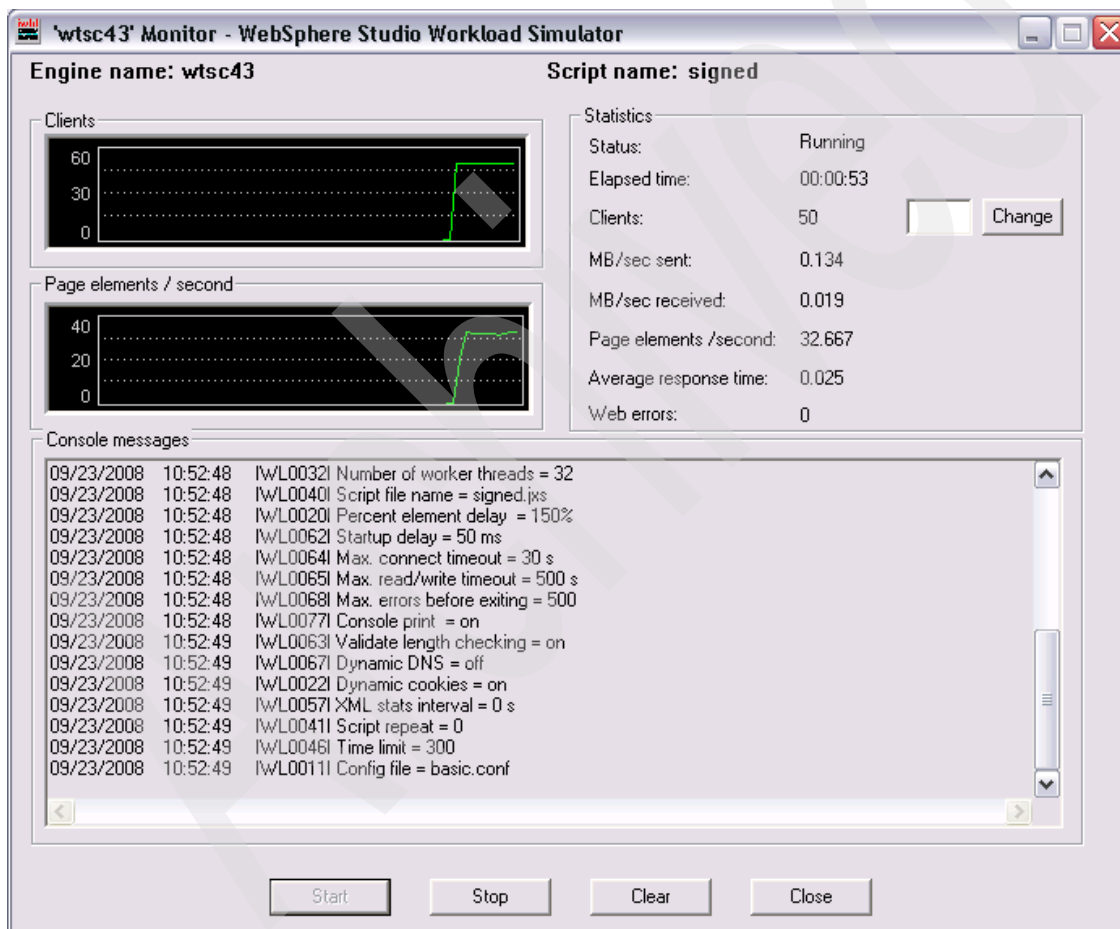


Figure 10-5 WebSphere Studio Workload Simulator running with 50 clients

10.3.2 Breakdown of results

When the workload has completed, we dump the SMF records and save them away for later analysis. CICS Performance Analyzer is used to analyze the results.

We chose to run the CICS PA report via JCL rather than using the ISPF panels because it is easier to show the necessary steps in book form. We used the JCL in Example 10-3 for all the CICS PA reports, changing only the SMF dataset and CICS APPLID as necessary. While we captured the detailed output, we do not present that data here, only each summary as necessary.

Example 10-3 CICS PA example JCL for a performance report of transactions CWXN, CPIH, ORDR and ORDS

```
//DPXZ1001 JOB MSGCLASS=H,NOTIFY=&SYSUID
// JCLLIB ORDER=CIWS.CICS.JCL
//CICSPA EXEC PGM=CPAMAIN,REGION=4M,PARM=NOSTAE
//STEPLIB DD DSN=CPA.V2R1.SCPALINK,DISP=SHR
//SYSPRINT DD SYSOUT=*
//LIST0001 DD SYSOUT=*
//CPAXW001 DD DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(200,50)),
// VOL=(, ,30)
//CPASWK01 DD DISP=(NEW,DELETE),UNIT=VIO,SPACE=(CYL,(90,50))
//CPASWK02 DD DISP=(NEW,DELETE),UNIT=VIO,SPACE=(CYL,(90,50))
//CPASWK03 DD DISP=(NEW,DELETE),UNIT=VIO,SPACE=(CYL,(90,50))
//CPASWK04 DD DISP=(NEW,DELETE),UNIT=VIO,SPACE=(CYL,(90,50))
//SYSOUT DD SYSOUT=*
//SMFIN001 DD DISP=SHR,DSN=SMFDATA.CICSRECS.G7883V00
//SYSIN DD *
CICSPA IN(SMFIN001),
APPLID(A6POS3C4),
LISTX(OUTPUT(LIST0001),
SELECT(PERFORMANCE(INCLUDE(TRAN(CWXN,CPIH,
ORDS,ORDR))))),
BY(START),
FIELDS(APPLID, CICS GENERIC APPLID
TRAN, TRANSACTION IDENTIFIER
TASKNO, TRANSACTION IDENTIFICATION NUMBER
START(TIMET), TASK START TIME
STOP(TIMET), TASK STOP TIME
CPU(TIME), CPU TIME
RESPONSE, TRANSACTION RESPONSE TIME
)),
SUMMARY(OUTPUT(SUMM0001),
SELECT(PERFORMANCE(INCLUDE(TRAN(CWXN,CPIH,
ORDS,ORDR))))),
BY(TRAN),
```

```

INTERVAL(00:01:00),
FIELDS(TRAN,
TASKCNT,
RESPONSE(AVE),
CPU(TIME(AVE))))
TRANSACTION IDENTIFIER
TOTAL TASK COUNT
TRANSACTION RESPONSE TIME
CPU TIME
/*

```

From the report output, we see that the following transactions are involved:

- CWXN** The CICS Web attach task
- ORDS** The transaction under which the provider application runs due to the context switch forced by the custom message handler CIWSMSG0
- ORDR** The transaction under which the pipeline runs; essentially a clone of CPIH initiated by the URIMAP settings

Metrics for individual transaction instances can be identified, however, it is generally more useful to view the summary (by transaction) at the end of the report.

Figure 10-6 is an example summary of a CICS PA report. We have chosen to display the *number of tasks*, *average response time*, and *average CPU time*.

V2R1M0		CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 3:57:42 9/23/2008		Data from 03:47:21 9/23/2008 to 03:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time
CWXN	1959	.0006	.0002
ORDR	1962	.0273	.0157
ORDS	1959	.0008	.0005
Total	5880	.0096	.0055

Figure 10-6 CICS PA report

10.3.3 Investigation of the limits

As more and more concurrent clients are simulated and the delay values are reduced, CICS becomes busier. Until now we have paid no attention to the CICS system settings or SIT parameters. These settings become more important as the workload increases. The following sections each describe a limiting factor discovered during our simulations.

MAXOPENTCBS

For a very small workload, there are no issues encountered. However, as the number of clients increases, we run into an unexpected problem. You can see the symptoms in WSWs from the screen capture in Figure 10-7.

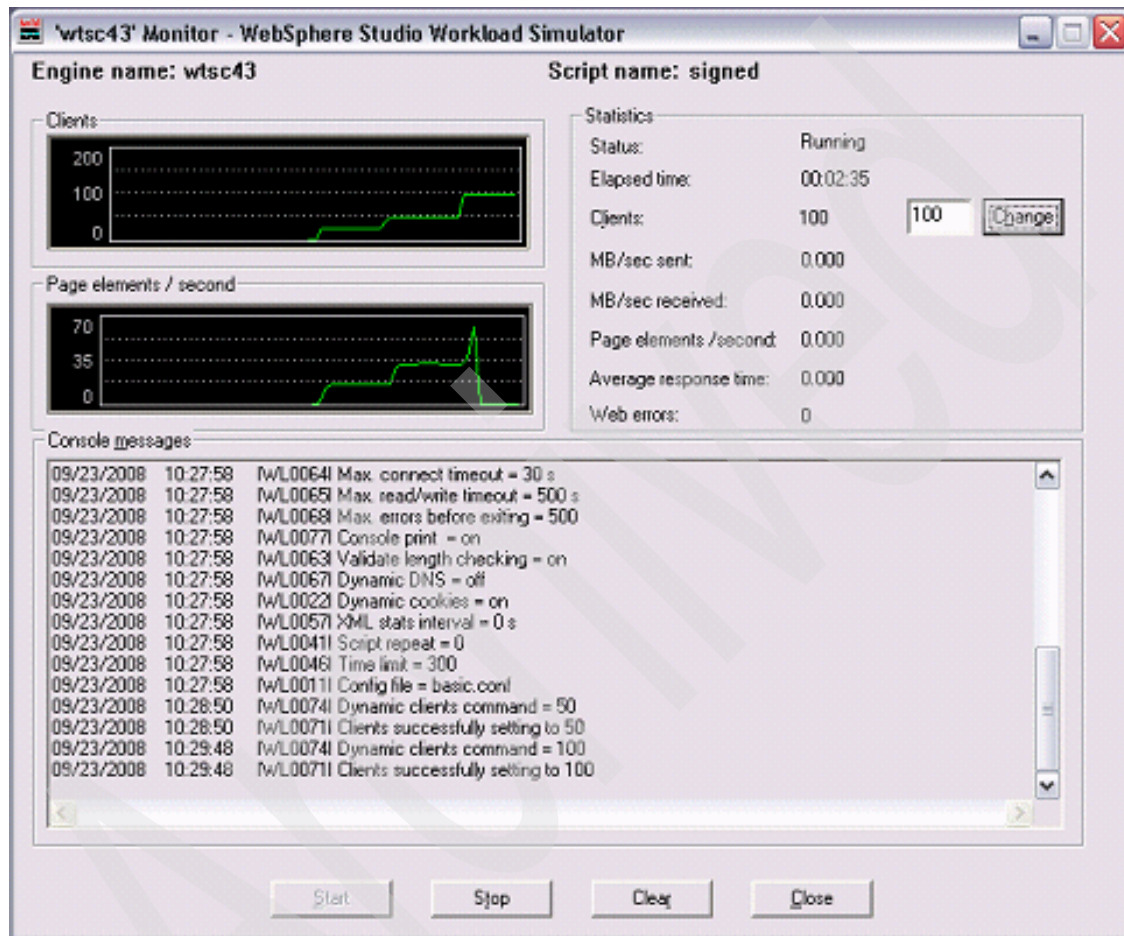


Figure 10-7 MaxOpenTCBs problem shown in WebSphere Studio Workload Simulator

The top graph in Figure 10-7 shows the number of concurrent clients, while the bottom graph shows the throughput. What we observe here is that when the number of clients reaches a certain value, the throughput stalls completely and no further requests are processed.

Inquiring on the tasks in CICS with **CEMT INQUIRE TASK** shows the results in Example 10-4.

Example 10-4 CEMT INQUIRE TASK output

```
Tas(0002799) Tra(CEMT) Fac(E030) Run Ter Pri( 255 )
  Sta(T0) Use(CICSR56 ) Uow(C30A140F69F31C87)
Tas(0012292) Tra(ORDR)          Sus Tas Pri( 001 )
  Sta(U ) Use(PRIVAT01) Uow(C30A149561624687) Hty(RZCBNOTI)
Tas(0012293) Tra(ORDR)          Sus Tas Pri( 001 )
  Sta(U ) Use(PRIVAT01) Uow(C30A14956175F687) Hty(RZCBNOTI)
Tas(0012294) Tra(ORDR)          Sus Tas Pri( 001 )
  Sta(U ) Use(PRIVAT01) Uow(C30A149561847207) Hty(RZCBNOTI)
Tas(0012295) Tra(ORDR)          Sus Tas Pri( 001 )
  Sta(U ) Use(PRIVAT01) Uow(C30A14956193D587) Hty(RZCBNOTI)
Tas(0012296) Tra(ORDR)          Sus Tas Pri( 001 )
  Sta(U ) Use(PRIVAT01) Uow(C30A149561A25187) Hty(RZCBNOTI)
```

We can see that something has caused all of our tasks to suspend: They are waiting on RZCBNOTI. RZCBNOTI is the request stream notification, so these tasks are waiting on another task. Using our knowledge of the set-up, transaction ORDR can only be waiting on one task, and that task is running the transaction ORDS, which is the CICS application handler (DFHPITP) and the end-user application. To understand why ORDS is not running, our first port of call is the dispatcher, so by using CEMT INQUIRE DISPATCHER, the information shown in Example 10-5 is obtained.

Example 10-5 CEMT INQUIRE DISPATCHER output

```
I DIS
STATUS: RESULTS - OVERTYPE TO MODIFY
Actjvmtcbs(000)
Actopentcbs(0010)
Actsslctcbs(0000)
Actxptcbs(000)
Aging( 00500 )
Maxjvmtcbs( 020 )
Maxopentcbs( 0010 )
Maxsslctcbs( 0008 )
Maxxptcbs( 005 )
Mrobatch( 001 )
Runaway( 0020000 )
Scandelay( 0100 )
Subtasks(000)
Time( 0001000 )
```

The problem becomes immediately obvious when we look at the number of OPEN TCBs. You should notice that we have hit the maximum number of OPEN TCBs allowed. Essentially what has happened is that enough ORDR transactions have entered the system at one time to take up all of the available OPEN TCBs, yet each ORDR is then waiting on an ORDS transaction to complete before it can relinquish the TCB it is using (which would allow ORDS to run). This is a deadlock situation.

To resolve the problem we can increase the number of OPEN TCBs either dynamically using **CEMT**, or by increasing the **MAXOPENTCBS** SIT parameter. When there are enough TCBs for the ORDS transaction to run, the deadlock clears itself.

Note: That if the number of clients is increased further, we run the risk of deadlock again.

In a production environment, we recommend that the number of active ORDR transactions is limited by use of a transaction class (TRANCLASS). Setting the **MAXACTIVE** value to a number less than the value of the OPEN TCBs should prevent the deadlock situation for any number of clients. We recommend setting it to half the value of **MAXOPENTCBS**, thus ensuring for each ORDR an ORDS can run without entering a deadlock on Open TCBs.

TCB performance discussion

In general, we want to avoid TCB switching. Our scenario switches TCBs as a result of a context switch: that is, we run under a new transaction, so a switch is unavoidable. However, there are many cases where a TCB switch occurs within the same transaction and it is avoidable.

For example, executed under our ORDR transaction is the custom pipeline handler **CIWSMSGO**. If this program ran in **USERKEY** we would need an L9 TCB in addition to the L8 TCB under which the pipeline handler was already running. If the end-user application running under ORDS was also defined as **USERKEY**, we would require another L9 TCB, in addition to the L8 TCB that **DFHPITP** (CICS application handler program) was running under. In this example each request would require four TCBs to complete; a very undesirable situation decreasing performance and increasing the risk of deadlock.

To avoid using more TCBs than required while improving the performance of CICS, we could ensure that the custom message handler program and the end-user application are both defined as **CICSKEY**. But **CICSKEY** does not give storage protection like **USERKEY**, and this must be taken into consideration. Provided that both the custom message handler and the end-user application have been thoroughly tested, however, this should be an acceptable situation in

a production system. Storage problems should be ironed out during test, where it is acceptable to run in USERKEY. For more information on this see the Redbooks publication, *Threadsafe Considerations for CICS*, SG24-6351.

EDSALIM

Now that we have increased the number of OPEN TCBs available to CICS, we can run more concurrent clients. However, doing so causes us to hit a further problem.

Our workload simulator reports an error (Example 10-6).

Example 10-6 Error caused by lack of EDSA

```
Status code 500(Internal Server Error) unexpected,  
server=9.12.4.75:14312, uri=/exampleApp/placeOrder, clientid=222
```

Looking at the CICS log gives us more information about the problem; in fact, it tells us all we need to know. The error in Example 10-7 is issued in the log.

Example 10-7 CICS Error message caused by lack of EDSA

```
DFHWB0728 09/23/2008 13:28:01 A6POS3C2 CWXN CICS Web attach processing  
detected a storage error within the Web receive module  
DFHWBSR. Host IP address: UNKNOWN. Client IP address:  
UNKNOWN.
```

Quite simply, CICS has run out of storage above the 16 Megabyte line. Increasing the EDSALIM cures this problem, the valid range is 10MB to 2047MB.

SOCKETS

The number of sockets available in CICS can limit the number of requests CICS can handle simultaneously. It is worth checking that your system allows a suitable number of clients to connect. This can be done by inquiring on the TCP/IP stack as shown here (see Example 10-8).

CEMT INQUIRE TCPIP

Example 10-8 CEMT INQUIRE TCPIP output

```
I TCP  
STATUS: RESULTS - OVERTYPE TO MODIFY  
Tcp Ope Act(00005) Max( 00005 ) Cic
```

In this example the maximum number of concurrent sockets (*Max*) is limited to 5, and all 5 are currently in use (*Act*). This setting could prevent you from achieving the throughput you might expect.

This setting, in conjunction with the connection backlog value defined on your TCPIPService, can determine whether clients are queued for an available socket or whether the socket connection is rejected. If, for example, your max sockets value is set to 5, and your connection backlog on the TCPIPService is also set to 5, then the first 5 clients will connect, the next 5 clients will be held waiting on the socket, and subsequent clients will be rejected with the message in Example 10-9.

Example 10-9 Unable to create sockets error message

```
+DFHS00126 W A6POS3C2 An attempt to create a socket has failed because  
the MAXSOCKETS limit has been reached.
```

Ensuring that the MAXSOCKETS value is specified correctly will prevent connections being rejected.

Note: MAXSOCKETS is specified as follows:

MAXSOCKETS=number

Specifies the maximum number of IP sockets that can be managed by the CICS sockets domain.

If the CICS region userid (the userid under which CICS is running) has superuser authority, the default value is 65535.

If the CICS region userid does not have superuser authority, the maximum possible value is the value of the MAXFILEPROC parameter in SYS1.PARMLIB member BPXPRMxx. If you specify a value greater than this in the MAXSOCKETS system initialization parameter (or by letting CICS use the default), CICS issues a message indicating the value that CICS has used.

MAXTASKS

Another parameter that can influence throughput is MAX TASKS (or MXT). This parameter determines how many active user tasks are allowed to run concurrently in CICS. Too low a value causes new tasks to be queued until running tasks have completed. Moreover, if you have tasks waiting on other tasks, as we do with ORDR and ORDS, you can enter a similar deadlock situation to that experienced with the MAXOPENTCBS described earlier.

You can determine your MAX TASKS value using the **CEMT INQUIRE SYS** command, the output of which is shown in Example 10-10.

Example 10-10 CEMT INQUIRE SYSTEM output

```
I SYS
STATUS: RESULTS - OVERTYPE TO MODIFY
Aging( 00500 )          Progautoctlg( Ctlgmodify )
Akp( 01000 )            Progautoexit( DFHPGADX )
Cicstslevel(030200)     Progautoinst( Autoactive )
Cmdprotect(Nocmdprot)   Reentprotect(Reentprot)
Db2conn()               Release(0650)
Debugtool( Nodebug )    Runaway( 0020000 )
Dfltuser(CICSUSER)      Scandelay( 0100 )
Dsalimit( 05242880 )    Sdtran(CESD)
Dsrtprogram( NONE )     Sosabovebar(Notsos)
Dtrprogram( DFHDYP )    Sosaboveline(Notsos)
Dumping( Sysdump )      Sosbelowline(Notsos)
Edsalimit( 0650117120 ) Storeprotect(Active)
Forceqr( Noforce )      Time( 0001000 )
Logdefer( 00005 )       Transolate(Inactive)
Maxtasks( 020 )
Memlimit(NoIlimit)
Mrobatch( 001 )
Oslevel(010900)
```

Here we see that we have a MaxTasks value currently set to 20.

In the screen capture from our simulator (Figure 10-8), you can see that with MAXTASKS set to 20, there are no problems at lower client numbers. The system copes with increasing numbers of clients until finally, in our example at 100 clients, too many ORDR transactions are in the system to allow ORDS to run, and therefore none of the tasks complete (the lower “throughput” graph drops to zero). We have a deadlock situation,

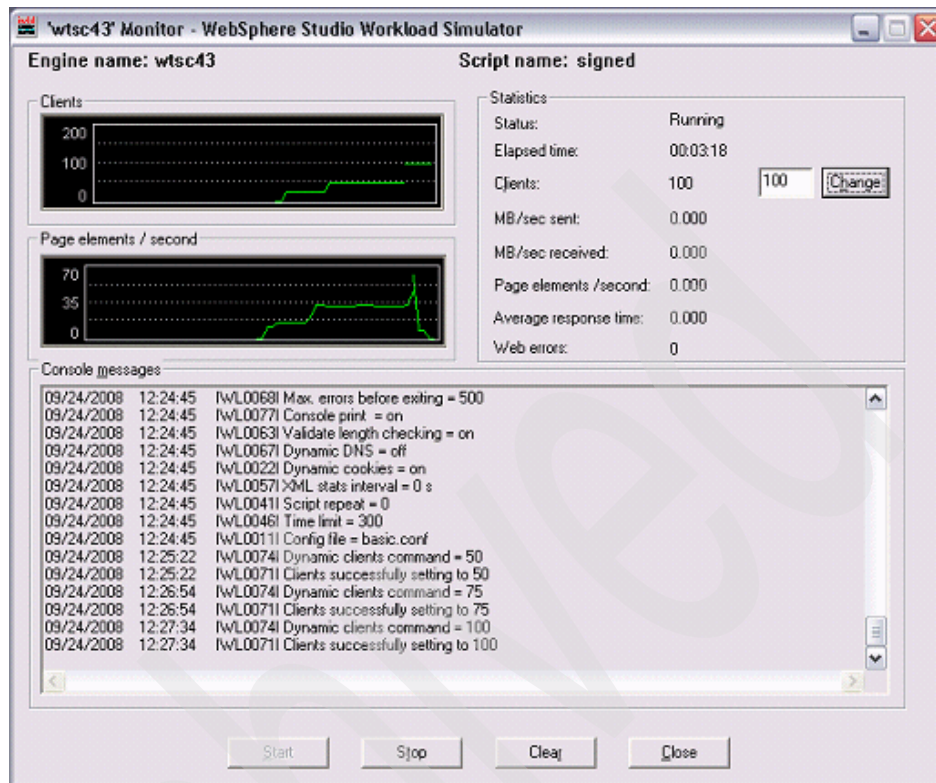


Figure 10-8 Deadlock results shown through WebSphere Studio Workload Simulator

Inquiring on TASK reveals exactly why the deadlock has occurred. From the following information, we can see that a number of ORDR transactions are waiting on the request-stream response (that is, ORDS to respond) but the ORDS transactions are also in a wait state—waiting on *MXT* (the maximum number of tasks). The ORDS transactions cannot run until there are more TASKS available to run under, but we have already reached the maximum allowed and those already in the system cannot complete; they are all waiting (forever) or until we increase MAXTASKS.

Example 10-11 CEMT INQUIRE TASK output

```
I TASK
STATUS: RESULTS - OVERTYPE TO MODIFY
Tas(0000031) Tra(CKAM)          Sus Tas Pri ( 255 )
Sta(SD) Use(CIWS3D ) Uow(C30924D44CFB5986)
Tas(0000038) Tra(OMEG)          Sus Tas Pri ( 255 )
Sta(S ) Use(CIWS3D ) Uow(C30924D88007DC89) Hty(USERWAIT)
Tas(0000039) Tra(OMEG)          Sus Tas Pri ( 255 )
```

```

Sta(S ) Use(CIWS3D ) Uow(C30924D88017BA89) Hty(USERWAIT)
Tas(0026736) Tra(CEMT) Fac(E027) Run Ter Pri( 255 )
Sta(T0) Use(CICSR6 ) Uow(C30B67880CFC0381)
Tas(0026866) Tra(ORDR)          Sus Tas Pri( 001 )
Sta(U ) Use(PRIVAT01) Uow(C30B67AA2A6F5B87) Hty(RZCBNOTI)
Tas(0026867) Tra(ORDR)          Sus Tas Pri( 001 )
Sta(U ) Use(PRIVAT01) Uow(C30B67AA2A81C387) Hty(RZCBNOTI)
Tas(0026868) Tra(ORDS)          Sus Tas Pri( 001 )
Sta(U ) Use(CIWS3D ) Uow(0000000000000000) Hty(MXT )
Tas(0026869) Tra(ORDS)          Sus Tas Pri( 001 )
Sta(U ) Use(CIWS3D ) Uow(0000000000000000) Hty(MXT )
Tas(0026870) Tra(CWXN)          Sus Tas Pri( 001 )
Sta(U ) Use(CIWS3D ) Uow(0000000000000000) Hty(MXT )

```

The minimum value of MXT to prevent deadlock can be calculated by the simple formula: $((n-1) \times c) + 1$

where:

n = tasks per client

c = maximum number of concurrent clients

In our example $n = 2$, and $c = 100$, therefore MXT would be a minimum value of 101. However, this is NOT a recommended value. For performance and throughput tuning the “+1” part of the formula should be substituted for a scaling factor (s). Where s determines the throughput you wish to achieve.

10.4 Results

In this section we present the results of our performance simulations. We have split the results into three sub-sections corresponding with the three scenarios that we want to compare (XML digital signature, DataPower with X.509 certificate, and DataPower with UsernameToken). Each sub-section presents the results of 10, 50, 75, and 100 clients, and for each of the client values we include the CICS PA Summary and the Workload Simulator summary.

10.4.1 CICS XML digital signature

Here we present the results of the CICS XML digital signature tests.

CICS XML digital signature: 10 clients

The CICS XML digital signature with 10 clients is shown in Figure 10-9.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 3:57:42 9/23/2008				Data from 03:47:21 9/23/2008 to 03:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	1959	.0006	.0002		
ORDR	1962	.0273	.0157		
ORDS	1959	.0008	.0005		
Total	5880	.0096	.0055		

Figure 10-9 CICS XML digital signature, 10 clients, 150% delay

Note: The 150% delay was set in WSWs.

Element delay: Enter a value from 0 to 9999, which represents a percentage of the original delay captured in the script. Element delay is the delay between the elements of a page. So a value of 100 executes the script with 100% of original delay (that is, use the original delay as recorded in the script).

CICS XML digital signature: 50 clients

The CICS XML digital signature with 50 clients is shown in Figure 10-10.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 3:48:17 9/23/2008				Data from 03:40:04 9/23/2008 to 03:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	9667	.0007	.0001		
ORDR	9665	.0468	.0162		
ORDS	9666	.0008	.0004		
Total	28998	.0161	.0056		

Figure 10-10 CICS XML digital signature, 50 clients, 150% delay

CICS XML digital signature: 75 Clients

The CICS XML digital signature with 75 clients is shown in Figure 10-11.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 3:41:05 9/23/2008				Data from 03:35:01 9/23/2008 to 03:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	14277	.0009	.0001		
ORDR	14275	.0621	.0168		
ORDS	14277	.0008	.0004		
Total	42829	.0212	.0058		

Figure 10-11 CICS XML digital signature, 75 clients, 150% delay

CICS XML digital signature: 100 clients

The CICS XML digital signature with 100 clients is shown in Figure 10-12.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 3:33:39 9/23/2008				Data from 03:24:23 9/23/2008 to 03:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	18411	.0011	.0001		
ORDR	18416	.1128	.0180		
ORDS	18411	.0008	.0004		
Total	55238	.0382	.0062		

Figure 10-12 CICS XML digital signature, 100 clients, 150% delay

Figure 10-13 shows CICS CPU consumption per request for the CICS XML digital signature tests with different numbers of simulated clients.

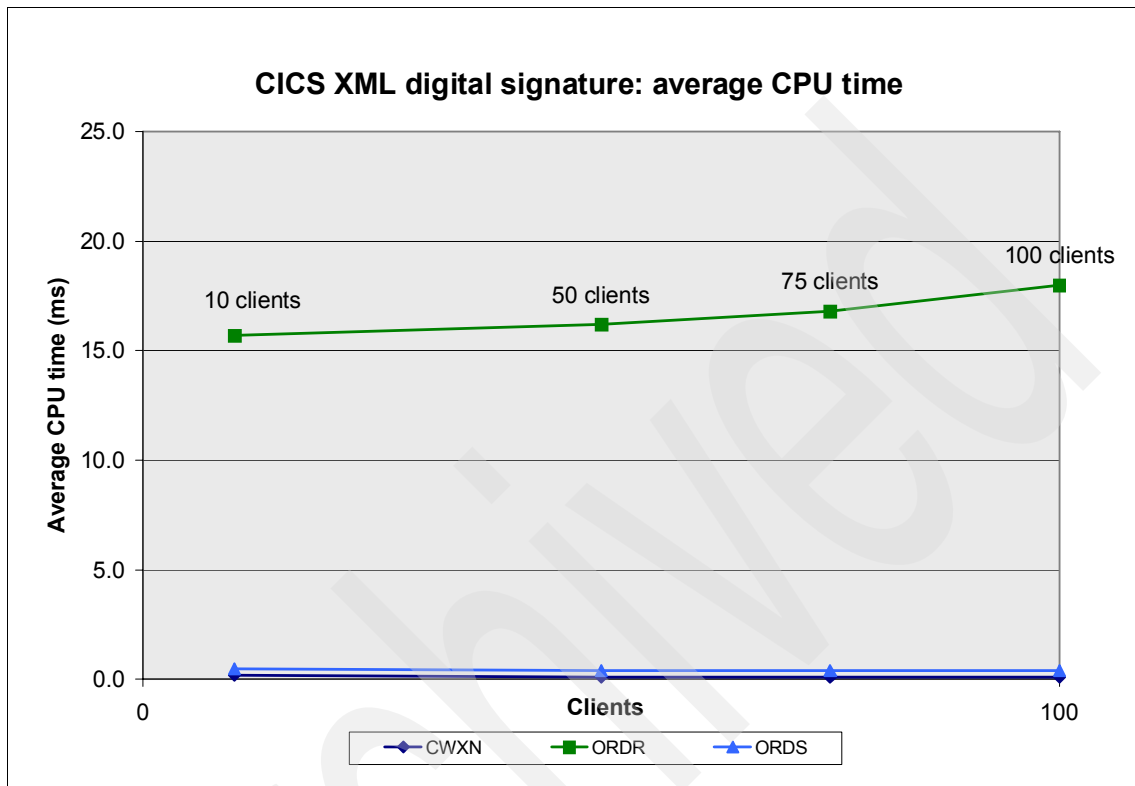


Figure 10-13 CICS XML digital signature: average CPU time

Figure 10-13 shows that the ORDR transaction is the most CPU intensive of the transactions in our scenario. This is expected, because the Security Handler runs under ORDR. The CPU time for transaction ORDR increases slightly as the number of concurrent clients increases. We observe that the ORDR CPU cost is between 15 ms and 18 ms, which is considered to be very CPU intensive for a CICS transaction. As a rough comparison, for the same Web service request without XML signature processing, we would expect between 1 ms and 2 ms of CPU time.

Figure 10-14 shows CICS XML digital signature tests with different numbers of simulated clients.

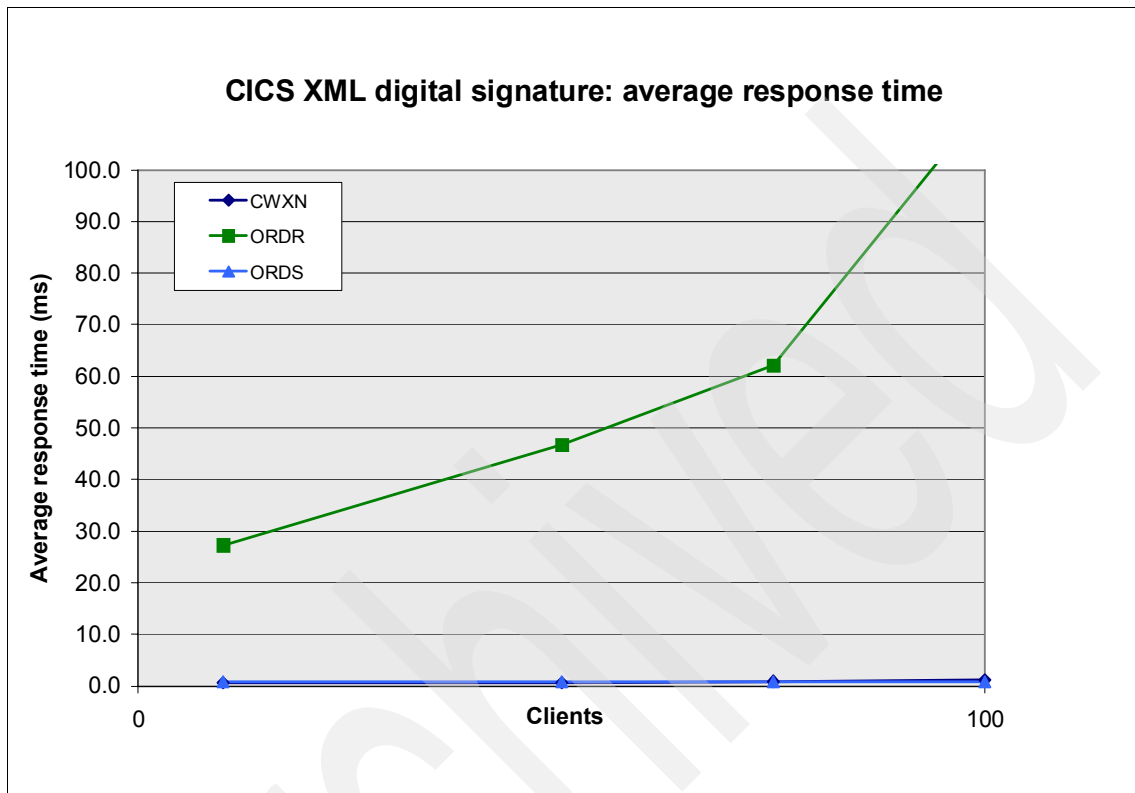


Figure 10-14 CICS XML digital signature: average response time

Figure 10-14 shows the response time as the number of concurrent clients increases. Transactions CWXN and ORDS are insignificant compared to ORDR. For ORDR up to around 75 clients, the response time increases linearly as the number of clients increase, however, beyond 75 clients we start to see the response times increase more dramatically.

10.4.2 DataPower with X.509 certificate

Here we present the results of the performance tests of the DataPower with X.509 certificate scenario.

DataPower with X.509 certificate: 10 clients

DataPower with X.509 certificate and 10 clients is shown in Figure 10-15.

V2R1M0				CICS Performance Analyze	
				Performance Summary	
SUMM0001 Printed at 4:59:20 9/23/2008				Data from 04:53:25 9/23/2008 to 04:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	1874	.0023	.0002		
ORDR	1869	.0189	.0091		
ORDS	1874	.0008	.0004		
Total	5617	.0073	.0032		

Figure 10-15 DataPower with X.509 certificate, 10 clients, 150% delay

DataPower with X.509 certificate: 50 clients

DataPower with X.509 certificate and 50 clients is shown in Figure 10-16.

V2R1M0				CICS Performance Analyze	
				Performance Summary	
SUMM0001 Printed at 4:52:39 9/23/2008				Data from 04:46:43 9/23/2008 to 04:	
		Avg		Avg	
Tran	#Tasks	Response	User	CPU	
		Time		Time	
CWXN	9177	.0010		.0002	
ORDR	9177	.0354		.0096	
ORDS	9177	.0010		.0004	
Total	27531	.0125		.0034	

Figure 10-16 DataPower with X.509 certificate, 50 clients, 150% delay

DataPower with X.509 certificate: 75 clients

DataPower with X.509 certificate and 75 clients is shown in Figure 10-17.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 4:45:46 9/23/2008				Data from 04:39:50 9/23/2008 to 04:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	13025	.0034	.0002		
ORDR	13025	.0529	.0099		
ORDS	13025	.0011	.0004		
Total	39075	.0191	.0035		

Figure 10-17 DataPower with X.509 certificate, 75 clients, 150% delayDataPower with X.509 certificate: 100 clients

DataPower with X.509 certificate and 100 clients is shown in Figure 10-18.

V2R1M0				CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 4:32:43 9/23/2008				Data from 04:26:42 9/23/2008 to 04:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time		
CWXN	16617	.0058	.0002		
ORDR	16617	.0477	.0095		
ORDS	16617	.0010	.0004		
Total	49851	.0182	.0034		

Figure 10-18 DataPower with X.509 certificate, 100 clients, 150% delay

Figure 10-19 shows CICS CPU consumption per request for the DataPower X.509 certificate tests with different numbers of simulated clients.

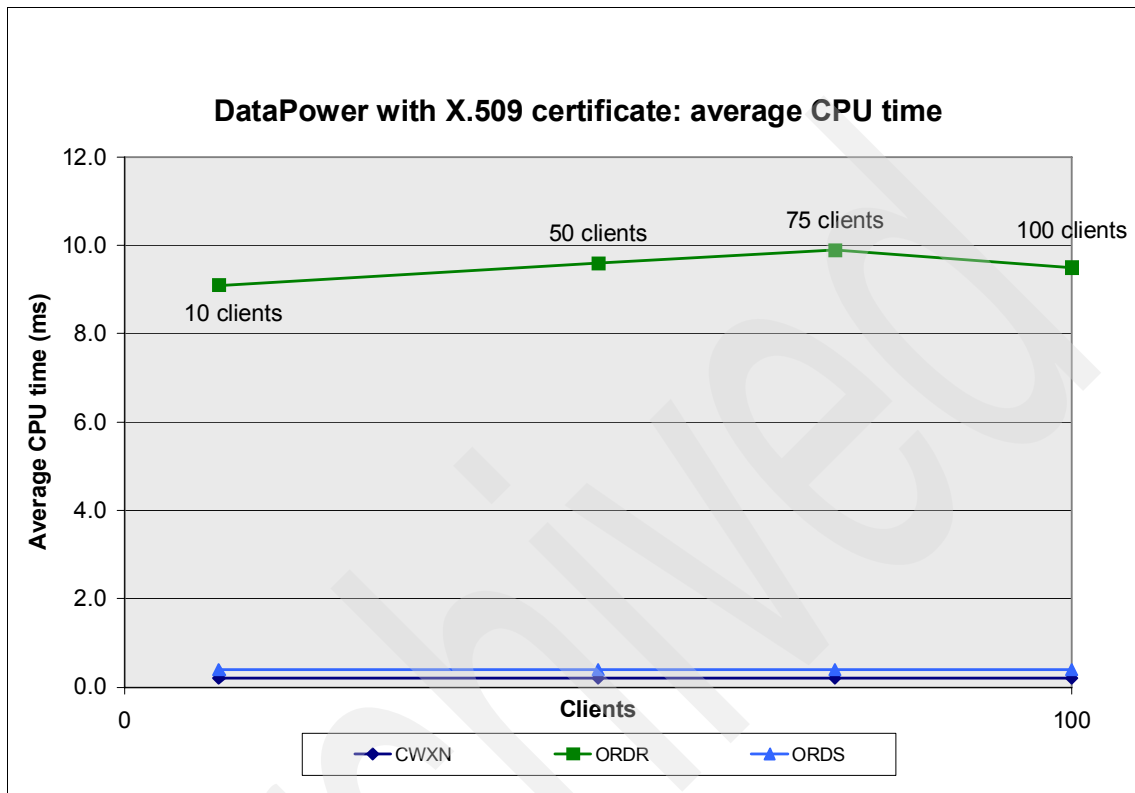


Figure 10-19 DataPower with x.509 certificate: average CPU time

Figure 10-19 shows a similar pattern to the CICS XML digital signature scenario. ORDR runs the CICS Security Handler just as it did in the previous scenario, however, because the processing of the digital signature is now performed by DataPower and not CICS, the CPU time is almost halved.

Figure 10-20 shows CICS transaction response times for the DataPower X.509 certificate tests with different numbers of simulated clients.

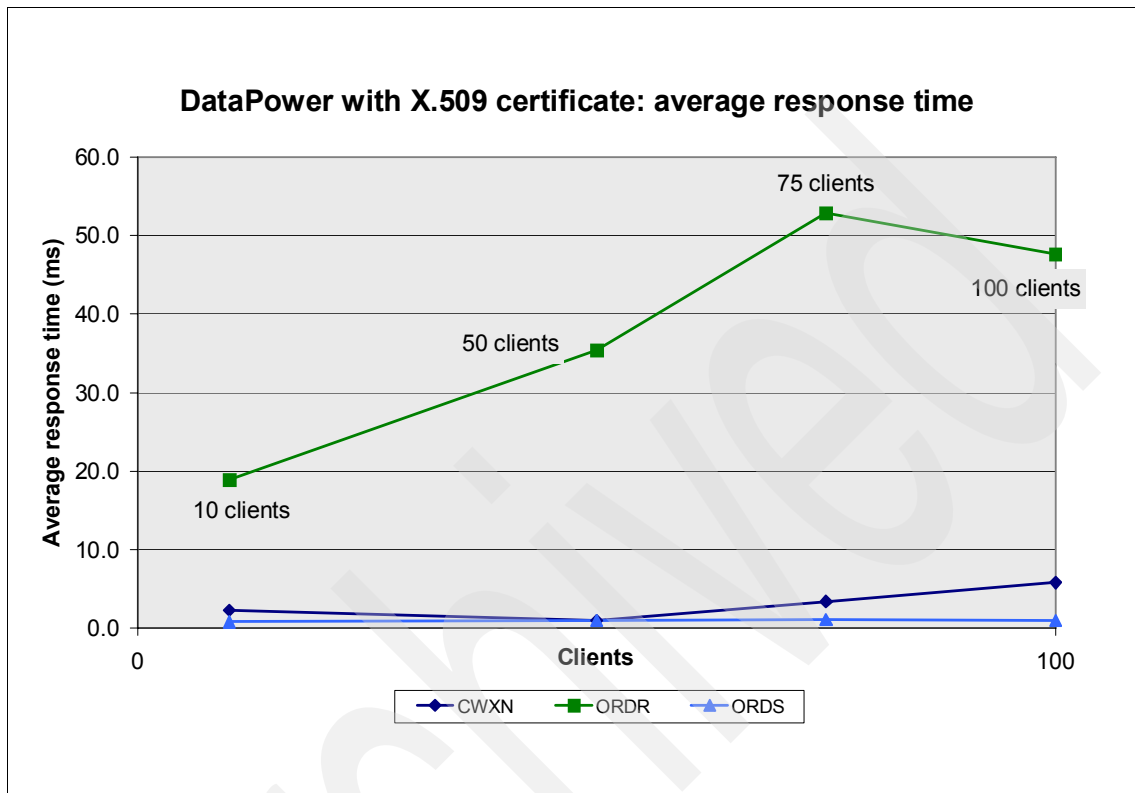


Figure 10-20 DataPower: XML digital signature with X.509 certificate: average response time

Figure 9-20 shows the response time as the number of concurrent clients increases. Again, ORDR is the transaction of significance, as the concurrent clients increase, so does the response time of the transaction. There is a slight anomaly for 75 clients, this can probably be explained as an increase in network traffic between the Poughkeepsie and Raleigh networks at the time of that test.

10.4.3 DataPower with UsernameToken

Here we present the results of the performance tests of the DataPower with UsernameToken scenario.

DataPower with UsernameToken: 10 clients

DataPower with UsernameToken and 10 clients is shown in Figure 10-21.

V2R1M0			CICS Performance Analyze	
			Performance Summary	
SUMM0001 Printed at 5:17:49 9/23/2008			Data from 05:10:00 9/23/2008 to 05:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time	
CWXN	1900	.0004	.0002	
ORDR	1900	.0023	.0010	
ORDS	1900	.0007	.0005	
Total	5700	.0012	.0006	

Figure 10-21 DataPower with UsernameToken, 10 clients, 150% delay

DataPower with UsernameToken: 50 clients

DataPower with UsernameToken and 50 clients is shown in Figure 10-22.

V2R1M0			CICS Performance Analyze	
			Performance Summary	
SUMM0001 Printed at 5:29:01 9/23/2008			Data from 05:22:11 9/23/2008 to 05:	
Tran	#Tasks	Avg Response Time	Avg User CPU Time	
CWXN	9315	.0005	.0001	
ORDR	9315	.0022	.0009	
ORDS	9315	.0007	.0004	
Total	27945	.0011	.0004	

Figure 10-22 DataPower with UsernameToken, 50 clients, 150% delay

DataPower with UsernameToken: 75 clients

DataPower with UsernameToken and 75 clients is shown in Figure 10-23.

V2R1M0			CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 5:37:49 9/23/2008			Data from 05:30:24 9/23/2008 to 05:	
Tran	#Tasks	Avg Response Time	Avg User Time	CPU
CWXN	13181	.0005	.0001	
ORDR	13181	.0021	.0008	
ORDS	13181	.0006	.0004	
Total	39543	.0011	.0004	

Figure 10-23 DataPower with UsernameToken, 75 clients, 150% delay

DataPower UsernameToken: 100 clients

DataPower with UsernameToken and 100 clients is shown in Figure 10-24.

V2R1M0			CICS Performance Analyze Performance Summary	
SUMM0001 Printed at 5:46:10 9/23/2008			Data from 05:39:19 9/23/2008 to 05:	
Tran	#Tasks	Avg Response Time	Avg User Time	CPU
CWXN	16435	.0007	.0001	
ORDR	16435	.0019	.0008	
ORDS	16435	.0006	.0003	
Total	49305	.0010	.0004	

Figure 10-24 DataPower with UsernameToken, 100 clients, 150% delay

Figure 10-25 shows CICS CPU consumption per request for the DataPower UsernameToken tests with different numbers of simulated clients.

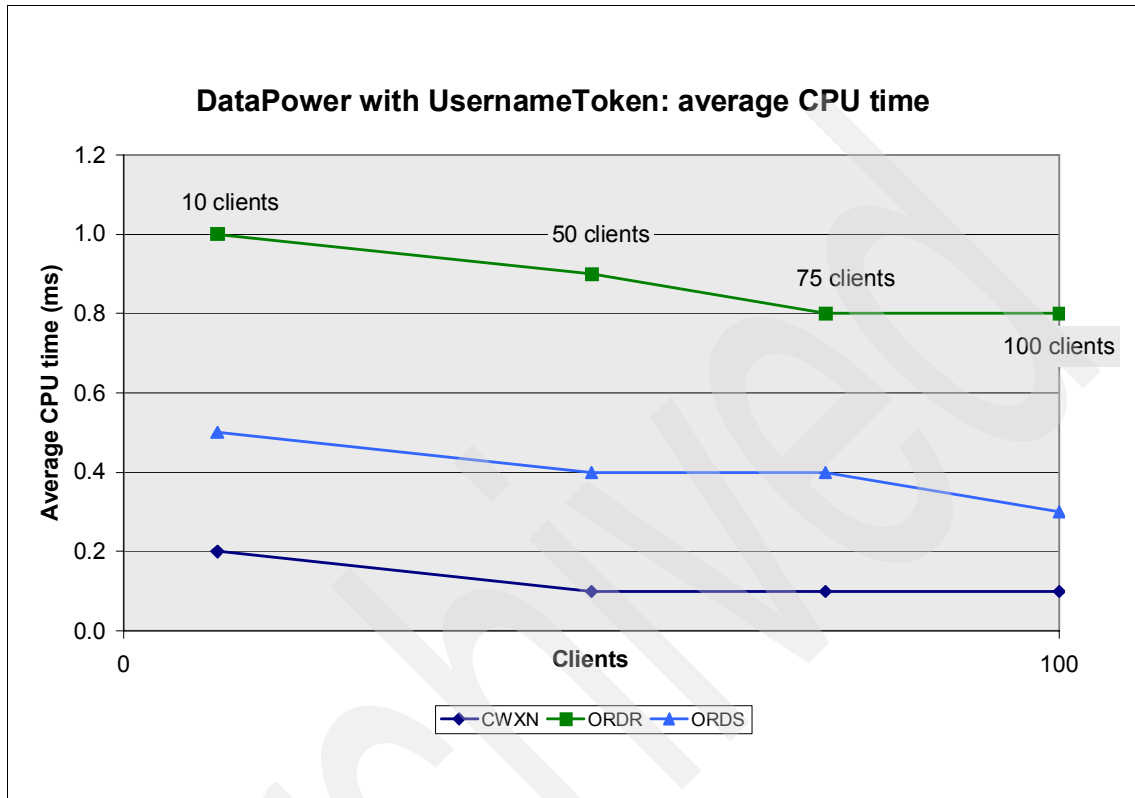


Figure 10-25 DataPower with UsernameToken: average CPU time

Figure 10-25 shows a dramatic reduction in CPU time for the transaction ORDR compared with the previous two scenarios. The scale of the graph has been changed to accommodate this reduction of CPU time in ORDR, and as a consequence CWXN and ORDS appear more significant (their values are actually consistent with the previous two scenarios).

The reduction in CPU time is expected because the Security Handler is no longer loaded. It has been replaced by an internal parsing and checking routine, made possible by the extra level of mapping in DataPower from an X.509 certificate to a UsernameToken.

Figure 10-26 shows CICS transaction response times for the DataPower UsernameToken tests with different numbers of simulated clients.

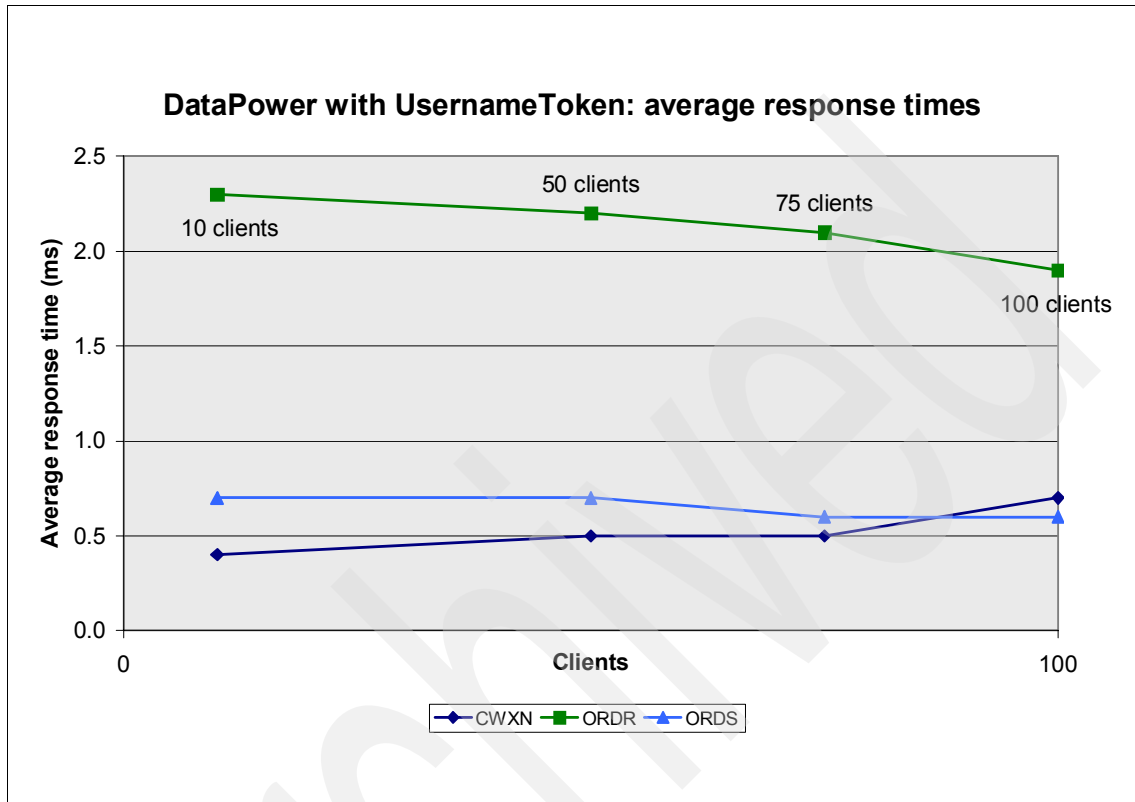


Figure 10-26 DataPower with UsernameToken: average response time

Figure 10-26 shows the average response time for transactions CWXN, ORDS and ORDR as the number of concurrent clients increases. ORDR exhibits significantly improved (quicker) response times compared to the previous two scenarios, a direct benefit of not loading the CICS Security Handler.

10.4.4 Comparison of CPU time

From the figures presented in the previous sections, we can see that the results for transactions CWXN and ORDS are consistent and identical across our scenarios. We expect this because the transactions do not differ in the task they perform.

However, ORDR shows quite a dramatic difference between scenarios. ORDR is a clone of CPIH (the CICS pipeline handler). It is in this transaction we expect to see differences, specifically between scenario 1 where all the XML digital signature processing is done by CICS, scenario 2, where only the parsing of the SOAP and identity checking is done under the Security Handler, and finally to scenario 3, where a lightweight internal routine does the parsing and identity checking.

The graph in Figure 10-27 shows the CPU time used for each of the scenarios as the number of clients is varied.

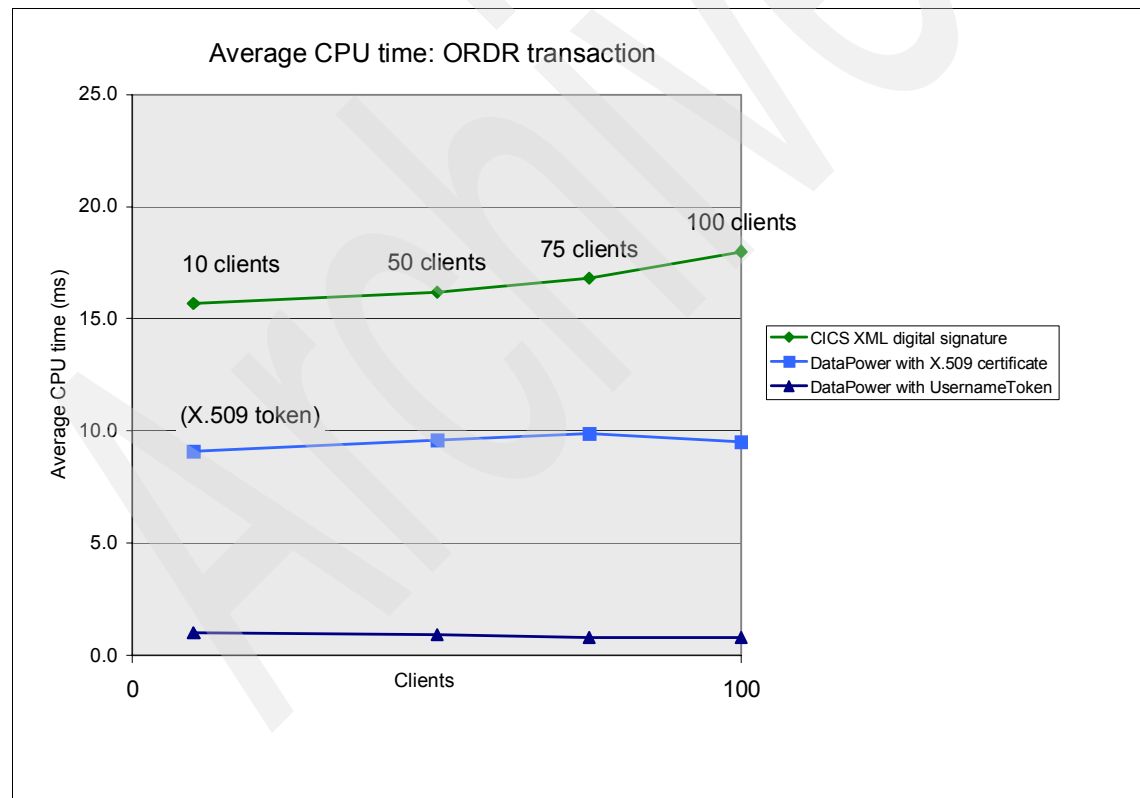


Figure 10-27 Comparison of security scenarios: average CPU time

Immediately we notice that the CICS XML digital signature processing is CPU intensive. Then, by offloading the signature verification to DataPower and forwarding the X.509 certificate to CICS the average CPU time for transaction ORDR is almost halved.

Even more interesting is the scenario with the CPU costs for DataPower with UsernameToken. These results show another significant reduction in CPU time; around a factor of 10 better than the DataPower with X.509 scenario, and around a factor of 20 better than CICS XML digital signature scenario.

The reason for such a dramatic improvement is that by providing CICS with a UsernameToken rather than an X.509 token, CICS can optimize the security handler. The security handler is known to have significant overhead loading the DLLs necessarily to provide the cryptographic function. In CICS TS V3.2, additional logic was introduced to support basic authentication type scenarios that did not require encryption or signature work. For more information on configuring CICS to use blind trust, see:

http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp?topic=/com.ibm.cics.ts.webservices.doc/reference/pipeline/dfhws_authentication.htmlmode

We are taking advantage of the optimized code in CICS TS V3.2.

10.4.5 Comparison of response time

Looking now at the response times for the same scenarios and for transaction ORDR, we can see a similar pattern of improvement. With lower numbers of clients, CICS XML digital signature processing responds slower than the two DataPower scenarios, and of the DataPower scenarios, the UsernameToken scenario performs the quickest.

However, there is then a trend for the DataPower X.509 certificate response times and CICS XML digital signature response times to increase at the same rate, yet the DataPower with UsernameToken response times remain constant. This suggests that the Security Handler is a large contributor to the overall response time.

ORDR response times in the CICS XML digital signature scenario start to be significantly impacted beyond about 75 concurrent clients. By contrast, both DataPower scenarios remain at consistent levels. See Figure 10-28.

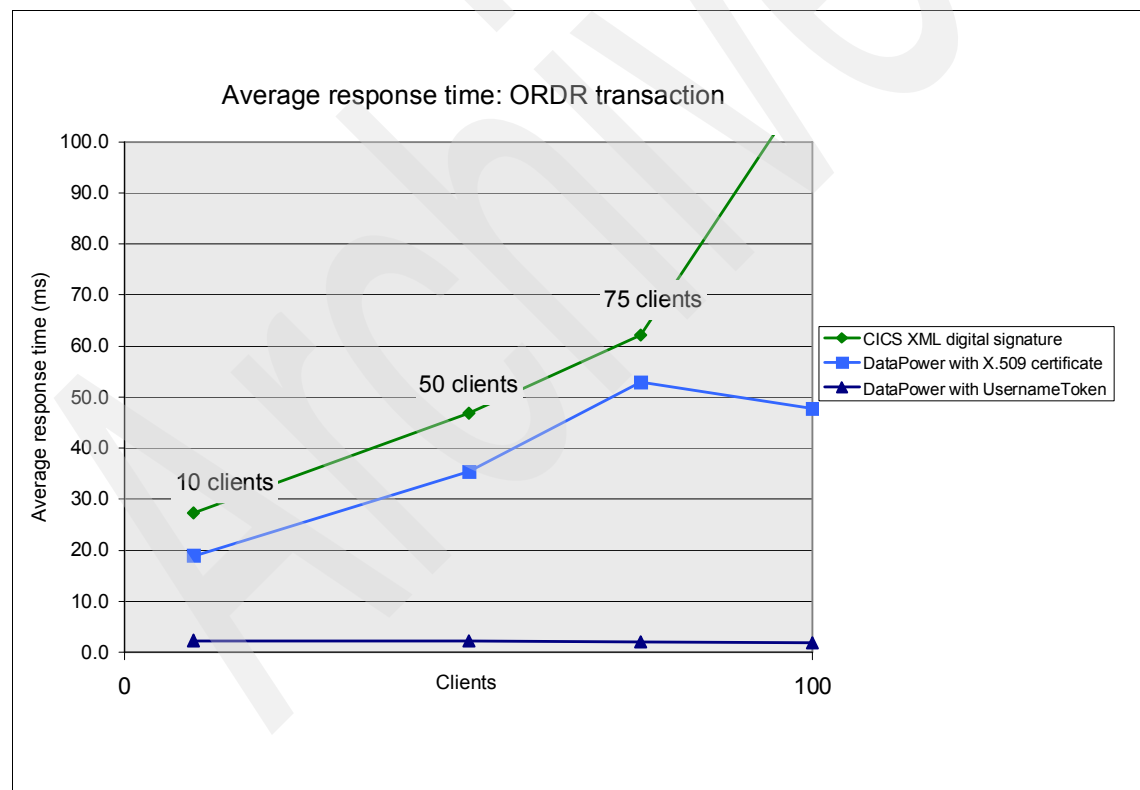


Figure 10-28 Comparison of security scenarios: average response time

10.5 Conclusions

Security is often at odds with performance, that is, the most secure technologies (for example, XML digital signature processing) are often expensive to implement. Inevitably, the need to implement a solution that meets the security requirements must be balanced against the need to meet the solution's performance objectives.

If you plan to use XML digital signature processing, our security scenario performance tests provide useful performance planning information, and we also suggest techniques that may help you to meet the security requirements but also minimize the impact on performance.

After you have a clear understanding of the security requirements and security implementation options, consider the following CICS performance related questions:

- ▶ Will you use transport based security or SOAP message security?

SSL/TLS is a mature technology that has been optimized over a long period of time, and there are ways of optimizing performance such as persistent TCP/IP connections and SSL session ID reuse. These optimizations mean that expensive security functions, such as SSL handshaking, can be avoided for service requests following the initial handshake.

WS-Security support, in comparison, is completely stateless, and expensive security functions, such as XML digital signature validation, are repeated for each service request.

In practice, the most optimum solution is often to use a combination transport based and SOAP message based security, for example, transport based security for confidentiality and data integrity and SOAP message based security for transport of security tokens.

- ▶ Where is authentication done and how many times?

The cost of authentication in CICS is dependent on the security token used in the authentication. Simple security tokens like UsernameTokens are less expensive than binary security tokens such as X.509 certificates.

If authentication is done by an intermediary server, where possible, the intermediary server should flow an asserted identity to CICS. This avoids the overhead of authenticating multiple times.

- ▶ Are you using hardware cryptographic devices in an optimal way?

Cryptographic hardware and ICSF (Integrated Cryptographic Hardware Facility) is a prerequisite for CICS XML digital signature and XML encryption processing. It is also required in order to maximize performance when CICS is configured to use SSL/TLS.

- Is there a need for an SOA appliance?

An SOA appliance such as WebSphere DataPower can be used in conjunction with CICS Web services to help secure the services and to offload expensive operations by processing the complex part of XML messages (such as an XML signature) at wirespeed.

WebSphere DataPower is also an ideal solution for other expensive tasks such as auditing and XML validation.



Using IBM Tivoli Monitoring Tools

In this chapter, we focus on using the IBM Tivoli Monitoring tools to help identify problems that can affect the performance of Web Services. We specifically use ITCAM for SOA and OMEGAMON XE for CICS on z/OS to show how these tools can be of benefit to identify the cause when the performance of a Web service in CICS becomes unacceptable.

11.1 Configuration

In this chapter we are using the application described in the MTOM scenario. This includes the inquireItem Web service extended to be a service requestor and invoking the imageServer Web service, We show how the monitoring can be used to identify the cause of slow response time.

In order to get the best out of any monitoring tools, it is important that they be configured to meet the goals of your environment. While the OMEGAMON and ITCAM products deliver product provided situations, it is unlikely that they would completely meet your needs.

11.1.1 Creating a situation

First you have to create a situation that generates an alert when the required performance is not being achieved. Use the following steps to create a new situation:

- 1. First select **Situations** from the Performance Summary navigator item for a CICS region. It does not matter which CICS region is selected (Figure 11-1).

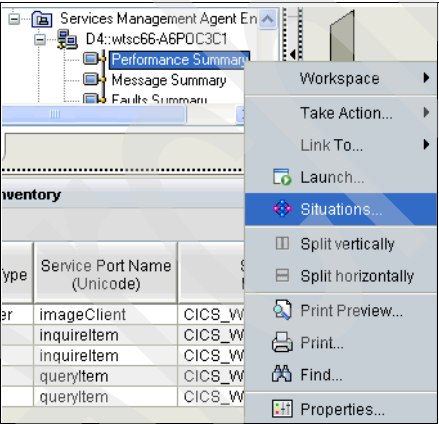


Figure 11-1 Situation Selection

- 2. Click the **Create Situation** icon (Figure 11-2).



Figure 11-2 Create Situation icon

3. This action presents a pop-up where you can name the situation (Figure 11-3). After entering the situation name and description, click **OK**.

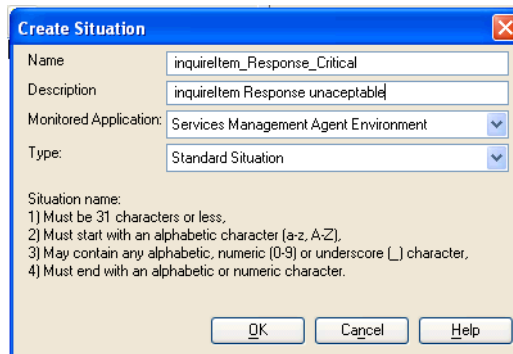
A screenshot of a 'Create Situation' dialog box. The dialog has a blue title bar with the text 'Create Situation' and a close button. It contains several input fields: 'Name' with the text 'inquireItem_Response_Critical', 'Description' with the text 'inquireItem Response unacceptable', 'Monitored Application:' with a dropdown menu showing 'Services Management Agent Environment', and 'Type:' with a dropdown menu showing 'Standard Situation'. Below these fields, there is a section titled 'Situation name:' followed by four numbered rules: 1) Must be 31 characters or less, 2) Must start with an alphabetic character (a-z, A-Z), 3) May contain any alphabetic, numeric (0-9) or underscore (_) character, and 4) Must end with an alphabetic or numeric character. At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

Figure 11-3 Situation name pop-up

4. Select the attributes. We selected the following attributes for this situation (Figure 11-4):
- *Average Elapsed Message Round Trip Time* is the mean time in milliseconds between the request and response. Because we are looking from the perspective of a service provider, then it is from the request being received and the response being sent.
 - *Interval Status* is simply an indication of whether a complete sample was taken. If a sample is not complete, then the values might not represent a true sample and should probably be ignored.
 - *Service Port Name (Unicode)* is the Web Service name as known to CICS. In this case we are looking at a particular Web service rather than all. It is also likely that different Web Services would have different performance objectives. It would therefore make sense to create situations for each Web Service that is to be managed.

5. Here we have set the values for the attributes that we would like to be checked for the situation.

Description

InquireItem Response unacceptable

Formula

	Average Elapsed Message Round Trip Time	Interval Status	Service Port Name (Unicode)
1	> 1000	== Complete	== 'InquireItem'
2			
3			

Click inside a cell of the formula editor to see a description of the attribute for that column and to compose the expression.

Add a condition by clicking **Add conditions** and selecting the situations to embed or attributes to include.

Situation Formula Capacity 18%

Sampling interval

00 : 05 : 00
ddd hh mm ss

Sound

☐ Enable critical.wav

Play Edit...

State

Critical

☒ Run at startup

Figure 11-4 Situation rules

6. We have set the sampling interval to 5 minutes and the situation state to Critical (Figure 11-5):
 - It makes little sense to have the value set to anything else, because that is the value of the interval length. Checking the situation more often would not result in any more data. Less often might miss some samples. Another point about situation sampling intervals is that if multiple situations are active against the same attribute group with the same sampling interval, then the ITM framework is able to combine the situations such that the data is only obtained once. This can definitely reduce the overhead of many situations.
 - The situation state has been set to **Critical** because we would like a red alert when this is true. Check the **Run at startup** box to insure that the situation starts automatically if the agent is recycled.

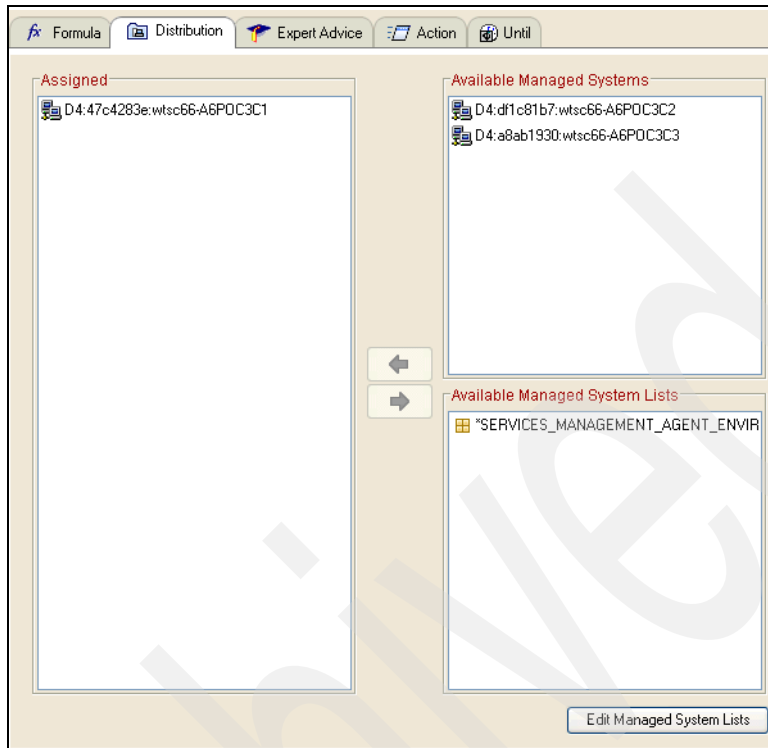


Figure 11-5 Situation assignment

- One important step is assignment. In this case we are looking at a specific Web Service in a specific CICS region. There is no need to distribute the situation to more managed systems. If this is not the case, then the situation should be distributed to all the places where it could be true.
 - If it is possible that it can be true in any place where ITCAM SOA is installed, use the system generated Managed System List (MSL) *SERVICES_MANAGEMENT_AGENT_ENVIR. The system will automatically maintain this as a list of all the managed systems.
7. Click **OK** to save the situation. Then right-click the situation name in the left pane and select start the situation.

11.1.2 Adding a dynamic workspace link

One of the powerful features of ITM is the ability to create links from one product workspace to another. These links can provide a useful tool in allowing you to navigate quickly from one view of a piece of work to another view of the same item but from a different perspective.

In some cases these workspace links are provided by the products themselves. If a link is not provided that makes sense for your environment, it is a relatively simple task to add it yourself.

In the following pages we demonstrate how to create a link from ITCAM for SOA to OMEGAMON XE for CICS.

In this case we start with the Operation Flow for an Application Server workspace in the Interaction detail display:

1. Highlight a Web service, then right-click and select **Link To..** → **Link Wizard..** (Figure 11-6).

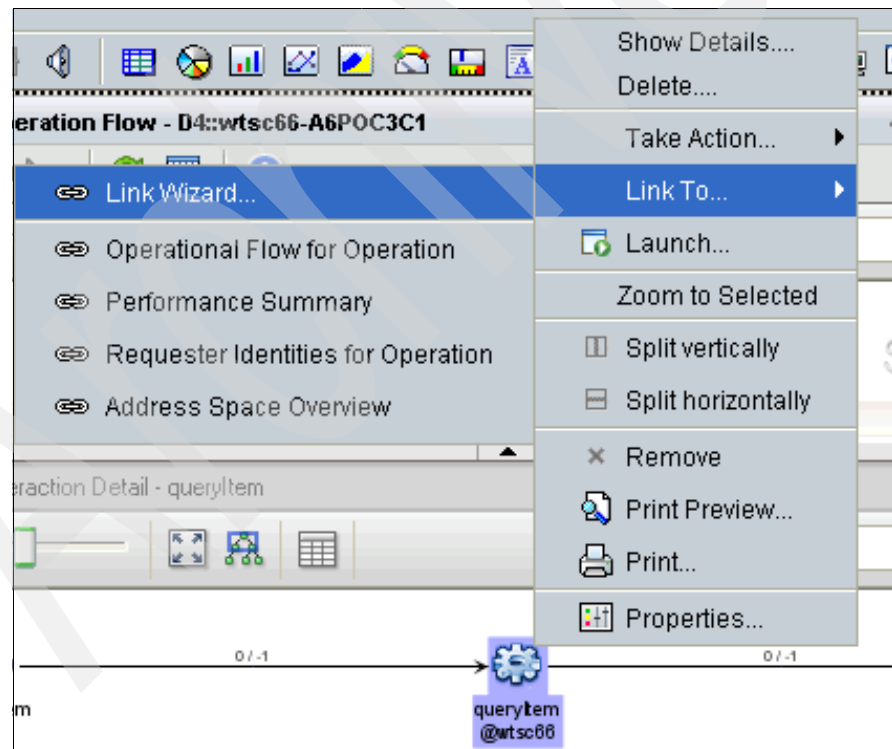


Figure 11-6 Link Wizard selection

2. Next check the **Dynamic** workspace link. This allows the TEP to dynamically select the target of the link based upon the context from where the link is selected (Figure 11-7).

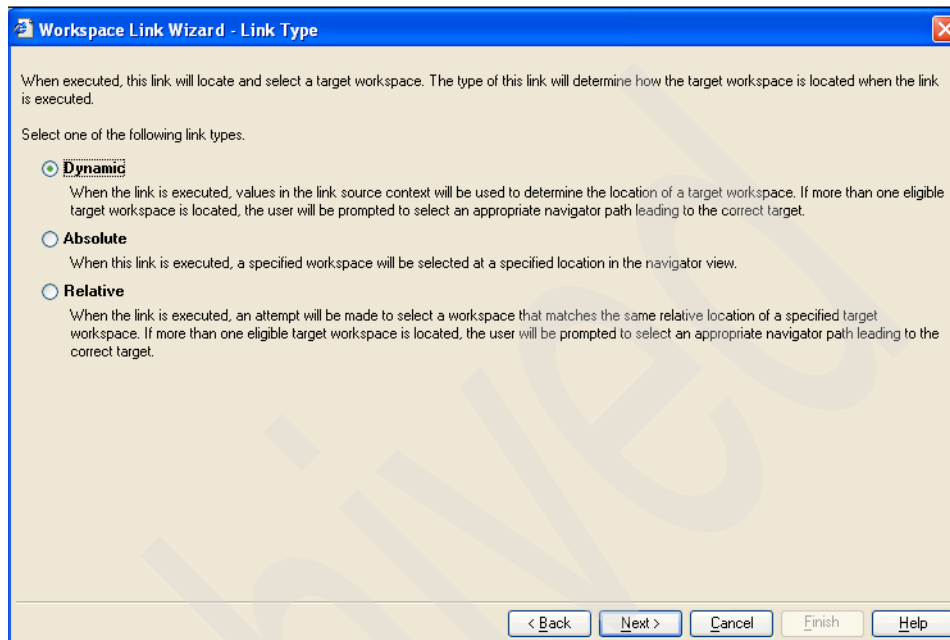


Figure 11-7 Link type selection

3. In the next panel (Figure 11-8):
 - a. Ensure that the *Navigator View* is set to **Physical**.
 - b. In the *Navigator* pane, then locate a CICS region and expand the available tree items. Select the **Web Services Analysis**. This will bring into the Workspaces view the available workspaces from this tree item.
 - c. Select the **Web Services: Detail** report.

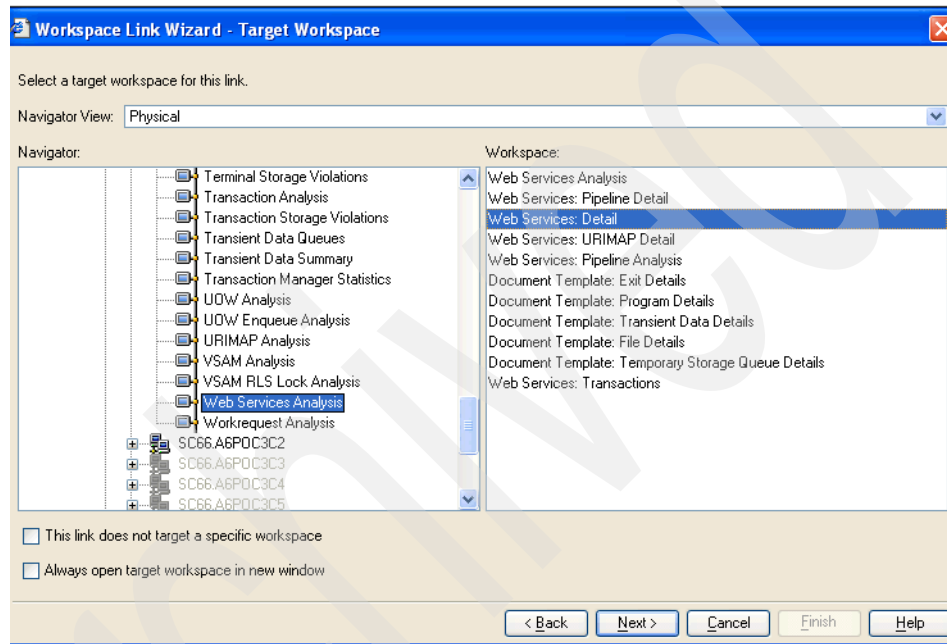


Figure 11-8 Workspace Selection panel

4. In the next sequence of panels (Figure 11-9):
 - a. In the Workspace Link Wizard - Target filters pop-up, highlight the *Managed System name* Filter and click the **Modify Expression** button.
 - b. In the Expression® Editor pop-up, type “*.’+ then click the **Symbol** button.
 - c. In the symbols pop-up, select the Application Server value and click **OK**.
 - d. Click **OK** in the Expression Editor.

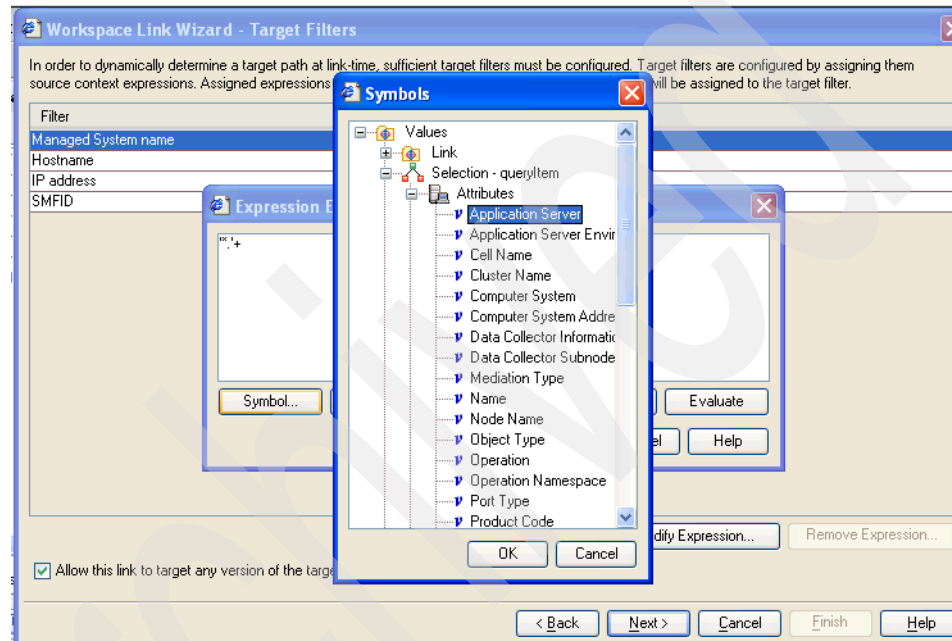


Figure 11-9 Managed system expression editor

5. The expression should look as follows (Figure 11-10).

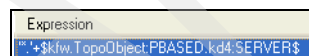


Figure 11-10 Managed System Name expression

6. Click the **Next** button to display the Workspace Link Wizard - Parameters pop-up. Here the LinksEnabled parameter determines if the link appears for an object or not. For this link we only want it to display if the Web Service is being hosted in a CICS region. This is accomplished by checking if the Application Server Environment is equal to ‘5’.

7. To set the LinkIsEnabled item, highlight the parameter and click the **Modify Expression** button. Click the **Symbol** button and select the **Application Server Environment** item. After being returned to the expression editor, append == '5' to the expression.
8. The expression can be checked by clicking the **Evaluate** button. This should provide the results shown in Figure 11-11.

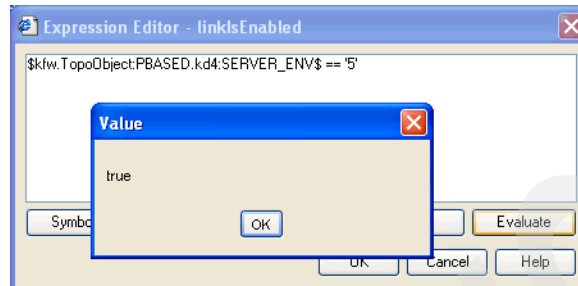


Figure 11-11 Evaluate expression results

9. Next, modify the expressions for CICSNAME and NAME with the symbols for Application Server and Port Name. The finished view should look as follows (Figure 11-12).

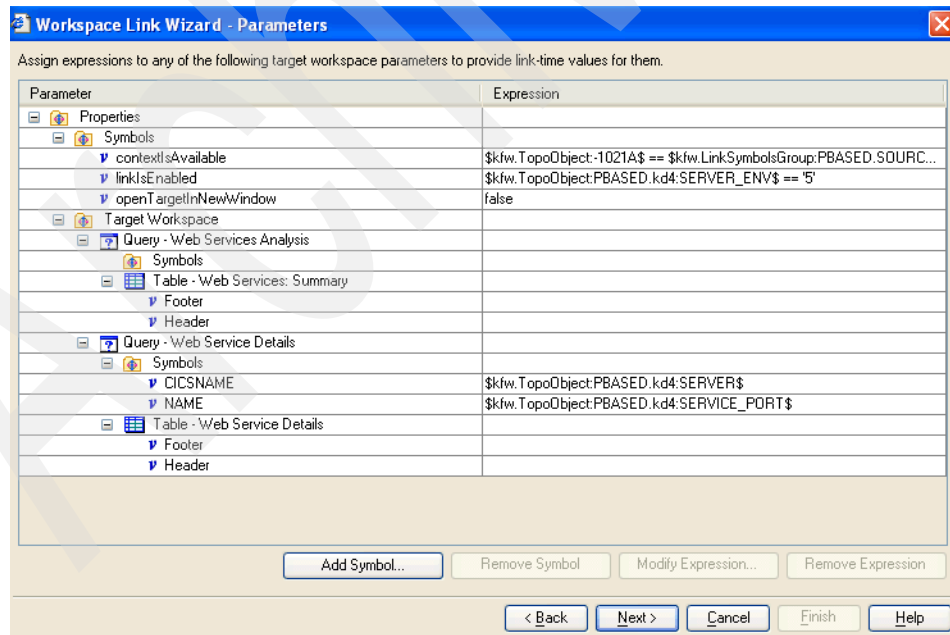


Figure 11-12 Workspace Link Wizard - Parameters pop-up

10. Click the **Next** button and then the **Finish** button.

The system is now configured with a Dynamic link between an Web Service in ITCAM for SOA that is hosted in CICS and the CICS display of the Web Service.

11. To test the Link, right-click a Web Service that is hosted in CICS. This does not need to be the same one that was used to create the link. Select the **Link to** → **CICS Web Services Details** link as shown in Figure 11-13.

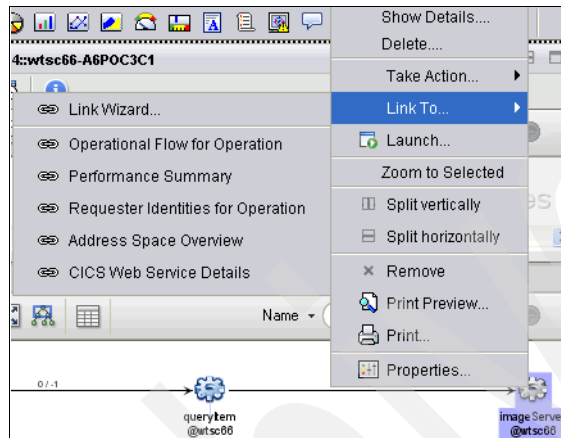


Figure 11-13 New CICS Web Services Details Link

Selecting this link then takes you to OMEGAMON XE for CICS view of the CICS Web Service (Figure 11-14).

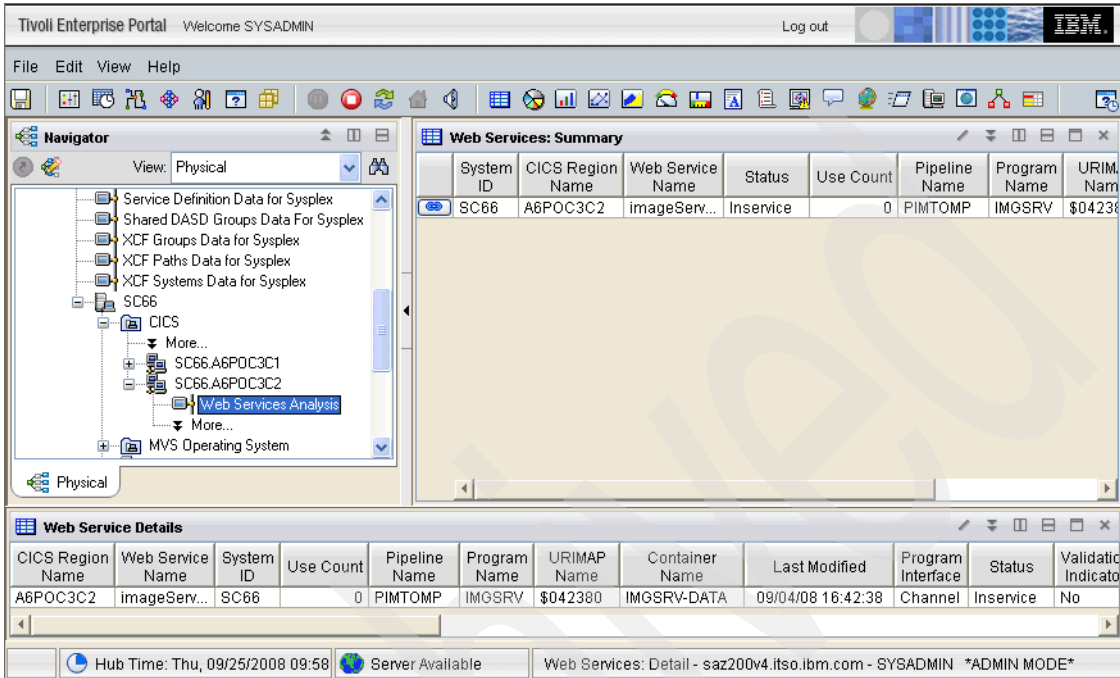


Figure 11-14 CICS Web Services details

11.2 CPU constraint scenario

In this example we look at a situation where the CICS region processing Web Service requests is simply using too much CPU. While many CICS applications can run a significant amount of work on open TCBs, there is still a portion of the workload that must run on the QR TCB. Many LPARs have many CPUs available to process the workload but the QR TCB is only able to go as fast as one CPU will allow regardless of the capacity of the LPAR.

In this scenario, the first indication that there is a problem would typically be an alert generated at the TEP (Figure 11-15).

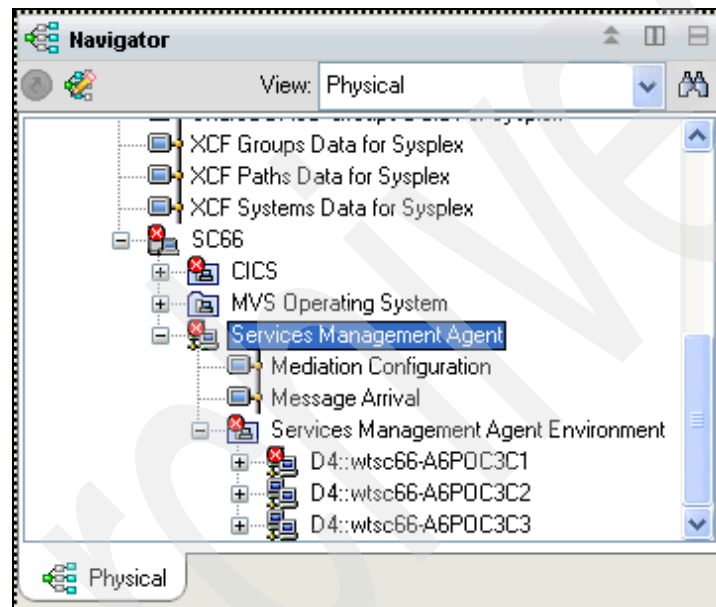


Figure 11-15 TEP Situation Alert

We follow these steps to investigate the problem:

1. Right-click the alert icon to determine the situation (Figure 11-16).

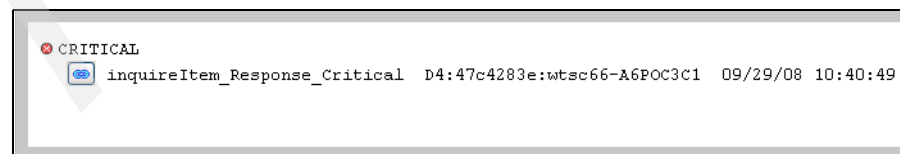


Figure 11-16 TEP situation message

The situation name is the one we created earlier. This tells us that the inquireItem Web Service is not meeting its expected performance criteria.

2. If we did not know about the inquireItem Web service, the next step would be to look at the operation flow for this request. The operation flow is selected via a Link to item on the Performance Summary workspace (Figure 11-17).

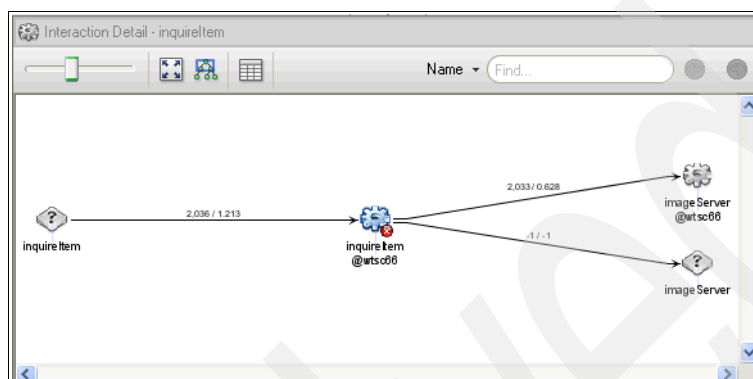


Figure 11-17 Operation flow for inquireItem

3. When we look at the operation flow, it becomes clear that about fifty percent of the response time is occurring in the invoked imageServer Web service rather than it being an issue directly with the CICS region running inquireItem (Figure 11-18).

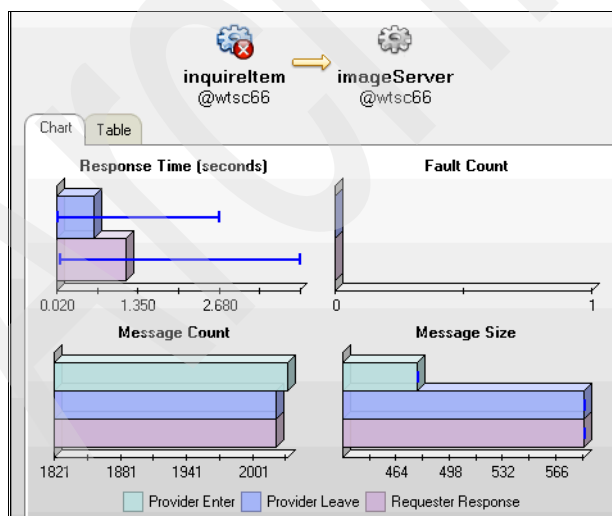


Figure 11-18 Metrics for interaction

4. Clicking the line between the inquireItem and the imageServer Web Services shows the metrics for the interactions. Looking at this interaction, we see that the response time from the requestor perspective is over a second:
 - The response time values show the times from both the provider and the requestor. It would be normal for there to be some difference between the two. This would be for network delays and also for the time taken for CICS to process the Web Service request and running the pipeline code to process the request.
 - For this interaction, we see that the response time from the requestor perspective is over a second. It is clear that this is the real cause of the delay. Just over half the time is spent in the Web Service provider and the rest lost in communication delay. Given that the two CICS regions are located on the same LPAR it would be unusual for a large network component. However, this would easily be explained in a situation where CICS was having difficulty dispatching work.
5. By right-clicking the imageServer icon and selecting **Link to → CICS Web Services Details**, we are taken to the Web Service information from OMEGAMON XE for CICS. From here, we can select **Web Service Transactions** to see details of recent transactions that ran for this Web Service (Figure 11-19).

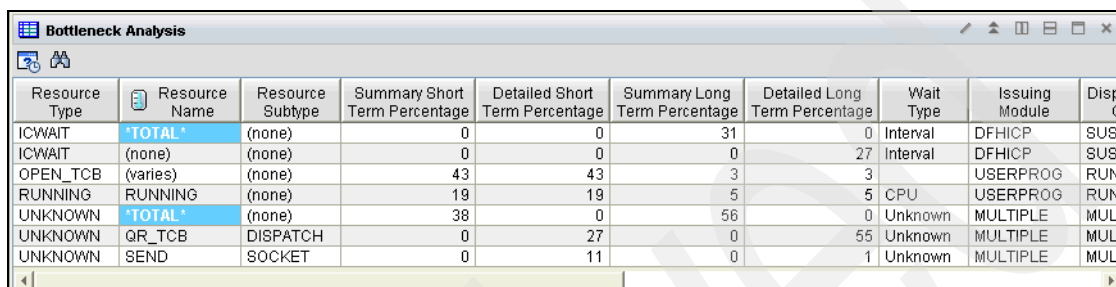
Web Service Name	Web Service Operation	Response Time	CPU Time	End Time	Transacti ID
imageServ...	imageServ...	00:00:01.129	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.177	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.125	00:00:00.05	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:01.046	00:00:00.06	09/29/08 10:51:06	CPIH
imageServ...	imageServ...	00:00:00.998	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.072	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.011	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:01.011	00:00:00.05	09/29/08 10:51:05	CPIH
imageServ...	imageServ...	00:00:02.039	00:00:00.05	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.03	00:00:00.07	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.952	00:00:00.06	09/29/08 10:51:04	CPIH
imageServ...	imageServ...	00:00:02.951	00:00:00.05	09/29/08 10:51:04	CPIH

CICS Region Name	Web Service Name	System ID	Use Count	Pipeline Name	Program Name	URIMAP Name	Container Name	Last Modified	Program Interface	Status	Validatio Indicator
A6POC3C2	imageServ...	SC66	19295	PIMTOMP	IMGSRV	\$042380	IMGSRV-DATA	09/04/08 16:42:38	Channel	Inservice	No

Figure 11-19 CICS Web Services Transactions Detail

The Web Service transactions display shows information about recent transactions for a service. In this case we do not learn much. We see that recent transactions are still taking a long time.

6. As there appears to be no obvious cause of the delay, we need to find some other way to determine that. OMEGAMON provides Bottleneck Analysis, which is an analysis of the causes of delays for the active tasks (Figure 11-20).



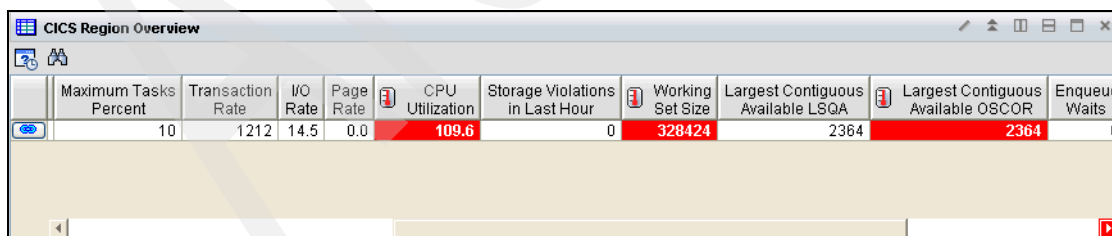
Resource Type	Resource Name	Resource Subtype	Summary Short Term Percentage	Detailed Short Term Percentage	Summary Long Term Percentage	Detailed Long Term Percentage	Wait Type	Issuing Module	Disp C
ICWAIT	*TOTAL*	(none)	0	0	31	0	Interval	DFHICP	SUSP
ICWAIT	(none)	(none)	0	0	0	27	Interval	DFHICP	SUSP
OPEN_TCB	(varies)	(none)	43	43	3	3		USERPROG	RUN
RUNNING	RUNNING	(none)	19	19	5	5	CPU	USERPROG	RUN
UNKNOWN	*TOTAL*	(none)	38	0	56	0	Unknown	MULTIPLE	MULT
UNKNOWN	QR_TCB	DISPATCH	0	27	0	55	Unknown	MULTIPLE	MULT
UNKNOWN	SEND	SOCKET	0	11	0	1	Unknown	MULTIPLE	MULT

Figure 11-20 Bottleneck Analysis shows QR_TCB as issue

The display shows a number of significant wait reasons, the highest being OPEN_TCB. However, this reason means the tasks are actively using the CPU on an open TCB. This would not typically be considered a cause of delay. The same is true for ICWAIT and this is where the application has requested a wait.

The next big item is QR_TCB with a resource type of DISPATCH. This is a problem in that it indicates that tasks are ready to be dispatched on the QR TCB but are unable to, either because other tasks are running, or the CICS region is not given enough cycles.

7. The Region Overview workspace can give us an indication of which it is (Figure 11-21).



	Maximum Tasks Percent	Transaction Rate	I/O Rate	Page Rate	CPU Utilization	Storage Violations in Last Hour	Working Set Size	Largest Contiguous Available LSQA	Largest Contiguous Available OSCOR	Enqueue Waits
	10	1212	14.5	0.0	109.6	0	328424	2364	2364	0

Figure 11-21 Region Overview shows CPU very high

From the Region overview report, it is pretty clear that the CICS region is getting plenty of CPU. The value represents the region CPU usage as a function of time. Because there are several TCBs and several processors, it is possible for this to exceed 100 percent. However, when it is this high, it would be extraordinary for this to be an issue with the CICS region not being given enough cycles. It is much more likely that the region is just trying to use more CPU than available.

8. We then look at the active transactions via the Transactions Analysis display (Figure 11-22).

S NT	Transaction ID	User ID	Terminal ID	Task Number	Resource Type	Resource Name	Task State	Elapsed Time	CPU Time	Program ID	Exce Th
	OSEC	CICSUSER	n/a	17023	USERWAIT	SR2WORK	Suspend	01:36:33.1	00:00:00	KOCSR2ZZ	No
	STRS	CICSUSER	E014	80300	ICWAIT	E014	Suspend	00:03:52.75	00:00:00.16	STRESSPG	No
	STRS	CICSUSER	n/a	85604	ICWAIT		Suspend	00:00:03.27	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85609	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85610	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85611	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85612	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85613	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85614	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85615	ICWAIT		Suspend	00:00:02.46	00:00:00	STRESSPG	No
	STRS	CICSUSER	n/a	85616	RUNNING	QR	Running	00:00:02.46	00:00:00.19	STRESSPG	No
	CPIH	CICSUSER	n/a	85663	DISPATCH	QR_TCB	Dispatbl	00:00:00.63	00:00:00.05	DFHPIDSH	No

Figure 11-22 Active transactions shows running tasks

In this display we see we currently have a task (STRS) using the CPU. There is also a Web Services transaction (CPIH) delayed because the QR TCB is not available.

Going to Task History or Online Data Viewing, we see more transactions (Figure 11-23).

End Time	Transaction ID	Task Number	Terminal ID	Transaction Type	User ID	Program ID	CPU Time	Response Time	Storage HWM	Re
19/29/08 10:59:26	STRS	87502	n/a	SYS	CICSUSER	STRESSPG	00:00:00.68	00:00:08.965	2176	
19/29/08 10:59:26	STRS	87501	n/a	SYS	CICSUSER	STRESSPG	00:00:00.68	00:00:09.659	2176	
19/29/08 10:59:30	STRS	87768	n/a	SYS	CICSUSER	STRESSPG	00:00:00.68	00:00:01.974	2176	
19/29/08 10:59:32	STRS	87766	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:04.056	2176	
19/29/08 10:59:31	STRS	87767	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:03.373	2176	
19/29/08 10:59:32	STRS	87765	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:04.773	2176	
19/29/08 10:59:33	STRS	87764	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:05.751	2176	
19/29/08 10:59:27	STRS	87500	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:10.342	2176	
19/29/08 10:59:28	STRS	87499	n/a	SYS	CICSUSER	STRESSPG	00:00:00.67	00:00:11.059	2176	
19/29/08 10:59:30	CPIH	87738	n/a	TRM	CICSUSER	DFHPIDSH	00:00:00.07	00:00:02.781	15763984	
19/29/08 10:59:30	CPIH	87736	n/a	TRM	CICSUSER	DFHPIDSH	00:00:00.06	00:00:02.764	15763984	

Figure 11-23 Transaction History shows High CPU tasks

Here we can see a lot of STRS transactions and they all are approximately 0.6 CPU seconds. It appears that they are running about one a second. This would therefore equate to around 60 percent of the CPU. This extra workload appears to be seriously affecting the capability of providing the Web Services throughput that we were aiming for. This CICS region is trying to use more CPU with the QR TCB than the machine can offer. The simplest solution would be to ensure that some of the workload runs in another CICS region. This way the work that must run on the QR can be run in parallel on another CPU in the LPAR.

What we have seen in this scenario is how using ITCAM for SOA we were able to be alerted of a failure to meet performance objectives for the Web Service we make available to our customers. We were also able to see that the Web Service also invokes an internal Web Service and that was the real cause of the delay. With that information we could look at the CICS region in OMEGAMON and see where the bottleneck was and the most likely cause of the delay.

11.3 File constraint scenario

In this scenario we look at another type of delay in CICS and how the features provided by OMEGAMON and ITCAM can aid problem resolution:

1. Again we start with an alert from the ITCAM for SOA situation that we created earlier. Navigating to the Operation Flow we can see that the delay this time is in the inquireItem Web Service and not the invoked imageServer (Figure 11-24).

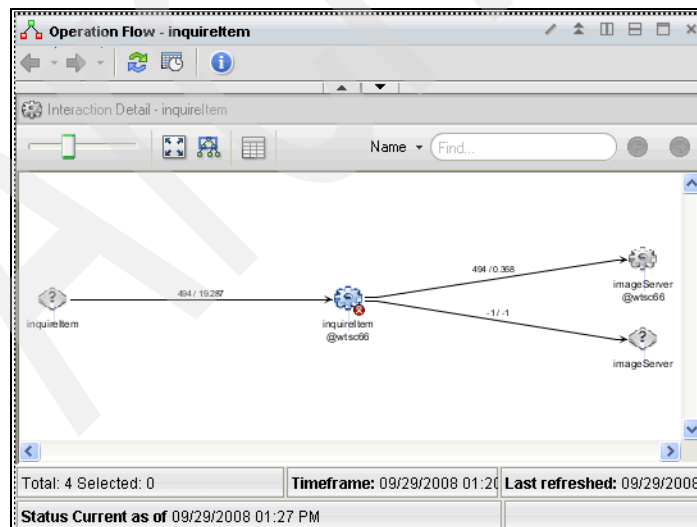


Figure 11-24 Delay in inquireItem not invoked service

The interaction detail clearly shows we have a very long response time to inquireItem but that imageServer is performing as expected.

The metrics pop-up for inquireItem confirms the long response time. It does not show a requestor response time because this is a Web Service that is being provided to the customers. The requestor in this case is outside the scope of the monitoring. While such a set up means we are unable to provide a complete picture of how this is being used, it does show that ITCAM for SOA can still be useful in such situations (Figure 11-25).

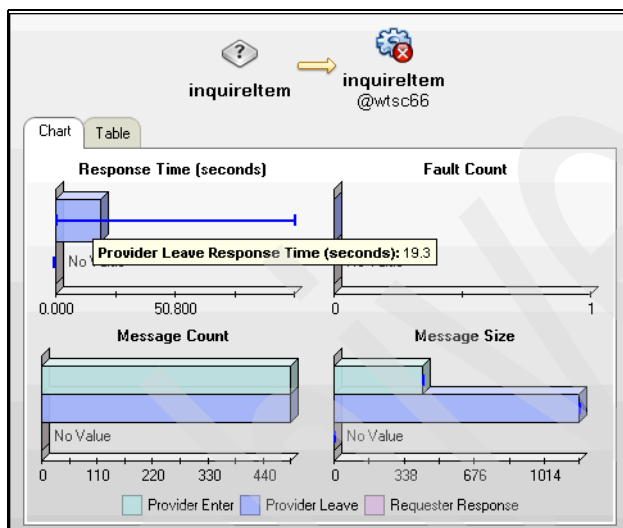


Figure 11-25 Metrics for inquireItem Web Service

- Knowing we have very long response times in this Web Service, we can now look at the transactions details to dig a little deeper. Using the link created earlier, we can link to OMEGAMON CICS view of the Web Service and from there look at transactions for this particular Web Service (Figure 11-26).

In this case we see quite a varied response time for the transactions. Some are taking forty seconds or more while others are much quicker.

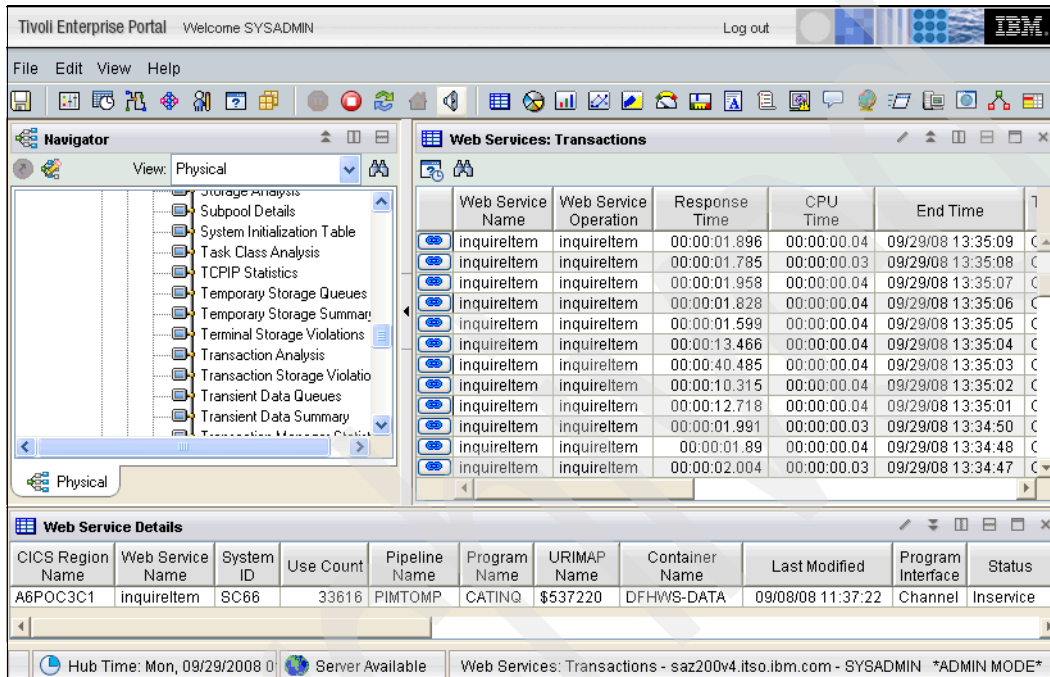
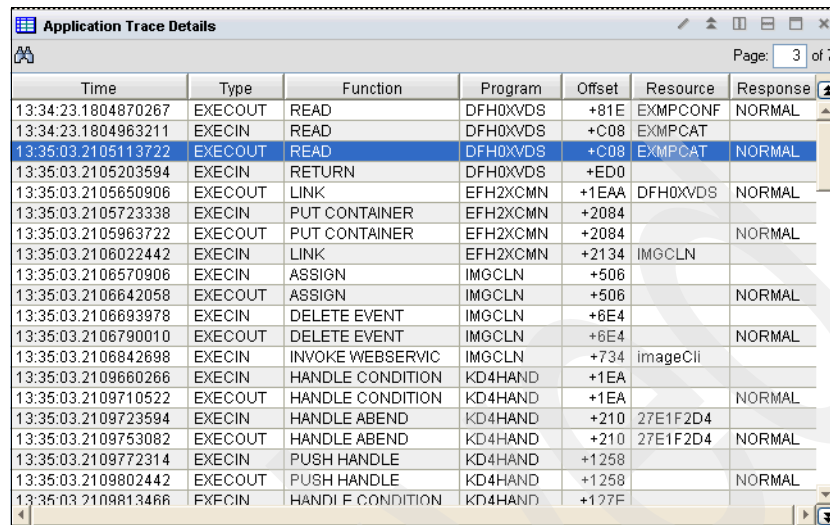


Figure 11-26 Web Service transactions

- Clicking the link icon takes us to the Application Trace Details workspace (Figure 11-27).



The screenshot shows the 'Application Trace Details' workspace. It features a table with 7 columns: Time, Type, Function, Program, Offset, Resource, and Response. The table contains 20 rows of trace data. The row with Time '13:35:03.2105113722' is highlighted in blue. The table is part of a larger application window with a title bar and standard window controls. A 'Page: 3 of 7' indicator is visible in the top right corner of the table area.

Time	Type	Function	Program	Offset	Resource	Response
13:34:23.1804870267	EXECOUT	READ	DFH0XVDS	+81E	EXMPCONF	NORMAL
13:34:23.1804963211	EXECIN	READ	DFH0XVDS	+C08	EXMPCAT	
13:35:03.2105113722	EXECOUT	READ	DFH0XVDS	+C08	EXMPCAT	NORMAL
13:35:03.2105203594	EXECIN	RETURN	DFH0XVDS	+ED0		
13:35:03.2105650906	EXECOUT	LINK	EFH2XCMN	+1EAA	DFH0XVDS	NORMAL
13:35:03.2105723338	EXECIN	PUT CONTAINER	EFH2XCMN	+2084		
13:35:03.2105963722	EXECOUT	PUT CONTAINER	EFH2XCMN	+2084		NORMAL
13:35:03.2106022442	EXECIN	LINK	EFH2XCMN	+2134	IMGCLN	
13:35:03.2106570906	EXECIN	ASSIGN	IMGCLN	+506		
13:35:03.2106642058	EXECOUT	ASSIGN	IMGCLN	+506		NORMAL
13:35:03.2106693978	EXECIN	DELETE EVENT	IMGCLN	+6E4		
13:35:03.2106790010	EXECOUT	DELETE EVENT	IMGCLN	+6E4		NORMAL
13:35:03.2106842698	EXECIN	INVOKE WEBSERVIC	IMGCLN	+734	imageCli	
13:35:03.2109660266	EXECIN	HANDLE CONDITION	KD4HAND	+1EA		
13:35:03.2109710522	EXECOUT	HANDLE CONDITION	KD4HAND	+1EA		NORMAL
13:35:03.2109723594	EXECIN	HANDLE ABEND	KD4HAND	+210	27E1F2D4	
13:35:03.2109753082	EXECOUT	HANDLE ABEND	KD4HAND	+210	27E1F2D4	NORMAL
13:35:03.2109772314	EXECIN	PUSH HANDLE	KD4HAND	+1258		
13:35:03.2109802442	EXECOUT	PUSH HANDLE	KD4HAND	+1258		NORMAL
13:35:03.2109813466	EXECIN	HANDLE CONDITION	KD4HAND	+127F		

Figure 11-27 Application trace view

- By paging through we are able to see an excessive wait on the read of EXMPCAT file. In this case it amounts to almost the entire transaction response time. This clearly indicates a problem reading the file EXMPCAT (Figure 11-28).

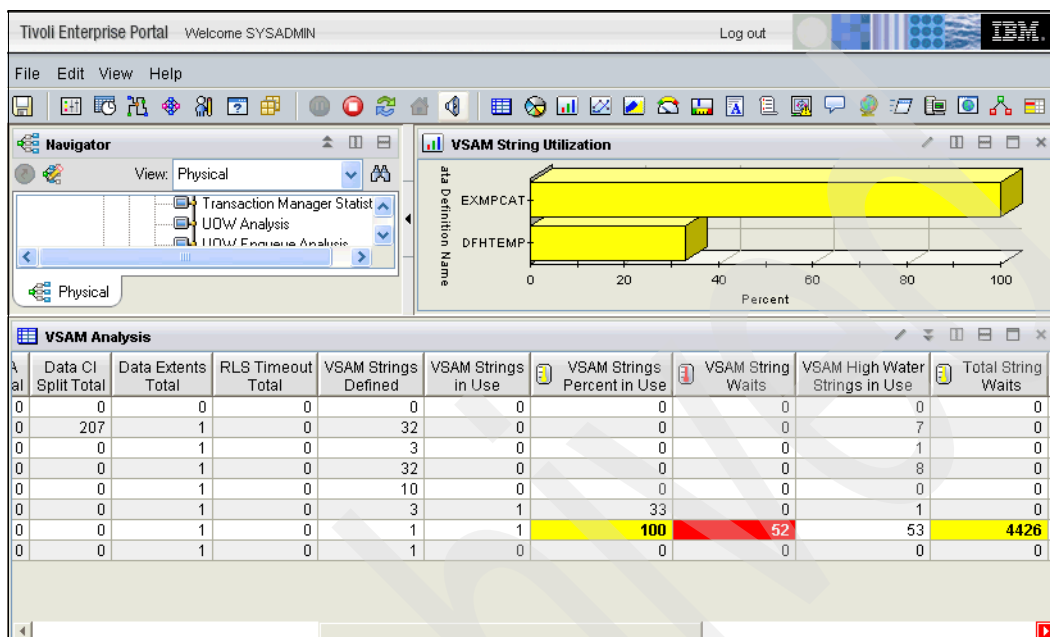


Figure 11-28 OMEGAMON VSAM Analysis

- The next logical step is to look at the VSAM Analysis for this CICS region. By scrolling to the right we can see a large number of VSAM string waits and that the number of strings defined is one (Figure 11-29).

The screenshot shows the 'Historical Transaction Overview' table. The table has columns for Transaction ID, Task Number, Terminal ID, Transaction Type, User ID, Program ID, CPU Time, Response Time, Storage HWM, File Requests, and Te. The table contains 12 rows of transaction data.

Transaction ID	Task Number	Terminal ID	Transaction Type	User ID	Program ID	CPU Time	Response Time	Storage HWM	File Requests	Te
READ	88004	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:05.457	2224	36	
READ	88005	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:04.146	2224	36	
READ	88006	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:03.098	2224	36	
CWXN	88012	n/a	TRM	CICSUSER	DFHWEBXN	00:00:00	00:00:00.263	304	0	
READ	88007	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:02.049	2224	36	
CWXN	88010	n/a	TRM	CICSUSER	DFHWEBXN	00:00:00	00:00:00.265	304	0	
READ	87971	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:11.01	2224	36	
CWXN	88008	n/a	TRM	CICSUSER	DFHWEBXN	00:00:00	00:00:00.298	304	0	
CPIH	87913	n/a	TRM	CICSUSER	DFHPIDSH	00:00:00.04	00:00:32.51	15766576	4	
READ	87972	n/a	SYS	CICSUSER	READPROG	00:00:00	00:00:09.961	2224	36	
CWXN	87996	n/a	TRM	CICSUSER	DFHWEBXN	00:00:00	00:00:00.264	304	0	

Figure 11-29 Historical transactions shows READ transaction with file access

So here we have lots of transactions that are running in the same CICS region as our Web Services. These transactions are processing many browse requests against a file that is also read by our Web Service. The problem is that Browse requests hold a VSAM string for the length of the operation. It is clear that we are seeing large numbers of string waits. It would almost certainly make the situation much better if a significant number of extra strings were defined for the file.

Again, in this scenario, we have seen how ITCAM for SOA was able to show exactly which component of the Web Service architecture was the cause of the delay. OMEGAMON CICS was then able to show us details of which requests in the task were responsible for the bulk of the delay. Knowing that the offending issue was with a VSAM file, the resource monitoring provided by OMEGAMON shows the issue to be with the number of defined VSAM strings. Using the task history details for other transactions, we are able to confirm that there are a number of requests occurring that hold VSAM strings for an extended period of time.

The combination of the products within the same GUI interface make problem resolution a much quicker task than using a combination of offline tools.



Part 4

Appendixes

Archived

The modified catalog manager application

In this appendix, we describe the changes that we applied to the catalog manager application to suit our test scenario for MTOM.

These are the basic steps we performed:

- ▶ Enable the catalog manager for channels and containers
- ▶ Enable the Web service interface for channels and containers
- ▶ Create the image retriever program
- ▶ Create a Web service provider and requester for the image retriever to be called by the catalog manager

Introduction to the catalog manager application

The example application is a catalog management, purchase order style application. The base application, with its 3270 user interface provides functions with which you can:

- ▶ List the contents of a stored catalog: Inquire catalog
- ▶ Select an item from the list: Inquire single
- ▶ Enter a quantity to order: Place order

After an order, the application updates a VSAM catalog to reflect the new stock levels. The application has a modular design, which makes it easier to extend the application to support newer technology, such as Web services.

We do not discuss the base application in more detail in this book. For further details on installation and configuration, refer to the *CICS Transaction Server for z/OS V3.2 Web Services Guide, SC34-6838*.

Figure A-1 shows the application environment.

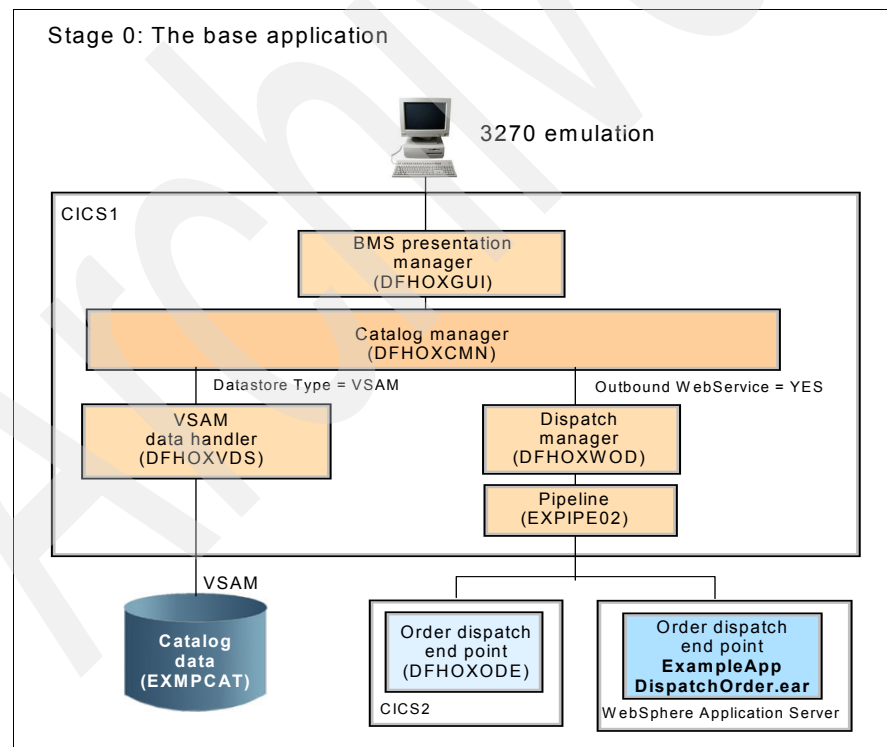


Figure A-1 CICS catalog manager example application

Module DFH0XGUI provides the presentation logic, which exclusively manages the process of sending and receiving the required BMS maps. DFH0XGUI links to the catalog manager program DFH0XCMN in order to perform the supported functions against the catalog. DFH0XGUI uses an EXEC CICS LINK command that passes the COMMAREA structure shown in Example A-1 on page 310.

The catalog manager module DFH0XCMN provides the business logic, which in turn links to the VSAM data handler stub DFH0XVDS. The catalog manager program has other functions that it also implements, such as the outbound Web service function. They do not play a role in the channels and containers migration project, therefore they are not discussed in this section.

As previously mentioned, the 3270 user interface and the provided Web client front end can call the catalog manager example business logic. We provide a wrapper program for Web service requestors that invokes the catalog manager program by passing a channel rather than a COMMAREA.

See Figure A-2.

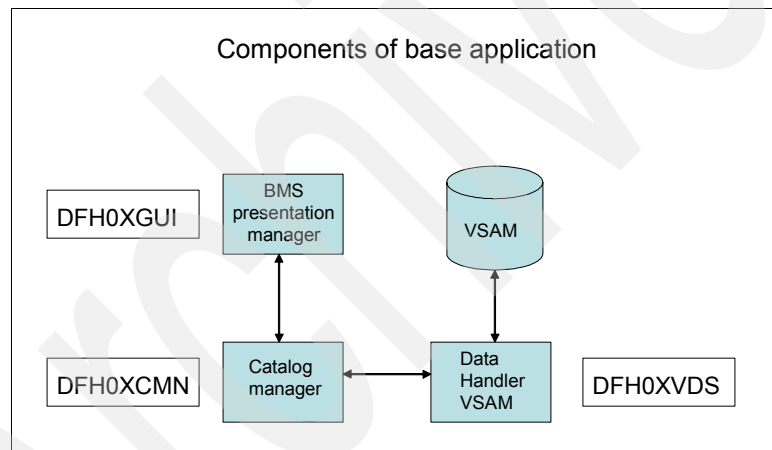


Figure A-2 Components of the base application

Providing channel and container interface to the base application

You do not want to migrate the presentation logic DFH0XGUI to the new function. The presentation logic is using the COMMAREA and it is inconceivable that you would get any constraints with it in the future. Therefore, the presentation logic remains unaffected. At this point, the presentation logic uses an EXEC CICS LINK command that passes a COMMAREA to call the catalog manager module DFH0XCMN.

For the new inquiry, which also returns an image, you use a wrapper program for receiving Web service requests that uses an EXEC CICS LINK command to the catalog manager program. Because of their sizes, the relevant images cannot pass between the wrapper program and the catalog manager using COMMAREAs. Therefore, you need to migrate the catalog manager module DFH0XCMN to EFH0XCMN with the new channels and container functionality.

There is an alternative solution to avoid migrating the presentation logic of the catalog manager example program. You can create an additional routine that the presentation logic DFH0XGUI can call in order to separate the COMMAREA structure to individual containers.

Figure A-3 shows the new structure of the base application. The presentation logic of the application calls the data separation logic EFH0XSEP first. The data separation logic then calls the business logic using an EXEC CICS LINK command that is passing a channel rather than a COMMAREA.

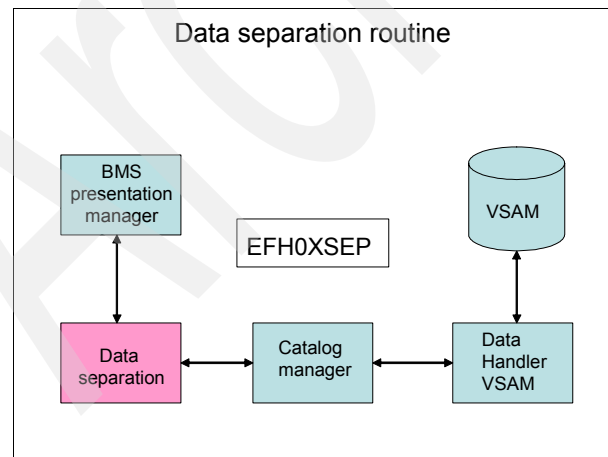


Figure A-3 Data separation module EFH0XSEP

It is possible to use a channel with a single container to replace the COMMAREA that passes between the presentation logic and the catalog manager. This is the simplest and quickest way to migrate the catalog manager application. However, we do not recommend replacing the COMMAREA with just one single container as a best practice. See Figure A-4.

We recommend that the process to replace the existing COMMAREA structure must be done according to the following statements:

- ▶ Use separate containers for input and output.
- ▶ Use a dedicated container for error information.
- ▶ Use separate containers for each structure.
- ▶ Use a copybook that records the name of the channel, the names of the containers used, and defines the data fields that map to the containers. Include the copybook in both the client and the server program.

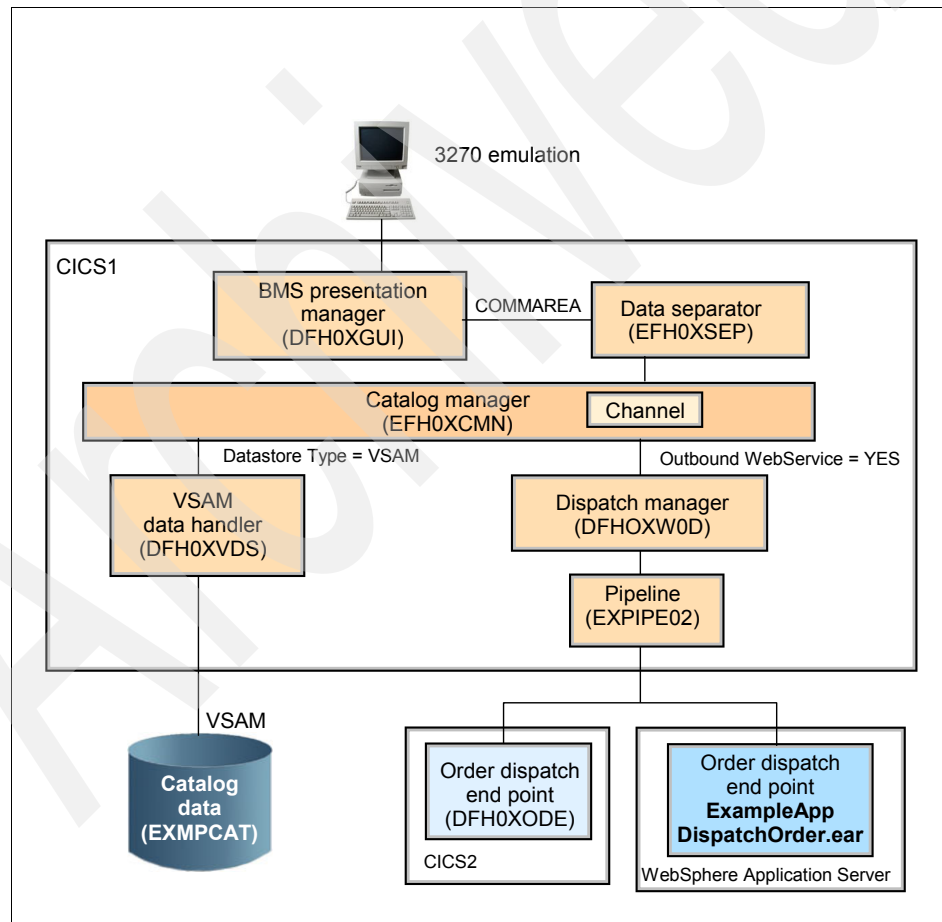


Figure A-4 Catalog manager application with channel and container interface

Create a new module EFH0XSEP that executes the structure illustrated in Figure A-5.

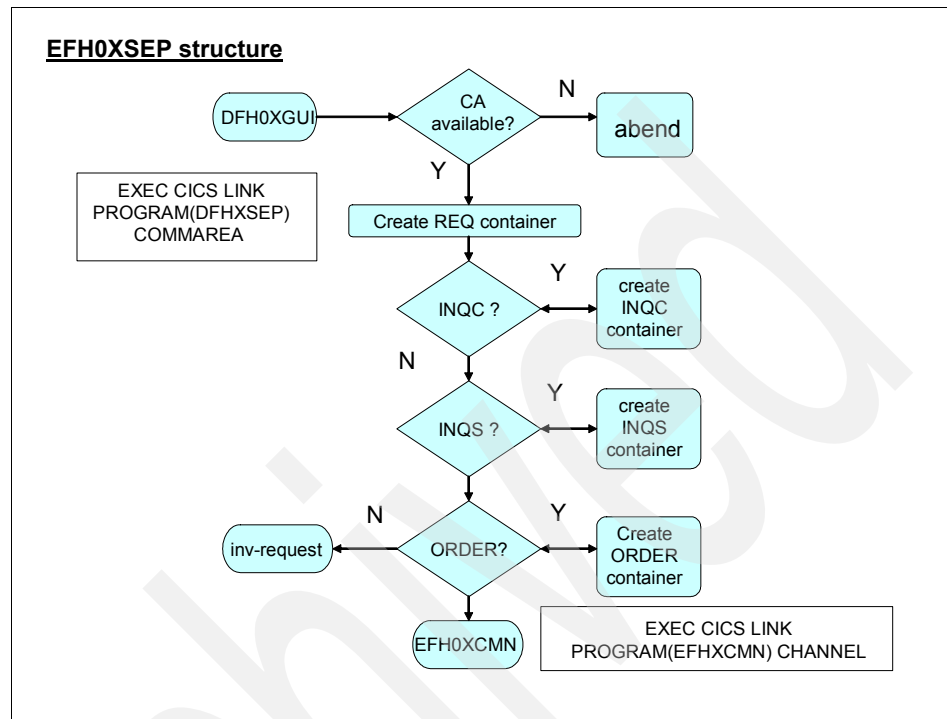


Figure A-5 EFH0XSEP structure

The following steps discuss the logic of the structure:

1. EFH0XSEP gets control from DFH0XGUI through the EXEC CICS LINK. DFH0XGUI passes a COMMAREA.
2. If there is no COMMAREA available, write an error message and issue an abnormal end EXCA through EXEC CICS ABEND.
3. If you have a COMMAREA available, create a separate input container for the required request, which can be inquire catalog, inquire single, or place order.

4. Do not create a separate input container for return codes and messages at present. The catalog manager module creates this later.
5. After that, evaluate the request type and create another input container according to the relevant request.

When the evaluation of the request type is complete, call the catalog manager using an EXEC CICS LINK command passing the channel holding the containers. Do not create any output containers in the data separation logic. We recommend to create the output containers in the server program, which in our case is the modified catalog manager module EFH0XCMN.

Following is a more practical discussion about how you can separate the COMMAREA structure.

COMMAREA structure

Example A-1 on page 310 shows the COMMAREA structure that the CICS catalog manager example program uses. The structure consists of the following parts:

- ▶ A general part:
This part contains the request type and structure items for the return code and the response message. The general part of the structure also contains a field for the result of the different requests that is CA-REQUEST-SPECIFIC.
- ▶ Fields used in inquire catalog function:
This section of the COMMAREA structure is used for the inquire catalog function. It redefines CA-REQUEST-SPECIFIC and uses a table to store 15 catalog items.
- ▶ Fields used in inquire single function:
This section of the COMMAREA structure is used to describe the fields used for the inquire single function. It also redefines CA-REQUEST-SPECIFIC.
- ▶ Fields used in place order function:
This section of the COMMAREA structure is used for the place order function. It also redefines CA-REQUEST-SPECIFIC.

See Example A-1.

Example: A-1 COMMAREA structure of catalog manager example

```
* Catalogue COMMAREA structure
03 CA-REQUEST-ID          PIC X(6).
03 CA-RETURN-CODE         PIC 9(2).
03 CA-RESPONSE-MESSAGE    PIC X(79).
03 CA-REQUEST-SPECIFIC    PIC X(911).

* Fields used in Inquire Catalog
03 CA-INQUIRE-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
   05 CA-LIST-START-REF    PIC 9(4).
   05 CA-LAST-ITEM-REF     PIC 9(4).
   05 CA-ITEM-COUNT        PIC 9(3).
   05 CA-INQUIRY-RESPONSE-DATA PIC X(900).
   05 CA-CAT-ITEM REDEFINES CA-INQUIRY-RESPONSE-DATA
       OCCURS 15 TIMES.
       07 CA-ITEM-REF      PIC 9(4).
       07 CA-DESCRIPTION   PIC X(40).
       07 CA-DEPARTMENT    PIC 9(3).
       07 CA-COST          PIC X(6).
       07 IN-STOCK         PIC 9(4).
       07 ON-ORDER         PIC 9(3).

* Fields used in Inquire Single
03 CA-INQUIRE-SINGLE REDEFINES CA-REQUEST-SPECIFIC.
   05 CA-ITEM-REF-REQ      PIC 9(4).
   05 FILLER               PIC 9(4).
   05 FILLER               PIC 9(3).
   05 CA-SINGLE-ITEM.
       07 CA-SNGL-ITEM-REF  PIC 9(4).
       07 CA-SNGL-DESCRIPTION PIC X(40).
       07 CA-SNGL-DEPARTMENT PIC 9(3).
       07 CA-SNGL-COST      PIC X(6).
       07 IN-SNGL-STOCK     PIC 9(4).
       07 ON-SNGL-ORDER     PIC 9(3).
   05 FILLER               PIC X(840).

* Fields used in Place Order
03 CA-ORDER-REQUEST REDEFINES CA-REQUEST-SPECIFIC.
   05 CA-USERID            PIC X(8).
   05 CA-CHARGE-DEPT       PIC X(8).
   05 CA-ITEM-REF-NUMBER   PIC 9(4).
   05 CA-QUANTITY-REQ      PIC 9(3).
   05 FILLER               PIC X(888).
```

Data separation of the structure

According to the best practice approach, to implement channel and containers, you can separate the COMMAREA structure in to the following functional parts:

- ▶ The request ID:

CA-REQUEST-ID is part of the general section and you must place it in a single input container.

- ▶ Return code and response message:

We recommend that you place return codes and response messages in a separate output-only container.

- ▶ Keep the CA-INQUIRE-REQUEST structure in a separate container:

You can separate the CA-INQUIRE-REQUEST structure further. However, we decided to keep the structures for each function in a single container rather than using further containers.

- ▶ Keep the CA-INQUIRE-SINGLE structure in a single container.

- ▶ The CA-ORDER-REQUEST structure also goes to a single container.

- ▶ The catalog manager module creates the result container.

Copy the result information for each function to a separate container. It is good practice to use different containers for input and output processing.

Figure A-6 illustrates the separation of the COMMAREA structure to different containers.

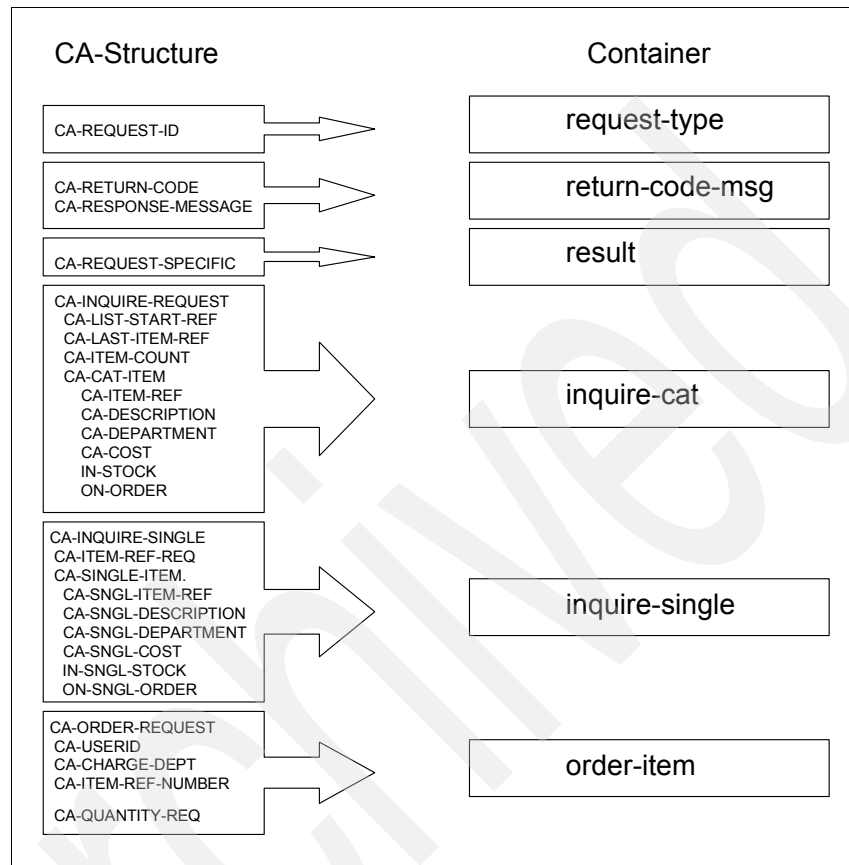


Figure A-6 Data separation of the COMMAREA structure

After separating the COMMAREA structure according to best practices in order to migrate the application to the new function, check if you have completed the following recommendations:

- ▶ Used different containers for input and output operations. This simplifies the copybook structure and makes the program easier to understand.
- ▶ Separated the structures of the individual functions into their own containers. Input containers that the server program has not changed are not returned to the caller.
- ▶ Used a dedicated container for return code and response messages.

EFH0X02 copybook

It is best practice to create a copybook that records the name of the channel and the names of the containers that you use. The copybook must define the data fields that map to the containers.

Example A-2 shows the copybook that our example uses for the data separation routine and for the catalog manager module EFH0XCMN.

The first part of the structure defines the use of the channel name and the container names to migrate the application. The data separation routine EFH0XSEP creates the following containers:

- ▶ request-type
- ▶ return-code-msg
- ▶ inquire-cat
- ▶ inquire single
- ▶ order-item

The catalog manager module creates the result container. Use the other containers in the structure when you add the item image support to the catalog manager which is discussed later.

See Example A-2 that illustrates the creation of copybook EFH0X02.

Example: A-2 Copybook EFH0X02

```
* Channel name
01 CMN-CHANNEL PIC X(16) VALUE 'cmn-channel'.

* Container names
01 REQ          PIC X(16) VALUE 'request-type'.
01 RC-MSG       PIC X(16) VALUE 'return-code-msg'.
01 INQC         PIC X(16) VALUE 'inquire-cat'.
01 INQS         PIC X(16) VALUE 'inquire-single'.
01 ORDR         PIC X(16) VALUE 'order-item'.
01 RESULT       PIC X(16) VALUE 'result'.
01 IMAGE-DATA.
03 IMG-CHANNEL  PIC X(16) VALUE 'IMG-CHANNEL'.
03 IMGSRV-REQUEST PIC X(16) VALUE 'IMGSRV-DATA'.
03 IMGSRV-RESPONSE PIC X(16) VALUE 'IMGSRV-DATA'.
03 IMGPROG      PIC X(16) VALUE 'IMGCLN'.

* name of container containing picture
03 IMGDATA-CONT PIC X(16) VALUE 'IMGDATA-CONT'.
01 SINGLE-CNT   PIC X(16) VALUE 'single'.

* Define the data fields used by the program
01 XCMNPROG     PIC X(8) VALUE 'EFH0XCMN'.
01 CATALOG-SERVER PIC X(8) VALUE 'CTLGSRV'.
```

```

01 CHN-NAME          PIC X(16).
01 IMG-REF-NUMBER    PIC 9(2) VALUE 10.
* Request-ID container structure
01 REQUEST-ID        PIC X(6).

* Return code & MSG container structure
01 RETCODE-MSG.
03 RC                PIC 9(2).
03 RESPONSE-MESSAGE  PIC X(79).

* Inquire single container structure
01 INQUIRE-SINGLE.
03 ITEM-REF-REQ      PIC 9(4).
03 FILLER            PIC 9(4).
03 FILLER            PIC 9(3).

* Inquire catalog container structure
01 INQUIRE-CAT.
03 LIST-START-REF    PIC 9(4).
03 LAST-ITEM-REF     PIC 9(4).
03 ITEM-COUNT        PIC 9(3).

* Order request catalog container structure
01 ORDER-REQUEST.
03 USERID            PIC X(8).
03 CHARGE-DEPT       PIC X(8).
03 ITEM-REF-NUMBER   PIC 9(4).
03 QUANTITY-REQ      PIC 9(3).

```

Creating the separator program EFH0XSEP

This section discusses steps that we used in our example to create the data separation module EFH0XSEP. You can also do the following steps:

1. Separate the existing COMMAREA structure of the catalog manager example program into different containers.
2. Copy the structure of copybook EFH0X02 to module EFH0XSEP.
3. Check in the mainline section of the program to see if you were passed a valid COMMAREA from DFH0XGUI. Prepare a suitable error message and issue an EXEC CICS ABEND if there is no COMMAREA available.
4. The first container contains only the request type. Therefore, move CA-REQUEST to the structure that maps the request ID container and put it into the REQ container.

5. Create the container that takes the return code and response messages named return-code-msg. Move the return code and response message to the structure that maps the container.
6. Evaluate the request type and set up the appropriate container that corresponds to the request.

Example A-3 shows the code snippet our example uses to create the request type and the return code container and how to call the request specific paragraphs.

Example: A-3 Set up request ID and return code container and call request specific paragraphs

```
* Set up request-id container
  MOVE CA-REQUEST-ID TO REQUEST-ID.

* Create request container
  EXEC CICS PUT CONTAINER(REQ) CHANNEL(CMN-CHANNEL)
    FROM(REQUEST-ID)
    END-EXEC.

* Set up return code & Msg container
  MOVE CA-RETURN-CODE TO RC.
  MOVE CA-RESPONSE-MESSAGE TO RESPONSE-MESSAGE.

* Create return code and message container
  EXEC CICS PUT CONTAINER(RC-MSG) CHANNEL(CMN-CHANNEL)
    FROM(RETCODE-MSG)
    END-EXEC.

  EVALUATE CA-REQUEST-ID
    WHEN '01INQC'
      * prepare request container for catalog inquiry
      PERFORM CATALOG-INQUIRE

    WHEN '01INQS'
      * prepare request container for inquire single
      PERFORM INQUIRE-SIN

    WHEN '01ORDR'
      * prepare request container for place order request
      PERFORM PLACE-ORDER

    WHEN OTHER
```

The following three examples show the paragraphs our example uses to set up the structures that map to the data fields of the containers.

Example A-4 shows the creation of the container that takes the structure for the inquire catalog request. Move two parameters from the COMMAREA to the INQUIRE-CAT structure. The parameters are LIST-START-REF and LAST-ITEM-REF. They define the start of the item list and are a reference you need to pass the displayed item.

Example: A-4 Create inquire catalog container

```
CATALOG-INQUIRE.  
* Set up inquire catalog container structure  
    MOVE CA-LIST-START-REF TO LIST-START-REF.  
    MOVE CA-LAST-ITEM-REF TO LAST-ITEM-REF.  
  
* Create inquire catalog container  
    EXEC CICS PUT CONTAINER(INQC) CHANNEL(CMN-CHANNEL)  
        FROM(INQUIRE-CAT) FLENGTH(LENGTH OF INQUIRE-CAT)  
        END-EXEC.  
  
EXIT.
```

Example A-5 shows the paragraph our example uses to create the container for the inquire single request. In order to execute the inquire single request, you have to pass one parameter. Move CA-ITEM-REF-REQ from the COMMAREA to the INQUIRE-SINGLE structure, which maps the data fields of the container.

Example: A-5 Create inquire single container

```
INQUIRE-SIN.  
* Set up inquire single container structure  
  
    MOVE CA-ITEM-REF-REQ TO ITEM-REF-REQ.  
  
* Create container for inquire single  
    EXEC CICS PUT CONTAINER(INQS) CHANNEL(CMN-CHANNEL)  
        FROM(INQUIRE-SINGLE) FLENGTH(LENGTH OF INQUIRE-SINGLE)  
        END-EXEC.  
  
EXIT.
```

Example A-6 shows the PLACE-ORDER paragraph that our example uses to create the container for the order request. Move four fields from the COMMAREA to the ORDER-REQUEST structure that was created in copybook DFH0S02. The four parameters that our example uses for the order request container are:

- ▶ USERID: The name of person that places the order
- ▶ CHARGE -DEPT: Name or name of the department

- ▶ ITEM-REF-NUMBER: The reference number of the catalog item
- ▶ QUANTITY-REQ: The number of items to order

Example: A-6 Create place order container

PLACE-ORDER.

* Set up request and container name

```
MOVE CA-USERID TO USERID.
MOVE CA-CHARGE-DEPT TO CHARGE-DEPT.
MOVE CA-ITEM-REF-NUMBER TO ITEM-REF-NUMBER.
MOVE CA-QUANTITY-REQ TO QUANTITY-REQ.
```

* Create container for place order

```
EXEC CICS PUT CONTAINER(ORDR) CHANNEL(CMN-CHANNEL)
      FROM(ORDER-REQUEST) FLENGTH(LENGTH OF ORDER-REQUEST)
      END-EXEC.
```

EXIT.

Example A-7 shows how the example links to the catalog manager. Issue an EXEC CICS LINK command passing the channel that owns the following containers:

- ▶ request-type
- ▶ return-code-msg
- ▶ inquire-cat
- ▶ inquire-single
- ▶ order-item
- ▶ result (created by the catalog manager EFH0XCMN)

Example: A-7 Link to catalog manager module

```
EXEC CICS LINK PROGRAM(XCMNPROG)
      CHANNEL(CMN-CHANNEL)
      END-EXEC.
EXEC CICS GET CONTAINER(RC-MSG) CHANNEL(CMN-CHANNEL)
      INTO(RETCODE-MSG)
      END-EXEC

MOVE RC TO CA-RETURN-CODE.
MOVE RESPONSE-MESSAGE TO CA-RESPONSE-MESSAGE.
```

Example A-8 on page 318 shows EFH0XSEP when the catalog manager EFH0XCMN returns. It expects the result of the relevant request in the result container. Therefore, evaluate the request ID in order to move the data fields of the result container to the corresponding structure in the COMMAREA. After that EFH0XSEP returns to DFH0XGUI using the updated COMMAREA.

EVALUATE CA-REQUEST-ID

```
        WHEN '01INQC'
EXEC CICS GET CONTAINER(RESULT) CHANNEL(CMN-CHANNEL)
        INTO(CA-INQUIRE-REQUEST)
END-EXEC
```

```
        WHEN '01INQS'
EXEC CICS GET CONTAINER(RESULT) CHANNEL(CMN-CHANNEL)
        INTO(CA-INQUIRE-SINGLE)
END-EXEC
```

```
        WHEN '01ORDR'
EXEC CICS GET CONTAINER(RESULT) CHANNEL(CMN-CHANNEL)
        INTO(CA-ORDER-REQUEST)
END-EXEC
```

The next section describes the process used to migrate the catalog manager DFH0XCMN.

Extending the catalog manager module

This section describes the migration of the catalog manager module to the new function. So far, you have developed the data separation module that you use as a converter from COMMAREA to channels and container. The presentation logic DFH0XGUI uses the data separation module to convert its COMMAREA structure to the channel and container design that the catalog manager module accepts. Copy the existing module DHF0XCMN to EFH0XCMN before applying any changes.

Here are some general considerations before we discuss the migration steps:

- ▶ The original catalog manager module takes the COMMAREA from the presentation logic DFH0XGUI.
- ▶ The catalog manager module links to a VSAM data handler module DFH0XVDS in order to perform one of the three request types, which are inquire catalog, inquire single, or place order.
- ▶ Use an EXEC CICS LINK passing a COMMAREA to call the VSAM data handler. You do not need to migrate the VSAM data handler to the function, because it is not possible to get any COMMAREA constraints with it in the future.

After you have completed the previously mentioned checks, you have to migrate the catalog manager module, so that it is able to perform the following tasks:

- ▶ When the data separation module DFH0XSEP or the Web client front end calls the catalog manager, it expects a channel.
- ▶ The catalog manager module calls the VSAM data handler DFH0XVDS using an EXEC CICS LINK command, using a COMMAREA.

Example A-9 shows the start of the mainline section within the original catalog manager module. The initial action is to perform a check against EIBCALEN. If the length of the COMMAREA is zero, then you can set up a response message and issue an EXEC CICS ABEND.

Example: A-9 DFH0XCMN: check for a valid COMMAREA

```
* If NO COMMAREA received issue an ABEND
  IF EIBCALEN IS EQUAL TO ZERO
    MOVE ' NO COMMAREA RECEIVED' TO EM-DETAIL
    PERFORM WRITE-ERROR-MESSAGE
    EXEC CICS ABEND ABCODE('EXCA') NODUMP END-EXEC
  END-IF

* Initialize COMMAREA return code to zero
  MOVE '00' TO CA-RETURN-CODE.
  MOVE EIBCALEN TO WS-CALEN.
```

As shown in Example A-10, replace the code that examines the COMMAREA by a check whether a current channel exists. Use an EXEC CICS ASSIGN CHANNEL command that returns the 16-character name of the program's current channel, if one exists; otherwise blanks. If there is no channel available, set up an error message and issue an EXEC CICS ABEND.

Example: A-10 Assign channel command

```
EXEC CICS ASSIGN CHANNEL(CHN-NAME)
END-EXEC
IF CHN-NAME = ' '
  MOVE ' NO CHANNEL RECEIVED ' TO EM-DETAIL
  PERFORM WRITE-ERROR-MESSAGE
  EXEC CICS ABEND ABCODE('EXCH') NODUMP END-EXEC
END-IF
```

If you have received a channel, you can issue an EXEC CICS GET CONTAINER(REQ) command to retrieve the required request ID. Our example specifies CA-REQUEST-ID on the INTO parameter, which sets up the COMMAREA structure used to call the VSAM data handler later on.

Depending on this request ID, perform an evaluation to call the corresponding function:

- ▶ The inquire catalog request 01INQC uses paragraph CATALOG-INQUIRE.
- ▶ The inquire single request 01INQS uses paragraph CATALOG-INQUIRE-S.
- ▶ The place order request 01ORDR uses paragraph PLACE-ORDER.

Example A-11 shows how to get and evaluate the request ID in order to call the corresponding paragraph.

Example: A-11 Evaluate request ID

```
EXEC CICS GET CONTAINER(REQ)
      INTO(CA-REQUEST-ID)
END-EXEC

MOVE FUNCTION UPPER-CASE(CA-REQUEST-ID) TO CA-REQUEST-ID
EVALUATE CA-REQUEST-ID
  WHEN '01INQC'
*    Call routine to perform for inquire
    PERFORM CATALOG-INQUIRE
  WHEN '01INQS'
*    Call routine to perform for inquire for single item
    PERFORM CATALOG-INQUIRE-S
  WHEN '01ORDR'
*    Call routine to place order
    PERFORM PLACE-ORDER
  WHEN OTHER
*    Request is not recognised or supported
    PERFORM REQUEST-NOT-RECOGNISED
END-EVALUATE
```

For every paragraph, perform the following steps:

1. Retrieve the request specific data from a container from the channel.
2. Insert data into COMMAREA structure.
3. Call the VSAM data store program with this COMMAREA.
4. Retrieve the result from the COMMAREA and put it into an output container on the channel.

Example A-12 shows the CATALOG-INQUIRE paragraph that our example uses to get the inquire catalog request parameter from the INQC container. Specify INQUIRE-CAT on the into parameter which is the structure defined in the EFH0X02 copybook to map the data fields of the INQC container. To set up the COMMAREA for the VSAM data handler move the three parameters from the

INQC input container structure to the corresponding fields in the COMMAREA. After this, call the VSAM data handler using an EXEC CICS LINK command passing the COMMAREA. The result after linking to the VSAM data handler is in the field CA-INQUIRE-REQUEST that is put into the result container.

Example: A-12 Paragraph catalog-inquire

```
CATALOG-INQUIRE.  
    MOVE 'EXCATMAN: CATALOG-INQUIRE' TO CA-RESPONSE-MESSAGE.  
    EXEC CICS GET CONTAINER(INQC)  
                INTO(INQUIRE-CAT)  
    END-EXEC  
  
    MOVE LIST-START-REF TO CA-LIST-START-REF.  
    MOVE LAST-ITEM-REF  TO CA-LAST-ITEM-REF.  
    MOVE ITEM-COUNT     TO CA-ITEM-COUNT.  
  
    EXEC CICS LINK    PROGRAM(WS-DATASTORE-PROG)  
                    COMMAREA(WS-CHN-DATA)  
    END-EXEC.  
  
    EXEC CICS PUT CONTAINER(RESULT)  
                FROM(CA-INQUIRE-REQUEST)  
    END-EXEC
```

Example A-13 shows the paragraph CATALOG-INQUIRE-S that issues an EXEC CICS GET CONTAINER(INQS) command to store the parameter in the INQUIRE- SINGLE structure. The inquire single request only takes one parameter. Therefore, move ITEM-REF-REQ to relevant COMMAREA field CA-ITEM-REF-REQ. The result after linking to the VSAM data handler is in the field CA-INQUIRE-SINGLE that is put into the result container.

Example: A-13 Paragraph catalog-inquire-s

```
CATALOG-INQUIRE-S.  
    MOVE 'EXCATMAN: CATALOG-INQUIRE' TO CA-RESPONSE-MESSAGE  
    EXEC CICS GET CONTAINER(INQS)  
                INTO(INQUIRE-SINGLE)  
    END-EXEC  
    MOVE ITEM-REF-REQ TO CA-ITEM-REF-REQ.  
  
    EXEC CICS LINK    PROGRAM(WS-DATASTORE-PROG)  
                    COMMAREA(WS-CHN-DATA)  
    END-EXEC  
  
    EXEC CICS PUT CONTAINER(RESULT)
```

```
FROM(CA-INQUIRE-SINGLE)
END-EXEC
```

Example A-14 shows the paragraph place-order that does an EXEC CICS GET CONTAINER(ORDR) command that maps the required input parameters to the ORDER-REQUEST structure, which is specified on the INTO parameter. Our example uses four parameters for the place order request type. You can move them from the ORDER-REQUEST structure to the relevant fields in the COMMAREA.

- ▶ USERID
- ▶ CHARGE-DEPT
- ▶ ITEM-REF
- ▶ QUANTITY-REF

When the parameters are set, issue an EXEC CICS LINK with COMMAREA command to link to the VSAM data handler in order to perform the request. The result after linking to the VSAM data handler is in the field CA-ORDER-REQUEST that is put into the result container.

Example: A-14 Paragraph place-order

```
PLACE-ORDER.
    MOVE 'EXCATMAN: PLACE-ORDER' TO CA-RESPONSE-MESSAGE.
    EXEC CICS GET CONTAINER(ORDR)
        INTO(ORDER-REQUEST)
    END-EXEC

    MOVE USERID TO CA-USERID.
    MOVE CHARGE-DEPT TO CA-CHARGE-DEPT.
    MOVE ITEM-REF-NUMBER TO CA-ITEM-REF-NUMBER.
    MOVE QUANTITY-REQ TO CA-QUANTITY-REQ.

    EXEC CICS LINK PROGRAM(WS-DATASTORE-PROG)
        COMMAREA(WS-CHN-DATA)
    END-EXEC.
    EXEC CICS PUT CONTAINER(RESULT)
        FROM(CA-ORDER-REQUEST)
    END-EXEC
```

When the VSAM data handler returns, move the return code and response message contents to the structure that maps the data fields of the RC-MSG container. After that issue an EXEC CICS PUT CONTAINER(RC-MSG) command in order to provide the return and response messages to the caller of the catalog manager.

Example: A-15 Set up return code container

```
MOVE CA-RETURN-CODE TO RC.  
MOVE CA-RESPONSE-MESSAGE TO RESPONSE-MESSAGE.
```

```
EXEC CICS PUT CONTAINER(RC-MSG)  
FROM(RETCODE-MSG)  
END-EXEC.
```

Figure A-7 shows a diagram of the modified catalog module.

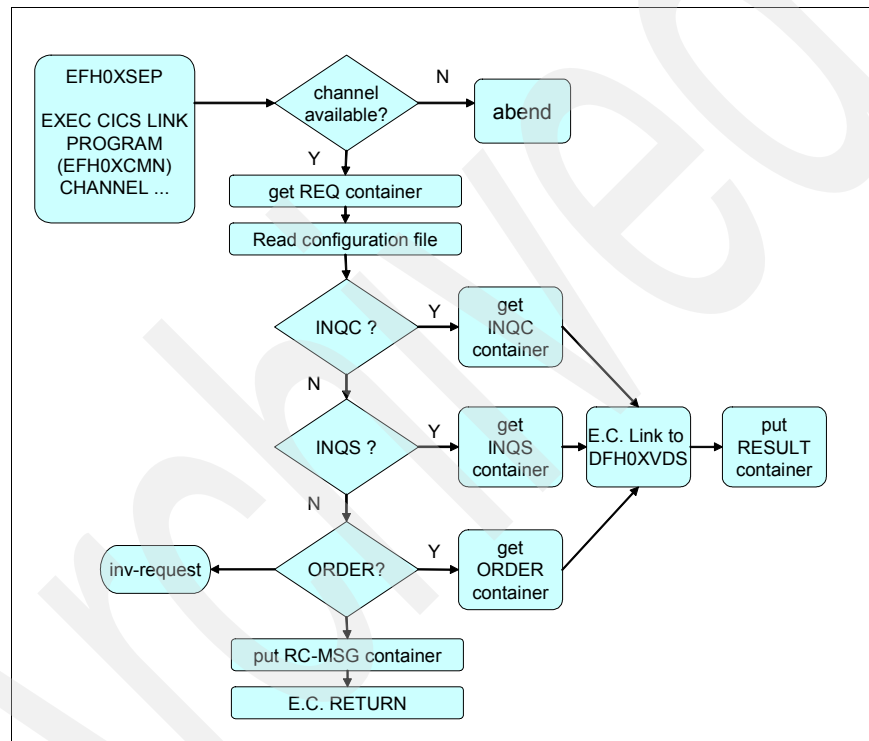


Figure A-7 EFH0XCMN structure

Running the 3270 application with channels and containers

After successfully compiling and linking EFH0XSEP and EFH0XCMN, you can now test if the application is working. To invoke the changed modules, use the CICS catalog manager configuration transaction ECFG and set the program name for “Catalog Manager” to EFH0XSEP as shown in Figure A-8. This will cause the presentation module DFH0XGUI to call the separator instead of the original catalog module.

CONFIGURE CICS EXAMPLE CATALOG APPLICATION

```
Datastore Type ==> VSAM                STUB!VSAM
Outbound WebService? ==> NO             YES!NO
Catalog Manager ==> EFHOXSEP
Data Store Stub ==> DFHOXSDS
Data Store VSAM ==> DFHOXVDS
Order Dispatch Stub ==> DFHOXSOD
Order Dispatch WebService ==> DFHOXWOD
Stock Manager ==> DFHOXSSM
VSAM File Name ==> EXMPCAT
Server Address and Port ==>
Outbound WebService URI ==>
```

Figure A-8 ECFG configuration to call modified catalog module

To make sure that the migrated version of the application works as expected, check the flow of the programs.

The Web service interface to catalog manager

The CICS catalog manager already comes with Web service interfaces for its three functions.

This includes three WSDL files to describe the service (default in /usr/lpp/cicsts/cicsts32/samples/webservices/wSDL) and three corresponding WSBIND files to be deployed to CICS (default in /usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider):

- ▶ inquireSingle.wSDL & inquireSingle.wsbind
- ▶ inquireCatalog.wSDL & inquireCatalog.wsbind
- ▶ placeOrder.wSDL & placeOrder.wsbind

These services have been created using the CICS Web Services Assistant in bottom up mode which means that the existing program structures were used as input to create WSDL and WSBIND for each service. Bottom up also means that everything from the structures is used for the Web service interface. So you might end up sending fields for response message and return code in the request where you do not need them.

Therefore, we recommend a better approach called “meet in the middle” rather than “bottom up.” In this case the Web service description has been defined first with only the fields that are really required and using meaningful names.

These WSDL files have been used as input for the CICS Web Service Assistant to create three corresponding WSBIND files as well as input and output structures. Those structures are used by three wrapper programs that:

1. Receive a Web service request like defined in the WSDL in a container.
2. Use the input to fill a COMMAREA to link to the original catalog manager module (DFH0CXCMN).
3. Transform the received answer conforming to the WSDL and put it back in a container on a channel.

Table A-1 shows the name of the provided artefacts.

Table A-1 Provided artefacts for meet in the middle approach

WSDL (*.wsdl)	WSBIND (*.wsbind)	Structures	Wrapper program
inquireSingleWrapper	inquireSingleWrapper	DFH0XWC3 DFH0XWC4	DFH0XISW
inquireCatalogWrapper	inquireCatalogWrapper	DFH0XWC1 DFH0XWC2	DFH0XICW
placeOrderWrapper	placeOrderWrapper	DFH0XWC5 DFH0XWC6	DFH0XPOW

The locations of those files are as follows:

- ▶ WSDL: Default in /usr/lpp/cicsts/cicsts32/samples/webservices/wsdl
- ▶ WSBIND: Default in /usr/lpp/cicsts/cicsts32/samples/webservices/wsbind/provider
- ▶ Wrapper program and structures: SDFHSAMP of your CICS installation

To avoid overriding of existing programs, we copied and renamed all three WSDL files. For more consistency we globally replaced all names within the WSDL relating to the file name (that is, in inquireSingleWrapper we globally replaced the string inquireSingle by queryItem). Furthermore we reduced the number of bindings to one postfix SoapBinding (that is, queryItemSoapBinding).

We also regenerated the WSBIND and structure files (Table A-2). For more details on using the CICS Web Service Assistant to perform this action, refer to the *CICS Transaction Server for z/OS V3.2 Web Services Guide*, SC34-6838.

Table A-2 WSBIND and structure files

WSDL (*wsdl)	WSBIND (*wsbind)	Structures	Wrapper program	URI
queryItem	queryItem	CATQQ01 CATQP01	CATQUERY	/catalog/queryItem
listCatalog	listCatalog	CATLSQ01 CATLSP01	CATLIST	/catalog/listCatalog
orderItem	orderItem	CATORQ01 CATORP01	CATORDER	catalog/orderItem

To create new wrapper programs, you can either modify the existing ones or take copies of EFH0XSEP which is what we recommend. In the latter case you have to perform the following steps:

1. Change the program to receive a container with the request rather than a COMMAREA.
2. Hard-code the request id depending on the function provided.
3. Only keep the lines required for the function.
4. Call the new catalog manager module EFH0XCMN similar to EFH0XSEP
5. Fit the answer back into the Web service response container.

To enable these programs as a Web service, we use a simple pipeline called PIPEP (which is basically a copy of the provided EXPIPE01 from the provided group DFH\$EXWS) that points to an HFS directory where our WSBIND files have been copied to. For configuration, it uses the basicsoap12provider.xml from the CICS install directory. (See Figure A-9.)


```

CEDA DEFine PIpeline(PIPEP )
PIpeline      : PIPEP
Group         : C3C3PERF
Description   :
Status        : Enabled          Enabled | Disabled
Respwait      : Deft             Default | 0-9999
Configfile    : /CIWS/C3C1/config/basicsoap12provider.xml
(Mixed Case)  :
              :
SHelf         : /CIWS/C3C1/shelf
(Mixed Case)  :
              :
Wsdir         : /CIWS/C3C1/PIPEP/wsbind/
(Mixed Case)  :

```

SYSID=C3C1 APPLID=A6P0C3C1

Figure A-9 Definition of the provider pipeline PIPEP

After the pipeline is installed, the three new Web services exposing the modified wrapper programs should be available. Use CEMT INQUIRE WEBSERVICE (Figure A-10) and CEMT INQUIRE URI (Figure A-11) to check.

```

INQUIRE WEBSERVICE
STATUS: RESULTS - OVERTYPE TO MODIFY
Webs(listCatalog          ) Pip(PIPEP )
   Ins Ccs(00000) Uri($305400 ) Pro(CATLI  ) Cha Xopsup Xopdir
Webs(orderItem            ) Pip(PIPEP )
   Ins Ccs(00000) Uri($301450 ) Pro(CATORDER) Cha Xopsup Xopdir
Webs(queryItem            ) Pip(PIPEP )
   Ins Ccs(00000) Uri($255280 ) Pro(CATQUERY) Cha Xopsup Xopdir

```

Figure A-10 The WEBSERVICEs of the Channel & Container enabled Wrappers

```

INQUIRE URIMAP
STATUS: RESULTS - OVERTYPE TO MODIFY
Uri($255280 ) Pip Ena      Http
      Host(*)              ) Path(/catalog/queryItem )
Uri($301450 ) Pip Ena      Http
      Host(*)              ) Path(/catalog/orderItem )
Uri($305400 ) Pip Ena      Http
      Host(*)              ) Path(/catalog/listCatalog )

```

Figure A-11 The URIMAPs of the channel and container enabled wrappers

Adding the image retriever functionality

This section describes how to extend the catalog manager to allow for catalog item image support.

When a Web service client requests detailed information on a single item the extended catalog manager functionality provides an image in addition to the item details. Our example stores the item image files in an HFS directory.

Creating the image server

Create a CICS program that reads the image file from the HFS directory and puts its contents to an output binary container.

For performance tests we want to use an image retriever program that can be linked using channels and containers or that can be invoked as a Web service. A Web service gives us the possibility to install the image server in another region and to drive performance tests on a requester program later as well.

This gives us two possibilities to create the retriever program:

- Create it directly and expose it as a Web service later
- As this is a new program we can use the Web service top down approach to create a Web service provider directly using the CICS Web Services Assistant

We prefer the second option that needs fewer steps. Therefore we start with a definition of the image server's interfaces using Web Service Description Language (WSDL).

From this we use the CICS Web Services Assistant in top down mode (WS2LS) to generate:

- ▶ The WSBIND file imageServer.wsbind
- ▶ The request and response structures IMGSVQ01 and IMGSVP01

Our image server program is called IMGSRV and uses the CICS DOCTEMPLATE mechanism to retrieve the images from HFS. Therefore we need to install one DOCTEMPLATE for each image we might want to retrieve and associate it with the corresponding HFS file. A sample definition is shown in Figure A-12. DOCTEMPLATES are cached by CICS. The cache can be cleared by issuing a DISCARD, or refreshed by issuing a NEWCOPY.

```
CEDA DEFINE DOctemplate( 0010      )
DOctemplate      : 0010
Group            : C3C2PERF
DEscription      : Image for Ball Pens Black
FULL TEMPLATE NAME
TEmlatename      : 0010.gif
ASSOCIATED CICS RESOURCE
File             :
TSqueue          :
TDqueue          :
Program          :
Exitpgm          :
PARTITIONED DATA SET
DDname           :
Membername       :
HIERARCHICAL FILE SYSTEM
Hfsfile          : /u/cicsrs9/images/0010.gif
(Mixed Case)     :
TEMPLATE PROPERTIES
Appendcrlf       : No                Yes ! No
TType            : Binary            Binary ! Ebcdic
```

Figure A-12 DOCTEMPLATE definition for image number 10

To retrieve data from the DOCTEMPLATE, the following steps have to be performed (for more details, refer to the Appendix):

1. Use the DOCTEMPLATE to create a DOCTOKEN and find its file length
2. Get dynamic working storage for the file and copy it there
3. Put the file to a container
4. Free dynamic working storage and delete the DOCTOKEN

The program uses the following containers:

► Container *IMGSRV-DATA*

This container is used for input into the program to tell the image number and for output of the name of the container (in *imageData-cont*) holding the binary image.

► Container *imageData-cont*

This is the binary output container that contains the image file.

Extensions to the channel container copybook EFH0XS02

The copybook already contains the fields required for image interchange as shown in Example A-16.

Example: A-16 Fields required for image interchange

```
01 IMAGE-DATA.  
  03 IMG-CHANNEL      PIC X(16) VALUE 'IMG-CHANNEL'.  
  03 IMGSRV-REQUEST   PIC X(16) VALUE 'IMGSRV-DATA'.  
  03 IMGSRV-RESPONSE  PIC X(16) VALUE 'IMGSRV-DATA'.  
  03 IMGPROG          PIC X(16) VALUE 'IMGCLN'.  
  03 IMGDATA-CONT     PIC X(16) VALUE 'IMGDATA-CONT'.
```

The last field *IMGDATA-CONT* describes the name of the container in which the catalog manager will return the binary image data.

Extension to the catalog manager

The image is only retrieved for the inquiry of a single item. Therefore we add the code to call the image server program to the corresponding paragraph *CATALOG-INQUIRE-S*. Example A-17 shows the additions made to the program after the result container was put on the channel.

Example: A-17 Extended CATALOG-INQUIRE-S paragraph

```
      MOVE ITEM-REF-REQ OF INQUIRE-SINGLE TO  
          ca-item-ref-req OF ImageServerRequest      (1)  
  
      EXEC CICS PUT CONTAINER( IMGSRV-REQUEST )  
          CHANNEL( IMG-CHANNEL )
```

```

                                FROM( ImageServerRequest )
END-EXEC (2)

```

```

EXEC CICS LINK PROGRAM( IMGPROG )
              CHANNEL( IMG-CHANNEL )
END-EXEC (3)

```

```

EXEC CICS GET CONTAINER( IMGSRV-RESPONSE )
              CHANNEL( IMG-CHANNEL )
              INTO( ImageServerResponse )
END-EXEC (4)

```

```

EXEC CICS MOVE
      CONTAINER( imageData-cont of ImageServerResponse )
      CHANNEL( IMG-CHANNEL )
      AS( IMGDATA-CONT OF IMAGE-DATA )
END-EXEC (5)

```

1. Populate the request structure with the reference number of the required item
2. Put the request into the IMGSRV-REQUEST container on the IMG-CHANNEL
3. Link to the image program using this channel
4. Populate the response structure from the IMGSRV-RESPONSE container, which defines the name of the container holding the image in the imageData-cont field
5. Move the imageData-cont container from the IMG-CHANNEL to the current channel and rename it to IMGDATA-CONT (specified in EFH0X02 copybook)

The image container is now on the current channel in addition to the previous results.

Extensions to the Web service wrapper

The only part affected by the extended functionality is the queryItem Web service with the corresponding wrapper program CATQUERY. We copied and renamed the WSDL file from queryItem.wsdl to inquireItem.wsdl and added a field to hold the binary data in the response as shown in Figure A-13.

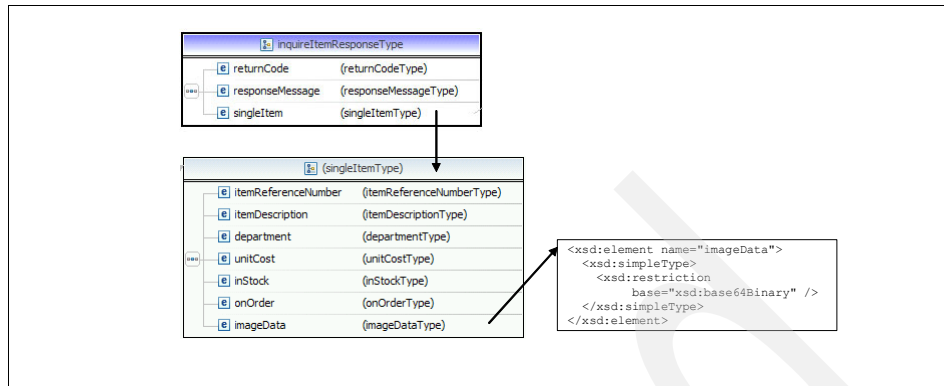


Figure A-13 Extension of inquireItem.wsdl by imageData field

A change in a WSDL file means we have to regenerate the WSBIND file (inquireItem.wsbind) and the input and output structures (CATINQ01, CATINP01). We copied and renamed the previous CATQUERY program to CATINQ and made the following additions to receive the image data (see Figure A-14):

- ▶ After receiving the result container, place the name of the image container into the response structure.
- ▶ Move the image container on the current channel to be available for Web service handling.

```

EXEC CICS GET CONTAINER(RESET) CHANNEL(CMN-CHANNEL)
        INTO( CA-INQUIRE-SINGLE ) END-EXEC
MOVE IMGDATA-CONT OF IMAGE-DATA TO
    imageData-cont of singleItem
* move the image container to the response channel
EXEC CICS MOVE
    CONTAINER( IMGDATA-CONT OF IMAGE-DATA )
    CHANNEL(CMN-CHANNEL)
END-EXEC.

```

Figure A-14 Single Inquiry: Image Data Handling by the Wrapper program

After copying and deploying the WSBIND file to our provider pipeline (PIPEP), we can test the Web service using SOAP generation tools like the Web Services Explorer provided in Rational Developer for System z. The result returned from the Web service is shown in Figure A-15.

```

<SOAP-ENV:Envelope
  xmlns:q0="http://www.exampleApp.inquireItemRequest.com"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <inquireItemResponse
      xmlns="http://www.exampleApp.inquireItemResponse.com">
      <returnCode>0</returnCode>
      <responseMessage>RETURNED ITEM: REF =0010</responseMessage>
      <singleItem>
        <itemReferenceNumber>10</itemReferenceNumber>
        <itemDescription>Ball Pens Black 24pk</itemDescription>
        <department>10</department>
        <unitCost>002.90</unitCost>
        <inStock>5948</inStock>
        <onOrder>2</onOrder>
        <imageData>
          <<<<< large binary data element >>>>>
        </imageData>
      </singleItem>
    </inquireItemResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

Figure A-15 SOAP response from inquireItem service

The CICS catalog example application is now accessible via channels and containers and returns an image during a single inquiry.

Extending the application with a Web service requester

For later tests we want to measure the behavior of CICS acting as a Web service requestor, as well as acting as a provider, so we adjusted the catalog manager application to perform a Web service request during single inquiry. The idea behind is that CICS calls an external image server to request the picture of the inquired item.

Deploying the image server Web service provider

In “Creating the image server” on page 328, we already created all necessary Web services resources of the image server by generating a top down provider from the WSDL file. We now move the program to another CICS region and deploy the corresponding WSBIND file there. The pipeline in this region is a copy of our previous pipeline, just with another pickup directory.

Creating the image client Web service requester

Instead of linking directly to the image server the catalog manager now links to an image client program that performs a Web service request to the image server. We created this client from the same WSDL file as the image server using the CICS Web Services Assistant in top down mode. It basically has to route through the containers and invoke the Web service as shown in excerpts in Example A-18.

Example: A-18 Image client code snippets

```
WORKING-STORAGE SECTION.
01 CONT-REQUEST PIC X(16) VALUE 'IMGSRV-DATA'.
01 CONT-RESPONSE PIC X(16) VALUE 'IMGSRV-DATA'.
01 RESPONSE.
    COPY IMGSRVP01.
    01 WS-WEBSERVICE PIC X(32) VALUE 'imageClient'.
01 WS-OPERATION PIC X(255) VALUE 'imageServer'.
01 WS-URI PIC X(255)
    VALUE 'http://9.12.4.75:14305/cics/services/imageServer'.
01 WS-CONTAINER PIC X(16) VALUE 'DFHWS-DATA'.
01 WS-CHANNEL PIC X(16) VALUE 'SERVICE-CHANNEL'.
01 RESP-CODE PIC S9(8) COMP VALUE 0.
```



```
PROCEDURE DIVISION.  
EXEC CICS MOVE CONTAINER(CONT-REQUEST)  
      AS(WC-CONTAINER) TOCHANNEL(WC-CHANNEL)  
END-EXEC.  
  
EXEC CICS INVOKE WEBSERVICE(WC-WEBSERVICE)  
      CHANNEL(WC-CHANNEL)  
      URI(WC-URI)  
      OPERATION(WC-OPERATION)  
      RESP(RESP-CODE)  
END-EXEC.  
  
EXEC CICS GET CONTAINER(WC-CONTAINER)  
      CHANNEL(WC-CHANNEL) INTO(RESPONSE)  
END-EXEC  
  
EXEC CICS MOVE CONTAINER(imageData-cont)  
      CHANNEL(WC-CHANNEL) AS(imageData-cont)  
END-EXEC
```

Deploy the imageClient.wsbind file to a basic requester pipeline as shown in Figure A-16.

```
CEDA DEFine Pipeline(PIPER )  
PIpeline      : PIPER  
Group         : C3C1PERF  
Description    :  
SStatus       : Enabled           Enabled | Disabled  
Respwait      : Deft             Default | 0-9999  
Configfile     : /CIWS/C3C1/config/basicsoap12requester.xml  
(Mixed Case)  :  
              :  
SHelf         : /CIWS/C3C1/shelf  
(Mixed Case)  :  
              :  
Wsdir         : /CIWS/C3C1/PIPER/wsbind/  
(Mixed Case)  :  
  
SYSID=C3C1 APPLID=A6POC3C1
```

Figure A-16 Definition of the basic requester pipeline PIPER

Changes to the base application to link to requester

The image server program and the image client requester have the same interfaces so simply exchange the name of the program linked by the catalog manager. Figure A-17 depicts the modified structure.

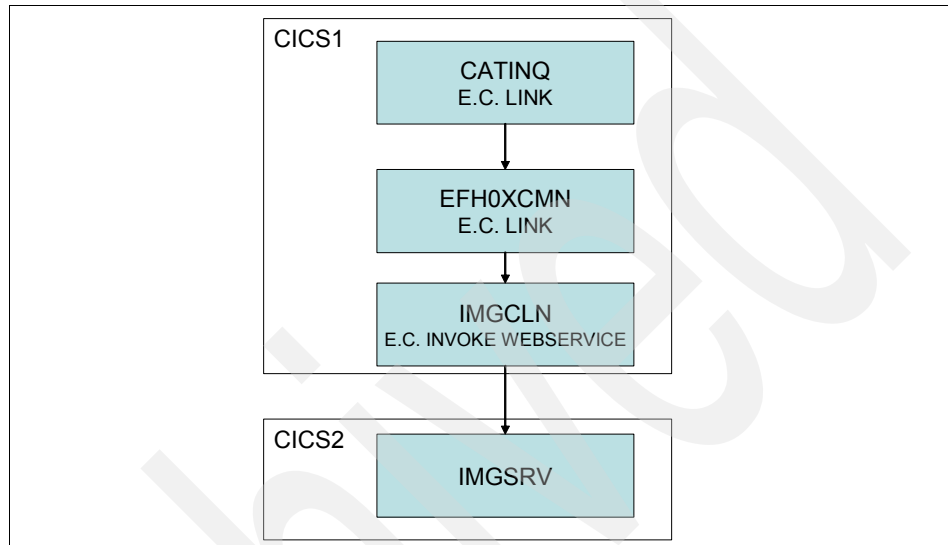


Figure A-17 Catalog manager extension with requester

XSL to add UsernameToken in DataPower

In this appendix, we show the XSL to add a UsernameToken in DataPower, and the XML for user mappings. Refer to Example B-1 on page 338 and Example B-2 on page 340.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
  -->
- <!-- j. rasmussen rasmussj@us.ibm.com
  -->
- <!-- Modified by Arnauld Desprets - arnauld_desprets@fr.ibm.com
  -->
- <!--
  -->
- <xsl:stylesheet version="1.1" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility
-1.0.xsd"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext
-1.0.xsd" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:dp="http://www.datapower.com/extensions"
xmlns:dpconfig="http://www.datapower.com/param/config"
extension-element-prefixes="dp" exclude-result-prefixes="dp dpconfig">
- <!--
  -->
  <xsl:output method="xml" />
- <!--
  -->
  <xsl:variable name="cert" select="concat('cert:',
/soapenv:Envelope/soapenv:Header/wsse:Security/wsse:BinarySecurityToken/text())" />
  <xsl:variable name="subject"
select="dp:get-cert-details($cert)/CertificateDetails/Subject/text()" />
- <!--
  -->
- <!-- New Arno
  -->
- <xsl:message dp:priority="debug">
  User DN:
  <xsl:value-of select="$subject" />
</xsl:message>
- <!-- Load the mappings file
  -->
- <xsl:variable name="UserMappings">
  <dp:url-open target="local:///user_mappings.xml" />
</xsl:variable>
- <!-- Extract username based on userDN
  -->
  <xsl:variable name="uid"
select="$UserMappings/mappings/mapping[user_dn=$subject]/uid" />
```

```

- <xsl:message>
  UID:
  <xsl:value-of select="$uid" />
</xsl:message>
- <!-- New Arno
  -->
- <xsl:template match="/">
  <xsl:apply-templates />
</xsl:template>
- <!--
  -->
- <xsl:template match="soapenv:Header/wsse:Security">
- <xsl:copy>
  <xsl:copy-of select="@*" />
- <wsse:UsernameToken>
- <wsse:Username>
  <xsl:value-of select="$uid" />
</wsse:Username>
</wsse:UsernameToken>
  <xsl:apply-templates />
</xsl:copy>
</xsl:template>
- <!--
  -->
- <!-- Changed Arno
  -->
- <xsl:template match="@*">
  <xsl:copy />
</xsl:template>
- <xsl:template match="*">
- <xsl:copy>
  <xsl:copy-of select="@*" />
  <xsl:apply-templates select="node()" />
</xsl:copy>
</xsl:template>
- <!-- Changed Arno
  -->
</xsl:stylesheet>

```

Example: B-2 User mapping xml

```
<?xml version="1.0" encoding="UTF-8" ?>
- <mappings xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <mapping>
  <user_dn>C=France, ST=Herault, L=Montpellier, O=IBM, OU=ITSO,
title=CWASWINCert01-certificate, CN=WASWINCert01</user_dn>
  <uid>USERWS01</uid>
</mapping>
</mappings>
```

Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the Web server for IBM Redbooks publications. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/MTOM Code.zip>

Alternatively, you can go to the IBM Redbooks publications Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks publications form number, SG247687.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
MTOM Code .zip	Zipped Code Samples

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 47kb

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

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 publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 344. Note that some of the documents referenced here might be available in softcopy only:

- ▶ *Effective zSeries Performance Monitoring using Resource Management Facility*, SG24-6645
- ▶ *Getting Started with IBM Tivoli Monitoring 6.1 on Distributed Environments*, SG24-7143
- ▶ *IBM Tivoli Composite Application Manager Family Installation, Configuration, and Basic Usage*, SG24-7151
- ▶ *Securing CICS Web Services*, SG24-7658
- ▶ *Threadsafe Considerations for CICS*, SG24-6351
- ▶ *Architecting Access to CICS within an SOA*, SG24-5466

Other publications

These publications are also relevant as further information sources:

- ▶ *CICS Web Services Guide*, SC34-6838
- ▶ *IBM Tivoli Monitoring: Installation and Setup Guide*, GC32-9407
- ▶ *IBM Tivoli Monitoring: Administrators Guide*, GC32-9408
- ▶ *IBM Tivoli Monitoring: User's Guide*, GC32-9409
- ▶ *Composite Application Manager for SOA V7.1.0 Release Notes*, GI11-4096
- ▶ *CICS Transaction Server for z/OS RACF Security Guide*, SC34-6454

Online resources

These Web sites are also relevant as further information sources:

- ▶ CICS Transaction Server for z/OS, Version 3 Release 2 Information Center, located at
<http://publib.boulder.ibm.com/infocenter/cicsts/v3r2/index.jsp>
- ▶ Information about XML:
<http://www.w3.org/XML/>

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, IBM Redpaper publications, Technotes, draft publications, and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services

Index

Numerics

3270 interface 178

32K limit 180

A

action method 121

AddFltrCntrl 121

AddMntrCntrl 121

advanced program-to-program communication (APPC) 73

APF authorized 142

ASCII 113

ASCII characters 49

Automatic Initiate Descriptors (AIDs) 150

B

base application 305

 new structure 306

base RTE 126

base64 11

base64Binary 179

BinarySecurityToken 238

Bottleneck Analysis 158

Business Transaction Services 150

C

C 32, 34

C++ 32, 34

CANSD4 133

catalog manager 305

 application 307

 DFH0XCMN 317

 module 305–306, 309, 318

 example 305

 business logic 305

 program 306, 309

 migration 305

 module 305

 program 305

CEDA 203

 commands 178

channel

 name 313

CICS

 CA certificate 235

 PIPELINE 189

 resources in RACF 199

 server certificate 235

 SSLCACHE 45

 trace 248

CICS Business Transaction Services (BTS) 73

CICS Journals 150

CICS JVMs 150

CICS monitoring facility 55

CICS Monitoring Facility (CMF) 72

CICS monitoring facility (CMF) 55

CICS PA

 editing a report form 78

 example JCL 250

 ISPF dialog 74

 Performance Summary report 81

 summary report form 80

 Wait Analysis report. 76

CICS Transaction

 CEDA 203

CICS Web Services monitoring 169

CICSDFLT 59

CICSUSER 198, 200

CIWSMSGH 204

CIWSMSGO 205

client-side interception 109

COBOL 32, 34

commands

 SETPROG 133

COMMAREA 180

 structure 307

Composite application management 104

composite applications 104

CONFIGFILE 207

container

 command 317

 name 313

CPIH 59, 67, 83, 198

CPU 40

Cryptographic hardware 231

CT engine 124

CWXN 59, 67, 83

D

data collector

See DC

DATAREG 172

Datasets 61

DB2 CLI 89

DB2 subsystem 150

DB2 thread activity 150

DB2 transactions 150

DC 123

DelFitrCntrl 121

DelMntrCntrl 121

DFH\$ECFN JCL 178

DFHDELIM 148

DFHECAT JCL 178

DFHEIENT 172

DFHLS2WS 32, 41

DFHPITP 253–254

DFHRPL 148

DFHWS2LS 32

DFHWS-TRANID 205

DFHWS-WEBSERVICE 204

Digital Signature 190

DisableDC 122

Dispatcher Summary 150

Dispatcher TCB modes 150

Dispatcher TCB pools 150

DSALIMIT 180

E

EBCDIC 49, 113

ECFG. 178

EDSALIM 255

EDSALIMIT 180

EGUI 178

element_delay 236

EnableDC 122

ERBRMF00. 61

ERBRMF04 61

ERBRMFxx 63

error

handling 16

error information 307

dedicated container 307

EXEC

CICS

ABEND 308

ABEND ABCODE 319

link 321

link command 306

link program 317

Exit programs 150

Extensible Markup Language (XML) 8, 11

F

Faults Summary workspace 118

FMID 124

FTP 8

full RTE 126

H

HFS directory

image file 328

HKCI310 124

HKD4600 124

HKDS360 124

HKLV190 124

HTTP 4, 7–8

I

I/O devices 61

IBM Redbooks publications Web site 344

IBM Tivoli Composite Application Manager (ITCAM)

88

implementation

ITCAM for SOA 122

IMS 72

input container 311

inquire

catalog request

01INQC 320

parameter 320

single request

01INQS 320

Interval control elements 150

IP connections 150

ISC connections 150

ISPF 54

panels 56

ISPF (TSO WLM). 56

ISPF/PDF 124

ITCAM for SOA 87, 104, 109

- Data collector(DC) 109
- Product components 108
- Product features 115
- Supported application environments 108
- Tivoli Enterprise Monitoring Agent 112
- action 121
- Basic architecture 106
- data collector 109
- implementation 122
- installation on z/OS 122
- overview 104
- Product features 115
- server-side interception 109
- situation 98
- workspace 116
- ITCAM for Web Resources 104
- ITM for Virtual Servers 104
- IWMNTFY 55
- IWMRPT 55

J

- JCL 73
 - DFH\$ECFN 178
 - to create Workload Activity Report 64
 - Workload Activity Report 64
- JCL DFHECAT 178
- JES 59
- JMS 8
- JVM classcaches 150
- JVM™ profiles 150

K

- KCIINST 124
- KD4 config subdirectory 113
- KD4 DCA Archive subdirectory 113
- KD4 DCA Cache subdirectory 113
- KD4 log subdirectory 113
- KD4USSJB 132
- KDSPROC1 133
- KDSPROC2 133
- key ring 200
- KOCCI 143
- KOCRMCLL 172
- KSDS VSAM 178

L

- Log streams 150

- LPAR 5, 151
- LSR pools 150

M

- max_clients 236
- MAXACTIVE 254
- MAXOPENTCBS 252, 254
- MAXSOCKETS 180
- MAXTASKS 256
- meet in the middle 41
- Message Arrival Summary workspace 117
- message exchange pattern (MEP) 8
- Message Transmission Optimization Mechanism (MTOM) 4
- Messages Summary workspace 118
- MINIME boundary 179
- MNSUBSYS SIT 55
- MQ status 150
- MRO connections 150
- MTOM 11, 177
- MTOM/XOP 36, 42
- MTOP/XOP 180
- multi-region operation (MRO) 73
- mustUnderstand 14
- MVS workload manager (WLM) 55
- MVS™ TCBs 150

N

- Namespaces 13
- Navigator Views 153
- NOREPORT 55

O

- OMEGAMON II for CICS
 - CUA interface 146
 - Menu System 142
- OMEGAMON XE
 - monitoring products 142
- OMEGAMON XE for CICS 137
 - Components 138
 - Features 149
 - Resource Monitoring 149
 - Automatic Initiate Descriptors (AIDs) 150
 - Monitoring Agent 138
- Omegamon XE for CICS 87
- Omegamon XE for Messaging 104

Online Data Viewing (ONDV) 162
OPEN TCBs 254
Optimized Packaging (XOP) 179
output container 309

P

PARMLIB 61
Performance Summary workspace 117
persistent connections 39
persistent data store 132
PIPELINE 83, 200
 defining 206
 definition 206
pipeline
 configuration file
 customizing 204
PL/I 32, 34
place order request
 01ORDR 320
PLACE-ORDER 320, 322
PLTPI 148
PLTSD 148
Port definition 29
presentation logic
 DFH0XGUI 306
Processors 61
PROCLIB 142

Q

QUANTITY-REQ 310, 314, 317, 322

R

RACF 189
 database 200
 definitions 177
 identity. 190
 user ID. 189
Redbooks Web site
 Contact us xv
remote procedure call (RPC) 17
Reporting Class 59
request ID 311, 314
Resource limiting (RLIM) 173
Resource Measurement Facility (RMF) 53
response message 311, 315, 319, 322
RESPWAIT 180
return code 311

 separate input container 309
 structure items 309
RKANMOD 133, 148
RKANMODL 133
RKANPARU 146
RKANSAM 132
RMF
 definitions 62
 measures 54
 Address space usage 54
 Channel activity 54
 Coupling Facility (CF) activity 54
 Cross Coupling Facility (XCF) activity 54
 Detailed system paging 54
 Detailed system workload 54
 Device activity 54
 Enqueues 54
 Page and swap data set 54
 Processor usage 54
Monitor 61
monitors 61
 Monitor I 61
 Monitor II 61
 Monitor III 61
overview 53
Startup Procedure 62
Workload Activity Report 64–65
RMF Overview 54
RMF records 55
RMFGAT 62
root-cause 105
RPC 17
 style 30
RTE 124
runtime environment
 See RTE

S

SCA 107
separate container 307, 311
 CA-INQUIRE-REQUEST structure 311
server program 307, 309
 output containers 309
server-side interception 109
Service Component Architecture
 See SCA
Service Management Agent workspace 116
SETPROG command 133

- sharing RTE 126
- single container 307, 311, 316
- SIT parameters 198
 - SEC 198
 - SEC=YES 198
 - SECPRFX 198
 - SECPRFX=YES 198
 - XTRAN 199
 - XTRAN=YES 198
 - XUSER 199
 - XUSER=YES 198
- situation 98
- Situations 120
- SMF 55
 - 101 records 72
 - 110 records 72
 - 111 records 72
 - 116 records 72
 - 88 records 72
- SMP/E 123
- SMTP 8
- SOAP 8
 - binding 30
 - body 15
 - communication styles 17
 - document 17
 - RPC 17
 - encodings 17
 - literal 17
 - SOAP encoding 17
 - envelope 12, 14
 - fault 16
 - headers 14
 - intermediary 14
 - introduction 12
 - message 65
 - messaging mode 18
 - MustUnderstand 15
 - namespaces 13
- SOAP Body 41
- SOAP Envelope 41
- SOAP Headers 41
- SOAP Message Transmission Optimization Mechanism (MTOM) 36
- SOCKETS 255
- SSL 45
- SSLDELAY 45
- Startup_delay 236
- Storage 61

- Storage DSA usage 150
- Storage usage 150
- SYSEVENT 55
- System initialization values 150
- System Modification Program/Extended
 - See SMP/E
- system resources manager (SRM) 54
- System z 190
- System z10 222

T

- TCP/IP port 181
- TCPIP activity 150
- TCPIP services 150
- TCPIPSERVICE 181
 - attributes
 - PORTNUMBER 203
 - PROTOCOL 203
 - definition 203
- Temporary storage 150
- TEMS 140
- TEPS
 - 89
 - database 89
- Tivoli Data Warehouse (TDW) 91
- Tivoli Enterprise Monitoring Agents (TEMA) 91
- Tivoli Enterprise Monitoring Server 88
- Tivoli Enterprise Portal Server 89
- Tivoli Enterprise Portal Server (TEPS) 89
- Tivoli Management
 - Services components 87
 - Services resources 87
- TLS 45
- TRANCLASS 254
- Transient data 151

U

- UNIX System Services 132
- UOW disposition 151
- UOW enqueues 151
- updateLogging 122
- updateTracing 122
- UpdMntrCntrl 122
- URI 30
- URIMAP 83
- USERKEY 254
- UTF-8 49, 113

V

Virtual Storage Access Method (VSAM) 319

VSAM

data handler

DFH0XVDS 319

module 318

module DFH0XVDS 318

return 322

stub DFH0XVDS 305

datasets 178

file 178

VSAM (Virtual Storage Access Method) 319

VSAM files 151

VSAM RLS conflicts 151

VTAM 158

W

Warehouse Proxy Agent (WPA) 92

Web service

Interoperability 10

properties 7

Web service provider 188

Web Service Proxy 191

Web Services Description Language (WSDL), 7

Web Services monitoring 169

Web Services Trust Language 11

WebSphere Service Registry and Repository (WSRR) 106

WebSphere Studio Workload Simulator (WSWS) 232

WLM 56

Classification Rules 58

definition panel 57

main panel 50, 56

Workload Activity Report 64

Workload Activity report 66

WS-Atomic Transaction 11

wsbind files 177

WSDIR 207

WSDL 18

binding 19, 27

bindings 30

definition 24

document 19

anatomy 20

message 19, 25

namespaces 23

operation 19, 26

port 19, 29

port type 19, 26

service 19

service definition 29

SOAP binding 30

type 19

types 24

Web Services Description Language 9

WSDL 2.0 34

WSDL Document 19

WS-Security 43, 187, 230

X

X.509 certificate 187, 230

X509 certificate 189

X509 certificate to CICS 189

XCF 61

XML 7

complexity 37

XML digital signature 188

XML-binary Optimized Packaging (XOP) 36

XSL 194

Z

z/OS 4

Considerations for CICS Web Services Performance

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages

Considerations for CICS Web Services Performance

Performance advantages of MTOM

Recommendations and tools

Security scenarios

The Web services support in CICS Transaction Server Version 3 enables your CICS programs to be Web service providers or requesters. CICS supports a number of specifications including SOAP Version 1.1 and Version 1.2, and Web services distributed transactions (WS-Atomic Transaction).

This IBM Redbooks publication reviews CICS Web Services performance in two particular areas:

- ▶ **MTOM/XOP:** Here we focus on performance benefits that can be achieved by taking advantage of the MTOM/XOP standard to transmit large binary data objects. The single inquiry function of the CICS catalog manager application has been modified to return an image of the requested item in addition to its details. This function is exposed through a Web service.
- ▶ **Security:** There are a number of ways to secure your CICS Web Service messages. Here we look at two technologies: WS-Security and DataPower®. We compare the performance and relative merits of using WS-Security with and without DataPower, and discuss what factors might influence your choice of technology.

We highlight tools that can help you to understand the performance profile of Web service interactions with CICS, such as:

- ▶ RMF™ z/OS Resource Measurement Facility (RMF)
- ▶ IBM CICS Performance Analyzer for z/OS (CICSPA)
- ▶ ITCAM for SOA
- ▶ OMEGAMON® XE for CICS on z/OS®

This book considers performance by using different scenarios including Security, MTOM/XOP, and the use of the IBM Tivoli® Monitoring tools to help identify problems that can affect the performance of Web Services. Specifically, we use ITCAM for SOA and OMEGAMON XE for CICS on z/OS to show how these tools can be of benefit to identify the cause when the performance of a Web service in CICS becomes unacceptable.

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