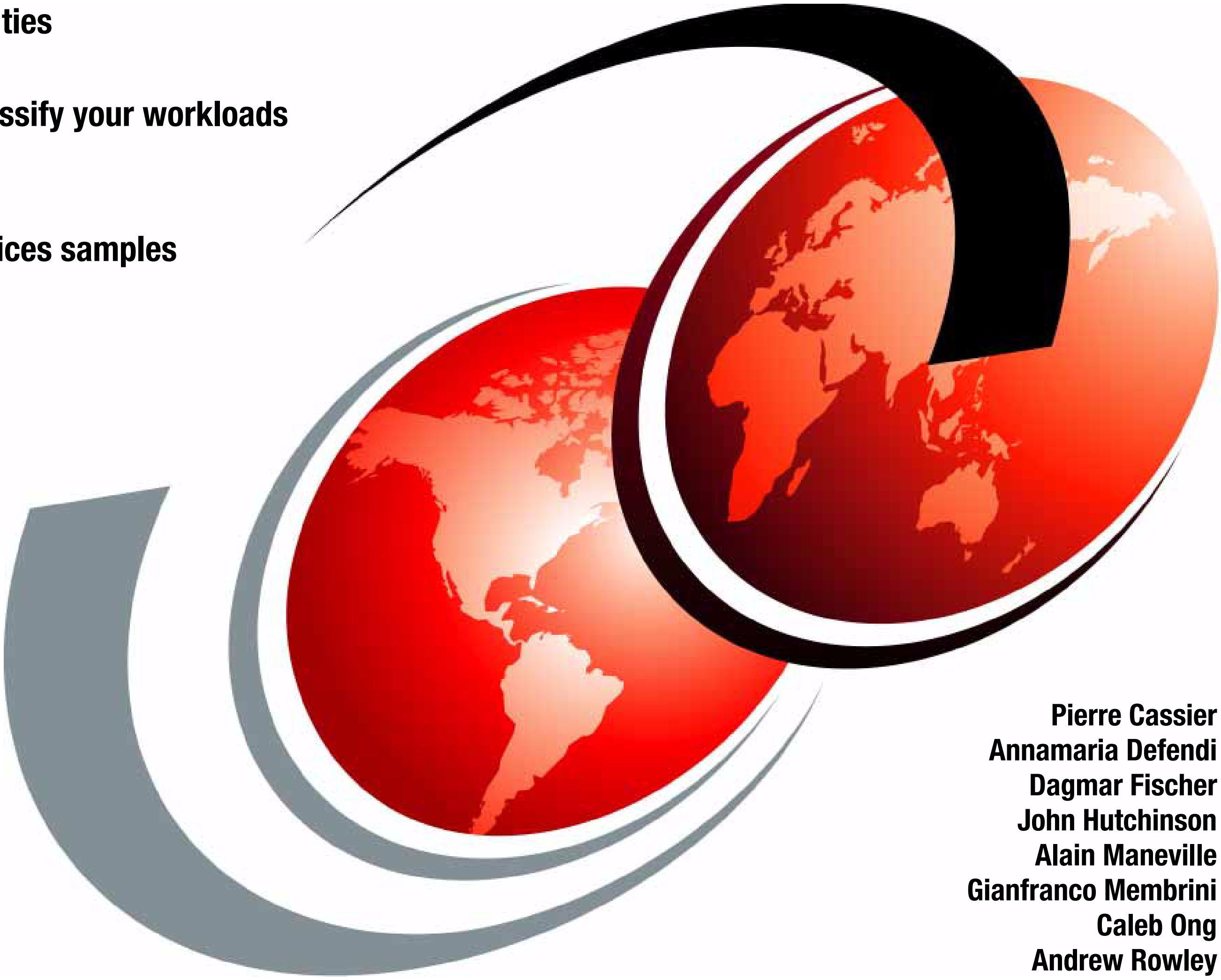
System Programmer's Guide to: Workload Manager

Workload Manager overview and functionalities

How to classify your workloads

Best practices samples



Pierre Cassier
Annamaria Defendi
Dagmar Fischer
John Hutchinson
Alain Maneville
Gianfranco Membrini
Caleb Ong
Andrew Rowley

Redbooks

ibm.com/redbooks

International Technical Support Organization

System Programmer's Guide to: Workload Manager

March 2008

SG24-6472-03

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

Forth Edition (March 2008)

This edition applies to Version 1, Release 8 of z/OS (product number 5694-A01).

Contents

Notices xi
Trademarks xii

Preface xiii
The team that wrote this redbook. xiii
Become a published author xiv
Comments welcome. xv

Chapter 1. Introduction. 1
1.1 Introduction to workload management 2
1.1.1 Basic concepts 2
1.1.2 Service Level Agreement 6
1.2 Workload Manager components 7
1.2.1 Service definition and service policies 8
1.2.2 Workload classification 9
1.2.3 Service classes 10
1.2.4 Importance of a goal 12
1.2.5 Resource Group 12
1.2.6 Application Environment 13
1.2.7 Scheduling environment 14
1.2.8 Sampling dispatchable unit states 14
1.3 Performance management metrics 15
Average transaction response time 15
External throughput rate (ETR) 17
Resource utilization 18
Saturation design point (SDP) 19
1.3.1 Processor performance metrics 20
CPU time 20
Cycle time 21
Millions of instructions per second (MIPS). 21
MFLOPS 21
CPU Service Units: System Resource Manager (SRM) constant 22
Internal throughput rate (ITR) 22
Large Systems Performance Reference (LSPR). 23
1.3.2 I/O performance and metrics. 25

Chapter 2. How WLM works 29
2.1 Workload management. 30
2.2 z/OS dispatchable units (DUs) 30
WLM Business Unit of Work 31
The idea of dispatchable units. 32
Enclave 34
2.3 Understanding the algorithms 35
Workload management 35
2.3.1 Performance index 36
Sysplex PI and local PI 36
Performance index evaluation in a sysplex 37
2.3.2 Policy adjustment function 40
Impact of the number of service class periods. 42

Donor and receiver determination 43
Selecting a receiver candidate. 45
Find a receiver bottleneck 45
2.3.3 CPU management 46
Dispatching priorities assignment 47
Discretionary goals 50
2.3.4 Swap control 51
Swap protect time 51
2.3.5 Storage targets 51
2.3.6 I/O management 52
I/O queues 52
DASD I/O priority 53
TAPE I/O priority 53
Sysplex I/O priority management. 53
I/O priority management decisions 54
Channel subsystem I/O priority queuing (CSS IOPQ) 54
Dynamic Channel Path Management (DCM). 55
Parallel Access Volumes (PAV). 55
2.4 Transactions and management. 61
2.4.1 How WLM treats work in the system. 61
2.4.2 Enclave management 64
Classification 64
Application Environment enclave server 64
Application Environment enclave server topology 64
2.4.3 Execution delay monitoring. 65
Monitoring environment or performance block (PB) 66
Services 66
Server topology 67
Server management 71
Transaction phase 71
Summary 73

Chapter 3. WLM functions 75
3.1 Protecting work 76
3.1.1 Resource Groups 76
Types of Resource Groups 76
Defining Resource Groups. 77
Resource Group maximum management 78
Service ramps up. 80
RMF reporting 81
Resource Group usage examples 82
3.1.2 CPU critical 85
CPU CRITICAL 86
CPU CRITICAL in action 86
Who should use CPU CRITICAL. 87
Where to specify CPU CRITICAL 87
3.1.3 Storage critical 88
Who should use STORAGE CRITICAL 88
Where to specify STORAGE CRITICAL 88
3.1.4 Initiator dispatch priority 90
Initiator behavior 90
3.1.5 WLM CPU preemption 90
3.1.6 ENQ management 90
3.2 Server address space management 91
3.2.1 Application Environment 92

Application Environment queues . . . 93
Transaction flow in an Application Environment . . . 93
Application Environment and transaction flow example. . . 93
3.3 Workload routing support . . . 95
3.3.1 VTAM generic resources. . . 95
VTAM exit customization . . . 96
3.3.2 Sysplex routing . . . 97
Calculation of server weight. . . 98
Sysplex routing user: z/OS Load Balancing Advisor . . . 99
Sysplex routing user: DB2 Connect. . . 100
3.4 Soft capping . . . 101
3.4.1 Defined capacity . . . 101
3.4.2 Group capacity . . . 102
3.5 Intelligent Resource Director. . . 102
Policy considerations for IRD. . . 103
Importance and goal of a service class . . . 103
Minimum number of logical processors . . . 103
3.5.1 Linux for System z LPAR management . . . 103
Setting the cluster name . . . 103
Setting the system name . . . 103
Setting the goal . . . 104
3.6 Specialty processors. . . 104
3.6.1 Internal Coupling Facility Processor (ICF) . . . 105
3.6.2 Integrated Facility for Linux Processor (IFL) . . . 105
3.6.3 zSeries Application Assist Processor (zAAP) . . . 106
zSeries Application Assist Processor and Java code flow. . . 106
3.6.4 zSeries Application Assist Processor positioning . . . 107
3.6.5 WLM zAAP DUs states . . . 108
3.6.6 z9 Integrated Information Processor (zIIP) . . . 109
z9 Integrated Information Processor and DB2 code flow. . . 109
3.6.7 z9 Integrated Information Processor positioning. . . 110
3.6.8 WLM zIIP DU states . . . 110

Chapter 4. Implementation and workload classifications . . . 113
4.1 Overview . . . 114
4.2 WLM implementation . . . 114
4.2.1 IEAOPT*xx* usage . . . 114
4.2.2 WLM ISPF application . . . 119
Security access . . . 119
Starting your WLM ISPF application . . . 120
4.2.3 WLM couple data set management . . . 121
Allocating the WLM couple data set. . . 121
Managing your WLM CDS . . . 122
4.2.4 Specifying your service definition . . . 123
4.2.5 Service policy definition . . . 126
4.2.6 Workload specification . . . 126
4.2.7 Resource Group specification. . . 127
4.2.8 Working with service classes . . . 128
4.2.9 Classification rules . . . 129
4.2.10 Service definition coefficient . . . 138
4.2.11 Report class . . . 139
4.2.12 Application Environment . . . 140
4.2.13 Scheduling environment . . . 141
4.2.14 Policy overrides. . . 142

4.2.15 How to use policy overrides 143
Adding and removing a Resource Group. 143
Overriding a Resource Group definition. 146
4.2.16 Evaluating your SMF records 149
4.2.17 Installing and activating your service definition. 150
4.3 Current WLM users upgrading to a new OS 151
4.3.1 WLM compatibility and coexistence 151
4.3.2 WLM enhancements matrix 152
4.4 WLM tools 153
4.4.1 Service Definition Formatter 153
4.4.2 WLM Work Queue Viewer. 153
4.4.3 WLM Service Definition Editor 154
Getting started 155

Chapter 5. Setting goals 167
5.1 Using service classes 168
5.2 Defining and maintaining service goals. 169
5.2.1 Using historical data to set goals 170
5.2.2 Using response time goals 172
5.2.3 Using response time goals for CICS and IMS. 174
5.2.4 Using execution velocity goals 176
Considerations for using velocity goals 177
5.2.5 Using the SYSSTC service class 178
5.2.6 Using service class periods. 179
5.2.7 Using importance levels 180
5.2.8 Using discretionary goals 182
When discretionary work receives too little service 183
Capping overachieving work 183
5.2.9 Using report classes 183
5.2.10 Using Resource Groups 184
5.2.11 Using CPU and storage critical 185
5.3 Managing service definitions. 186
5.3.1 Revisiting service definitions. 187
5.3.2 Reevaluating service definition 187
5.3.3 When things are not working 187

Chapter 6. Batch considerations 191
6.1 General considerations 192
6.1.1 WLM-managed initiators 192
How WLM batch initiator management works 192
JES2 service class management. 194
Jobs with system affinity 195
Considerations for multiple JES2 MAS complexes in a sysplex 195
Operational changes 196
6.1.2 Scheduling environment 197
6.1.3 TWS integration with WLM 197
Implementation tasks. 200
6.2 Types of goals. 203
6.2.1 Response time goal 203
Good candidates for this type of goal 204
6.2.2 Velocity goal 204
Good candidates for this type of goal 204
6.3 Classification rules 204
Classification 205

6.4 Best practices 206
6.4.1 WLM-managed initiators 206
Fragmentation 206
Balancing 207
JES2 parameter 207
6.4.2 Quiescing a batch job 208
6.4.3 Choosing the right velocity 208
Test case 208
6.4.4 Usage and number of periods. 210
6.5 Resource Group usage 210
6.5.1 First test case 211
6.5.2 Second test case 211
6.6 Impact of the number of engines on velocity. 212
6.6.1 First case results. 212
6.6.2 Second case results 213

Chapter 7. TSO, STC, and APPC workloads 217
7.1 General considerations for TSO 218
7.2 Types of goals for TSO 218
7.3 Best practices for TSO 220
7.4 Classification rules for TSO. 220
7.5 Reporting for TSO. 221
7.6 General considerations for STC 224
SYSTEM service class. 224
SYSSTC service class. 225
7.7 Classification rules for STC. 226
7.8 Types of goals for STC 227
7.9 Best practices for STC 227
7.10 Reporting for STC 229
7.11 General considerations for APPC 230
7.12 Classification rules for APPC 230
7.13 Types of goals for APPC. 231
7.14 Best practices for APPC 231
7.15 Reporting for APPC. 231

Chapter 8. DB2 workload considerations 233
8.1 General considerations 234
8.2 DB2 address spaces 234
8.2.1 DB2 address space velocity goals 234
8.2.2 WLM bufferpool management. 235
8.2.3 How DB2 assigns I/O priorities 235
8.3 DB2 distributed environment. 236
8.3.1 Thread and enclave relationship. 236
8.3.2 Duration of an enclave 236
8.3.3 DDF enclave creation 237
8.3.4 Classification rules for a distributed environment 238
Considerations for JDBC requests. 239
8.3.5 Types of goal 239
8.3.6 Best practices 240
8.3.7 Enclave reporting 240
8.4 DB2 Stored Procedures 242
8.4.1 What are stored procedures 242
8.4.2 WLM considerations for DB2 Stored Procedures 245
Considerations for enforcing WLM goals. 246

8.4.3 WLM definition for DB2 Stored Procedures 246
Using WLM-established address spaces. 246
Define your classification rules 250
Defining your service classes 251
Performance goals for the server address spaces 251
8.5 DB2 sysplex query parallelism 251
8.5.1 Query parallelism 251
8.5.2 How sysplex query parallelism works 252
8.5.3 WLM considerations for sysplex query parallelism 253
8.5.4 WLM definition for sysplex query parallelism 254
Classification rules. 254
Types of goals 256

Chapter 9. WebSphere Application Server workload considerations 257
9.1 General considerations 258
9.2 WebSphere on z/OS and WLM. 259
9.2.1 Queuing services 259
WLM temporal affinity 260
9.2.2 Enclaves 261
9.2.3 Application Environments 261
9.3 Classifications rules 263
9.3.1 Classification rules for STC subsystem. 264
9.3.2 Classification rules for CB subsystem. 265
Assigning transaction classes 266
9.4 Types of goals. 279
9.5 Reporting. 279

Chapter 10. UNIX System Services considerations. 283
10.1 General considerations 284
10.1.1 Unix System Services terminology 284
10.1.2 Unix System Services address spaces. 284
10.1.3 USS and WLM 286
UNIX Services fork and spawn function calls 286
Fork, spawn, and enclave propagation 288
10.2 Controlling the performance of your USS work. 289
10.2.1 Types of goals and classification rules 290
Definitions for USS started tasks 290
Definitions for OMVS subsystem type work. 290
OMVS and TSO/E response time 292
10.2.2 Control using UNIX functions 293
10.3 Reporting. 293
SDSF Processes panel 293
RMF III OPD 294
RMF Postprocessor. 295

Chapter 11. Transactional workload considerations 297
11.1 General considerations for CICS 298
11.2 Types of goals. 298
11.2.1 Region management goal. 299
CPU and storage-critical assignment in region management mode 299
Service class reporting in region management mode 300
11.2.2 Transaction response time management 300
CPU and storage-critical assignment in transaction management mode 302
11.3 Classification rules 303

11.4 CPU contention considerations . . . 303
11.4.1 CICS and batch jobs contention analysis . . . 304
11.4.2 CICS regions contention analysis . . . 306
11.4.3 CICS response time contention analysis . . . 307
11.4.4 Contention analysis summary . . . 307
11.5 Reporting . . . 307
11.5.1 Reporting enhancement . . . 307
Service definition used for this test . . . 308

Chapter 12. Uniprocessor considerations . . . 311
12.1 CPU contention in a uniprocessor system . . . 312
12.1.1 Dispatching priorities . . . 312
12.1.2 Test case . . . 313
Test one: Very high goals . . . 314
Test two: Reduced goals . . . 315
Test three: Discretionary work . . . 316
Test four: The effect of goals on capping . . . 317
Test five: Resource Groups . . . 318
Conclusions . . . 319
12.2 Recommendations . . . 320
12.2.1 Number of service classes . . . 320
12.2.2 Goals . . . 320
12.2.3 SYSSTC . . . 320
12.2.4 CPU Critical . . . 321
12.2.5 Discretionary work . . . 321

Chapter 13. Enterprise Workload Manager . . . 323
13.1 Introduction to EWLM . . . 324
ARM-instrumented middleware . . . 325
13.1.1 EWLM and zWLM policies . . . 327
13.1.2 Platform support in EWLM 2.1 . . . 329
Domain managers . . . 329
Managed servers . . . 329
ARM-instrumented middleware . . . 329
13.2 Getting started . . . 329
13.2.1 How EWLM interfaces with z/OS . . . 330
13.3 Using the EWLM Control Center . . . 333
13.3.1 Monitoring the use of the EWLM Control Center . . . 336

Related publications . . . 341
IBM Redbooks . . . 341
Other publications . . . 341
Online resources . . . 341
How to get IBM Redbooks . . . 342
Help from IBM . . . 342

Index . . . 343

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 illustrates 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. 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 IBM's application programming interfaces.

Trademarks

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

Redbooks (logo) ™
i5/OS®
z/Architecture®
z/OS®
z/VM®
zSeries®
z9™
AIX 5L™
AIX®
CICS®
Distributed Relational Database Architecture™
Domino®
DB2 Connect™
DB2 Universal Database™
DB2®
DRDA®
Enterprise Storage Server®
ESCON®
FICON®
IBM®
IMS™
Lotus®
MQSeries®
MVS™
NetView®
OS/390®
OS/400®
Parallel Sysplex®
POWER5™
PR/SM™
Redbooks™
RACF®
RMF™
System z™
System z9™
SOM®
SOMobjects®
Tivoli®
VSE/ESA™
VTAM®
WebSphere®

The following terms are trademarks of other companies:

ABAP, SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates.

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

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

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, 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, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook gives a broad understanding of the Workload Manager (WLM) component of the z/OS® system. It also provides suggestions about how to classify the different workloads to WLM so it can best manage the different types of units of work.

This book assumes that the starting environment is a z/Architecture® running a z/OS V1R8 level of operating system. For this reason, there are no references to components, for example, expanded storage, compatibility mode, and so forth, that no longer exist.

This book also provides a brief introduction to the Enterprise Workload Manager and describes the current interactions with z/OS WLM.

The team that wrote this redbook

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

Pierre Cassier is a Certified IT Specialist working for IBM France in Integrated Technology Services. He has more than 21 years of systems programming experience in mainframe environments on MVS™, OS/390®, and z/OS platforms. His areas of expertise include z/OS, Parallel Sysplex®, WLM, and performance tuning. He teaches WLM and Performance Management courses.

Annamaria Defendi is an Advisory IT Specialist working for IBM Italy in Integrated Technology Services. She has 15 years of experience in the MVS, Parallel Sysplex, and WLM areas. Since 2001 she has been a member of the z/OS EMEA Back Office Team, providing Level 2 support to the EMEA Clients. She teaches software courses on different MVS components for clients and for IBM internal education.

Dagmar Fischer is an IT Specialist working at Informatik Zentrum (IZB) in Munich, Germany. She has more than 22 years of experience in mainframe environments on MVS, OS/390, and z/OS platforms.Her areas of expertise include Parallel Sysplex, WLM and performance tuning issues, and capacity planning.

John Hutchinson (Hutch) is a Certified Senior Consulting IT Specialist at the Washington Systems Center (WSC) in Gaithersburg, Maryland, where he is the Team Leader for the WebSphere® for z/OS technical support group. He has over 36 years of experience with the installation, migration, and management of large, complex MVS systems, complexes, and networks. Hutch joined the WSC in 1976, and served as lead JES2 Technical Support leader, consultant to NJE Protocol Review Board, and IBM Representative to JES2 Projects at the GUIDE and SHARE user groups.

Alain Maneville is an IT Specialist working for IBM France in Field Technical Sales Support for zSeries® and z/OS. He has more than 20 years of systems programming experience in mainframe environments on MVS, OS/390, and z/OS platforms. Alain joined IBM in 2001 and worked for 13 years as a Consultant System Engineer for Amdahl Corp. His areas of expertise include zSeries hardware, z/OS, Parallel Sysplex, WLM and performance tuning, and z/Architecture. He has been involved in migrations to WLM in various client environments.

Gianfranco Membrini is an Advisory IT Specialist working for IBM Italy in Integrated Technology Services. He has 18 years of experience in MVS, OS/390, z/OS, WLM, and the Parallel Sysplex environment. His areas of expertise include zSeries hardware configuration and performance tuning.

Caleb Ong is an IT Specialist from IBM Philippines, Integrated Technology Services division. He holds a degree in Electronics and Communication Engineering from the University of Santo Tomas. He has been in the IT industry for more than 15 years.

Andrew Rowley has 14 years experience as a Systems Programmer on MVS, OS/390, and z/OS. He is particularly interested in Workload Manager and z/OS performance. He is the owner of Black Hill Software Pty. Ltd. located in Ballarat, Australia. The Web site is:

`http://www.blackhillsoftware.com/zos`

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

Paola Bari, Richard Conway, Bob Haimowitz
International Technical Support Organization, Poughkeepsie Center

Peter Baeuerle
IBM Germany

Laura Blodgett
IBM Poughkeepsie

Mike Cox
IBM Gaithersburg

Craig Day
IBM Poughkeepsie

Franco Meroni
IBM Italy

John Kinn
IBM Poughkeepsie

Alvaro Salla
Consultant

Pascal Tillard
IBM France

Robert Vaupel
IBM Germany

Dieter Wellerdiek
IBM Germany

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@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

1

Introduction

This chapter introduces the concept of workload management and provides a basic description of the Workload Manager (WLM) z/OS component.

In this chapter, we discuss the following topics:

- Introduction to workload management
  - Basic concepts
  - Service Level Agreement (SLA)
- WLM components
  - WLM application
  - Workload classification
  - Service definition and service policies
  - Sampling units of work states
  - Service classes
  - Importance of a goal
  - Resource Group
  - Application Environment
  - Scheduling environment
- Performance management metrics

1.1 Introduction to workload management

This section provides an introduction to the workload management concept, focusing on its goals and the problems that it solves.

1.1.1 Basic concepts

One of the strengths of the zSeries platform and the z/OS operating system is the ability to run multiple workloads at the same time within one z/OS image or across multiple images. The function that makes this possible is dynamic workload management, which is implemented in the Workload Manager component of the z/OS operating system.

The idea of z/OS Workload Manager is to make a contract between the installation (user) and the operating system. The installation classifies the work running on the z/OS operating system in distinct service classes and defines goals for them that express the expectation of how the work should perform. WLM uses these goal definitions to manage the work across all systems of a sysplex environment.

In order to understand how WLM works, we can look at the traffic in a city, as shown in Figure 1-1 on page 3. Usually, when we drive a car between two locations, the time that we need is determined by the constant speed we are allowed to go, the amount of traffic on the streets, and the traffic regulations at crossings and intersections. Based on this, we can already identify the time as the basic measure to go from a start to a destination point; we can also easily understand that the amount of traffic is the determining factor of how fast we can go between these two points. While the driving speed is usually regulated and very much constant, the number of cars on the street determines how long we have to wait at intersections and crossings before we can pass through them. As a result, we can identify the crossings, traffic lights, and intersections as the points where we encounter delays on our way through the city; a traffic management system can now use these considerations to manage the running traffic. For example, dedicated lanes for busses and taxis allow them to pass the waiting traffic during rush hours and therefore to travel faster than passenger cars. This is a common model of prioritizing a certain type of vehicle against others in the traffic systems. So far, we see that it is possible to use the time as a measure between two points for driving in a city, we identified the points where contention might occur, and we found a simple but efficient method of prioritizing and managing the traffic.

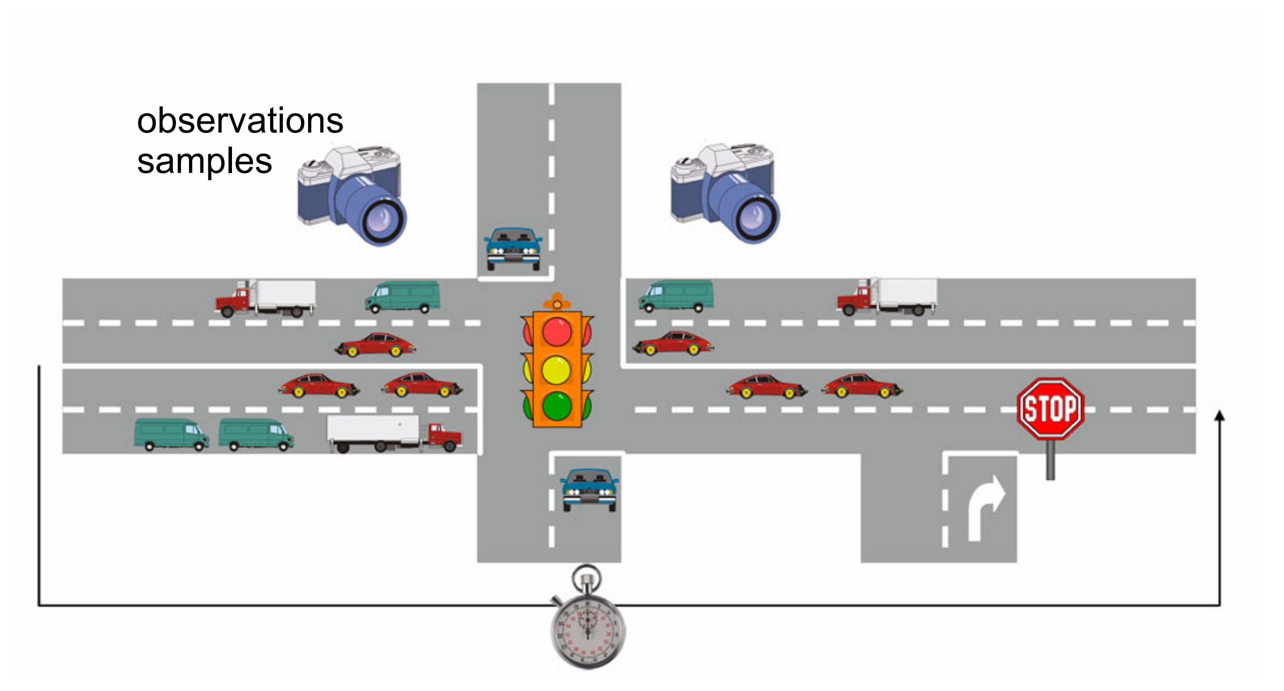


Figure 1-1 Traffic management concepts

These concepts can be easily ported to a system; all we need is a technique to:

- Identify the work running in the system.
- Measure the time it runs.
- Find out the contention points.

The identification of work requests is supported by the middleware and the operating system; they tell WLM when a new unit of work enters the system and when it leaves the system. WLM provides constructs to separate the work into distinct classes. The name of this is *work classification,* and it allows an installation to deal with different types of work running on the system.

Contention occurs when the work wants to use the system resources. These are the CPUs, the I/O devices, and storage, but also software constructs, such as processes or address spaces, which provide the capability to execute programs and serialization points that allow the programs to access resources or serialized control areas. WLM monitors these resources to be able to understand how many resources the work requires or will wait for.

The work classification and WLM observation of how the work uses the resources provide the base to manage the system. This management is done based on the goals that the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class; the goal determines how much service the work in the class is able to receive. These classes of work are named *service classes*.

Beside the goal, it is also necessary to define which work is more important than other work. In the case where several service classes do not achieve their target goals, the *importance* helps WLM decide which service class it should help get resources.

Figure 1-2 on page 4 describes the WLM constructs used for performance management. In the service policy, we have workloads. For each workload, we have service classes indicating the goal and the importance of the associated work. Then, we have the classification rules to link a workload to a service class, optionally the Resource Groups; and, in addition, Application Environments and scheduling environments. In the next sections, we analyze each component in detail.

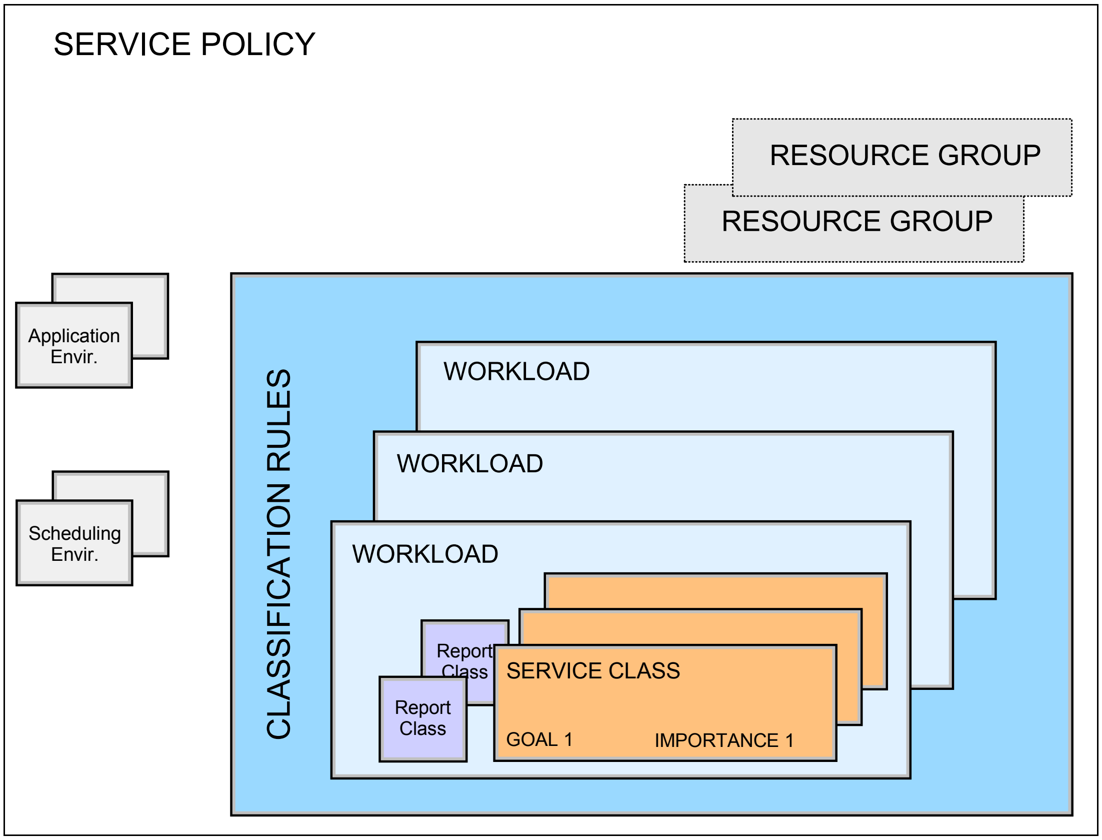


Figure 1-2 WLM constructs

Now we have all the basic constructs to define a policy, but we have to clarify the goals of our service classes. From the traffic example, we see that the most natural goal is the time that is required for a car to go between two points. The same is true for work running in a system: The time that the work needs between the points when it enters and leaves the system can be used as a base to manage it. Because a system deals with many work requests running in parallel, the time of an individual request becomes less important. The average time the work needs to fulfill its task on the system is more interesting, because this is also the time the user can observe while waiting for a response from the system. This time is named the *Response time*.

As we will see later, the average response time can be influenced by very few long-running transactions. In order to become more independent for such events, we want a more stable goal definition. For example, we want a certain number of work requests to end within a predefined time. Other work requests can run much longer, but we will consider the goal as achieved when the defined number of requests ends in the expected time period. This goal is named a *percentile response time* goal and defines the percentage of all work requests that must end within a predefined time limit.

Using the time as a measure to manage the work on a system requires that the system has the ability to capture it. Usually, an operating system provides processes and address spaces to execute the application programs; these constructs are often started once and live for a very long time. That means that WLM needs assistance from the middleware to capture the response time. Indeed, WLM provides interfaces that capture the time the work needs when it runs on z/OS. WLM also supports an infrastructure that allows the middleware to identify individual work requests, capture their response times, and thus allow WLM to manage them in their own service classes. Therefore, on z/OS, WLM is able to manage service classes of transactions that belong to the same application and that are executed by a set of server address spaces.

At this point, the question that arises is what the system can do when it is not possible to capture the response times of the transactions. There are many reasons why this is necessary: Middleware might not be able to tell WLM the response times of the middleware transactions, or there are too few and too sporadic transactions, so that it does not make sense to manage them toward a response time goal. Also, many address spaces provide services in the system that cannot be easily attributed to a certain type of transaction. For those address spaces, it is also necessary to understand how fast they progress in the system and to define a goal for them. Because the system is not able to capture the response time of this type of work, it must use the information that it is able to collect, such as information about the delay and using values that it observes when the work tries to use resources in the system. This information can be obtained by observing where delay situations occur and constantly taking snapshots. After a certain time period, these snapshots are summarized and the result helps to identify how often individual work and the service classes were delayed for a resource and how often they were able to use it. The speed in the system can now be expressed by the acceptable amount of delay for the work when it executes. The speed is named *execution velocity*, a number between 0 and 100, where 100 means that all measured samples are using samples and that the work did not encounter any delays for resources managed by WLM, and 0 means that the work was completely delayed over the measurement interval.

Let us now try to compare the potential goals that can be defined for work in the system. On one side, we can have a response time goal. Independently of whether we use an average or percentile response time goal, we can always immediately understand what this means with respect to how fast the work progresses in the system. On the other side, we have an execution velocity goal. Clearly, we see that an execution velocity of 100 means no delay and therefore it means the highest speed. But what does an execution velocity goal of 30 mean? With respect to the traffic example, we know that it is always possible to go very fast by car in a city on a Sunday morning, but not during rush hours where a fast speed always means that the car has to wait at intersections and traffic lights. That also means that the best we can do during rush hours implies that we have to wait quite frequently and that a speed expressed as an execution velocity will be very low. If we want to define a reliable goal that can be achieved under all circumstances, we have to understand how the system behaves during rush hour. Such goals can be easily achieved during other times, because there is no contention and everybody and everything can progress very fast. When we have two service classes of work, one for which we can measure the response time, and the other for which we cannot measure the response time, we need a way to compare the potential goal settings against each other. Because it is very difficult to understand how an execution velocity translates into a response time, we have to capture the execution velocity of the work being managed toward response time goals simply because we need to understand what this means as an execution velocity.

And last, in addition to setting response time or velocity goals, you might want to have work running in the system for which no concrete goal must be achieved. This work should only run when plenty of resources are available. This work is classified to service classes with a *discretionary* goal and thus tells the system that it should only run when service classes with higher importance do not need the resources to achieve their goals.

In the next sections, we analyze in more depth the WLM concepts and the constructs that WLM uses for workload management. The starting point of the workload control process is to define a Service Level Agreement between installation and the users.

1.1.2 Service Level Agreement

The perception of the performance of a system is often subjective and inaccurate. In order to make it more objective and more effective for business needs, the concept of a *Service Level Agreement* (SLA) was introduced. SLA is a fundamental step for performance analysis and capacity planning disciplines.

An SLA is a contract that objectively describes:

- Average transaction *response time* for network, I/O, CPU, or total
- Distribution of response time (for example, 90% Time Sharing Option (TSO) trivial at less than 0.2 of a second)
- Throughput
- System availability (percentage of time that the system is available to users)
- System load

Note: We do not recommend that you include the external throughput rate (ETR) in an SLA, because ETR includes variables not under control of the system administrators, such as the number of users and user "think time." Refer to "External throughput rate (ETR)" on page 17 for more information about ETR.

A transaction or a business unit of work is an interaction (online or not) with a CICS® subsystem, a IMS™ subsystem, a Web server, a batch job, and so on.

Figure 1-3 on page 7 shows an example of an SLA where the business staff and the system administrators of the data center agree that the installation should provide the following level of service for the application ABC:

- Ninety-five percent of all the data entry transactions must have an average response time less than 0.3 seconds.
- This service level must be kept during the time frame between 08:00 am - 17:00 pm of the working days.
- The application ABC has to be available for 96% of the service hours.

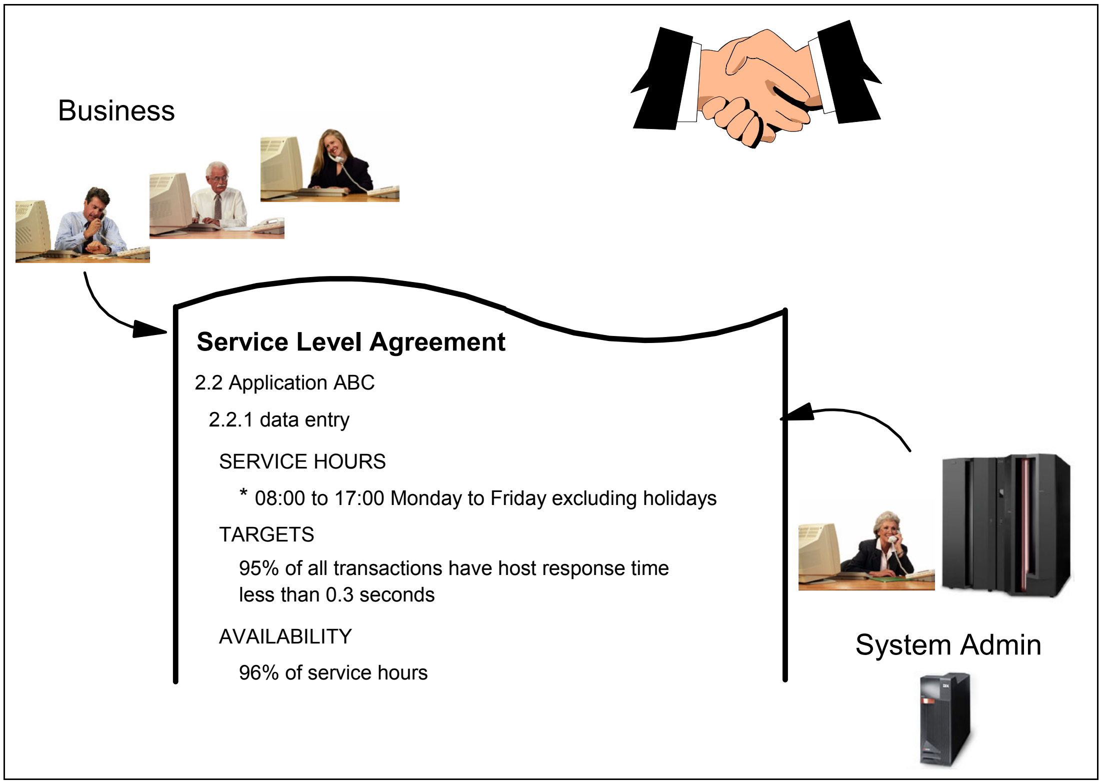


Figure 1-3 Service Level Agreement example

1.2 Workload Manager components

All the business performance requirements of an installation are stored in a *service definition*. There is one service definition for the entire sysplex.

The service definition contains the elements that WLM uses to manage the workloads. These include:

- One or more *service policies*.
- *Workloads*: Arbitrary names used to group various service classes together for reporting and accounting purposes. At least one Workload is required.
- *Service classes*: Where you define the goal for a specific type of work.
- *Report classes*: Aggregate set of work for reporting purpose.
- *Performance goals*: The desired level of service that WLM uses to determine the amount of resource to give to a unit of work.
- *Classification rules* and *classification groups*: Used to assign the incoming work to a service class and, if needed, to a report class.
- *Resource Groups:* Used to assign a minimum and a maximum amount of CPU SU/sec to one or more service classes.
- *Application Environments*: Set of address spaces with the same characteristics suitable for transactions requested by a client and executed in a set of server address spaces. WLM can dynamically manage the number of server address spaces to meet the performance goals of the work making the requests.

- *Scheduling environments*: A list of resource names and their required state. The scheduling environment allows you to manage the scheduling of work in an asymmetric sysplex where the systems differ in installed applications or installed hardware features.

Figure 1-4 shows the hierarchy of the WLM elements within the service definition.

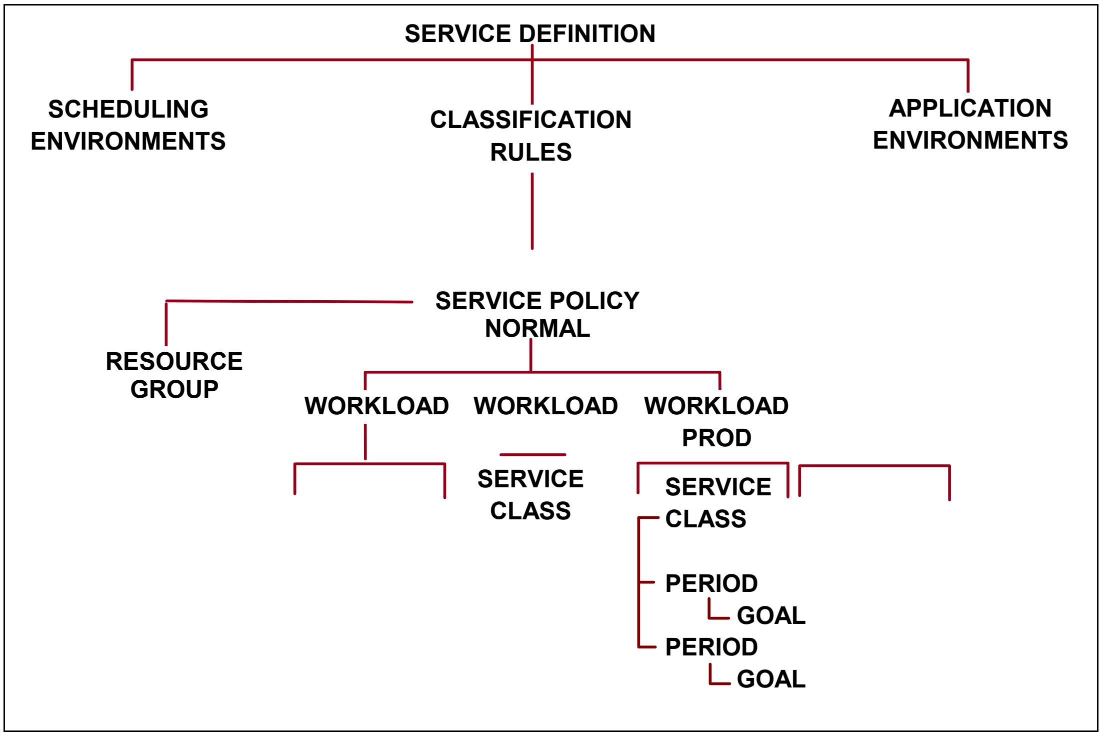


Figure 1-4 Service definition hierarchy

1.2.1 Service definition and service policies

The service definition is defined to WLM through the WLM ISPF application or through the batch interface.

Refer to 4.2.5, “Service policy definition” on page 126, for more information about creating a service definition.

There is only one service definition for the entire sysplex. The definition is given a name, and it is stored in the WLM couple data set accessed by all the z/OS images in the sysplex.

In addition, there is a work data set used for backup and for policy changes.

The service definition consists of one or more *service policies*. There is only one active service policy at a time in the sysplex. A service policy is a named collection of performance goals and processing capacity bounds. It is composed of workloads, which consist of service classes and Resource Groups. Two different service policies share the same set of service classes where the performance goals can be different.

The active policy can be switched by the operator command below to reflect changes in performance objectives. This command is a sysplex-wide command.

```
V WLM,POLICY=policyname[,REFRESH]
```

A workload is a named collection of service classes to be tracked and reported as a unit. (Note that it does not affect the management of work.) You can arrange workloads by subsystem (such as CICS or IMS), by major workload (for example, production, batch, or office), or by line of business (ATM, inventory, or department).

The Resource Measurement Facility (RMF™) Workload Activity report groups performance data by workload and also by service class periods within workloads.

1.2.2 Workload classification

z/OS is able to manage many types of workloads, each with different business importance and processing characteristics. In order to accomplish this, the installation must categorize the incoming work to the system using the c*lassification rules*.

Classification rules are the filters that WLM uses to associate a transaction's external properties, also called *work qualifiers* (such as LU name or user ID), with a goal.

Refer to Figure 1-5 for a view of the classification rules concept.

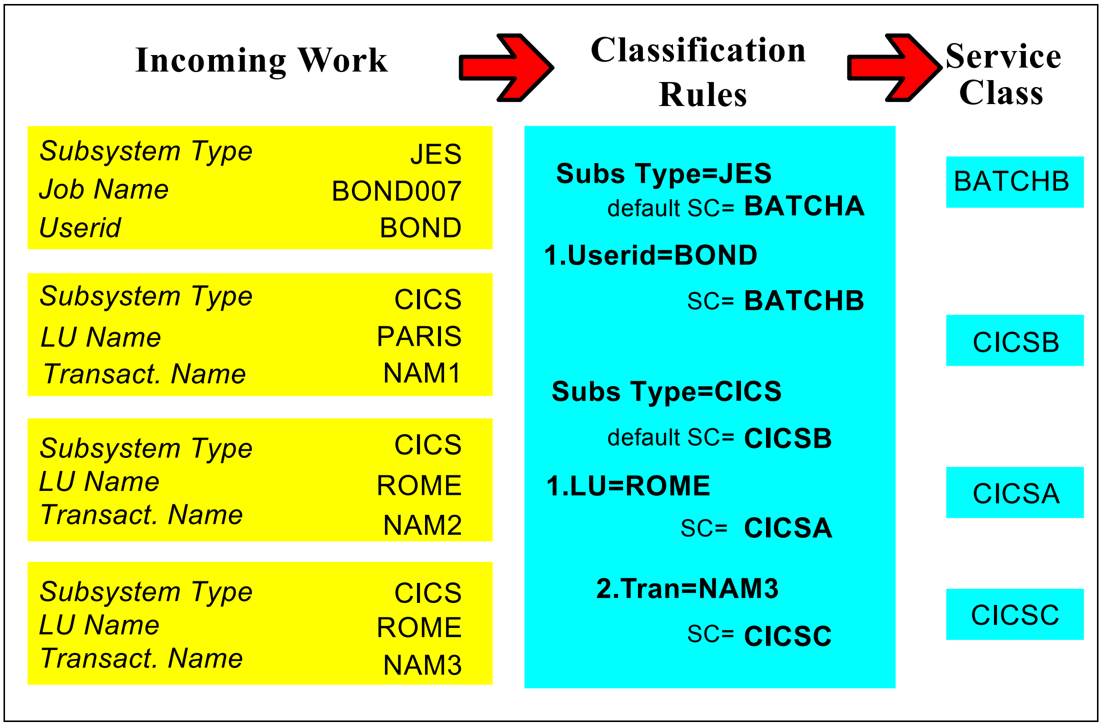


Figure 1-5 Classification rules

Following are some examples of work qualifiers, which are used in classification rules:

- Subsystem type, such as IMS, JES2/JES3, TSO/E, STC, CICS, OMVS, DB2® (parallel queries), Distributed Data Facility (DDF), and others. Work qualifiers are specific to the subsystem originating the transaction.
- LU name/netid
- Transaction name/job name, user ID
- DB2 work qualifiers such as correlation ID, collection name, package name, and others

For a complete list of work qualifiers, refer to 4.2.9, "Classification rules" on page 129.

A group of classification rules consists of a group of nested rules, where the order of nesting determines the hierarchy of the classification rules.

Consider Example 1-1: The PAYR transaction coming from Paris is going to be associated to service class CICSC and report class ABC. A PAYR transaction coming from Rome is going to run in the CICSB service class by default.

Example 1-1 Sample classification for CICS transactions

```
* Subsystem Type CICS - Use Modify to enter YOUR rules

    Default Service Class is CICSB
    There is no default report class.

      Qualifier  Qualifier      Starting       Service  Report
    # type       name           position       Class    Class
    - ---------- ------------- ---------      -------- --------
    1 LU         Paris                         CICSA
    2 . TN       . PAYR                        CICSC    ABC

  Descriptions:
```

For more information about classification rules, refer to 4.2.9, “Classification rules” on page 129.

1.2.3 Service classes

The service class (SC) describes a group of work within a workload with similar performance characteristics. A service class is associated to only *one* workload and it can consist of one or more periods (specified through the DURATION keyword).

There are three system-provided service classes: SYSSTC, SYSTEM, and SYSOTHER:

- SYSTEM: Used as the default service class for certain system address spaces. It does not have a goal, and it is assigned fixed DP=x’FF’ and fixed IOP= 255.
- SYSSTC: The default service class for system tasks and privileged address spaces. It does not have a goal, and it is assigned fixed DP=x’FE’ and IOP=254.
- SYSOTHER: As default service class for non-STC address spaces when no classification rules exists for the subsystem type. It is assigned a discretionary goal.

A service class has the following parameters:

- Importance

  The Importance is the most important definition and distinction you can make in the service class definition. First of all, you need to define what is more and what is less important to your business, and this defines the service classes (at least for work of a similar nature).

- Performance goal

  The goal types are:

  - Response time (average and percentile)

    Average response time is the expected amount of time required to complete the work submitted under the service class, in milliseconds, seconds, minutes, and hours. This only includes the time spent on z/OS, for example, it will not include network time.

    Percentile is the percentage of work in that period that should complete within the response time (for example, 80% of transactions ended in 0.5 of a second).

If you have a bad RT distribution, using a percentile goal is better than using an average goal. For example, nine transactions with a response time of 0.5 seconds and one transaction with a response time of 3 seconds have an average response time of 0.75, but 90% of them have a response time of 0.5 seconds.

Workload management does not delay work, or limit it, to achieve the response time goal when extra processing capacity exists, unless you are explicitly using a capping function.

Work that is appropriate for a response time goal should have at least 10 transaction completions within 20 minutes. If less, it might be difficult for WLM to react correctly, because a single transaction might have a big impact and affect the calculated response time.

> **Note:** You must specify a host response time goal, not end-to-end. That is, workload management does not control all aspects of system performance, so the response time scope is confined to the time workload management has control of the work. This time includes the time that the work is using or waiting for CPU, storage, I/O service, or server address spaces.

– *Velocity* is the measure of how fast work should run when ready, without being delayed for WLM-managed resources. Velocity goals define the acceptable amount of delay for work when work is ready to run. The formula for calculating the execution velocity is:

$$\text{Velocity} = \text{Using samples} / (\text{Using samples} + \text{Delay samples})$$

Where:

- *Using samples* include all types of processors using samples and I/O using samples.
- *Delay samples* include all types of processor delays, I/O delays, storage delays, and queue delays.

To include the I/O figures in both Using samples and Delay samples, you need the option I/O management set to `YES` in the service policy.

Starting with z/OS 1.6, a zSeries system can have regular processors and zSeries application assist processors, and both types of processors are included in processor delays and using delays. Processor delays also include processor capping delays, which come from Resource Group capping. Storage delays consists of a variety of different delay reasons, and queue delays can either be JES queue delays (for batch work) or address space queue delays for applications such as WebSphere or DB stored procedures, which exploit the WLM queuing services.

> **Note:** The execution velocity formula does *not* include the *unknown* state; as mentioned before, this state includes the delay not tracked by WLM, such as locks or enqueues. The response time goal instead does include the elapsed time that corresponds to unknown samples. So, unknown delays affect WLM decisions for the response time goal, but the same delays do not affect velocity goals.

– *Discretionary*: WLM-defined goal. Work is run when system resources are available. The performance index of the discretionary goal is always equal to 0.81 and has a fixed I/O priority of 248. It is the goal of the SYSOTHER service class.

Associate this goal with work for which you do not have a specific performance goal.

A service class can have additional characteristics, such as belonging to a Resource Group and having the CPU Critical or Storage Critical attribute:

- Resource Group (optional)

  A Resource Group defines:

  – Minimum CPU service required (protection)
  – Maximum CPU service limit (capping)

- CPU Critical (optional)

  This can be defined for a service class for long-term CPU protection for critical work. For more details, refer to 3.1.2, “CPU critical” on page 85.

- Storage Critical (optional)

  This can be defined for a service class to assign long-term storage protection to critical work. For more details, refer to 3.1.3, “Storage critical” on page 88.

1.2.4 Importance of a goal

Each service class is associated to an *Importance* level that specifies how important it is to your business that this workload is meeting its goal. The importance defines how work is treated by the system. It indicates which “unhappy” service class period should receive resources in order to achieve the target goal. The Importance level values can be:

- Highest (1)
- High (2)
- Medium (3)
- Low (4)
- Lowest (5)

The absolute value specified is meaningless. What matters is the *relative* value. Importance is ignored when you are meeting your goals.

Discretionary work has implicitly no importance while SYSTEM and SYSSTC work are considered the highest importance works because of their top dispatching priority.

1.2.5 Resource Group

The Resource Group (RG) definition describes the amount of CPU capacity available to transactions associated to a specific service class.

You can use an RG to:

- Limit (capping) the amount of CPU capacity available to the service classes. Capping is used in situations where you want to deny the access of CPU cycles to one or more service classes.
- Guarantee some minimum CPU capacity to the service classes when work in the group is missing its goals.

Refer to Appendix B of *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602, for further details.

If one of the associated service classes has a discretionary goal, WLM achieves the minimum capacity only if this action does not impact other goals.

Refer to 4.2.7, “Resource Group specification” on page 127, for more information about RGs.

1.2.6 Application Environment

Application Environments (AE) are used to automatically handle and control the number of server address spaces for work managers (such as DB2, SOM®, WebSphere Application Server, or MQ) exploiting the Queuing and Routing Manager WLM services.

Application Environments are defined in the WLM service definition and allow WLM to:

- Help meet the goals of a set of transactions by dynamically changing the number of server address spaces.
- Do this without harming other, more important workloads on a system.

The Application Environment can be thought of as a set of server address spaces having system or sysplex scope, belonging to the same subsystem instance, started by the same procedure, and with the same capabilities (in terms of program libraries, accessible data, and security authorization). An AE can be shared across the members of the sysplex.

The transactions in an AE can have different goals and consequently different service classes. Each unique combination of AE and service class defines an *Application Environment queue*. WLM creates queues for the transactions coming from the work manager, and the application servers will fetch work from these queues. WLM samples the queues to measure the time delay in order to decide if a new server address space for the AE needs to be created. WLM creates at least one server address space for each Application Environment and service class combination.

When defining an AE, you can specify:

- Application Environment name: Unique in the sysplex
- Work Manager: Identifies the work manager by type and subsystem name
- STC procedure name: The JCL in SYS1.PROCLIB to start the server address space
- Start parameters: Optional parameters to be passed to the server during address space creation (such as user ID for security checking)
- WLM management options: Special options governing WLM control of address spaces

Example 1-2 shows the Application Environment panel in the WLM application, although this task might not be required any longer since the introduction of the Dynamic Application Environment capability. The Dynamic Application Environment allows the work managers to dynamically deploy an Application Environment without the need of the explicit definition in the WLM policy.

Example 1-2 Application Environment definition panel

```
Appl Environment Name . . DBDMWLM1
Description . . . . . . . Workload generator
Subsystem type  . . . . . DB2
Procedure name  . . . . . DBDMWLM1
Start parameters  . . . . DB2SSN=DB8E,APPLENV=DBDMWLM1



Limit on starting server address spaces for a subsystem instance:
 No limit
```

1.2.7 Scheduling environment

A *scheduling environment* is a list of resource requirements that ensures that transactions are sent to systems that have those resources. This is particularly important in a sysplex installation where the systems are not identical.

The defined resources can be physical entities (such as hardware features or subsystem instance) or intangible entities, such as certain periods of time (working hours, off shift hours, and so on).

The scheduling environment lists the resources and their required state, ON or OFF.

Example 1-3 shows the scheduling environment definition panel.

Example 1-3 Scheduling environment definition panel

```
IWMAPAD              Create a Scheduling Environment          Row 1 to 2 of 2
 Command ===> ____________________________________________________________
 Scheduling Environment Name    DB2LATE________  Required
 Description  . . . . . . . . . Offshift DB2 Processing_________
 Action Codes: A=Add  D=Delete

                         Required
 Action  Resource Name     State      Resource Description
   __    DB2A              ON_______   DB2 Subsystem
   __    PRIMETIME         OFF______   Peak Business Hours
```

Refer to 1.2.7, “Scheduling environment” on page 14 to get more information about this function.

1.2.8 Sampling dispatchable unit states

In order to enforce goals and to track the performance of a sysplex, WLM samples the states of dispatchable units every 250 milliseconds.

Within each sample, a unit of work can be in one of the following states:

- *Using*, when using CPU or DASD I/O
- *Delayed*, when delayed by CPU, storage, or I/O
- *Idle*, when without work (such as TSO or OMVS without transactions, an initiator without a job, APPC wait, or a server in STIMER wait)
- *Unknown,* when delayed by a non-tracked WLM resource (such as ENQ or operator) or idle for a reason other than those listed under the idle state above
- *Quiesced*, when the unit of work is quiesced by the RESET operator command

RMF in the Workload Activity report shows these states as a percentage, and they should add up to 100%. However, in some cases, there might be an overlap between Using states and Delay states, causing the sum to be greater than 100%. For further details about how to interpret the report, refer to *z/OS V1R8.0 Resource Measurement Facility (RMF) Report Analysis,* SC33-7991*.*

The *using* and *delay* states are measured as follows:

- CPU

  WLM adds to the address space or enclave *using CPU* count, which is the number of dispatchable units (multiple state) actively executing on a CPU at every sample.

WLM adds, to the address space or enclave *delay CPU* count, the number of dispatchable units (multiple state) ready but delayed by CPU at each sample.

- Storage (paging and swapping)

  There is no *using* storage count.

  WLM adds, to the address space *delay* storage count, the number of dispatchable units (multiple state) that have a storage delay (such as a page fault).

  However, if the address space is swapped out in the OUTR queue (by an WLM decision) or is in the process of being paged in due to a swap in, the delay storage counter is increased by *one* (single state). This is done at every sample.

- DASD I/O (optional)

  The measurement of I/O *delay* and I/O *using* is optional.

  WLM adds, to the address space or enclave *using I/O* count, the number of dispatchable units that have an I/O operation in the connect state that is moving data through the channel. This is done at every sample.

  WLM adds, to the address space or enclave *delay I/O* count, the number of dispatchable units (multiple state) that have an I/O operation in the IOS queue or in pending state (delayed by channel, control unit, or shared DASD device) in the channel subsystem, that is, the channel program that is delayed. This is done at every sample.

> **Note:** After APAR OW47667, *disconnect time* is no longer counted as productive I/O time. It is also not counted as I/O delay, because there is nothing WLM can do to reduce disconnect time.

1.3 Performance management metrics

Before going deeper into understanding how WLM works, we need to introduce some metrics concepts in order to better understand the performance management process.

In performance management, there are metrics to verify the system behavior toward SLA and performance in general. The following metrics apply to the following resources: CPU processors including ICFs, IFLs, zAAPs, zIIPs, and SAPs, processor storage, and channels. The SLA is a contract that objectively describes:

- Average transaction Response Time (Tr)
- External Throughput Rate (ETR)
- Resource Utilization (U)

In your performance management activity, you might also want to consider the concept of saturation design point (SDP). For details, see “Saturation design point (SDP)” on page 19.

Average transaction response time

A *transaction* is a business unit of work implemented to solve a problem for the business. It is produced by the interaction of an user (online or not) and the system. So, the transaction time has two pieces: Computer response time and the user thinking time.

The computer response time is made of:

- Host response time
- Network time
- Other servers time

Response time for an online transaction is the time between when the user presses Enter and when a panel with the required full data is presented. The time in other servers (your workstation, for example) is due to Web applications, for example, where a servlet or JSP™ returns an HTML page, including many GIFs.

Response time for a batch job transaction is the time between the JOB submit and the final execution of the last job step where the time for sysout printing and purging is not included.

Response time in general is made of service time (the time actual work is done) and waiting time (the time waiting for resource):

$$Tr = Ts + Tw$$

Similarly for computer response time, transaction service time (Ts) and transaction wait time (Tw) are made up of the individual resource service and wait times, as shown in Figure 1-6:

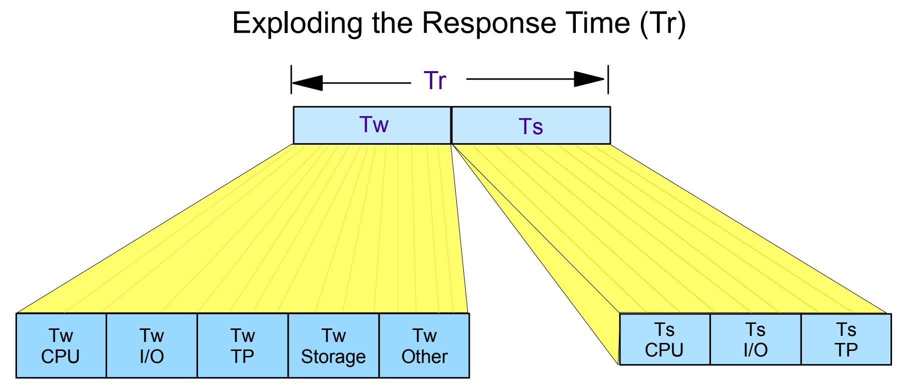
$$Ts = Ts\,(CPU) + Ts\,(I/O) + Ts\,(TP) + Ts\,(Other)$$

$$Tw = Tw\,(CPU) + Tw\,(I/O) + Tw\,(TP) + Tw\,(Storage) + Tw\,(Other)$$

You should understand that a high Tw is not a cause for unacceptable Tr, but a consequence of high activity in the server. Then, we can decrease Tw by decreasing Ts and ETR; refer to “External throughput rate (ETR)” on page 17.

Another way to split response time (Tr) on the host is by subsystems. For example, in a CICS transaction, the response time is composed of:

- TCP/IP time
- CICS processing time
- DB2 access time



Tw (CPU): Time with ready dispatchable units, but not enough dispatching priority.
Tw (I/O): Time with I/Os delayed in UCB or channel subsystem.
Tw (TP): Time in the VTAM or TCP/IP queues.
Tw (Storage): Time suffering a page fault or being swapped-out.
Tw (Other): Time delayed by operator, ENQ, data set recall, server AS service.
Ts (CPU): Time with dispatchable units executing CPU.
Ts (I/O): Time with I/Os being executed (connected or disconnected).
Ts (TP): Time being transported in the network.

Figure 1-6 Average transaction response time

Transaction response time is the best way to characterize the performance of a transaction, because it is directly connected to business needs and it is easily understood by people. The response time measured by z/OS and reported by RMF is the host response time portion only.

If the Tw is zero and the SLA is not accomplished, then you need to decrease the Ts. This can be done either by buying more resources on the processors (CPU, channel, or SAP®) or improving the logic of the application program processing the transaction. There is a simple relationship between time in the queue (Tw) and the average length of the queue, and it is obtained by sampling.

RMF offers different ways of viewing response times:

- The Monitor III and WLM sample the state of the address space and enclaves keeping the result in state counters:
  - Using state is the sampled view of service time (Ts).
  - Delay state is the sampled view of wait time (Tw).
- Workflow and execution velocity are metrics based in the using and delay sampled numbers. These measurements provide a detailed view of what happens in the system, and they can help you to understand system performance. You can see which address spaces, enclaves, or workloads are performing well and which ones are not. Use these measurements to learn the reasons for performance problems occurring in workloads or resources.

Example 1-4 shows a sample of the Monitor III Group Response Time report where you can see several time components of a transaction response time. For example, the OMVS service class transactions have on average a response time of 19.39 sec. This response time is broken into pieces, such as 6.46 secs using CPU, 0.64 secs using DEV (I/O), 0.64 secs being delayed by CPU, and so forth.

Example 1-4 Monitor III Group Response Time report

```
                        RMF V1R5   Group Response Time

Samples: 100     System: AQTS  Date: 12/22/04  Time: 16.18.20  Range: 100   Sec

Class: OMVS         Period: 1    Description:
Primary Response Time Component: Delayed for unmonitored reasons

                                                   TRANS    --- Response Time ----
WFL    Users    Frames   Vector   EXCP   PGIN   Ended    -- Ended TRANS-(Sec) -
 %    TOT  ACT   %ACT     UTIL    Rate   Rate   Rate       WAIT  EXECUT  ACTUAL
 91    16    1      1        0   772.4    0.0   0.110     0.003   19.39   19.39

                           -AVG USG-   -------------Average Delay--------------
                   Total   PROC  DEV   PROC   DEV  STOR  SUBS  OPER   ENQ OTHER
Average Users      2.130   0.71 0.07   0.07  0.00  0.00  0.00  0.00  0.01  1.31
Response Time ACT  19.39   6.46 0.64   0.64  0.00  0.00  0.00  0.00  0.09  11.9

                     ---STOR Delay---  ---OUTR Swap Reason---  ---SUBS Delay---
                     Page  Swap  OUTR    TI    TO    LW    XS   JES   HSM   XCF
Average Users        0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
Response Time ACT    0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
```

External throughput rate (ETR)

ETR is a measurement of the number of ended transactions per elapsed time, as pictured in the following formula:

$$\text{ETR} = \frac{\text{Number of Transactions}}{\text{Elapsed time}}$$

ETR is also called *transaction rate.* ETR is associated with a system's throughput.

In the Workload Activity report, the elapsed time is the RMF interval. In Example 1-5, the interval set was 30 minutes (1800 seconds), where 1879 transactions from service class TSOHIGH ended, causing an ETR of 1.04 (field END/S in the report).

Example 1-5 Workload Activity report (extract)

```
TRANSACTIONS (Service Class TSOHIGH)
   AVG       4.42
   MPL       4.39
   ENDED     1879
   END/S     1.04
   #SWAPS    3154
   EXCTD        0
   AVG
   ENC       0.32
   REM
   ENC       0.12
   ENC       0.01
```

There is also one formula relating the Tr with ETR:

$$\mathrm{ETR} = \mathrm{N}/(\mathrm{Tt} + \mathrm{Tr})$$

Where:

N Is the average number of users generating transactions (logged on)
Tt Is the average thinking time of these users
Tr Is the average computer response time

Some considerations regarding this formula:

- The variables that more intensively affect the ETR are N and Tt due to their usual numeric values. Therefore, SLAs based on ETR are difficult to attain, because the only variable that the IT department can directly manage is Tr.
- Any bottleneck, internal or external to the system, affects the ETR. Examples include I/O constraints, tape mount delay, and paging delay.

Resource utilization

Resource utilization measures how much a resource delivers service in a timely basis. The general formula is:

$$\mathrm{U\%} = \mathrm{Busy\_Time} \times 100/\mathrm{Elapsed\_Time}$$

We can derive (for just one server):

$$\mathrm{Busy\_Time} = \mathrm{Ts} \times \#\ \mathrm{Transactions}$$

Replacing Busy_Time in the first formula:

$$\mathrm{U\%} = \mathrm{Ts} \times \#\ \mathrm{Transactions} \times 100/\mathrm{Elapsed\_Time}$$

$$\mathrm{U\%} = \mathrm{Ts} \times \mathrm{ETR} \times 100$$

This formula shows that the U increases when Ts, ETR, or both elements rise.

For example, if in a bank branch office with only one teller, there is an average arrival rate of two clients per minute with a service time (Ts) of 0.25 minute, we can say that the teller average utilization is U% = 0.25 x 2 x 100 = 50%.

- The product: Ts x ETR is also called *traffic*.

- The product: Tr x ETR is called *intensity*.

Saturation design point (SDP)

CPU is either in busy state or in wait state. In other words, either it is 100% busy or 0% busy. Figure 1-7 on page 19 shows this fact for a duration of a few minutes. The shaded area indicates that the CPU is busy. Performance monitors group these states in a specific amount of time, for example, a one-minute interval. In Figure 1-7 on page 19, the left portion of the graphic shows a minute time scale. In the first minute, the average utilization is 40% and in the second minute it is 65%.

Usually, these values are grouped in intervals of hours. An hourly average is calculated and presented to human analysis (50%, 80%, and 90% in the right part of the graphic). As you decrease the granularity (from one minute to one hour), you might lose key details. The major question is, “If my hourly utilization is 75%, how often did I reach 100% in the *per minute* average within this hour?”

To answer this question, the concept of saturation design point (SDP) was developed. SDP is the maximum average utilization in a larger interval (one hour, for example), where the installation is sure that in the smaller interval (one minute, for example) the average utilization was never 100%. If the current average is greater than the SDP, then the 100% busy was reached in the smaller interval. Following is the formula for the hour/minute relationship:

$$\text{SDP\%} = \text{Average Hourly Utilization} \times 100 / \text{Peak Minute Utilization}$$

Imagine that, in some hour interval, the average hourly utilization was 80%, and, in that hour interval, the highest minute average peak was 92%. Then:

```
SDP% = 80 x 100 / 92 = 87%
```

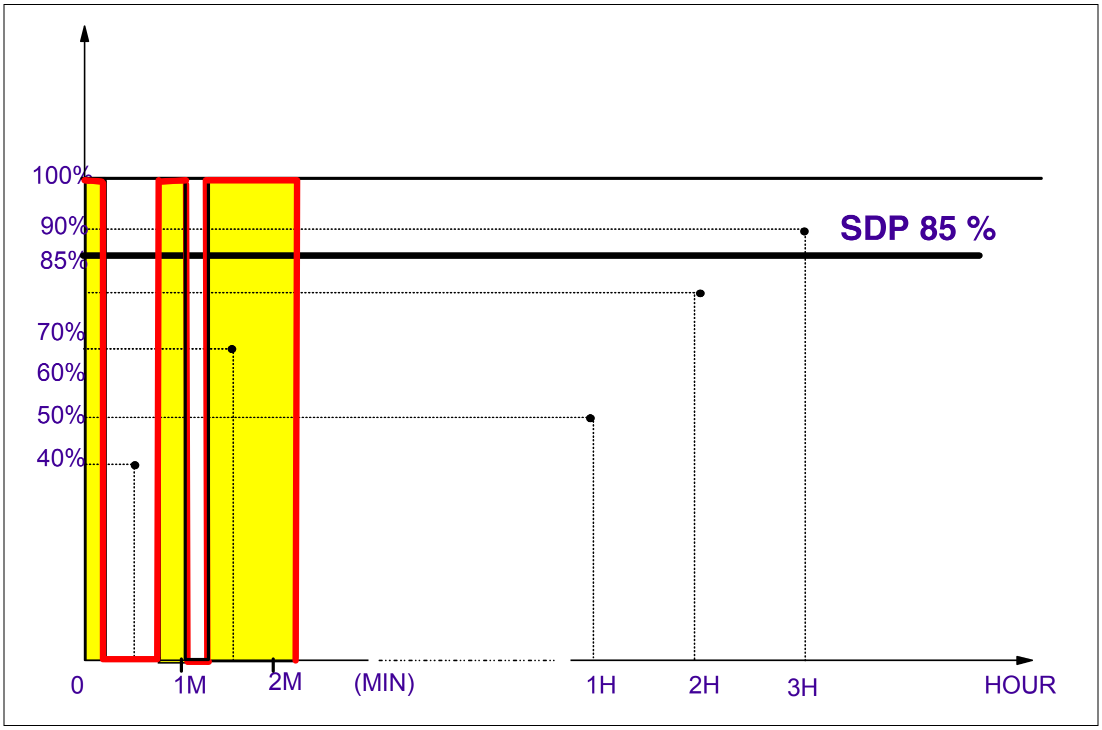


Figure 1-7 Utilization curve with SDP

Observing the formula, it is clear that when SDP is equal to the average hourly utilization, then at least in one minute, the 100% minute average was reached.

In Figure 1-7, the SDP is worth 85%; then, when hour utilization is 90% (as in the third hour), there is the guarantee that for some minute average within that hour the 100% mark was reached.

If a client knows its SDP using theoretical (usually 70%) or experimental approaches, the client can use this information in:

- Performance management for avoiding the case where the hour average exceeds the SDP
- Capacity planning for sizing the processors to not exceed the SDP limit

1.3.1 Processor performance metrics

Below are metrics used in performance management and capacity planning to evaluate the processor speed capacity:

- CPU time
- CPU busy %
- Millions of instructions per second (MIPS)
- Millions of floating point instructions per second (MFLOPS)
- CP service units, System Resource Manager (SRM) constant, millions of service units per hour (MSU)
- External Throughput Rate (ETR)
- Internal Throughput Rate (ITR)
- Large Systems Performance Reference (LSPR) - RPP

CPU time

CPU time of a transaction is the processing time consumed to execute the program logic of the transaction. In z/OS, the program is associated with a dispatchable unit, a task (TCB) or a service request (SRB) at an address space or enclave basis. The transaction CPU time is measured by z/OS and is called *captured time*.

The RMF Workload Activity report shown in Example 1-6 shows several components of the CPU time.

Example 1-6 RMF Workload report extract

```
---SERVICE----   --SERVICE RATES--
 IOC     56040    ABSRPTN       598
 CPU      1680K   TRX SERV      594
 MSO      2938K   TCB         271.3
 SRB     56695    SRB           9.2
 TOT      4731K   RCT           4.4
 /SEC     2626    IIT           5.4
                  HST           0.0
                  APPL %       16.1
```

One of the installation's targets in performance management is to reduce such an amount. The CPU time value for the execution as a transaction can also be theoretically derived through the formula:

$$\text{CPU time} = \text{Cycle\_Time} \times \text{CPAI} \times \text{Path\_Length}$$

Let us now discuss the elements of this formula.

Cycle time

Cycle time can be defined in one of the following ways:

- Time to execute the fastest instruction of the CPU instruction set. All the instructions are executed in a time multiple of the cycle time.
- Time elapsed between two consecutive pulses generated by the internal clock in the processor. Numerically, it is the inverse of the frequency that is measured in MHz.

Path length depends on the instruction set and the compiler.

Cycles/instruction (CPAI) depends on the design (for example, microcoded instructions or pipeline).

Cycle time depends on the technology (for example, CPU model).

Thus, cycle time by itself is not a complete indicator of CPU speed.

Cycle per average instruction (CPAI)

CPAI is the number of cycles, on average, needed to execute one instruction. CPAI depends very much on the set of executed instructions. These are very simple instructions executed in just one cycle. The z990 can even execute more than one instruction in one cycle; this property is called *superscalar*. Alternatively, there are complex instructions that need many cycles in order to execute, biasing consequently the CPAI to a higher value.

The multiplication of CPAI x Cycle_Time produces the average time to execute one instruction, and its inversion produces the MIPS metric.

Path length

Path length is the number of instructions needed to execute the transaction program. Obviously, if you have a shorter path length, this produces shorter CPU time. Path length depends on:

- Processor architecture, which defines the set of instructions
- Quality of the compiler that generates the object code

The goal of an architecture is to offer powerful instructions in order to implement a program with a short path length.

Millions of instructions per second (MIPS)

This term depends on cycle time and the number of cycles per instructions. It is only valid for comparison if the instruction sets are the same. In common usage today, MIPS are used as a single number to reflect relative processor capacity. As with any single-number capacity indicator, your experience can vary considerably. These numbers are often based on vendor-provided announcements.

Using MIPS as a comparison between two processors is fair only when the two processors execute exactly the same set of instructions executing the same logic associated with the same transaction program.

We do not recommend using this method; you should instead use the LSPR method described later.

MFLOPS

MFLOPS mean *millions of floating point instructions per second*. This value is used only in numerically intensive computing (scientific and technical). There is one important benchmark available in the industry that measures MFLOPS performance, the LINPACK benchmark.

Here, many computer companies are in a tough battle to have the worldwide fastest processor. You can find details in:

http://www.linpack.com

CPU Service Units: System Resource Manager (SRM) constant

CPU Service Units is a metric used by z/OS to measure the CPU consumption by transactions executing under z/OS dispatchable units, such as tasks (TCB) or service requests (SRB).

A CPU Service Unit should be more or less *repetitive* and only include the *productive* CPU execution. Below is an explanation of these terms:

- Productive: The consumption of CPU due to overheads such as page fault, swapping, and dispatcher is not accounted for in the transaction CPU Service Unit.
- Repetitive: The same value is roughly measured for the same transaction (executing the same logic with the same amount of I/O records) in any CPU.

The System Resource Manager constant is a number derived by product engineering to normalize the speed of a CPU, so that a given amount of work would report the same service unit consumption on different processors.

The relationship between CPU SUs and SRM constant is illustrated in the following formula:

$$\text{Service units} = \text{CPU seconds} \times \text{SRM constant}$$

Service Units (SUs) are typically used by the WLM as a base for resource management.

Tip: The SUs delivered in an LPAR configuration are dependent on the number of physical shared processors on the CPC and not on the number of online processors at LPAR IPL time.

With IRD, the SRM constant used for an image is equal to the INITIAL CP definition for the LPAR even if the number of logical CPs is changing because of the IRD effect. For example, if there are nine initial CPs and IRD leaves only four CPs, the SRM constant used will correspond to a nine-way machine.

The SRM constants are contained in internal tables within z/OS and are published in the *z/OS MVS Initialization and Tuning Guide,* SA22-7591. The SUs are reported in many Monitor I and Monitor III reports, or you can find them at:

http://www.ibm.com/servers/eservers/zseries/srm/

Many installations that have to charge the users will do so on the basis of SUs. This is a reasonable approximation of relative capacity. But, of course, any single-number metric for processor capacity comparison can be misleading; therefore, Large Systems Performance Reference (LSPR) numbers, based on your own workload mix, are the best basis to use.

Internal throughput rate (ITR)

ITR determines the capacity of a processor in terms of the number of transactions per CPU second.

$$\text{ITR} = \text{Number of Transactions} / \text{CPU Time}$$

ITR is a function of:

- CP speed. For example, faster the CP, higher the ITR.
- Operating system. So, for example, the better the operating system, the higher the ITR.

- Transaction workload characteristics. For example, short trivial transactions produce higher ITR.

If we make constant the operating system (z/OS, for example) and the transaction workload variable, we can use ITR to measure the CPU speed in terms of transactions. Comparatively, ITR is more suitable for such comparisons than ETR, because ETR depends on a much larger amount of variables. Refer to “External throughput rate (ETR)” on page 17.

IBM calculates ITRs for several workloads and machines using the Large Systems Performance Reference (LSPR) methodology. ITR provides a reliable basis for measuring capacity and performance. The relationship between ETR and ITR is:

$$\text{ETR} = \text{ITR} \times \text{CPU Utilization}$$

Then, when the CP utilization is 100%, the ETR is equal to ITR.

Using ITRs, you can define an ITR ratio (ITRR) between two processors:

$$\text{ITRR} = \frac{\text{ITR(Proc2)}}{\text{ITR(Proc1)}}$$

Large Systems Performance Reference (LSPR)

Although CPU time consumed by one transaction is the best metric to measure the speed of a CPU, the formula seen in “CPU time” on page 20 is not practical because path length and CPAI can hardly be measured in your workload.

To address that problem, IBM implemented a benchmark method to estimate the CPU capacity needed for your workload. A benchmark is a well-defined set of measurements for specific applications. A benchmark can be online transaction processing, batch jobs, WebSphere applications, and others. Due to the fact that benchmarks are based on applications (rather than just using a MIPS value for a processor, for example), it is easier for installations to understand benchmark results, and then project these results onto their own environments.

The IBM benchmark project is called *Large Systems Performance Reference (LSPR)* and measures ITRs of different machines with different workload types and different operating systems (z/OS, z/VM®, VSE, and ACP/TPF). The idea behind LSPR is that in most cases it is not useful to characterize a processor by only one number (or a set of numbers for different workloads), but instead by comparing two processors. Therefore, if you have installed one specific processor and you want to upgrade to another one, you can use the LSPR numbers to see how much more capacity you can get. LSPR tables always show ITRR values; one selected processor is defined as base with the value ITRR=1.0, and the values for all other processors show the relative performance compared to the base processor. All the capacity numbers are relative to the zSeries 2084-301. Table 1-1 shows the ITRR value for some z990 processors.

Table 1-1 LSPR table z990 referred to z/OS Release 1.6

Processor	#CP	Mixed	CB-L	CB-S	WASDB	OLTP-W	OLTP-T
2084-301	1	1.00	1.00	1.00	1.00	1.00	1.00
2084-302	2	1.90	1.95	1.80	1.91	1.93	1.93
2084-303	3	2.78	2.88	2.58	2.80	2.82	2.82
2084-304	4	3.63	3.80	3.32	3.68	3.69	3.68
...	...	...	...	...	...	...	...
2084-315	15	12.32	12.70	9.36	12.45	11.22	10.72

Processor	#CP	Mixed	CB-L	CB-S	WASDB	OLTP-W	OLTP-T
2084-316	16	12.98	13.40	9.70	13.16	11.72	11.26
Ratios for 17-way and above are based on running multiple OS images.							
2084-317	17	13.62	14.08	10.05	13.85	12.24	11.79
2084-318	18	14.24	14.73	10.38	14.52	12.74	12.31
...	...	...	...	...	...	...	...
2084-323	23	17.05	17.71	11.65	17.61	14.94	14.82
2084-324	24	17.56	18.24	11.83	18.17	15.32	15.30

These ratios represent the IBM assessment of relative processor capacity in an unconstrained environment for the specific benchmark workloads and system control programs specified in the tables. Ratios are based on measurements and analysis. The data shown in the table is based solely on IBM measurements and analysis of the processors in the tables.

Each individual LSPR workload is designed to focus on a major type of activity, such as interactive, online database, or batch. The LSPR does not focus on individual pieces of work such as a specific job or application. Instead, each LSPR workload includes a broad mix of activity related to that workload type. Focusing on a broad mix can help assure that resulting capacity comparisons are not skewed. Some of the workloads are:

- OLTP-T: Traditional Online Workload
- OLTP-W: Web-enabled Online Workload
- WASDB: WebSphere Application Server and Database
- CB-L: Commercial Batch Long Job Steps
- CB-S: Commercial Batch Short Job Steps

Relative Processor Power (RPP) is the average ITR ratio of the workloads described by the Mixed column in Table 1-1 on page 23. The mixed workload consists of an equal mix of OLTP-T, OLTP-W, WASDB, CB-S, and CB-L.

RPP is normalized to a base machine. So, if a given transaction takes a certain amount of RPPs, you can estimate how much it would consume on a different machine by using the formula:

$$\text{Utilization (CPU new)} = \frac{\text{RPP(CPU old)}}{\text{RPP(CPU new)}} \times \text{Utilization (CPU old)}$$

In summary, various numbers can commonly be used as rough indicators of CPU capacity. Remember that most of these are very rough approximations only, and your actual capacity can vary significantly. To get the most accurate assessment of CPU capacity for your workload, differences in workload processing characteristics must be taken into account using a methodology such as that described in the Large Systems Performance Reference. You can find the most current version at:

http://www.ibm.com/servers/eserver/zseries/lspr/zSeries.html

In addition, we recommend that you do not use the LSPR ratios directly, but rather to build workload mixes based on the work that your installation is running. There is a flash with the methodology about how to define a workload mix when building capacity sizing plans, and it gives information about when it is appropriate for its usage. You can find the document at:

http://www.ibm.com/servers/eserver/zseries/lspr/lsprmixwork.html

1.3.2 I/O performance and metrics

As we did with CPU, now we visit some performance management considerations and metrics associated specifically with an I/O operation.

As CPU speed increases, the I/O response time (I/O Tr) become the determinant factor in the average transaction response time. The I/O response time is shown in the formula:

$$\text{I/O Tr} = \text{I/O Ts} + \text{I/O Tw}$$

The average I/O response time is one of the fields shown in Example 1-7, the Shared Device Activity report.

Example 1-7 Shared Device Activity report

```
SMF            DEVICE    AVG  AVG    AVG  AVG  AVG  AVG  AVG    %      %     %     AVG
 DEV  DEVICE   VOLUME PAV  SYS  IODF LCU  ACTIVITY  RESP IOSQ   CMR  DB   PEND DISC CONN   DEV    DEV   DEV   NUMBER
 NUM  TYPE     SERIAL      ID   SUFF      RATE      TIME TIME   DLY  DLY  TIME TIME TIME   CONN   UTIL  RESV  ALLOC


 8004 33903    PAVTS1      *ALL          4043.964   1.0  0.0   0.2  0.0   0.4  0.0  0.7  33.03  33.03   0.0   8.9
                        8  SYS1  29  00A9 1139.045   1.0  0.0   0.2  0.0   0.4  0.0  0.7   9.33   9.33   0.0   3.0
                        8  SYS2  29  00A9 1466.329   1.0  0.0   0.2  0.0   0.4  0.0  0.7  11.96  11.96   0.0   3.0
                        8  SYS3  29  00A9 1438.591   1.0  0.0   0.2  0.0   0.4  0.0  0.7  11.74  11.74   0.0   2.9
```

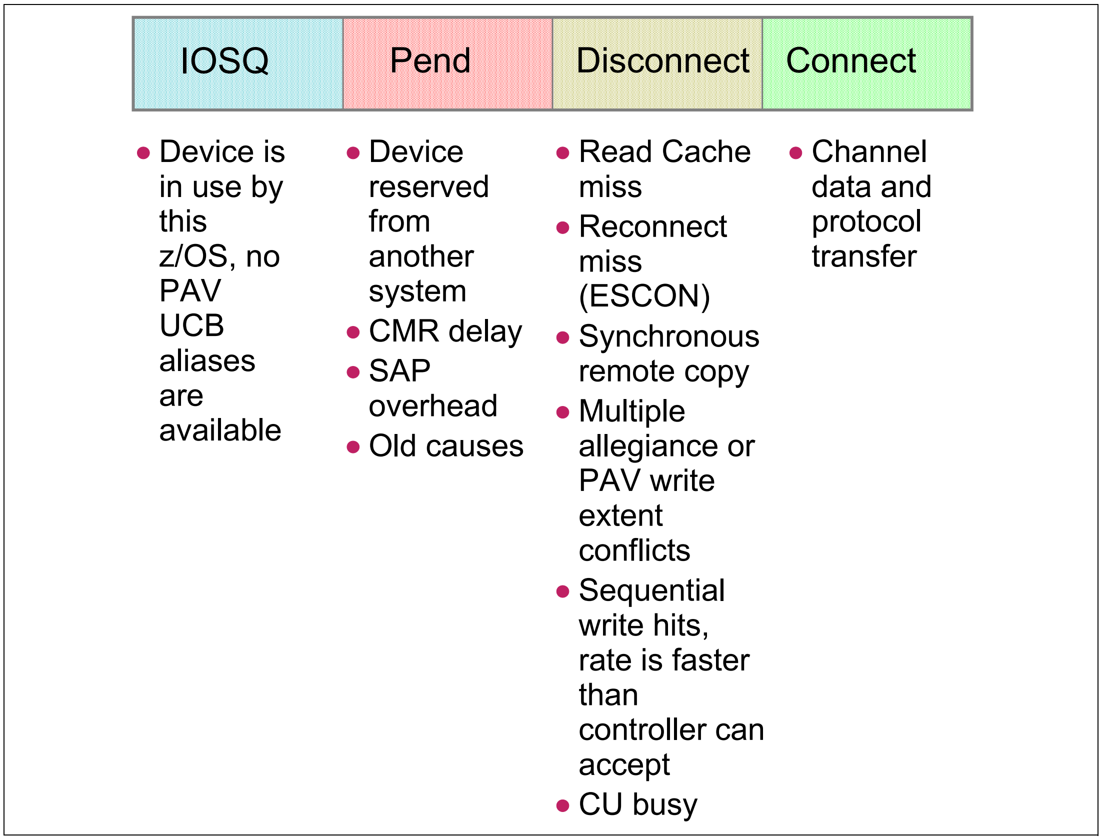
So, obviously, you can get excellent transaction response time by reducing I/O wait time (Tw) and I/O service time (Ts). Turning into I/O performance metrics, the I/O Tr is formed by four components:

$$\text{I/O Tr} = \text{IOSQ Time} + \text{Pending Time} + \text{Connect Time} + \text{Disconnect Time}$$

Where:

- I/O Tw = IOSQ time + pending time
- I/O Ts = Connect time + disconnect time

Figure 1-8 depicts the current four stages of an I/O operation followed by an explanation of each of the fields.



IOSQ	Pend	Disconnect	Connect
• Device is in use by this z/OS, no PAV UCB aliases are available	• Device reserved from another system • CMR delay • SAP overhead • Old causes	• Read Cache miss • Reconnect miss (ESCON) • Synchronous remote copy • Multiple allegiance or PAV write extent conflicts • Sequential write hits, rate is faster than controller can accept • CU busy	• Channel data and protocol transfer

Figure 1-8 The four stages of an I/O operation

The stages and fields are:

- IOSQ: Time waiting for the UCB available in the z/OS operating system. It is drastically reduced by the implementation of dynamic PAV.
- Pending: Time from the SSCH instruction (issued by z/OS) until the start of the dialog between the channel and the I/O controller. The pending time is a product of the following components: All channel path busy time, Director Port Switch (DPS) busy time, control unit (CU) busy time, Device (DB) busy time, and SAP overhead time. DPS, CU, and DB time were presented individually in the RMF DASD Activity report and in the IOQUEUE Activity report. The other two values were derived by subtraction from the total pending time. Due to FICON® and modern controllers, some of these time components disappeared and some were moved to disconnect time; as such, FICON channel is almost always available, FICON switches do not present DPS busy, modern controllers hide the CU busy situation, and Shark multiple allegiance avoids shared device busy. Then, with today's technology, the pending time is formed by:
  - Command Reply (CMR) time. It measures how long the channel waits for the controller at the start of the I/O dialog (command response delay). It is shown in RMF Device activity and IOQUEUE reports. It is an indication of how busy the controller is. This applies only to FICON technology.
  - Device busy time. It indicates how long the device is busy because a hardware reserve has been issued by another system. The other cases of potential device busy, together with CU busy and DPS busy, were moved to disconnect time. Device busy time is shown in the RMF Device Activity report, and it can be an explanation for Missing Interrupt Handling messages, together with bad performance.

  - SAP overhead. It indicates the time that SAP needs to handle the I/O request. This should be around 0.1 to 0.3 milliseconds depending on the SAP speed, which depends on the CPC model. SAP utilization is now reported on an IOQ Activity report.
  - Old reasons such as: The existence of ESCON® directors causing DPS busy. These times are not individually portrayed in the DASD activity report, but you still can see them in the IOQUEUE Activity report on a logical control unit basis.
- Disconnect: Time that the I/O operation has already started, but the channel and I/O controller are not in a dialog. In a FICON protocol, there is no disconnect time concept; however, a numerically equivalent figure is accounted by the hardware component Channel Measurement Facility. The reasons for disconnect time are:
  - Read Cache miss: The I/O block is not in the controller cache.
  - Reconnect miss: The time to dynamically reconnect an ESCON channel.
  - Synchronous remote copy (Peer-to-Peer Remote Copy (PPRC)): The mirroring is done within the disconnect time.
  - Multiple allegiance or parallel access volume (PAV) write extent conflicts: It was reported as PEND (pending timed) within the device busy time.
  - A sequential write hits faster than the controller can accept in the cache.
  - CU busy: It was previously reported as CU busy in pending time; modern controllers move into disconnect time.
- Connect: Time when the channel is transferring data from or to the controller cache or exchanging control information with the controller about one I/O operation. That is, both are in a dialog. Then, connect time includes the real productive time of an I/O operation, that is, the data transfer. However, there is still the overhead of the dialog protocol. For example, in sequential processing, if you decrease the number of SSCHs (by better buffering), even though transferring the same amount of data, your connect time decreases consistently because there is less overhead.

There are other I/O metrics, such as:

- I/O Intensity = I/O Tr x I/O Rate

  Measures the activity in the device in ms/sec, including the queue time.
- I/O Traffic = I/O Ts x I/O Rate

  Measures the activity in the device in ms/sec, excluding the queue time. In one device, it is numerically equal to the utilization of such device.
- I/O Density = I/O Rate per DASD space capacity

  Measured in I/Os per sec per GB. Depends on type of access, size of your files, and how many gigabytes per device (if not a PAV device).

2

How WLM works

This chapter provides insight into the workload management structures in a z/OS environment. The purpose of this chapter is to briefly review the SRM algorithms and focus on the WLM decision-making process, on the transaction management, and on functions that comprise the workload performance management tasks.

SRM is still a live component in the z/OS operating system and cooperates with WLM to manage the performance of the workloads. In today’s environment, you no longer need to customize the SRM component, because all the SRM functions are driven through WLM. WLM manages work execution by dynamically managing system resources to achieve the goals associated with the work requests.

The WLM component can be viewed as an enabler providing services that allow subsystem work managers to supply response time information. WLM is also responsible for exchanging the performance data between systems in a sysplex and enables sysplex-wide performance management.

For simplicity, in this book we do not distinguish between WLM and SRM functions because they are tightly integrated; we only use WLM.

2.1 Workload management

Workload management can be divided into several functions:

- A z/OS component, WLM, formed by the following routines:
  - Service administration application
  - Administration policy management
  - System and sysplex operation

All these routines are involved with WLM policy management:

- WLM services constituted by the following set of routines:
  - Services for the subsystem work managers
  - Report manager
- A WLM component formed by two major functions:
  - Policy adjustment
  - Resource adjustment
- Subsystem work managers supporting the WLM services

Throughout this book, the expression *subsystem work managers* means subsystems such as CICS, IMS, IWEB, and others, exploiting WLM services.

2.2 z/OS dispatchable units (DUs)

Before going deeper into the WLM algorithms discussion, it might be useful to briefly revisit the concept of a dispatchable unit in z/OS.

A z/OS dispatchable unit (DU), as seen in the literature, is a *serial execution of computer logic code to achieve a business need*. WLM transaction management introduces a concept called Business Unit of Work and a new idea of dispatchable units, which is fully described in “The idea of dispatchable units” on page 32 and in 2.4, “Transactions and management” on page 61. In this section, we describe the WLM management of its dispatchable units.

- There are four types of DUs in z/OS:
  - Preemptible Task (TCB)
  - Nonpreemptible Service Request (SRB)
  - Preemptible Client Service Request (client SRB)
  - Preemptible Enclave Service Request (enclave SRB)
- Each Dispatchable Unit (DU) is associated with:
  - State
  - Resources
  - Priorities
  - Account
  - Addressing
  - Security

A program and a DU are different entities. In z/OS, you have control blocks describing the DU (TCB and SRB) with a description of their state and held resources. A *program* is a sequence of instructions that are executed on behalf of a DU on a z/OS Central Processor (CP).

Note: With z890, z990, and z9™-109 processor types, there are new processing units called zSeries Application Assist Processor (zAAP) and z9 Integrated Information Processor (zIIP) available. These processing units are used to offload certain types of DUs from the central processor, which is often referred to as the standard or general purpose processor. The zAAP is available for executing Java™ instructions under the control of JVM™ starting with z/OS V1R6 and with the z890, z990, and z9-109 processor types. DU running JVM (from AS/enclave) can be executed on these processor types. The JzIIP is available for the execution of specific DB2 DUs starting with DB2 Version 8 and z9-109 processor types.

Preemption

A preemptive operating system follows strict CPU dispatching priority (DP). When a dispatchable unit (DU) is active and another DU with a higher DP becomes ready, the active DU is instantaneously suspended (its state is saved) and the new ready becomes active (its state is restored).

z/OS is a reduced preemptive dispatcher. There are situations in z/OS where preemption does not occur:

- Code running hardware disable for interrupts in the program status word (PSW).
- Nonpreemptible dispatchable units (such as traditional SRBs). Once dispatched, they continue to run until they incur a voluntary suspension (such as lock held or page fault) or they complete. External and I/O interrupts are serviced (usually SRBs are interrupt-enabled), but the SRB is redispatched after each interrupt is processed.

Resources needed for DUs

There are three basic resources of a system that are required in order to execute dispatchable units (TCBs and SRBs). These resources are the processor, the processor storage, and I/O. WLM manages these resources and the transactions using them by measuring the various delays that dispatchable units are experiencing.

WLM Business Unit of Work

WLM uses the concept of Business Unit of Work (BUoW) to identify a transaction or a unit of work.

A Business Unit of Work can be:

- An address space. The life of the transaction is the life of the address space.
- An action within an address space, for example, a TSO user issuing a command. This requires interfaces that allow the address space to tell WLM when a transaction starts and when it stops. Note that this can be at the CLIST/REXX command level when CNTCLIST=YES is specified (as we generally recommend) in IEAOPT*xx*.
- An activity that spans multiple address spaces generally through Cross Memory. It is typically the case for transaction monitors, such as CICS, IMS, WebSphere, DB2 Stored Procedures, and so forth.

Let us have a look at the transaction flow shown in Figure 2-1 on page 32.

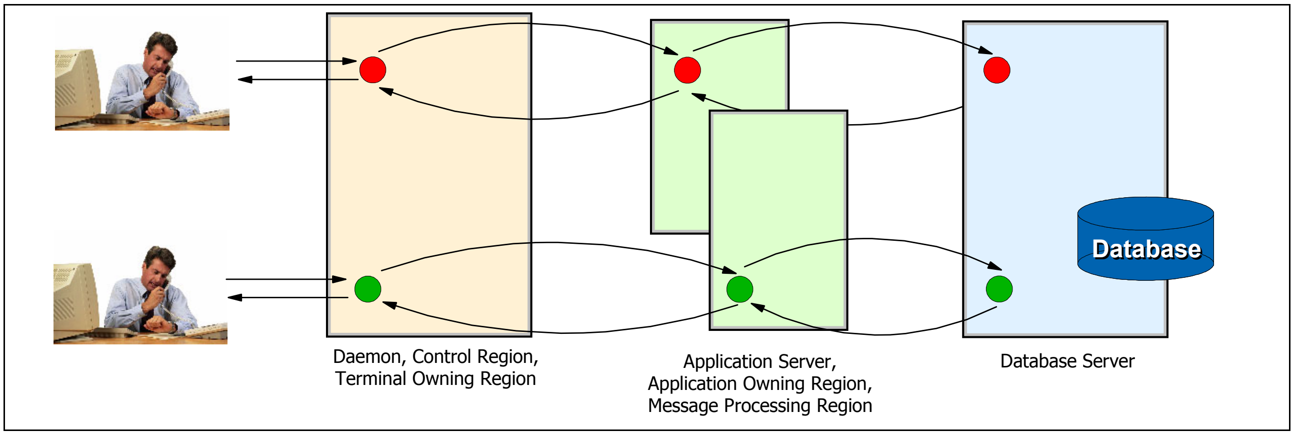


Figure 2-1 Transaction flow

Figure 2-1 depicts the general structure of transaction processing on z/OS. Depending on the middleware or subsystem, one or multiple address spaces provide a connection to external clients.

For CICS, this is typically a Terminal Owning Region (TOR) and for WebSphere, for example, a Control Region. External clients can be everything; for example, a user sitting at a terminal or a program on a different server. The address space that receives the work request on z/OS primarily checks its authority and resource requirements, and if the application can process it, the region sends it to an application processing region. For CICS, these are named Application Owning Regions (AOR) and for IMS, Message Processing Regions. The application region starts a program that processes the request, and the program invokes other middleware, operating system services, or services from other subsystems to complete the request. One typical example is that the request accesses a database, in many cases, DB2 on z/OS.

The idea of dispatchable units

In a modern multiple address space application scenario such as client/server, we might have:

- Multiple tasks in one address space executing requests from other tasks from other address spaces
- A task in one address space executing programs (in cross-memory mode) in other address spaces
- A client preemptible SRB in a loop in a server address space utilizing an entire CPU
- A transaction requiring the serial execution of multiple tasks in several address spaces
- A global SRB with a priority above any task
- An SRB scheduled from one address space and running in other address spaces

Because the business unit of work or user transaction might span and mix different MVS dispatchable units in terms of TCB or SRB, you need to consider from an accounting and priority point of view:

- The TCB in the server address space, even working on behalf of a client TCB, would be charged for its resource consumption.
- The client requests are managed according to the server address space TCB priority and not to the TCB client goals as inherited from the user transaction. So, the server can

execute tasks on behalf of clients who have different goals, but all tasks execute with the goal of the server address space.

The dispatchable units in modern multi-address space applications

In order to address such problems, new dispatchable units have been defined based on the following requirements:

- Preemptible processes to avoid someone dominating CPU resources
- Easy parallelism of processes in one query
- Accounting (by transaction and globally) in the client
- Priority of the client
- Low overhead (less costly than ATTACH)
- Retaining the capability of defining service class or performance group periods
- Retaining the concept of Resource Group, if in goal mode

These dispatchable units are built on preemptible class SRB and a construct called *enclave* is introduced. Preemptible-class SRBs combine the characteristics of SRBs and tasks and include the following types:

- Client SRBs
- Enclave SRBs

All preemptible-class SRBs share certain attributes, for instance, the property of being preemptible. Preemptible dispatchable units, such as tasks and preemptible-class SRBs, can be suspended by the dispatcher at any time to run other dispatchable units at the same or a higher priority.

Any dispatchable unit that is CPU bound is expected to be preemptible in order for the dispatcher to be able to run other units of work.

Therefore, the preemptibility of the new SRB types, together with the lower overhead of creating an SRB instead of a TCB, makes the preemptible SRBs a viable replacement for tasks.

A *client SRB* is a preemptible dispatchable unit that runs a program in one address space (server) but executes work on behalf of some other address space (client). In other words, the priority and account are the ones from the client address space. All dispatching controls are derived from the client address space, including the dispatching priority. The CPU time consumed by a client SRB is accumulated back to the client address space and is included as a CPU service in the client address space's current WLM transaction. When the client address space switches to a new performance period, any client SRBs running on behalf of the client address space also switch to a new performance period. Service accumulated by client SRBs while a client address space is swapped out is not lost; rather, it is accumulated to the client address space when the client is swapped in again. As a consequence, the service accumulated by client SRBs while the client address space was swapped out might trigger a period switch at swap in, but it does not do so *before* swap in.

You create client SRBs by using an option on the macro IEAMSCHD. This macro schedules an SRB routine for asynchronous execution. The SRB can be scheduled for execution in any address space.

Enclave SRBs are preemptible dispatchable units, where the unit of priority and account is an enclave and not an address space (client or server). However, due to the lack of addressability in the enclave, the enclave SRBs must still run in an address space.

Enclave SRBs are also created by the macro IEAMSCHD. This macro creates the SRB and joins it into the enclave. Refer to "Enclave" on page 34 for more information about enclaves.

Enclave

Enclaves introduced a much more direct way of managing transactions that span multiple regions. An *enclave* is an entity that encapsulates the execution units (TCBs and SRBs) that execute programs on behalf of the same work request.

You can also think of an enclave as a BUoW without address space boundaries. It is close to the user view of a transaction.

The types of enclaves

The types of enclaves are:

- *Independent enclaves*: These are true WLM transactions separately classified and managed in a service class. Distributed Data Facility (DDF) uses only independent enclaves. They represent a new transaction that has no association with a particular address space from a priority and account point of view.
- *Dependent enclaves*: These are a logical extension of an existing address space and inherit service class from the owner address space. They continue an existing transaction that is running under dispatchable units not associated with the current home address space.

When an enclave spans a system boundary in a sysplex, it is called a *multisystem enclave*. Multisystem enclaves are made with the original enclave and *foreign enclaves*, which we describe in the next section.

Enclave characteristics

The enclave characteristics are:

- Enclaves are created by WLM on behalf of an address space owner, and one address space can own many enclaves.
- Enclaves maintain the priority and account information of the associated dispatchable unit.
- One enclave can include many dispatchable units (SRBs and Tasks) executing concurrently in multiple address spaces called *participants;* although, it is much more common to have only a dispatchable unit per enclave. Enclave SRBs are preemptible (like tasks) by higher priority work. All its dispatchable units are managed as a group.
- WLM manages enclave dispatching and I/O priority.
- Enclaves do not own storage.
- CPU accounting in SMF records:
  - For a dependent enclave, the CPU service consumed is accumulated in the SMF30 record of the owning address space and the SMF72 record of the owning address space service class period.
  - For an independent enclave (representing an individual WLM transaction), transaction count and resource usage are recorded in the SMF72 record (service class and report class). SMF30 records are accounted to the address space where the enclave runs.
- Multisystem enclaves is the sysplex scope of an enclave. This implementation requires a coupling facility structure named SYSZWLM_WORKUNIT, as shown in Figure 2-2 on page 35.

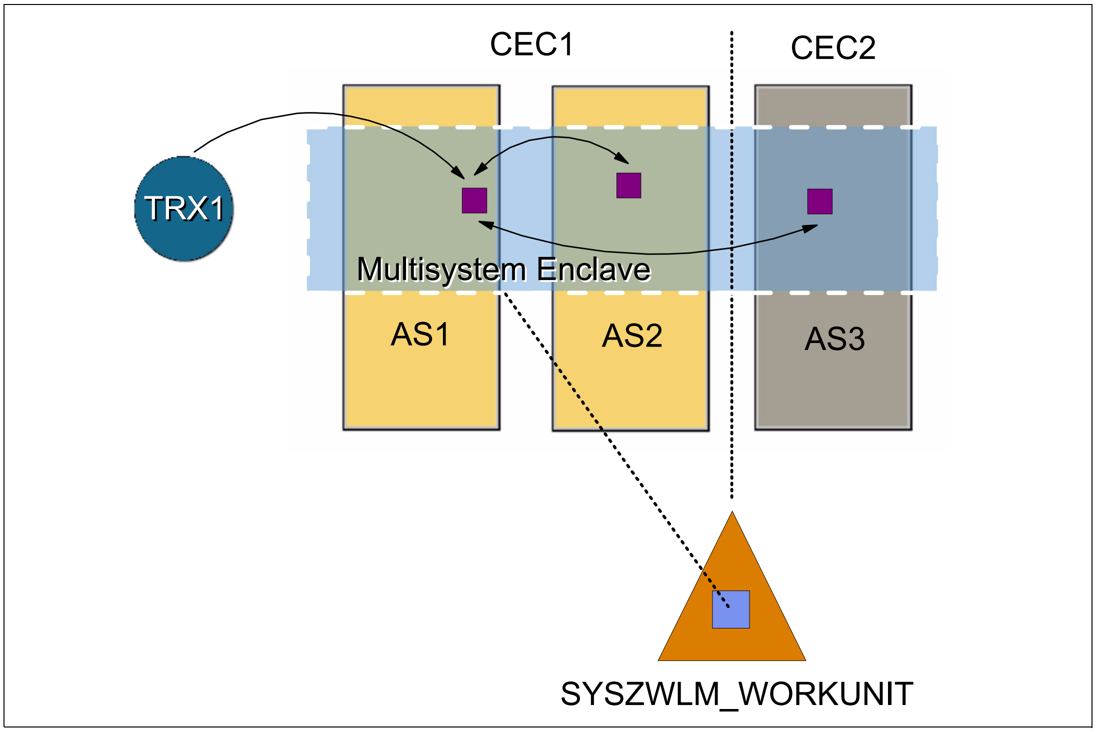


Figure 2-2 Multisystem enclaves

2.3 Understanding the algorithms

In this section, we provide explanations about what resources need to be managed and which algorithms are used to understand their usage. The resources involved during the decision-making process are CPU, I/O, and storage management.

Workload management

Workload management is a process of tracking the delay states and granting or denying access to system resources through the use of priorities. The definition of the delays for each type of resource are:

- CPU delays: Are ready to run, but not dispatched because of active work at the same or higher dispatching priority.
- Storage delays: There are several reasons that cause a delay for storage. Among them are:
  - Delayed for a page-in operation from auxiliary storage (AUX)
  - Swapped out, either logically or physically, with no MPL available for a swap in
  - Swapped out physically and delayed because of AUX page-in operations
- I/O delays: I/Os UCB Queue, Channel Subsystem busy, Director Port busy, Control Unit Busy, or Device busy conditions.

Beside the physical resource delays, such as CPU, storage, and I/O, there are other tracked delays such as a job that is delayed because of the lack of an appropriate initiator, or in general, a transaction that is delayed because of the lack of a server address space. We cover these delays later in this chapter.

There are different algorithms to manage the different resource types. WLM controls the use of CPU through dispatching priority and the use of I/O through I/O priority. Storage usage is controlled using storage targets, a swap protect time target, and the MPL swap mechanism.

2.3.1 Performance index

WLM maintains a *Performance Index* (PI) for each service class period to measure how the actual performance varies from the goal.

Because there are different types of goals, WLM needs to compare how well or how poorly workloads in one service class (SC) perform compared to other workloads. The comparison is possible through the use of the PI. The PI allows WLM to compare a workload having a velocity goal with a workload having an average response time goal and with a workload having a percentile response time goal in order to determine which is farthest from the goal (the highest PI).

The following list explains the meaning of the PI values:

- PI = 1 means that the SC period is exactly meeting its goal.
- PI > 1 means that the SC period is missing its goal.
- PI < 1 means that the SC period is beating its goal.

In the sysplex, every SC period can have two types of PIs:

- One or more *local PI*. There is one local PI for every z/OS image where the SC period work is running and represents its performance in each local system in the sysplex.
- One *sysplex PI.* It represents the SC period global performance across all the systems in the sysplex.

The global PI is reported in the RMF Workload Activity report and can be easily interpreted. If it is below 1 the goal is overachieved; if it is equal to 1 the goal is exactly met; and if it exceeds 1, then the goal is missed. Depending on the RMF options, the local PI can also be reported.

Through services provided by WLM, performance data is periodically exchanged between sysplex systems in goal mode. The information is sent between the systems and allows WLM to construct an approximate view of the status of the sysplex through the calculation of sysplex PIs. The aggregate view enables WLM algorithms such as policy adjustment to make trade-offs within each individual system. The data exchanged includes:

- Response times and execution using and delay information for each service class period to allow each local WLM to calculate the sysplex PIs for each service class period
- CPU service consumption information for each Resource Group, to globally enforce the capping and the protection algorithms
- Information identifying the active service policy

Sysplex PI and local PI

The *sysplex PI* tells us how well the work is performing in the sysplex. This information is displayed in most performance monitors and is used for adjusting resources.

The *local PI* tells how well the work is performing on the local system. This information is available in performance monitors and is also used for adjusting resources.

They are both important. The existing algorithms treat them similarly:

- Work in a service class that is achieving its goal on one system will not ignore the fact that work in the same service class is not achieving the goal on other systems.

- The sysplex PI is used first for importance 1 work; then the local PI is used for importance 1 work. This could be a local PI for other service class periods than the one selected because of the high global PI. Then, this search continues for other importance works.
- So, the local PI is always used and not overruled by the sysplex PI.

This procedure helps you manage installations where heterogeneous transactions run in the same service class but on different systems. A service class is identified as a receiver when its global PI of an Importance 1 service class period is less than 1 (“happy”) but the local PI is greater than 1.

Performance index evaluation in a sysplex

The PI is periodically calculated. The local PI and the sysplex PI are maintained for each service class period.

The policy adjustment loop (see the algorithm parts 1 on page 41 through 4., “The process is repeated for each lower Importance level.” on page 41) uses the sysplex PI at the beginning of each interval (every 10 seconds) to decide whether there is an adjustment that it can take to improve the sysplex PI for the most important service class period. The policy adjustment loop is performed a second time, this time focusing on the local service class period PI. An adjustment is made, if possible, to help a service class period from the local system perspective. This is to compensate for overachievement on one system cancelling out the effects of poor performance on other systems processing the same service class periods.

Performance index calculation

This section explains how the PI is calculated according to the type of goal. For every type of goal, the achieved value is then compared to the goal value. The achieved value is calculated, and the goal value is set by the user in the WLM ISPF application. Now, we explain the calculation of the achieved value:

- Average response type goal PI calculation

  The average response time value is calculated by dividing the sum of response time by the number of ended transactions.

  This gives the Average response time field, which corresponds to the achieved performance of the service class.

  Then, this field is divided by the goal response time to give the performance index of this service class period.

  In Example 2-1 on page 37, the ACTUAL field (under TRANS.-TIME) shows that the average response time is 0.157 seconds for the 432072 ended transactions.

  The goal of the service class is 0.180, as shown in the GOAL field. The PI is calculated for the sysplex (*ALL) and for each system in the sysplex by dividing the achieved average response time (either in the sysplex or on the single system) with the target average response time as illustrated in the formula:

$$PI = \frac{\text{Average Response Time}}{\text{Goal Average}}$$

Example 2-1 Average response time RMF report

```
 TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
AVG       0.00   ACTUAL                 157   SSCHRT   0.0   IOC          0   ABSRPTN         0   SINGLE     0.0   AVG        0.00
MPL       0.00   EXECUTION                0   RESP     0.0   CPU          0   TRX SERV        0   BLOCK      0.0   TOTAL      0.00
ENDED  432072    QUEUED                   0   CONN     0.0   MSO          0   TCB           0.0   SHARED     0.0   CENTRAL    0.00
END/S  720.32    R/S AFFINITY             0   DISC     0.0   SRB          0   SRB           0.0   HSP        0.0   EXPAND     0.00
£SWAPS       0   INELIGIBLE               0   Q+PEND   0.0   TOT          0   RCT           0.0   HSP MISS   0.0
EXCTD        0   CONVERSION               0   IOSQ     0.0   /SEC         0   IIT           0.0   EXP SNGL   0.0   SHARED     0.00
```

```
   AVG ENC  0.00   STD DEV             6.153                                   HST           0.0   EXP BLK    0.0
   REM ENC  0.00                                                                APPL %        0.0   EXP SHR    0.0
   MS ENC   0.00

           ---RESPONSE TIME---  EX   PERF   AVG  --USING%--   ------------ EXECUTION DELAYS % -------------  ---DLY%-- -CRYPTO%-    %
           HH.MM.SS.TTT         VEL  INDX  ADRSP   CPU   I/O  TOTAL                                          UNKN IDLE  USG  DLY QUIE
   GOAL    00.00.00.180   AVG
   ACTUALS
   *ALL    00.00.00.157         N/A   0.9    0.0   0.0   0.0    0.0                                           0.0  0.0  0.0  0.0  0.0
   OSOB    00.00.00.125         N/A   0.7    0.0   0.0   0.0    0.0                                           0.0  0.0  0.0  0.0  0.0
   OSOC    00.00.00.214         N/A   1.2    0.0   0.0   0.0    0.0                                           0.0  0.0  0.0  0.0  0.0
   OSOD    00.00.00.150         N/A   0.8    0.0   0.0   0.0    0.0                                           0.0  0.0  0.0  0.0  0.0
   OSOE    00.00.00.163         N/A   0.9    0.0   0.0   0.0    0.0                                           0.0  0.0  0.0  0.0  0.0
```

- Velocity goal type PI calculation

  The execution velocity percentage is calculated by dividing the number of the using sample by the sum of the using and delay samples.

  The using and delay samples are found in the Execution velocity RMF report.

  In Example 2-2, the using samples percentages are found in the fields USING % CPU and USING % I/O. Their sum is 20.1% + 21.7%, or 41.8%. The total of all the delay samples is given in the field EXECUTION DELAYS % TOT; the value is 40.0%.

  The achieved execution velocity is then 41.8% / (41.8% + 40.0%) = 51.1%.

  The goal is an execution velocity of 30%.

  The PI is then calculated by dividing the goal execution velocity by the achieved execution velocity. In our example, the calculation is 30% / 51.1%, for a total of a PI of 0.6 (the result of the division is rounded). This formula is:

  $$\text{PI} = \frac{\text{Goal execution Velocity}}{\text{Actual execution Velocity}}$$

Example 2-2 Execution velocity RMF report

```
TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
   AVG      2.97   ACTUAL            1.45.029   SSCHRT 899.2   IOC     223323    ABSRPTN     58154   SINGLE     0.0   AVG     3966.86
   MPL      2.97   EXECUTION         1.44.561   RESP      0.9   CPU      8820K   TRX SERV    58154   BLOCK      0.0   TOTAL   11780.8
   ENDED      12   QUEUED                 468   CONN      0.6   MSO     77155K   TCB         425.0   SHARED     0.0   CENTRAL 11780.8
   END/S    0.02   R/S AFFINITY             0   DISC      0.0   SRB     153458    SRB           7.4   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE               0   Q+PEND    0.3   TOT     86352K   RCT           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION             262   IOSQ      0.0   /SEC    172707    IIT           3.1   EXP SNGL   0.0   SHARED     0.00
   AVG ENC  0.00   STD DEV             50.588                                   HST           0.0   EXP BLK    0.0
   REM ENC  0.00                                                                APPL %       87.1   EXP SHR    0.0
   MS ENC   0.00

   VELOCITY MIGRATION:   I/O MGMT  51.1%    INIT MGMT 51.0%

           ---RESPONSE TIME---  EX   PERF  AVG   --USING%- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
           HH.MM.SS.TTT         VEL  INDX ADRSP   CPU  I/O  TOT CAPP  CPU  I/O                     UNKN  IDLE USG  DLY  USG  DLY  QUIE
   GOAL                        30.0%
   ACTUALS
   ITS1                        51.1%  0.6   1.0  20.1 21.7 40.0 18.0 15.4  6.6                     18.1  0.0  0.0  0.0  0.0  0.0  0.0
```

- Average response time with percentile goal type PI calculation

  The calculation of the achieved response time is somewhat complicated for this type of goal (RMF does it for you). In this section, we explain it.

  Let us take Example 2-3 on page 39 as a base reference.

  WLM keeps response time distribution data in *buckets* for service class periods that have a response time goal specified. These buckets are counters that keep track of the number of transactions ended within a certain response time range. The response times are stored in a structure that contains 14 buckets. These buckets exist per service class period:

  – Bucket range values

The bucket’s value ranges from *half of the goal response time value* to *four times the goal response time value*. In our case, the goal is 70% of transactions with a response time of 0.600 seconds. The buckets value will range from 0.300 seconds to 2.400 seconds.

The sixth bucket contains the number of transactions that ended within the goal response time value, 0.600 seconds in our case. In our example, 84.6% of the accumulated transaction number ended within the goal response time value. The bucket itself shows that 475 transactions representing 1.6% of the total number of transactions have a response time of 0.600 seconds.

The thirteenth bucket contains the number of transactions that ended within *four times the goal response time value (2.400 seconds).*

The last bucket contains the transactions that ended at more than *four times the goal response time value (more than 2.400 seconds).*

- The NUMBER OF TRANSACTION CUM TOTAL field accumulates the transactions at each bucket.
- The maximum value for a PI in a percentile goal is 4.0 due to the way the buckets are organized. Bucket 14 always corresponds to 400% of the value of the goal. The minimum value for the PI is obviously 0.5 due to the bucket range organization.
- The RESPONSE TIME DISTRIBUTION CUM TOTAL field contains the accumulated percentage of transactions at each bucket. To calculate the PI, WLM checks this field until it finds the goal percentile or the immediate above value. In our case, the goal percentile is 70%. The percentile immediately equal to or above is 74.9%, located in the first bucket. Then, WLM picks the response time of this bucket, 0.300 seconds in our case, and divides it by the goal response time, which is 0.600 seconds. The PI of this service class is then 0.5, as shown in the RMF report.
- The general formula is then:

$$\text{PI} = \frac{\text{Percentile actual}}{\text{Percentile goal}}$$

Example 2-3 Average Response Time with Percentile RMF Report

```
---RESPONSE TIME---  EX    PERF
           HH.MM.SS.TTT         VEL  INDX
   GOAL    00.00.00.600  70.0%
   ACTUALS
   *ALL                 84.6%  N/A   0.5
   NA01                  100%  N/A   0.5
   PA01                  100%  N/A   0.5
   QA01                 84.4%  N/A   0.5

                                          ----------RESPONSE TIME DISTRIBUTION----------
      ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
      HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  !....!....!....!....!....!....!....!....!....!....!
   <  00.00.00.300        22925            22925         74.9        74.9  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   <= 00.00.00.360        23681              756         77.3         2.5  >>
   <= 00.00.00.420        24338              657         79.5         2.1  >>
   <= 00.00.00.480        24900              562         81.3         1.8  >>
   <= 00.00.00.540        25420              520         83.0         1.7  >>
   <= 00.00.00.600        25895              475         84.6         1.6  >>
   <= 00.00.00.660        26283              388         85.8         1.3  >
   <= 00.00.00.720        26662              379         87.1         1.2  >
   <= 00.00.00.780        26992              330         88.1         1.1  >
   <= 00.00.00.840        27271              279         89.0         0.9  >
   <= 00.00.00.900        27508              237         89.8         0.8  >
   <= 00.00.01.200        28313              805         92.4         2.6  >>
   <= 00.00.02.400        29727             1414         97.1         4.6  >>>
   >  00.00.02.400        30626              899          100         2.9  >>
```

Performance index of discretionary goal type

A discretionary service class period is defined to have a PI equal to *0.81*. A discretionary service class period is also called a *universal donor*, because its PI indicates that it is always beating its goal.

2.3.2 Policy adjustment function

WLM constantly measures the systems. At certain intervals, it combines the measurement results to a complete picture, calculates how work progresses, and compares the actual achieved results with the goal definitions to find out whether adjustments are necessary.

The measurement of the system state takes places *every 250 milliseconds*. At these points, WLM captures the *delay* and *using* states of all dispatchable units in the system and resources that it manages. WLM basically manages CPU, DASD I/O devices, the system storage, server address spaces, and to some extent, resource serialization such as GRS enqueues. The states for these resources can be quite varied. While for CPU, there is currently one using and one delay state, it is possible to observe multiple cases why work is delayed because of storage, for example, paging delays, shortages of the page data sets, and much more.

Every 10 seconds, WLM summarizes the system state information and combines it with measured data such as CPU service or transaction end times. At this point, WLM *learns the behavior* of the workloads and the system. It keeps a certain amount of historical data in storage and maintains graphs learning the relationship between dependent functions in the system.

The following describes an example for such a graph. It is possible to learn from the time that work requests were delayed because they were swapped out, how many address spaces ideally should be in storage. The delay per transaction while the work unit was not in storage can be measured. This is a function of the fraction of ready users that are in storage, usually named the Multiprogramming Level (MPL). By plotting the MPL delay per transaction over the number of ready users with MPL slots, it is possible to learn at which point few MPL slots will cause a drastic increase in MPL delay. In addition, it can be learned at what point adding more MPL slots stops having a positive effect on achieving the service class goals. Such a plot can be used in a decision algorithm to determine the effect of increasing or decreasing the number of ready users in storage.

There are many more examples. At this level, notice that everything WLM does is based on measured and sampled data, which is combined to learn the relationship between workloads and resources in the system. After this step, the achieved response times and achieved execution velocities are compared with the goal definitions. The result is called the performance index (PI), which is the base for all further decisions in the system. The PI is defined as follows:

$$\text{Execution Velocity Goal : PI} = \frac{\text{Execution Velocity Goal}}{\text{Achieved Execution Velocity}}$$

$$\text{Response Time Goal : PI} = \frac{\text{Achieved Response Time}}{\text{Response Time Goal}}$$

The formula for percentile response time PI is more complex, and we will discuss it later.

The next step is to visit all goal achievements and to identify service classes that do not meet their goals as potential receivers for goal adjustments. Figure 2-3 on page 41 shows the basic decision flow. All service classes are logically sorted by their importance and their goal achievement. In order to find the service class period receiver, WLM running on one z/OS needs to determine the global PI for all active service class periods in its system. These figures are calculated by all the WLMs in the sysplex by exchanging information data through XCF. After that, the receiver service class period is selected through the algorithm:

1. Importance=1 and global PI > 1.
2. Importance=1 and local PI > 1.
3. Importance=2 and global PI > 1.
4. The process is repeated for each lower Importance level.

Then, WLM searches one or multiple donors that also use the resource. While receivers are searched from highest to lowest importance, donors are searched by examining discretionary work first and then from the lowest to the highest PI of “happy” service class periods. Importance does not contribute to the algorithm when the service class periods are “happy”.

If donors are found, WLM calculates and assesses the changes for the receivers and donors. At this point, it uses the internally captured and learned data, for example, the plots as previously described. The assessment logic calculates the improvement for the receiver and the expected degradation for the donors. If the logic concludes that the change is beneficial, it executes it and the adjustment is completed for this interval.

Figure 2-3 shows this algorithm.

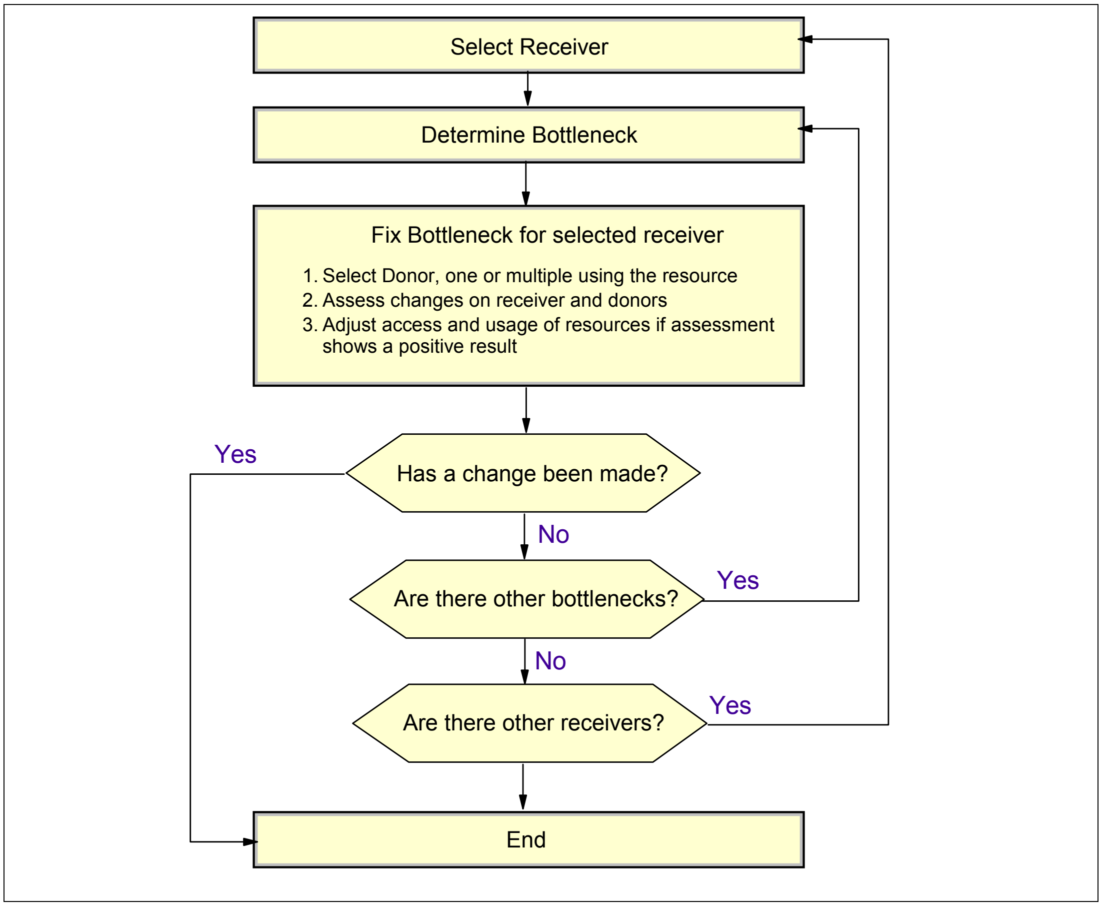

Select Receiver
Determine Bottleneck
Fix Bottleneck for selected receiver
1. Select Donor, one or multiple using the resource
2. Assess changes on receiver and donors
3. Adjust access and usage of resources if assessment shows a positive result
Yes
Has a change been made?
No
Are there other bottlenecks?
Yes
No
Are there other receivers?
Yes
End

Figure 2-3 The WLM policy adjustment algorithm

Note: Only one RECEIVER is managed at a time.

There are many possibilities why a change is beneficial and can be made. The easiest possibility is that the receiver is an important service class and the donors have lower importance. In that case, it can be safely assumed that the change is meaningful. But, it is also possible that a more important service class can become a donor for a less important service class, for example, if the more important service class overachieves its goals and the projections show that an adjustment will improve the less important work and still allow the more important work to easily meet its goals. There are also interesting cases where the receiver and the donor have the same importance and both do not meet their goals. In these cases, WLM will try to bring the goal achievement of both closer together, thus, improving the overall throughput of the system.

When a change has been made, the logic completes, new data is collected, and the evaluations and assessment run after ten seconds again. If no change was possible, WLM first tries to examine other potential bottlenecks of the receiver and then continues with lesser important work until one or no change could be made.

Impact of the number of service class periods

According to the description of the Policy Adjustment algorithm, it is obvious that the number of service classes periods can impact the efficiency of the process.

Too many service classes can cause some work to not get attention in time, so that the system makes trade-offs between work with similar goals and characteristics.

There has been quite a bit of confusion regarding the number of service classes that you might want to define in a WLM policy. The value that has been stated often by many is 30, and the number has been related to WLM overhead. To clarify this a bit, the value refers more specifically to the number of ACTIVE service class periods, and it is not an issue of overhead. The recommended value is to keep the number of active service class periods between 25 and 35.

Because of the sampling nature of WLM, the philosophy has been to combine workloads into fewer service classes and allow WLM to manage similar work appropriately. The issue with respect to the number of service class periods is that WLM only makes one change during every 10-second interval. If you have too many service class periods active, then lower importance work might get delayed because WLM is unable to make adjustments in a timely manner. So, it is a matter of responsiveness, not overhead, that causes us to recommend 25–30 service class periods.

Having too many service class periods with little activity in these periods means that there is not enough work defined to these service class periods. In these cases, you might experience highly variable performance indexes; combining similar work into fewer service classes helps alleviate this situation. You can have a policy with more than 30 service class periods; however, the number of active service class periods in a system should not be more than 30.

Figure 2-4 is the sample JCL to run the RMF Postprocessor Sysplex report by service class period. Run this report for a whole production cycle on your system. In this example, we run the report for a 9 hour interval on December 12, 2006.

```
//JOBNAME  JOB (ACCOUNT),'PGMRNAME',CLASS=A
//RMFPP    EXEC PGM=ERBRMFPP
//MFPMSGDS DD   SYSOUT=*
//SYSIN    DD   *
  DATE(12122006,12122006)
  DINTV(0900)
  RTOD(0000,0900)
  SYSRPTS(WLMGL(SCPER))
  SYSOUT(A)
/*
```

Figure 2-4 RMF Postprocessor Sysplex report sample

Example 2-4 shows that there is no activity in service class DDFSP7 in the sysplex.

Example 2-4 RMF report showing no activity for SC DDFSP7

```
z/OS V1R8               SYSPLEX SANDBOX            START 12/12/2006-00.00.00 INTERVAL
009.00.00   MODE = GOAL RPT VERSION V1R8 RMF       END   12/12/2006-09.00.00

REPORT BY: POLICY=WLMPOL     WORKLOAD=DATABASE   SERVICE CLASS=DDFSP7 RESOURCE GROUP=DDF
                                                CRITICAL     =NONE

 ----------        ALL DATA ZERO        ----------
```

Combine work that is similar. You can use report classes to analyze the performances of individual workloads running in the same service class.

Donor and receiver determination

This section discusses the donor and receiver determination in a general way. For more technical details, see “Selecting a receiver candidate” on page 45.

In order to meet the goals and requirements of the workload requests, the policy adjustment function determines:

- Which service class periods require more resources to attain their goals.
- WLM selects one receiver from these candidates. The receiver is the most important work in terms of service class period that needs help meeting its goal.
- Which resource bottlenecks (delays) are affecting the receiver. A service class period that needs help must already have detected delays. A service class period might be missing its goal, but if the delays are for resources that the WLM does not control, such as locks or enqueues, then the service class cannot be helped, and is not a receiver.
- Which donors can donate resource to the receiver or exchange priorities.

In the policy adjustment loop, WLM checks:

- If PI>1, then the service class period is eligible for help and is a candidate to be a RECEIVER.
- If PI<1, then the service class period is a potential DONOR.

Remember that a service class is selected by business importance and by worst PI.

The donor and receivers are address spaces and enclaves, but WLM manages resources for a receiver/donor service class period, not a receiver/donor address space or enclave.

WLM uses resource plots to project a recipient benefit and a donor cost. The net benefit must be positive before a resource adjustment takes place.

In order to derive the net value of the receiver/donor relationship, WLM creates the following plots to track how well work is processed:

- System paging delay plots
- Period MPL delay plot
- Period ready user average plot
- Period swap delay plot
- Period paging rate plot
- Period proportional aggregate speed plot
- Address space paging plot
- I/O plots

For workloads containing subsystem transactions, such as CICS and IMS, where the response time goal is assigned to the transactions using CICS/IMS subsystem classification rules (transaction mode), the priorities of the address spaces are determined dynamically by the performance goal in the service class (called external SC) of the transactions that these address spaces are processing:

- The priorities (I/O and CPU) of the address spaces are determined by the need of the most important and most unhappy (PI > 1.0) transactions running in those address spaces. Remember that:
  - The same transaction can run in multiple address spaces.
  - One address space can run transactions from multiple external service classes.

In this environment, there are chances of free rides, where less important happy transactions are running in the same address spaces and as a consequence have a lift in their performance. You can decrease the effect of the free ride by using the WLM option as a routing algorithm in the CP/SM; therefore, WLM will identify the best application-owning region (AOR) to route the transaction. WLM will try to locate the AOR with same goal transactions:

- Only average response time and percentile response time can be specified as goal. It is not possible to measure the Using and Delay states of the CICS transactions.
- CICS/IMS subsystem is used in classification rules to associate with a service class.
- Periods or Resource Group capping are not allowed, because it is not possible to derive the service units consumed by a CICS transaction.
- Subsystem work manager delays describing internal states of a transaction are reported by RMF.
- RMF shows transaction response time and transaction rates associated with the SC defined in transaction mode.

To implement these functions, WLM needs to implement two types of donors and receivers:

- Goals Receivers/Donors, with the external service classes periods described above.
- Resource Receivers/Donors, with the service class periods representing the CICS/IMS address spaces where the transactions are executed. These service classes are generated internally by WLM and are called *dynamic internal service (DIS) class periods*. The delay and using states of these service classes are proportioned to the external service class periods based on the time the address spaces (ASs) of the DIS class period

were serving transactions from the external service class period. Based in these delays and the PI of the external service class period, the priorities of the address spaces of the DIS class periods are raised (resource receiver) or decreased (resource donor).

In summary, the goal receiver and donor are the CICS/IMS transactions in the external service class with goals and the resource receiver and donor are the CICS/IMS address spaces with delays and priorities.

Selecting a receiver candidate

The following describes the criteria used to select a candidate service class to be a receiver.

If one of the criteria is met, then the search stops, because it is the aim of the routine to find *only one* receiver at each invocation. This is the order in which the evaluation takes place:

1. A service class period that is not meeting its goal that is running in a Resource Group that is running below the Resource Group service rate minimum. The best receiver is chosen by the largest amount below Resource Group service minimum and by most importance and by highest PI.
2. A service class period not meeting its goal, running in a Resource Group, between Resource Group service minimum and maximum or a service class period not meeting its goal and not running in a Resource Group. The best receiver is chosen by most importance and highest PI within importance.
3. A discretionary period (universal donor) with a Resource Group service below minimum. The best receiver is chosen by largest amount below Resource Group service minimum.
4. A service class period meeting its goal. The best receiver is chosen by largest PI.

As you can see, if you do not use Resource Groups, only steps 2 and 4 are considered.

Periods for certain service classes are not candidates for being a receiver because in the past they were defined as receivers without much improvement in their PI. Those service class periods are skipped for a while to avoid wasting effort. This usually only happens when the installation sets a very difficult goal.

Sometimes, a Resource Group is skipped even when the CPU consumption is below the minimum, because the analysis of the local data in every z/OS image indicates that another image can do better than this one. This avoids having all the systems overreact to such a situation.

If a receiver service class (ESC) is associated with a CICS address space or an IMS MPR address space (DISP) running important transactions with a PI greater than one, together with less important CICS transactions, then resources are given to the DISP to help the important ESC. Obviously, the less important ESCs also benefit from these resources. We refer to this as giving the low importance work a “free ride.” For more information about ESC and DISP, see “Server topology” on page 67.

Find a receiver bottleneck

Generally, in order to find a bottleneck in the receiver, the general resource delays are analyzed:

- Processor delay
- Swap page-in delay
- MPL (OUTR) delay
- I/O delay (DASD or Tape device delay)
- Server address space delay
- Paging delay (local, common, memory, or hiperspaces)

If none of these resources is a bottleneck, such as when the delay is due to capping, locks, enqueue, or operator, then the receiver cannot be helped and the algorithm selects another receiver candidate by going back to the *select a receiver* step.

An important distinction among the delay sample types is the way in which they are counted. CPU delay samples, I/O delay samples, and paging delay samples are counted by dispatchable units (multi-state). Swap-related delays (MPL and swapping) are counted per address space, regardless of the number of ready dispatchable units (single-state). Just as a reminder, RMF in workflow% calculations uses a single-state mode. This difference might cause the following effect, in a multi-task swappable workload, WLM tends to fix problems for swapped in work before allowing the swap in of swapped out work. This happens because CPU delays and paging delays tend to dominate over swap and MPL delay samples.

There are two cases, non-server and server, when selecting a receiver:

- Non-server case

  The primary way to help a non-served receiver is to alleviate one of the receiver's delay reasons.

  If a cross-memory delay is returned (page fault suffered by the PC target address space), then the target address space is identified. This is so that the cross-memory fix routine can help the target address space by protecting its working set, which in turn should help the receiver.

- Server case

  In order to find a bottleneck for a receiver-served service class (ESC), also called *goal period*, the following steps are executed:

  a. Use the topology to understand the proportion of time (weights) that each server DISP spent serving the chosen receiver ESC.

  b. Calculate the delay samples for the receiver ESC.

  c. The receiver ESC is not associated specifically with address spaces, and only address spaces can be sampled when looking for delay reasons. To handle this, the delay samples of the server DISP address spaces are apportioned to the receiver ESC based on the previous calculated weights. In other words, the delay samples for the server DISP address spaces are scaled by the proportion of time that the server DISP was serving this receiver ESC in relation to the amount of time that the server period was serving any DISP.

  d. Find the resource with the largest delay in the receiver's ESC. Select the related DISP. This is the service class period passed to the fix routines.

2.3.3 CPU management

Access to the CPU is controlled through dispatching priorities. The dispatcher maintains a queue of ready work units, which is sorted in priority order. This section explains how this queue is managed by WLM. WLM assigns dispatch priorities to service classes, based on:

- Need of the service class:
  - Goal achievement, expressed by performance index
  - CPU consumption
- Tries to assess what a change means:
  - Basically moves service classes to higher or lower dispatch priorities
  - Calculates the change in service consumption
  - Calculates the change in goal achievement

– Changes dispatch priorities if it is beneficial for more important (higher importance) work

Note: You cannot enforce a certain behavior by setting unrealistic goals. In general, you should base goals on observations and knowledge of the work. This is especially important for execution velocity goals.

Dispatching priorities assignment

A *dispatching priority* is a number from 0 to 255 associated with a dispatchable unit of work (TCBs or SRBs). It is placed in an address space or in an enclave control block (for more information about enclaves, refer to "Enclave" on page 34). It indicates the priority of these dispatchable units for getting CPU service. In goal mode, dispatching priorities are not assigned by the installation, but by WLM.

Table 2-1 shows the range of dispatch priorities prior to z/OS 1.8 while Table 2-2 shows the new dispatching priorities as of z/OS 1.8.

Table 2-1 Dispatching priorities prior to z/OS 1.8

255	FF	SYSTEM
254	FE	SYSSTC
253	FD	Small consumer
252-208	FC-D0	Dynamically-managed dispatch priorities
207-202	CF-CA	Not used
201-192	C9-C0	Discretionary
191	BF	Quiesce

Table 2-2 Dispatching priorities as of z/OS 1.8

255	FF	SYSTEM
254	FE	SYSSTC, SYSSTC1-5
253-249	FD-F9	Reserved
248	F8	Small consumer
247-203	F7-CB	Dynamically-managed dispatch priorities
202	CA	Not used
201-192	C9-C0	Discretionary
191	BF	Quiesce

The dispatch priority of 255 is reserved to keep the system running and to provide system work preferential access to the CPU. All system work should run in the predefined service class SYSTEM, which always gets a dispatching priority of 255.

Support work and system-related work can be associated with another predefined service class SYSSTC and SYSSTC1-SYSSTC5. Work in SYSSTC and SYSSTC1-5 gets dispatching priority 254.

Note: SYSSTC1-SYSSTC5 are called Process entitlement groups, and IBM does not recommend that you use them. You might want to classify work into them and use them as report classes. The dispatching priority is the same as for SYSSTC.

Dispatching priorities 253-249 are reserved for future use.

Note: The SPM qualifier in the classification rules helps automatically to define this work in the proper service classes.

Below this starts the range of dispatch priorities that are dynamically assigned to user-defined service classes. These are all service classes with an importance of 1 to 5. One special dispatch priority is used for service classes that consume very little CPU service. Because it is difficult for WLM and not beneficial for the whole system to deal with work that only requires very little CPU, WLM gives this work a dispatch priority just below the two system-related dispatch priorities.

All dispatchable units of the same service class period have the same base dispatching priority (with the exception of the mean-time-to-wait group for discretionary work).

The dispatching order of all dispatchable units with the same dispatching priority in the same or in different service class periods are periodically rotated.

Dispatching priority 202 is not used and the range from 192 to 201 is used for discretionary work. All discretionary work is managed by a mean time to wait algorithm, which simplified gives a higher dispatch priority for work using I/O and a lower dispatch priority to work demanding a lot of CPU. The last used dispatch priority is *191*. It is assigned to quiesced work. That also means that quiesced work can run if the CPU is not busy. It also means that all other work can access the CPU before it.

The interesting range is from 203 to 247. WLM assigns these dispatch priorities to the user-defined service classes that have a goal. The assignment is based on the CPU consumption, CPU demand, and goal achievement of the work. In the previous chapter, we saw that WLM determines through the performance index which service classes meet and which miss their goals. In the next step, WLM starts to look for a receiver by starting with the most important service class with the worst PI. Then, it determines the resource for which an adjustment is required. For CPU, that means that the service class must show a high number of CPU delays. Now, WLM looks for donors. Based on the captured data, WLM knows how much CPU is consumed by each service class and how much CPU is consumed by all work at the same dispatch priority. From the CPU delays and average time that the work used the CPU, it is possible to calculate the demand that the work has for the CPU. In order to assign a new dispatch priority to a service class, WLM logically moves the service class to higher dispatch priority and then projects the change of CPU consumption, demand, and delays for all service classes at a lower and the same new dispatch priority of the work that is moved up. If that is not sufficient, WLM looks in the next step whether some service classes can run with a lower dispatch priority and does the same calculations. The algorithm continues until a positive result can be projected, meaning that the performance of the receiver gets better and no service classes with a higher importance are badly affected, or until WLM concludes that no change is possible.

Note: If I/O Priority management is not activated, the DP of the service class is used as I/O priority. If I/O priority is activated, WLM calculates an I/O priority based on demand, goals, and goal achievement for each service class using I/O devices. The I/O priority is independent from the CPU dispatching priority. There are eight I/O dispatching priorities. An assessment between two or more service classes takes place when these service classes use the same set (or subset) of devices. So, if one service class misses its goal and policy adjustment finds out that I/O is the biggest problem, WLM tries to find other service classes that use the same set (or a subset) of the devices for the service class missing the goals. Then, the usual assessment logic starts to find out whether a change in I/O dispatching priority is possible between them.

An example of the CPU management algorithm

Figure 2-5 depicts this algorithm. The example assumes nine service classes, and the policy adjustment algorithm selected service class I as a potential receiver of the CPU service. In the first step, it moves *I* to higher dispatch priority. Together with I, there are two other service classes, G and H, at the same dispatch priority. After the first move up for *I*, the calculations show that *H* also has to move up because otherwise no overall value of the move for service class *I* can be projected. This projection is named *net value*, meaning that the overall change does not harm work of higher importance and is still beneficial for the receiver. But this move does not show enough receiver value, meaning that the change does not help *I* enough to improve its performance. In the next step, WLM tries to move work with higher dispatch priorities to lower dispatch priorities. While the move down for *C* is acceptable, the move down for *B* violates the net value test. Finally, the move down of *A* passes the net value test, but again we do not have sufficient receiver value for *I*. In the third step, WLM again attempts to move the receiver up. In this case, the move for service class *I* to dispatch priority 249 gives enough receiver value for *I* and all other classes pass the net value tests. Now, WLM concludes that the change is meaningful and adjusts the dispatch priorities of the work. In case no valuable change can be projected, all intermediate calculations are set back, and no change occurs.

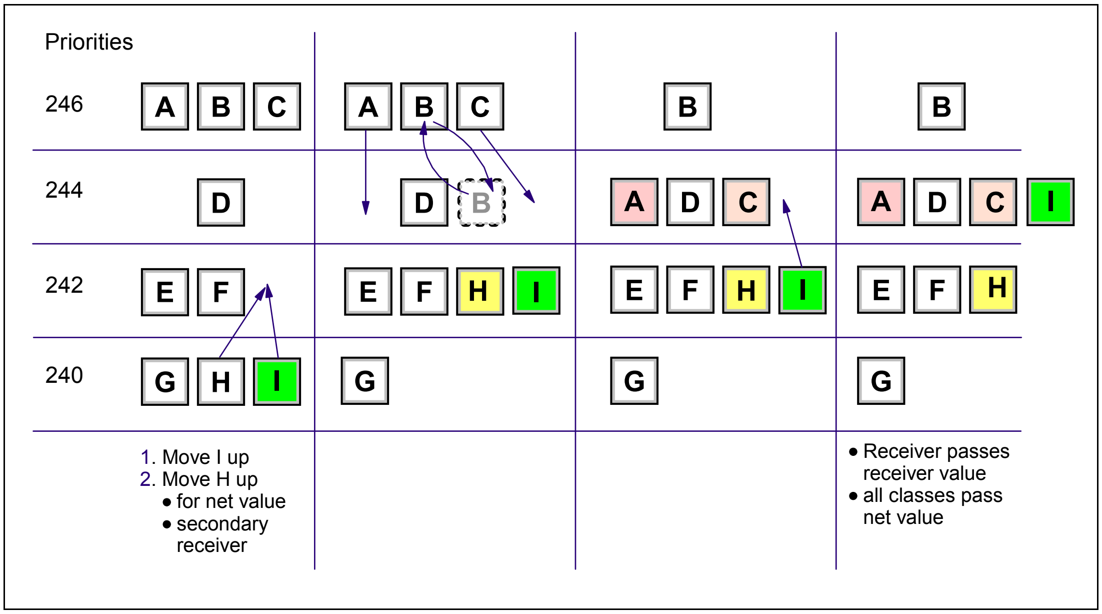


Figure 2-5 WLM priority adjustment

When we look at this example, we can also observe that together with service class I, service class H was moved to a higher dispatch priority. This became necessary because the first move showed that H would have been degraded too much. In the end, H is below I, but now competes with service classes E and F at the same dispatch priority and therefore has better and sufficient access to the CPU. In the end, the picture has changed to some extent. The moves are calculated based on the current and the expected CPU consumption, which is derived from the CPU demand of the service classes and the amount of CPU that is available at each dispatch priority level.

Note: This example also demonstrates why WLM only executes one change per policy adjustment interval. One change does not mean that only the resource access for one service class has been changed. It means that the access to a resource for a set of service classes is changed in order to help one service class.

Discretionary goals

A discretionary goal means “do the best that you can,” and usually applies to batch jobs.

Furthermore, this type goal can also offer other advantages, such as:

- It always runs in a low MTTW dispatching priority, thus improving CPU and I/O parallelism.
- It is the goal of the SYSOTHER service class (the non-STC default service class), ensuring that workloads arriving to be processed that do not have classification rules defined are not assigned to an important service class.
- It is only allowed in the last period of a service class, thus forcing long transactions to have a low priority (aging).
- It is a candidate for individual storage control to improve the use of central storage.
- Together with the specification of a Resource Group minimum, it can be used as a sort of throughput goal.

Note: Specifying a Resource Group to a service class can help it not become a universal donor.

A discretionary goal can be seen as a buffer of CPU resources available, and capacity planners should trigger actions when these service classes cannot have CPU resources.

How can a discretionary goal receive (at least) CPU resources:

- To assist discretionary work to get some CPU service when other types of work are overachieving their goals (PI consistently less than one), a design is implemented by dynamically capping the overachieving work, therefore freeing CPU cycles to be used by the discretionary type of work.
- In WLM discretionary goal management, the receiver has a discretionary goal and the donor does not. The donation is implemented through the use of capping. The following conditions need to be fulfilled by a service class period in order to be considered a donor to a discretionary service class period:
  - It cannot currently be a receiver or donor.
  - It cannot be a member of a Resource Group (RG).
  - It cannot be a small consumer (an internal WLM qualifier).
  - It cannot be the last uncapped nondiscretionary period.
  - If it has a velocity goal, the goal must be less than or equal to 30%.
  - If it has a response time goal, the goal must be more than one minute.
  - It has a PI less than 0.7.
  - It has not been a significant receiver candidate for CPU lately.

2.3.4 Swap control

The Multiprogramming Level (MPL) is computed internally by WLM (no more external specifications) as follows:

- MPL in_target (ITMPL)

  The number of service class period address spaces required to be swapped in to make the service class period meet its goal.

- MPL out_target (OTMPL)

  The upper limit of address spaces allowed to be swapped in without causing too much of a constraint to the central storage resource. It can be decreased to guarantee memory for the swapped in address spaces.

These numbers are dynamically adjusted through the following routines:

- Policy adjustment routine

  Based on the PI values and the reason for delay (MPL, for example), a service class period receiver is selected to increase its MPL targets. WLM selects a service class donor where MPL targets are decreased.

- Resource adjustment routine

  Based on the utilization of the system (underutilized or overutilized), a service class period is chosen to increase (underutilized) or to decrease (overutilized) its OTMPL by one.

Refer to 2.3.2, "Policy adjustment function" on page 40, for more information about this subject.

Swap protect time

In goal mode, swap protect time is the time in milliseconds that a swapped out address space remains in processor storage (central only in z/Architecture) before becoming a candidate for auxiliary storage (AUX). It is a value set specifically for each service class period by WLM.

The policy adjustment function uses the swap protect time to guarantee processor storage to service class periods by allowing address spaces to stay logically swapped out in processor storage for a certain amount of time, hoping that they become ready before being moved to AUX. Policy adjustment raises the swap protect time for a service class period receiver when its major delay is swap in delay and it is missing goals. This delay indicates that the address spaces belonging to the service class period are suffering delays due to working set page-in after a swap in WLM decision. The service class donors have a swap protect time decrease.

Resource adjustment also modifies swap protect time of some service class periods, depending on the availability of processor storage.

2.3.5 Storage targets

In WLM, the least recently used (LRU) algorithm was implemented in the resource adjustment function. However, in OPT keywords, there are still external options such as MCCAFCTH (which specifies the low and the OK threshold values for the central storage available queue).

Note: In z/Architecture, with logical partitions (LPARs) defined with large memory, take care to set the proper value of the MCCAFCTH parameter. Starting with z/OS 1.8, physical swaps targeting pageable storage shortage are eliminated, and potential shortage is resolved by frame exchanges. See APAR OA14409 for more details.

There are four types of storage targets:

- Protective Processor: A storage target to protect a target number of pages in processor storage. Do not migrate pages to auxiliary storage (AUX) if the current working set in processor storage is below this target.
- Restrictive Processor: A storage target to preferentially migrate pages to AUX, if the current working set in processor storage is above this target. It is greater or equal to the protective target.
- Protective Central: A storage target to protect a target number of pages in central storage. Do not steal pages from central storage if the current working set in central storage is below this target.
- Restrictive Central: A storage target to preferentially steal pages from central storage if the current working set in central storage is above this target. It is greater or equal to the protective target.

Storage protection in WLM

STORAGE CRITICAL is an attribute for an address space or for the classification rules for CICS and IMS to help be protected against page stealing.

Refer to “Where to specify CPU CRITICAL” on page 87 for information about the effects of setting this parameter.

2.3.6 I/O management

Each I/O request is performed by or on behalf of a z/OS dispatchable unit, either a TCB or an SRB. The dispatchable units are associated with:

- Address spaces
- Enclaves

Each address space-oriented unit of work and enclave-oriented unit of work is associated with a service class period, which becomes the focal point of the management algorithms. For the purposes of managing I/O, WLM controls a small fixed number of I/O priority levels.

As a reminder, the general formula of the response time is:

```
Response_Time = Service_Time + Queue_Time
```

The purpose of this chapter is to determine where WLM can use priorities to manage the Queue_Time (QT) in a DASD subsystem.

I/O queues

There are queues where priority management can apply:

- IOS UCB Queue: This is a local queue, because it is per device and all requests come from the same operating system. This queue can be optionally managed by WLM if I/O Priority management is set to YES.

- Channel Subsystem Queues: This is a global queue because the requests come from all LPAR devices connected to that CPC or machine. So there can be competition among all these LPARs to access particular devices.
- Control Unit Queue: This is a global queue because the requests come from all the LPARs of all the connected CPCs. IBM control units starting with 2105 are able to use the I/O priority passed by WLM in the Define Extent CCW.

DASD I/O priority

I/O priorities are managed in the range of X'F8' through X'FF', where:

- X'FF' is reserved for high priority system address spaces (service class SYSTEM).
- X'FE' is reserved for started tasks (service class SYSSTC).
- X'F8' is reserved for address spaces with a discretionary goal (service class SYSOTHER).
- The range of priorities between X'F9' and X'FD' is dynamically assigned to address spaces and enclaves, as determined by the algorithms.

Each service class period maps to an I/O priority level. This mapping is dynamically adjusted based upon how each service class period is meeting its goals and how the I/O contributes to this success or failure. When the I/O supervisor (IOS) starts the I/O request and places the I/O request on the unit control block (UCB) queue for the target device, the priority of the request is used to place the request in priority sequence with respect to other pending requests for the same device.

TAPE I/O priority

With z/OS 1.8, tape I/O priorities are managed in the range of X'F8' through X'FF', where:

- X'FF' is reserved for high priority system address spaces (service class SYSTEM or SYSSTC).
- X'FE' is reserved for future use.
- X'FD' is reserved for service class periods with importance = 1.
- X'FC' is reserved for service class periods with importance = 2.
- X'FB' is reserved for service class periods with importance = 3.
- X'FA' is reserved for service class periods with importance = 4.
- X'F9' is reserved for service class periods with importance = 5.
- X'F8' is reserved for service class periods with discretionary goals.
- The range of priorities between X'F9' and X'FD' is dynamically assigned to address spaces and enclaves, as determined by the algorithms.

Each service class period maps to an I/O priority level. This mapping is dynamically adjusted based upon how each service class period is meeting its goals and how the I/O contributes to this success or failure. When IOS starts the I/O request and places the I/O request on the UCB queue for the target device, the priority of the request is used to place the request in priority sequence with respect to other pending requests for the same device.

Sysplex I/O priority management

The goals of the sysplex I/O priority management support are:

- To separate I/O scheduling priority from CPU dispatching priority.
- To manage DASD I/O contention at the sysplex level.

- To provide algorithms for assigning I/O priorities to the various workloads, based upon information gathered through sampling techniques and data obtained from the IOS component. It is a donor receiver function when the major delay is the I/O delay.
- To collect information and distribute the resulting I/O priority adjustments across systems in a sysplex, using existing WLM cross-system communications (XCF) services.
- For each I/O request, IOS sets the request priority based upon a value established by WLM for the requesting dispatchable unit.

The following changes have been introduced in the I/O process:

- IOS queues requests on the UCB queue by I/O priority.
- The I/O priority of an enclave or an address space is adjusted by WLM. Neither the business transaction nor the WLM administrator has control over I/O priority.

I/O priority management decisions

To manage I/O priorities, information is needed about the delays of each DASD device. When WLM wants to increase a service class period I/O priority, it does it with donor/receiver logic; see “Donor and receiver determination” on page 43. Therefore, WLM must be able to determine:

- If a service class period can significantly improve its goal achievement by raising its I/O priority (a potential receiver).
- If lowering the I/O priority of a service class period (a potential donor) can improve the achievement of the receiver.

WLM needs two types of information to make decisions on I/O priority management:

- Device usage information

  The device usage time is the sum of device connect time and device disconnect time for each I/O request, as measured by the I/O subsystem.

  WLM can make decisions by comparing the level of device usage across the range of I/O priorities, and projecting the amount of device usage that might occur if I/O priorities are altered.
- Device delay information

  WLM can make decisions by comparing the level of device contention across the range of I/O priorities, and projecting the amount of contention that might occur if I/O priorities are altered. The device delay time is the sum of the time that each request was delayed by IOS (IOS queue time) and the time delayed in the I/O subsystem (pending time).

Note: Channel measurement data gives the device connect and disconnect time and the device pending time (including channel busy, controller busy, shared device busy). The IOSQ time is a sampled value. The disconnect time is not taken into account when calculating VELOCITY.

Channel subsystem I/O priority queuing (CSS IOPQ)

This is a part of the IRD feature available on System z machines in z/Architecture mode.

CSS IOPQ is an extension of I/O priority queuing by allowing I/O prioritization within an LPAR cluster through a Service Assist Processor (SAP).

Table 2-3 on page 55 shows the priority settings.

Table 2-3 Priority settings by type of work

Priority	Type of work
FF	System work
FE	Importance 1 and 2 missing goals
FD	Importance 1 and 2 missing goals
F9-FC	Meeting goals - adjusted by ratio of connect time to elapsed time
F8	Discretionary

You can read more information about the IRD CSS IOPQ algorithm and setup in 3.5, “Intelligent Resource Director” on page 103, or see the IBM Redbook, *z/OS Intelligent Resource Director*, SG24-5952.

Dynamic Channel Path Management (DCM)

This is a part of the IRD feature available on System z servers in z/Architecture mode. DCM applies only to DASD subsystems at this time.

DCM can be implemented to reduce the amount of channel and switch contention. It introduces a new term called *I/O velocity*. I/O velocity is calculated for each logical control unit (LCU) by dividing the amount of time that the LCU is used productively by this amount of time plus an approximation of the amount of time that the LCU is delayed by channel busy and Switch port busy. Example 2-5 shows how I/O velocity is calculated.

Example 2-5 I/O velocity calculation

```
                                Connect Time
I/O Velocity  =    ---------------------------------------------------
                    Connect Time + Pend Time - (CU busy + Device busy)
```

For planning and using DCM, see the IBM Redbook, *z/OS Intelligent Resource Director,* SG24-5952.

Parallel Access Volumes (PAV)

Parallel Access Volume allows the same z/OS to execute multiple I/O operations concurrently against the same 3390 volume.

There are two ways to implement PAV devices in your installation: Static and dynamic alias management, depending on the assignment process.

PAV devices

Your installation can manage PAV devices either in static or dynamic mode. If static PAV is used, there is no change in device alias assignment unless an alias is deleted or added using ESS Specialist.

In this section, we provide more details about the current implementation of the WLM algorithms for dynamic alias management with IBM Enterprise Storage Server® (ESS). WLM can be used to dynamically manage the assignment of alias devices to base devices and thus ensure that the concurrency in I/O processing provided by PAV is available where it is most needed.

ESS supports up to 16 logical subsystems (LSSs) for System z; the combined number of alias and base devices in each LSS is limited to 256. The pool of alias devices for an LSS is managed by WLM. Because aliases can be viewed as resources within ESS shared by multiple systems, the WLM management of aliases is at a sysplex level. The WLM policy specified by the administrator determines whether dynamic management of aliases for a z/OS sysplex is to be activated. With WLM managing aliases, your installation does not need to determine how many aliases to assign to each base device and does not need to manually change this assignment when workloads change. WLM performs these tasks automatically. WLM uses two mechanisms to manage the number of aliases. Both of these mechanisms use sysplex-level device activity information to drive the decision to transfer an alias from one device to another.

The first mechanism, based on the *goal algorithm*, attempts to allocate additional aliases to a base device (the receiver device) that is experiencing IOS queuing delay and is servicing a workload that is missing its client-specified goal in the WLM policy. A donor device is found among the devices used by workloads that have equal or less business importance than the receiver device. If such a donor device is found, then an alias can be transferred from it to the receiver even if it means increased IOS queuing for the donor. However, if the donor device is used by a workload of equal importance to the receiver, the donor device must not suffer increased queuing as a result of losing the alias. These restrictions simplify the algorithm, because then WLM does not need to predict how much an alias transfer will help or hurt the attainment of business goals. An alias transfer requested by the goal algorithm does not necessarily decrease overall IOS queue delay. It does help a higher importance workload do better, but it could be at the expense of a lower importance workload, which is consistent with the philosophy for other WLM-managed resources.

The second mechanism, based on the *efficiency algorithm*, transfers alias devices from low contention base devices to high contention base devices without regard to the business importance of the workloads involved. The objective is to minimize the total IOS queuing within an LSS. This algorithm transfers an alias only when the goal algorithm did not result in an alias being transferred, for reasons having to do either with the receiver or with the donor device.

The goal algorithm is invoked more frequently than the efficiency algorithm. This gives the attainment of business goals precedence over simply reducing total IOS queuing. High contention base devices are those that have significant IOS queue delay. Low contention base devices are those that have no significant IOS queue delay and can give up an alias with no increase in queue delay. The efficiency algorithm is conservative, because it does not make a transfer that can potentially cause more harm to the donor than benefit to the receiver. The WLMs in the sysplex act as peers with respect to the efficiency algorithm: Any of them can use the sysplex-level PAV device data to initiate alias transfers regardless of who is using the devices.

Sysplex view of device activity

Because moving an alias from one base device to another affects every system using those devices, the algorithm making that determination should have a sysplex view of the logical volume, including the appropriate performance data. To build a sysplex view, each z/OS system in the sysplex broadcasts its local performance data to all other z/OS systems. A peer approach is used by WLM for managing aliases under the efficiency algorithm.

Any system in the sysplex is able to make alias transfer decisions based on the sysplex view. The goal algorithm is *locally oriented* because each system tracks the PAV devices that are causing the most IOS queue delay for each service class on the local system. Both the goal algorithm and the efficiency algorithm are invoked periodically, although with a different period.

Efficiency algorithm

Although the goal algorithm is invoked before the efficiency algorithm, we discuss the efficiency algorithm first in order to introduce some new concepts that are more easily described in this context. The WLM efficiency algorithm takes aliases from base devices with low contention and transfers them to base devices with higher contention; the contention is measured by the IOS queuing delay. The result is to minimize overall IOS queuing against an LSS. In a sysplex, invocations of the algorithm are scheduled at least 60 seconds apart. Note that this algorithm is invoked less frequently than the goal algorithm, for which the period is about 30 seconds.

A table listing all base volumes, as well as the unbound aliases, in an LSS, sorted by IOS queue length, is used to find the most “needy” receiver devices as well as the “richest” donor devices (unbound aliases are treated as having zero queue length). When a receiver-donor pair is identified, IOS is instructed to unbind the alias from the donor device and bind it to the receiver device. The efficiency algorithm can transfer multiple aliases in one pass but does not transfer more than one alias to or from a particular base device more often than once per cycle. See Figure 2-6 on page 58 and Table 2-4 on page 58.

Searching for a receiver-donor pair

WLM creates a table listing all base devices in an LSS. The table is sorted in descending order by sysplex IOS queue length. For ties on IOS queue length, the table entries are in ascending order with respect to the number of aliases assigned to the base device. Devices at the top, at the high-IOS-queue-length end of the table, are potential receivers of aliases. A device must have a sysplex IOS queue length above a threshold of 0.5 to be considered as a receiver; for example, an average queue length of 0.5 is calculated when an IOS device queue is sampled 40 times and when, on 20 of those samples, there was one request queued. The value of 0.5 was empirically chosen after experimentation. If no receiver device is found on the first pass, that is, if there are no devices with an IOS queue length greater than 0.5, a second pass is made in which receiver devices with lower levels of queuing are considered.

Devices at the bottom, at the low IOS queue length end of the table, are potential donors. Unbound aliases are placed at the very end of the low-IOS-queue-length end of the table. Unbound aliases are the first choice as donors because they represent unused resources, possibly unbound from base devices now offline and therefore idle. The efficiency algorithm attempts to transfer one alias from a donor to a receiver, but only if taking the alias from the donor is not expected to cause a significant increase in queue delay for the donor device.

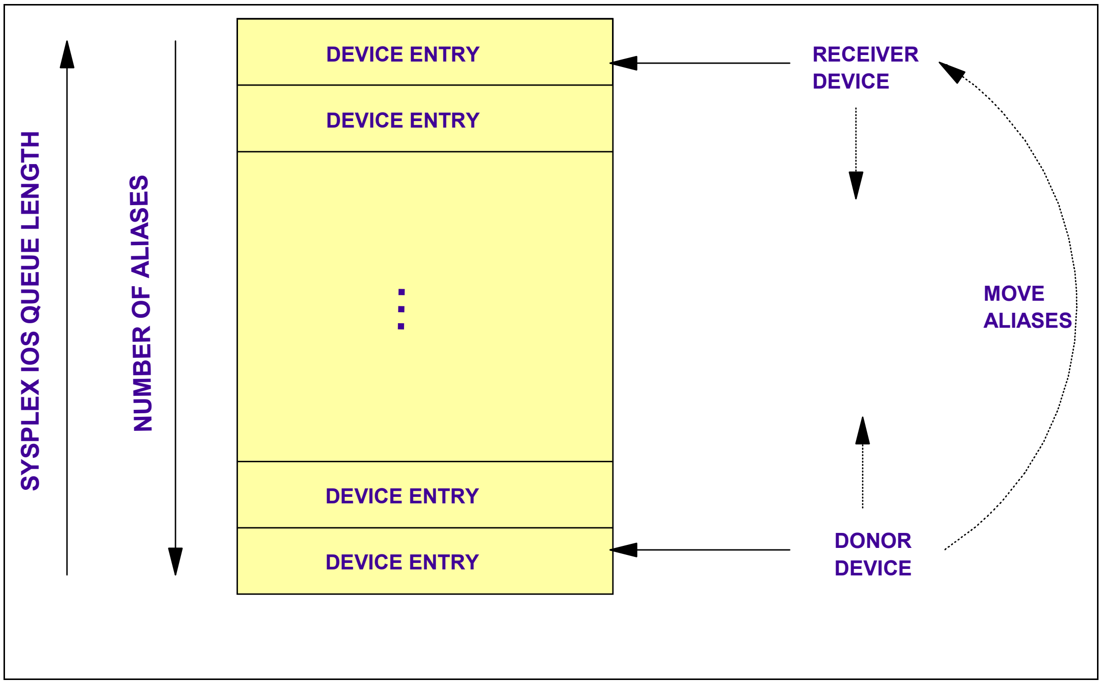


Figure 2-6 WLM PAV data table for a sysplex

Table 2-4 Alias donor effectiveness

Number of remaining aliases	Acceptable utilization
1	20%
2	35%
3	47%

Figure 2-6 shows the WLM PAV data table, which is the sorted table used for searching for receiver-donor pairs. The efficiency algorithm takes an alias from a donor device only if the action will not result in a significant increase in IOS queuing for the device. The more aliases that the device has, then the higher its utilization can be and still remain an acceptable donor. Table 2-4 shows an example of how the decision is made about the suitability of a donor.

According to this table, when a potential donor is taken down to one alias, its utilization must be below 20%. When the potential donor is taken down to two aliases, the current utilization must be below 35%. The table was generated from a multiserver queuing model used to estimate the impact on device queuing based on the number of aliases and the average utilization per alias. The utilization figures in the table were selected to ensure minimal impact on the performance of the donor device. Here, the definition of minimal impact on queuing is the point at which the probability of an I/O request encountering the base and all aliases in use is less than 20%. Thus, if the projected impact of removing an alias keeps the device below the utilization that would cause a 20% chance of queuing, then the alias can be transferred.

Goal algorithm

In many cases, the efficiency algorithm is sufficient to optimize alias placement in a logical subsystem and minimize overall IOS queue delay. However, there is one obvious case that the efficiency mechanism cannot handle. This is the situation in which an alias needs to be taken from a high activity device that is used only by work of lower importance and given to a high activity device that is delaying work of high importance. For example, suppose there is a transaction processing workload of high importance delayed by IOS queuing on a device, and there is a second device in the same logical subsystem that has multiple aliases and is used only by a batch workload of lower importance. Then, aliases should be taken from the second device and transferred to the first in order to decrease the queuing delays for the first device, even at the expense of increased queuing delays for the second device. The goal algorithm is designed to handle such cases. Moreover, the goal algorithm runs before the efficiency algorithm in order to ensure that when the number of donor devices is limited, goal-attainment alias transfers take precedence over efficiency transfers. Because the service-class-delay information used by the goal algorithm is only available to the local system, it is the local system that makes the alias transfer decision in order to help a local service class that is missing its goals.

The goal algorithm runs every 30 seconds on each system in the sysplex. Because the goal algorithm runs more frequently than the efficiency algorithm, it can be more responsive to situations where work of high importance is missing its goal. Specifically, the algorithm looks for a service class missing its goal and also experiencing a significant amount of IOS queue delay on the local system. For each such service class, an attempt is made to find additional aliases for up to three devices among those contributing the most to the delay of the service class on the local system. For each one of these three devices, the algorithm attempts to find a donor device.

Although the search for a donor is performed in the same order used in the efficiency algorithm, only devices used by service classes of the same or lower importance can qualify as donors. The device-to-service-class mapping is used to make this determination. If the donor device is used by less important service classes than the receiver, the transfer is made even if IOS queuing will increase on the donor device. If the donor is used by work of equal importance, a more conservative approach, similar to that used in the efficiency algorithm, is used. The alias transfer is made only if there will be no increase in IOS queuing on the donor.

Pacing of alias transfers

Alias transfers are paced in order to prevent overreaction to transient fluctuations in I/O workload. Any particular base device gets no more than one alias action, either added or removed, per minute. The exception is when an alias is needed by the goal algorithm to help a more important service class than the last service class helped in the subsystem. In this case, the minimum time between alias transfers is lowered to 30 seconds. This allows a more important workload to get first chance at taking aliases from devices that have had a recent alias transfer. If the pacing interval were one minute in all cases, the efficiency algorithm on one system in the sysplex could always be the first to run after the minute had expired; it could transfer several aliases from a donor base device and delay the goal algorithm on another system from helping work that is missing its goal.

Limiting the number of aliases

There is no prescribed maximum or minimum number of aliases that can be assigned to a base. The adjustment algorithms heuristically determine the optimal number of aliases needed to reduce or eliminate IOS queuing. WLM needs to recognize the point at which adding more aliases will not improve I/O performance. Additional aliases do not help if the IOS queuing is simply replaced by increased channel, control unit, or device contention. WLM will not add aliases to a device if its average pending time, control unit queuing time, or disconnect time is over a threshold. The current thresholds are 20 milliseconds for average control unit queuing plus disconnect time and 20 milliseconds for average pending time. (These threshold values were empirically derived based on experimentation.) The handling of the pending time threshold is different from handling of the CU queuing/disconnect time threshold. Average pending time for a PAV device is caused by a channel constraint local to a system. In this case, aliases are still added to reduce IOS queuing for the system with low pending time even though the other system has high pending time and cannot benefit from the additional alias. A pending device reservation is handled as a special case. WLM does not take action against delays that are a result of pending device reserves on the volume because adding an alias does not decrease this type of IOS queuing or pending time.

PAV management

There are two ways for the administrator to control dynamic alias management, either a sysplex level option in specifying the WLM policy or a device level option in the z/OS Hardware Configuration Definition (HCD). The sysplex level option in the WLM policy is a YES/NO corresponding to whether dynamic alias management is active (for the entire sysplex) or not. The default is NO, so that the administrator has a chance to prepare the configuration and upgrade to the required software release levels before switching on the dynamic alias management. The device level option in HCD enables or disables dynamic alias management for a particular PAV base device. A device is enabled by default. The device-level option in HCD is used to stop WLM from adjusting the number of aliases for a device that is shared by systems that do not support dynamic alias management. Such systems include systems from another sysplex or systems running an operating system other than z/OS, such as VM. If an installation were to allow dynamic alias management for such shared devices, the WLM alias transfer decisions would factor in only the device usage by the systems that are participating in dynamic alias management. These decisions, based on a partial view of device usage, are at risk of causing less than optimal alias allocation. The HCD device-level option is also used to disable dynamic alias management for base devices that are offline to some systems in the sysplex. If a device is online to some systems in the sysplex but offline to others, WLM does not make valid alias transfer decisions.

Note: Like OFFLINE=YES/NO, WLMPAV=YES/NO is specified in the HCD Define Device Parameters/Features panel and is specific to a system image. It is therefore possible to have a different WLMPAV specification for the same alias device on different system images. This is useful in situations where an installation has more than one sysplex with shared access to a 2105 logical control unit (LCU).

PAV dynamic alias management for paging devices

Your system availability might be impacted when certain events, such as an SVC dump, cause a burst of paging activity that locks out or slows down page fault resolution for your critical workloads. New, combined ASM/WLM exploitation of parallel access volumes (PAVs) on the IBM Enterprise Storage Server (ESS) allows the system to prevent page-outs from blocking page-in operations.

Enhancements in z/OS V1R3 allow Auxiliary Storage Manager (ASM) to state the minimum number of PAV dynamic aliases needed for a paging device. The more page data sets that you have on the volume, then the more alias devices are set aside. A minimum number of aliases is preserved by the system even in cases where aliases are moved to reflect changes in workload. You do not need to define any of this; simply enable the volume for dynamic alias management for your paging devices.

Note: All systems in your sysplex must be on a z/OS V1R3 or above level in order for this support to work.

Details about PAV metrics

The measurements used to drive the PAV adjustment algorithms are accumulated across a 60-second interval on each system in the sysplex. Thirty-second measurements are used if the goal algorithm needs to make an alias adjustment to help work of high importance before the full minute has elapsed. The values maintained are:

- Sysplex IOS queue length: IOS queuing on a base device is measured by WLM for dynamic alias management because it is the indicator that the device has more outstanding requests than its current number of aliases can handle concurrently. The average IOS queue length for each base on a single system is calculated by sampling the IOS queue every quarter second and averaging the queue length across the samples. The sysplex IOS queue length for each base is the sum of the average queue lengths on each system in the sysplex. The sum is used rather than an average across the systems to ensure WLM properly handles the case where only one system out of many in a sysplex has a nonzero queue length. Using a sum ensures appropriate action is taken for the device to relieve the contention within that one system. Averaging the queue length across the systems, especially in a large sysplex with many systems, can dilute the value so much that WLM does not recognize that action is needed.
- Total service time: The device utilization over the measurement interval is calculated for each system based on the total service time on the base device and all its current aliases. The sysplex device utilization is calculated as the maximum utilization observed by any system in the sysplex. This utilization is then used in a formula to determine if removing an alias increases IOS queuing for a device. See "Assessing impact on a donor device" in the section "Efficiency algorithm" on page 56 for more information.
- Average device delay time per request: This value includes control unit queue time and disconnect time. It is compared against the threshold of 20 milliseconds. If the sum is higher than the threshold, it indicates that the device cannot benefit from additional aliases, and WLM stops transferring aliases to the device.
- Average pending time per request: This value represents channel contention. It is compared against a threshold of 20 milliseconds. If this value is higher than the threshold on the local system, it indicates the device cannot benefit from additional aliases.
- Device-to-service-class mapping: This is an array indicating which service classes are using which PAV devices during the minute that is measured. Each service class is assigned an importance level in WLM policy. The goal algorithm determines the most important service class using a device and ensures that a donor device has a lower importance than or equal importance to the receiver device. Certain special I/Os are ignored for the purposes of creating this mapping so that the mapping represents only application-level I/O activity.

The automatic management of alias devices by WLM makes the most efficient use of this valuable resource in ESS under changing workload conditions. Letting WLM manage aliases saves the installation staff from having to continually make manual alias adjustments in order to match resources to work requirements. Because WLM takes into account the business priority of the work in the z/OS sysplex, it can make alias adjustments to ensure that business-critical work gets the resources that it needs to meet its goals, and that overall IOS queuing is minimized.

2.4 Transactions and management

This section describes how WLM handles and manages different units of work.

2.4.1 How WLM treats work in the system

What we can see from Figure 2-7 on page 63 is that a client work request spans across multiple address spaces on z/OS. In order to manage these requests, you must recognize the transaction flow and, based on this, manage the address spaces and requests to meet the user's expectation. WLM developed a set of programming interfaces that allows middleware and subsystems to identify work requests to WLM in order to help WLM to trace the work through the system and to identify the time the requests stay on z/OS. Over time, WLM develops various techniques to recognize work on the system and to manage it independently from or together with the address space where the programs run to process them:

- The most basic variant is the *address space*. The start of an address space is known to WLM, and if no more information is available about the work in the address space, WLM can manage it based on its resource consumption and demand. Address spaces are started and classified as started tasks, and the only goals that are applicable for them are execution velocity goals or discretionary work.
- *Batch work* reuses already started address spaces. These are called *initiators,* and they select work from the Job Entry Subsystems (JES). The beginning and end of each new batch job is signaled through a system interface to WLM. This allows WLM to recognize its duration, to provide classification rules for them, and to manage the initiators based on the currently active job. Batch work can be managed through execution velocity and response time goals, as well as discretionary work. Furthermore, WLM is able to start the initiators on demand based on the goals for the batch service classes and the batch jobs waiting on the JES input queues.
- A similar technique is used for *Advanced Program to Program Communication (APPC)*. APPC also has initiators available in the system, which receive incoming work and the initiators tell WLM when a new work request begins and ends.
- *TSO* uses a different technique. After a TSO user address space has been started, it waits on input from the TSO user terminal. When the user presses Enter, a new TSO transaction starts and its end is defined when the result is returned to the terminal. The beginning and end of the TSO transaction is signaled to WLM so that it is able to recognize the individual transactions and to manage the TSO address spaces based on them. Unix System Services (*OMVS*) uses the same interface to inform WLM about the beginning and the end of user requests.
- *CICS* and *IMS* regions register as work managers to WLM and create control information, the performance blocks, for all work that can be executed in parallel in the region. When a work request enters the terminal-owning region (TOR) or IMS Control Region, it is classified and associated with a performance block**.** State conditions are recorded in the performance block during execution, and when the request is passed from a TOR to an application-owning region (AOR), the continuation is maintained and information is provided to WLM to understand the relationship.

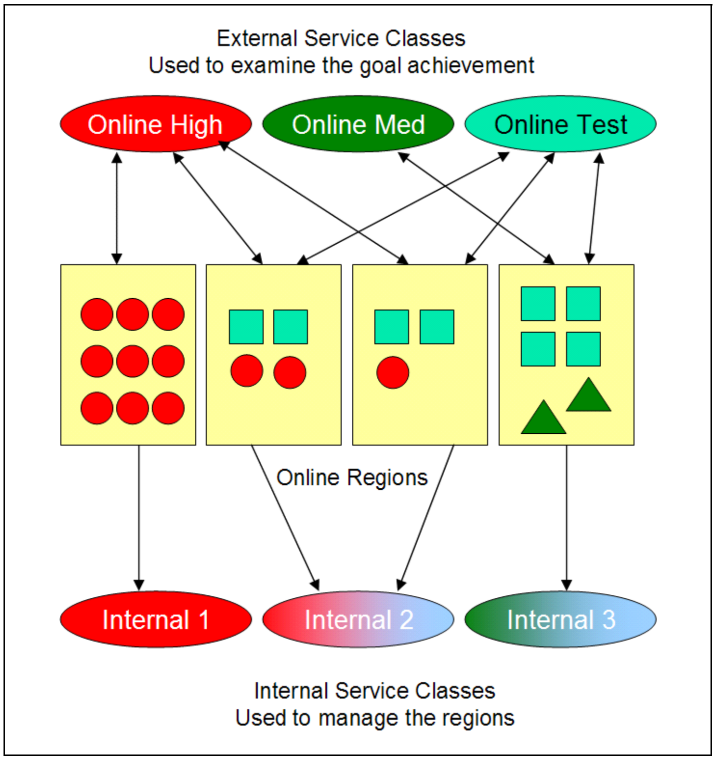


Figure 2-7 Online workloads

WLM continuously collects data from the performance block and captures the beginning and end of all work requests to understand which region processes which type of work. The type of work request is specified through work classification, which associates the work request with a service class. WLM creates a picture of the topology, including the server address spaces and the service classes of the work requests. It uses this topology to manage the server address spaces based on the goal achievement of the service classes for the work requests.

The picture on the left of Figure 2-7 depicts this topology. In the example, there are three external service classes for CICS or IMS work and four registered server regions that process the requests. One region processes only requests for service class Online High (red circles). As a result, one internal service class (ISC) is created, which is used to manage this region. Two regions process transactions depicted as red circles and turquoise rectangles. The second internal service class is used for these two server regions, and the last region is associated with the third internal service class.

Let us assume the external service class Online High misses its goals. Through the topology, WLM finds out that three server regions process work requests of this service class, which are associated with two internal service classes. Then, it calculates how much each of the internal service classes contributes to the goal achievement of the external service class. Let us assume WLM found out that the server regions of the internal class 2 need help. WLM will help the server regions of this service class in order to improve the goal achievement of the external service class Online High.

When we examine this example, we can see that in this case the transactions for Online Med running in the server regions for class Internal 2 also receive help even if they might not need it. This is a side effect of this type of transaction management. The benefit is that the installation can use response time goals for the work requests processed by CICS and IMS.

The example in Figure 2-7 on page 63 also demonstrates that too much granularity in defining service classes might not be helpful.

We have described here the monitoring environment that is available for CICS and IMS.

2.4.2 Enclave management

This section describes the WLM management of enclaves. We see the classification, enclave server, CPU and I/O management, storage management, and reporting aspect of this management.

Classification

For classification:

- All independent enclaves have to be classified (dependent enclaves inherit owning address space's service class).
- The work manager subsystem type determines which rules are searched.
- Failure to classify results in service class SYSOTHER (with a goal of discretionary).
- Any goal type is allowed, but not all make sense under certain conditions.
- Enclave transactions are subject to period switch.

Application Environment enclave server

Enclave servers are DDF, CB, IWEB, and NETV:

- The enclave server address space is managed toward the goal of the enclaves.
- This address space runs in Dynamic Internal service class Period (DISP):
  - The name has the format $SRMS*nnn*.
  - The DISP assignment is based on events such as Join/Leave.
  - DISP represents a group of address spaces serving the same set of external service classes.
- Assumption:
  - No significant CPU service is consumed outside of an enclave.
  - Such CPU service is *not* taken into account when workload management assesses the impact of CPU adjustment of enclave work.

Application Environment enclave server topology

Figure 2-8 on page 65 shows a server topology and the DISP:

- Topology is related to the number of enclave servers and their relation to external service.
- An Enclave server can serve:
  - A single period service class
  - A multi-period service class
  - Multiple service classes

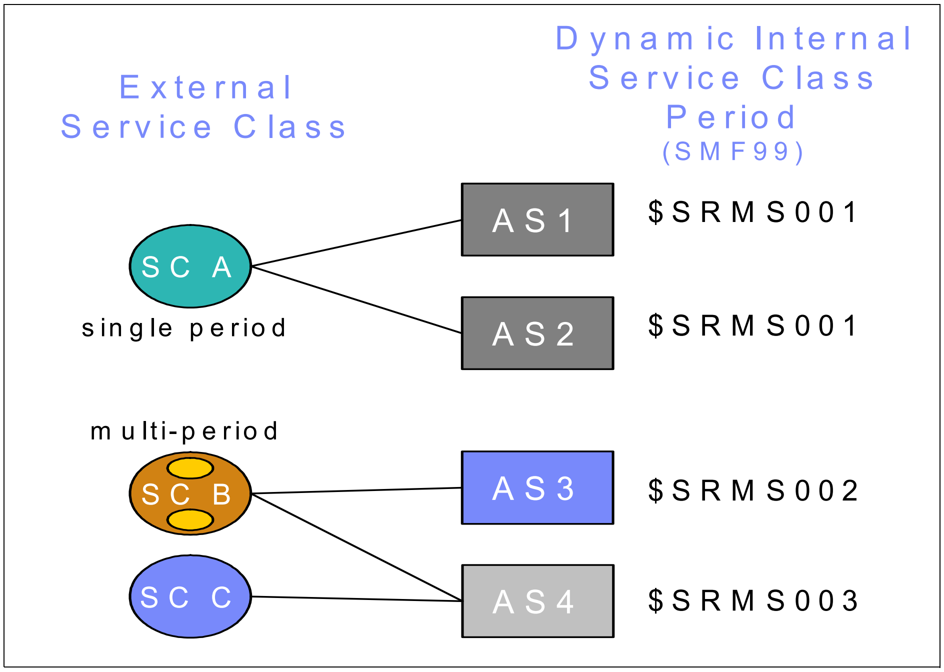


Figure 2-8 Enclave server topology

Note: The more complex the topology, the less sophisticated storage management you can perform.

Enclave exploiters

Enclave exploiters are:

- DB2 Universal Database™ for OS/390 and z/OS (DB2) - Dependent when client
- Distributed Data Facility (DDF) - Independent
- WebSphere HTTP Server for OS/390 (IWEB) - Independent
- LAN Server for MVS (LSFM) - Independent
- WebSphere (CB) - Independent
- MQSeries® Workflow (MQ) - Independent
- NetView® (NETV) - Independent
- Oracle® 9i (OSDI) - Independent
- SAP R/3® (SAP) - Independent

2.4.3 Execution delay monitoring

From the execution delay monitoring services, WLM knows how well work is executing, and where any delays are occurring. The execution delay monitoring services are for complex work manager configurations that process on a single system or across systems in a sysplex, but do not allow MVS to individually manage resource consumption of the transactions.

The services allow MVS to recognize additional address spaces that are processing transactions.

At address space initialization, the work manager address space issues the connect service to establish authorization for subsequent services. It then issues the create service to establish a monitoring environment that keeps track of the execution delays encountered by a work request.

When the execution delay monitoring services are used, z/OS can allocate resources for address spaces based on the behavior of the transactions that are serviced by them. The services also provide detailed execution delay information, so that you can determine where work is delayed. You can then adjust the work manager configuration to more consistently meet the goals.

> **Note:** Only response time goals can be used with execution delay services. If a subsystem needs to use velocity goals, discretionary goals, or period switch, it has to use enclave services instead.

The subsystem work manager uses the execution delay monitoring services to tell workload management about the subsystem work manager's view of the current state of a work request, such as ready state, idle state, or waiting state. The actual state might be different. For example, a work request can be active from the subsystem's view, but might be delayed by a page fault or for CPU access. The information is kept in *performance blocks*, also called *monitoring environments*.

The monitoring environments represent work wherever it executes: across multiple dispatchable units, address spaces, and systems.

Monitoring environment or performance block (PB)

In a monitoring environment or performance block:

- A PB is associated with a transaction, not a service class.
- It is created once by a work manager for each task that it owns.
- It contains transaction start time and transaction state data.
- PBs are sampled periodically by WLM as follows:
  - Every 10th second in regions managed toward *region goal*
  - Every quarter second in regions managed toward *transaction goal*
  - To find out the server topology
  - To collect subsystem work manager delays for reporting only

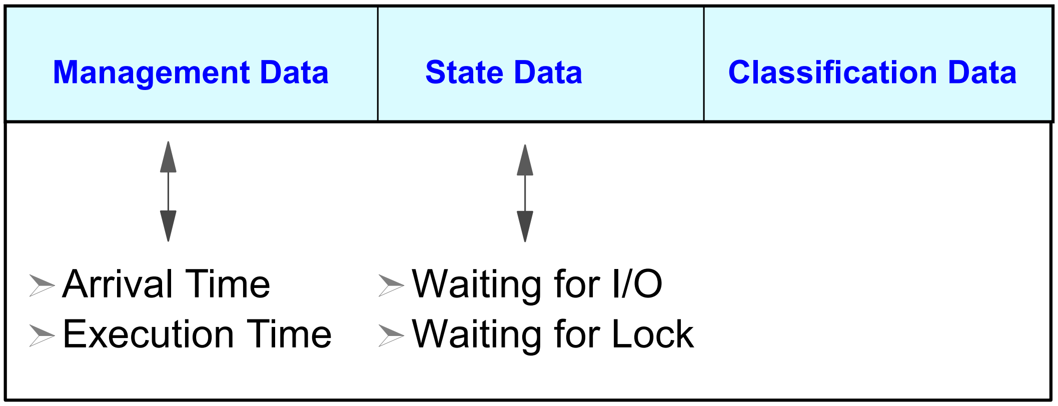


Figure 2-9 Performance block sampling

Services

Figure 2-10 on page 67 depicts the flow of execution delay monitoring services.

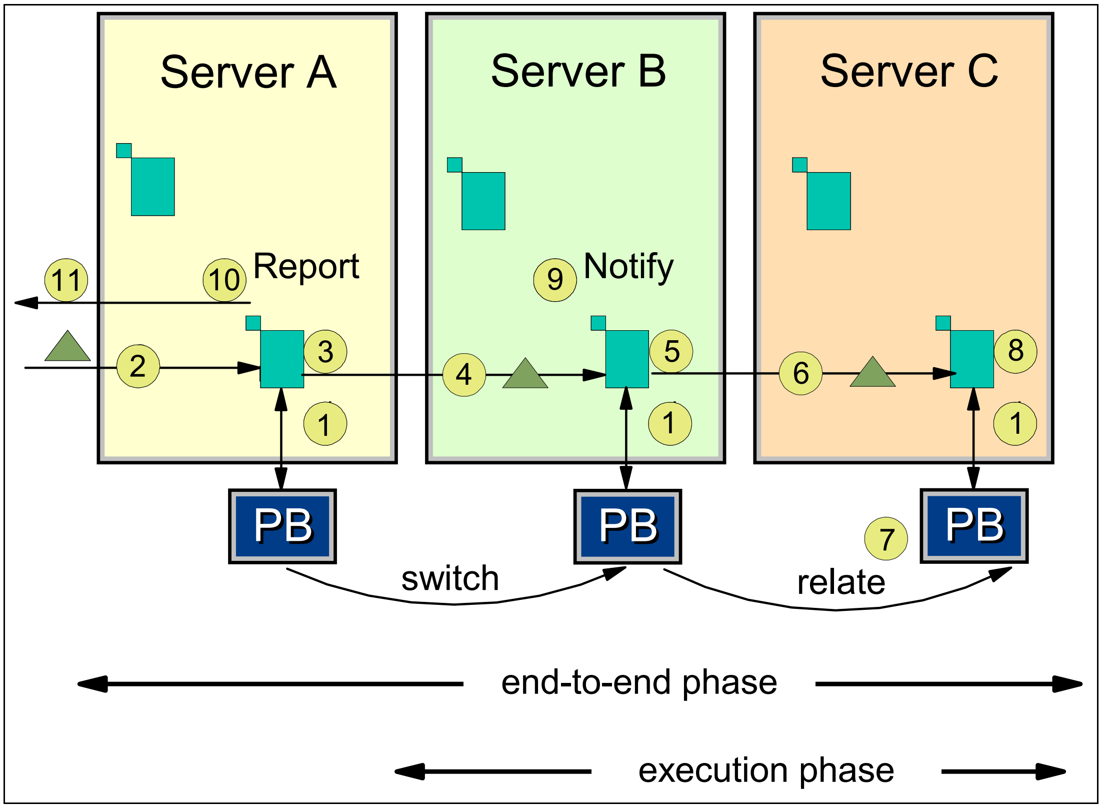


Figure 2-10 Execution delay monitoring services

Where:

- **1** → Work Managers (servers) create a PB for each task.
- **2** → The transaction arrives at the router (server A) and then:
  - Obtains service class
  - Initializes the monitoring environment (PB)
- **3** → The transaction states are recorded in the PB as it flows.
- **4** → Server A passes the transaction to the execution address space (server B) and the switch is recorded in the PB of the task in server A.
- **5** → PB is initialized, and transaction states are recorded.
- **7** → Server C relates the PB (dependent) of the receiving task to the PB (parent) of the sending task in server B.
- **8** → The transaction continues while the states are recorded until the transaction completes.
- **9** → Server B receives the result and notifies the end of the execution to WLM.
- **10** → Server A receives control back from server B and reports the end of the transaction to WLM.
- **11** → The results are sent to the originator.

Server topology

Server topology is an algorithm used to track, in a multiple address space subsystem work manager, the distribution of the transactions, and their service classes across these multiple address spaces. The priorities and resource management of these address spaces are decided by this distribution of the transactions and the goals of the transactions. Server topology relies on the information passed by the subsystem work manager through the use of WLM services interfaces.

The server topology is only implemented for CICS and IMS transaction response time goals. Other subsystems use enclave transactions instead.

A positive aspect of server topology is that it is completely installation-transparent, that is, you do not need to tell WLM in which address spaces your CICS or IMS transaction runs. The locality of the transaction is communicated to WLM by the subsystems.

The CICS and IMS address spaces (regions) are called *server address spaces*.

When any address spaces start, WLM views them as nonservers. Later, an address space is recognized as a server if it uses certain WLM services, such as:

- Either a report (IWMRPT) or notify (IWMMNTFY) for a transaction.
- Services associated with the use of a performance block (PB) that represents a transaction. The association was done via execution delay monitoring services.

When does an address space becomes a server

An address space becomes a server when:

- A PB exists that represents a transaction. This is done via WLM execution monitoring services.
- The address space notifies WLM about the end of transaction execution or reports the transaction completion to WLM.

Dynamic Internal Service Class Period (DISP)

For each set of servers found serving the same set of external service classes, an internal service class period (DISP) is created and assigned to them.

They are dynamically created based on sampling and called $SRM*nnn*. They are not visible externally (except for SMF99 trace data).

The number of DISPs depends on the number of external service classes and the combination of server address spaces.

Address spaces must belong to the same Resource Group (if any).

DISPs are rebuilt potentially once per minute.

Figure 2-11 on page 69 shows an example of three external service classes and three DISPs associated with this particular server topology.

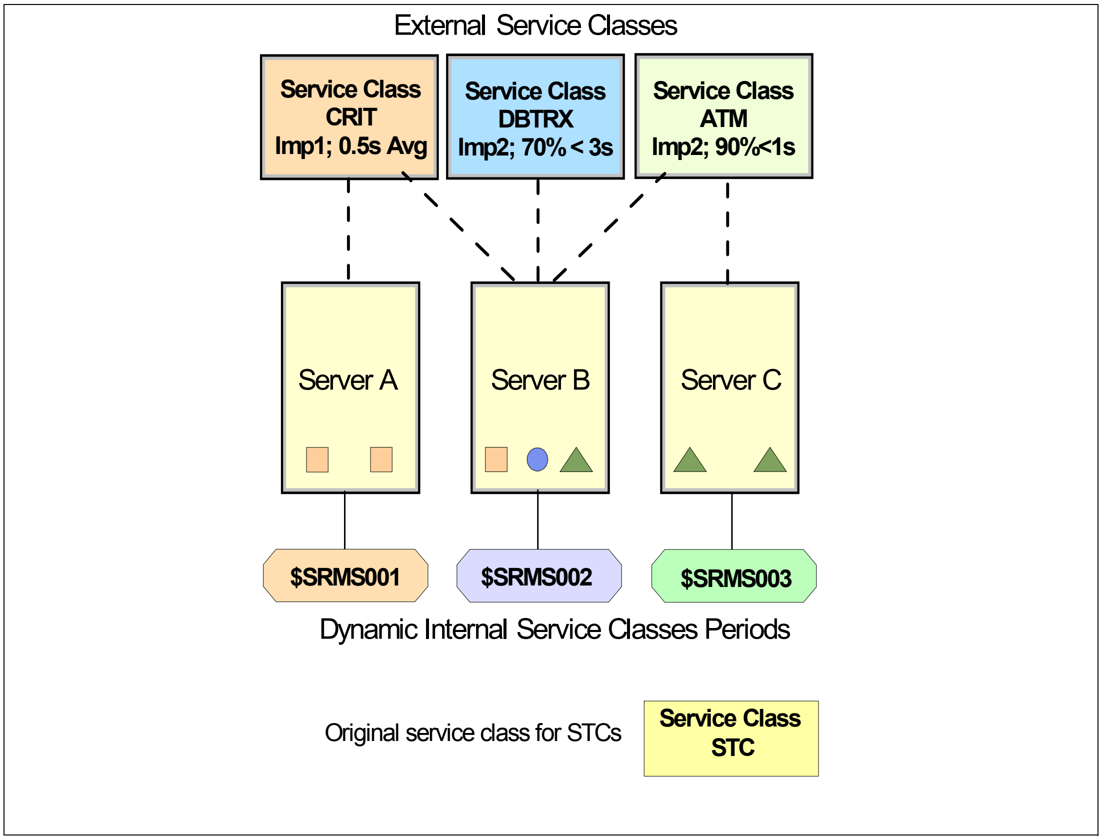


Figure 2-11 Dynamic Internal service class (DISP)

In order to evaluate the server topology, WLM must first understand which external service classes are served by each server. This is accomplished by WLM maintaining a served class history for each server address space.

Clients define service classes for transactions to be processed by server address spaces. These service classes are referred to as *external service classes (ESCs)*, and they have real client-defined goals. WLM groups some server address spaces into *internal* service classes. Some internal service classes are called *dynamic* when they are associated with server address spaces (such as CICS) serving the transactions with the external service classes. These dynamic internal service classes are used to manage to the goals of the external service classes. For example, assume that the WLM wants to help a given service class representing some CICS transactions. WLM must know which address spaces must be given more resources because the CICS transactions in that service class run in multiple address spaces. A DISP is the set of address spaces that serve a given set of ESCs, and it is the set of address spaces in the DISP that will be given the resources.

Server topology is the function used to create the DISP and map it to the external service classes representing transactions with goals.

Server topology determination

The following is an example of the methodology that is used to determine the server topology.

Imagine that address space server B was found servicing service classes CRIT, DBTRX, and ATM. Address space server A was found servicing service class CRIT. Address space server C is serving service classes ATM. The following list shows which server is serving what:

```
Server A {CRIT}

Server B {CRIT, DBTRX, ATM}

Server C {ATM}

Server D {CRIT}
```

To find the minimum number of dynamic internal service classes, it must be determined for each server address space set all other server address space sets that are equal (that is, those address spaces that serve the same set of external service classes, and that belong to the same Resource Group). For this example, this exercise would result in the following dynamic internal service classes getting built:

- DISP $SRMS001 - Server A+D
- DISP $SRMS002 - Server B
- DISP $SRMS002 - Server C

The topology is reassessed and the DISPs potentially rebuilt once per minute, based on a full PB sampling for topology or on the topology deduced from report and notify calls.

The dynamic internal service classes are used as accumulators of data (delay counters, for example) for managing servers. Because the same address space can be serving multiple external service classes, the counts are used to determine a weighting factor of not only which server is serving what, but also how much. This is important to know, because if your transaction does not reach its goal, WLM has to decide what the major delay for this transaction is and which address spaces get more resources.

Refer to Figure 2-11 on page 69 for an example of a server topology. There is an ESC for each service class representing the CICS transactions. If WLM wants to help the ESC for CICS transaction Type DBTRX in this figure, then the only address space that can be helped is the DISP representing server B. But, if WLM wanted to help CICS transaction Type CRIT, then the weighting factors would be used to determine whether to help the server A address space (DISP $SRMS001) or server B address space (DISP $SRMS002).

Server requiring help determination

In order to determine which server to help, WLM calculates a “weight” for all servers potentially targeted for this help.

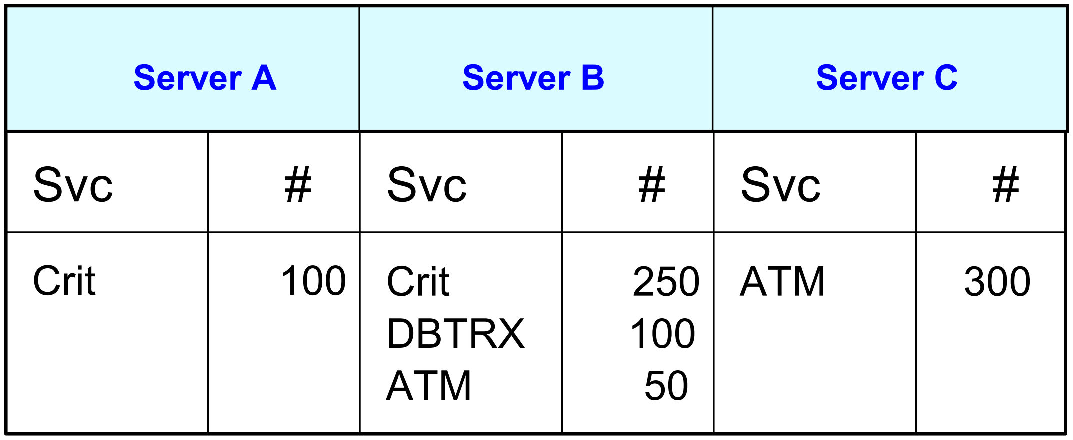
The weight is calculated according to the different delays that the servers are experiencing. The server with the biggest weight will be helped.

Let us evaluate this example based on Figure 2-11 on page 69.

Figure 2-12 on page 71 shows the ESC and the number of transactions per server:

- CRIT is missing its goal.
- CRIT is served by A and B.
- Based on PB-sampling:
  - A's share is about 30% computed as 100 / (250+100).
  - B's share is about 70%.

- WLM examines delays in $SRMS001 and $SRMS002 and might come to the conclusion that $SRMS002 needs help.
- Transactions in DBTRX and ATM will benefit from additional resources assigned to B.



Server A		Server B		Server C	
Svc	#	Svc	#	Svc	#
Crit	100	Crit DBTRX ATM	250 100 50	ATM	300

Figure 2-12 DISP scenario

Server management

WLM uses samples within the internal service class periods to determine how well work is progressing.

The servers are managed based on the PI of the transaction service classes served.

WLM sets dispatch priority and access to other resources for the server:

- WLM is unaware of the single transactions.
- WLM does not assign computing resources to transactions.

The external service class of a server is ignored, unless:

- Management toward the region-goal was specified in the classification rules for the server.
- The server is not serving any transaction (during start/stop or periods of longer inactivity).
- Dependent enclaves of type MONENV are used (DB2 Stored Procedures called by CICS).

Transaction phase

Figure 2-13 on page 72 shows transaction phases, for example, for a CICS transaction using DB2.

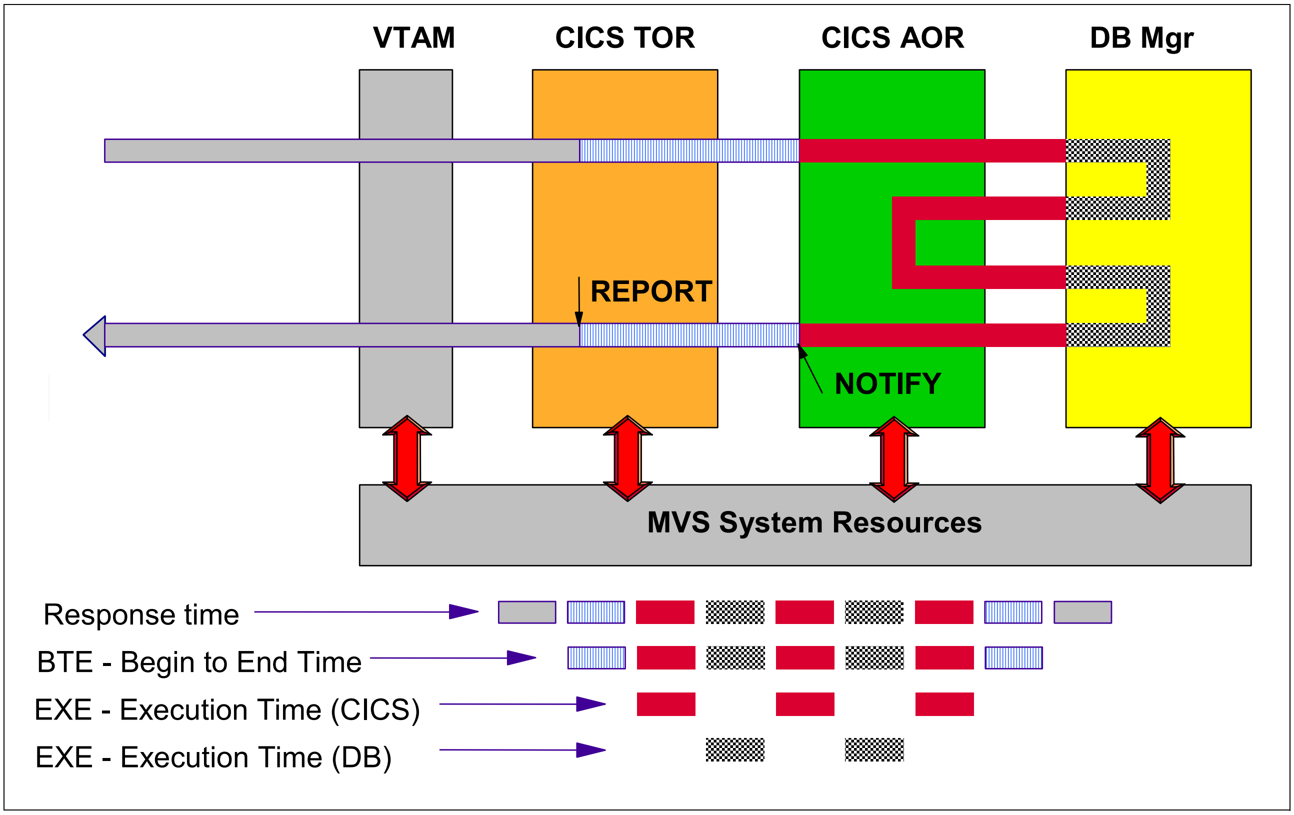


Figure 2-13 Transaction phases for a CICS transaction

The Begin to End (BTE) phase starts when a transaction starts in a terminal-owning region (TOR) and ends when the TOR receives it back from an application-owning region (AOR), finishing off. Its associated PB bits show TOR states along transaction execution.

BTE phase duration is roughly response time minus TOR queue time.

The EXEC phase PB has details about the different steps of the transaction out of the TOR. There are two types of EXEC phase:

- EXEC1, when the transaction is in the execution AS (AOR or IMS/MPR). It is roughly the AOR/MPR transaction service time. It is not contiguous; it is interrupted when the database AS is executed. Along AOR execution, there are two active and sampled PBs: the BTE (with the delay-for-conversation bit on) and the EXEC1 phase describing the states of the AOR execution.
- EXEC2, when the execution is done by the supporting database AS, such as DB2 or DL1. It is contiguous in time. Along with database execution are two active PBs: the BTE with (the delay-for-conversation bit on), and the EXEC2 phase describing the states of the database execution.

Example 2-6 on page 73 shows an example on a RMF report of a CICS transaction using DB2 and DBCTL (IMS) database managers.

Example 2-6   RMF Reporting on CICS DB2 IMS

```
REPORT BY: POLICY=ITSOPOL1 WORKLOAD=ONLPRD     SERVICE CLASS=ONLPRD     RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT
   AVG      0.00   ACTUAL                  90
   MPL      0.00   EXECUTION               89
   ENDED  101250   QUEUED                   0
   END/S   56.25   R/S AFFINITY             0
   £SWAPS      0   INELIGIBLE               0
   EXCTD    1110   CONVERSION               0
   AVG ENC  0.00   STD DEV                808
   REM ENC  0.00
   MS ENC   0.00

              RESP  -------------------------------- STATE SAMPLES BREAKDOWN (%) -------------------------------  ------STATE------
   SUB    P   TIME  --ACTIVE-- READY IDLE  -----------------------------WAITING FOR-----------------------------  SWITCHED SAMPL(%)
   TYPE        (%)   SUB  APPL             MISC PROD LOCK CONV  I/O LTCH                                          LOCAL SYSPL REMOT
   CICS  BTE  55.3  16.4   0.0  19.3  3.1   1.3 37.9 11.7  7.9  2.3  0.0                                            8.0   0.0   0.0
   CICS  EXE   1.1  20.5   0.0  17.3  0.5   0.7 52.0  0.0  3.9  5.1  0.0                                            980   0.0   0.0
   DB2   BTE  10.4  36.5   0.0   0.0  0.0  61.9  0.0  0.2  0.0  0.7  0.6                                            0.0   0.0   0.0
   DB2   EXE   0.6  29.6   0.0   0.0  0.0  68.7  0.0  0.0  0.0  0.9  0.9                                            0.0   0.0   0.0
   IMS   BTE   1.8   100   0.0   0.0  0.0   0.0  0.0  0.2  0.0  0.0  0.0                                            0.0   0.0   0.0
   IMS   EXE   0.0   100   0.0   0.0  0.0   0.0  0.0  0.0  0.0  0.0  0.0                                            0.0   0.0   0.0
```

ACTUAL RESPONSE TIME = 0.090 seconds = 100%

Note: The frequency of the RMF interval and the state and response time sampling can potentially cause these situations. If a transaction starts at RMF interval 1 but ends at RMF interval 2, the state sample is included in RMF interval 1 but the response time will not be included. In RMF interval 2 (when the transaction ends), the response time will be included but not the state sample. Never-ending tasks generate only state samples.

APAR OW52227 addresses this problem by showing the single states as percentages of total state samples instead of response time. This eliminates percentages >100 in the BREAKDOWN section. Thus, the RESPONSE TIME BREAKDOWN is replaced by a STATE SAMPLE BREAKDOWN. Accordingly, the STATE SWITCHED TIME is replaced by a STATE SWITCHED SAMPL(%) breakdown. Only the TOTAL column will be kept as percentage of response time, which is the current view. However, the field name will be changed to RESP TIME(%). Before this APAR, state samples were reported as a percentage of response time. The response time is calculated when a transaction completes. This could result in values >100% when samples for long-running transactions were included that did not complete in the RMF interval.

For example, RMF has information covering 93.4% of the response time of a subsystem BTE phase. Thus, the RESP TIME(%) field has a value of 93.4. Of the total, 89.2% state samples are of type waiting_for_conversion. Thus, CONV is reported with 89.2. This value accounts for 83.3% of the response time, because 100% of the total state samples correspond 93.4% of the response time (89.2 x 93.4 / 100 = 83.3). Before OW52227, CONV would have been reported with 83.3%.

Summary

WLM provides the ability to classify transactional work at either the region or transaction level.

WLM provides cross-subsystem management and reporting of transactions throughout system and sysplex.

WLM knowledge of server topology and WLM goals allow dynamic on-going tuning.

RMF provides an enhanced reporting of work manager delays.

The limitations compared to enclaves are:

- Only single period service classes with response time goals are allowed for transactions.
- No MVS resource accounting for transactions. Work manager specific means are required.
- All management is on a server basis.

  Resource adjustments for the server affect all transactions (in all service classes) executing in it.

3

WLM functions

This chapter describes the following Workload Manager (WLM) functions:

- Protecting the work
- Management of server address spaces
- Sysplex routing
- Intelligent Resource Director
- Management of specialty processors

3.1 Protecting work

WLM offers several features to protect specific work from being hurt by other work's behavior. These features apply to resources such as CPU, storage, and initiator controls.

3.1.1 Resource Groups

Resource Groups can be used to define a minimum and maximum amount of CPU service for particular work. Work is assigned to a Resource Group by specifying the Resource Group in the service class definition. A Resource Group can include multiple service classes, but a service class can only be assigned to one Resource Group.

The maximum capacity of a Resource Group is enforced. WLM takes various actions to stop work in a Resource Group from exceeding the capacity, including swapping, changing dispatching priorities, and capping the CPU consumption.

The minimum capacity is only used when work in the Resource Group is not meeting its goals. If work in a Resource Group is not meeting its goals, WLM will attempt to provide the minimum service defined in the Resource Group, even if that causes more important work to miss its goals.

Normally, you should avoid using Resource Groups. Try to set appropriate goals and allow WLM to manage the resources. Resource Groups restrict the ability of WLM to make changes to meet the goals you have defined. However, they are necessary in some circumstances. Your workload might have some characteristics that mean that normal WLM management does not work well. In that case, Resource Groups can be useful to ensure that work receives the service that it needs.

Types of Resource Groups

There are three types of Resource Groups:

Type 1 Capacity is specified in unweighted CPU service units per second (SU/s). The maximum and minimum limits have *sysplex* scope.

Type 2 Capacity is specified as a percentage of logical partition (LPAR) capacity from 0-99. The sum of all Resource Group minimums should not exceed 99. The maximum and minimum limits have *system* scope.

Type 3 Capacity is specified as a percentage of a single general purpose CP. 100 represents the capacity of one CPU. The number can range from 0 to 999999, but the sum of all Resource Group minimums should not exceed the number of processors x 100. The maximum and minimum limits have *system* scope.

Resource Group types 2 and 3 were introduced in z/OS release 1.8. Prior to that, only Resource Group type 1 was available.

Resource Group type 1 has sysplex scope; therefore, the limits will be applied based on the service those workloads receive across the entire sysplex. There are no guarantees about the amount of service that the workload achieves on a particular system. It is possible that a service class might receive most or all of its capacity limit on one system, and very little service on other systems.

Resource Group types 2 and 3 have system scope, which means that WLM applies the limits to each system independently. The amount of service consumed in the Resource Group on one system does not influence the amount of service that can be provided on the other systems in the sysplex.

Up to 32 Resource Groups can be defined in a service definition.

Defining Resource Groups

Resource Groups have a name and optionally a description.

Resource Group parameters are:

- Define capacity:

  Type 1 In service units (sysplex scope)

  Type 2 As a percentage of the LPAR share (system scope)

  Type 3 As a number of CPs times 100 (system scope)

- Minimum capacity

  Defines the minimum amount of service for service classes in this Resource Group.

  Service classes, which are not meeting their goals in Resource Groups, that are consuming less than their minimum get higher preference when new dispatch priorities are calculated, even if that causes higher importance work to miss its goals.

  Note: Minimum capacity is ignored when the work is meeting its goal.

- Maximum capacity

  Defines the maximum amount of service that the service classes are allowed to use.

  This can be accomplished by enforcing swapping, adjusting dispatch priorities, or switching the work in the Resource Groups to non-dispatchable in order to reduce CPU consumption.

Defining capacity in service units

Type 1 Resource Groups are defined in unweighted service units per second. The capacity limits apply to CPU used in both TCB and service request block (SRB) mode.

Type 1 Resource Groups define service limits across the entire sysplex. They can control how much service the workloads receive in total, but do not control how much service is received on each system in the sysplex.

Service units are intended to be a processor independent measure of service, so SU/s values required by a particular workload should remain reasonably consistent (but not necessarily the same) across hardware changes. The equivalent CPU percentages will change. If you want to keep the same percentage of the system across a hardware migration, you will need to calculate new SU/s values.

The service units per second that can be delivered by a CPU can be found on the SRM site at:

http://www.ibm.com/servers/eserver/zseries/srm/

The value to use depends on the number of CPUs in the LPAR, not the number in the central electrical complex (CEC). The exact value varies in order to account for the multiprocessor (MP) factor. This value is also reported on the Policy section of the RMF Workload Activity Report.

Defining capacity as a percentage of the LPAR share

Type 2 Resource Groups are defined as a percentage of the LPAR share. The LPAR share is normally the LPAR weight divided by the sum of all LPAR weights on the CEC. If the LPAR capacity changes, for example, due to soft capping, the capacity of the Resource Group also changes.

The value should be from 0-99, and the sum of all Resource Group minimums should not exceed 99.

Type 2 Resource Group limits are applied per system. If one LPAR in the sysplex has twice the capacity of another, its Resource Group service limits will also be double.

Type 2 Resource Groups will change capacity (measured in SU/s) with hardware changes, but will be the same percentage of the system.

Defining capacity as a number of CPs times 100

Type 3 Resource Groups are defined as a number of CPs times 100. This is equivalent to a percentage of a single CP. Values can be greater than 100 if the system has multiple CPUs.

The limits are applied per system. The Resource Group capacity in SU/s will change with hardware changes.

> **Note:** You cannot assign a Resource Group to service classes representing transaction-oriented work, such as CICS or IMS transactions. The ISPF application notifies you with an error message if you attempt to define one.

Resource Group usage consideration

Type 1 Resource Groups are sysplex-wide and can contain multiple service classes. Problems can occur when these service classes have different importances and when the sysplex hardware is different (for example, 9672 and z990).

The minimum service consumption and maximum service consumption of type 1 Resource Groups are sysplex-wide, but work in the service classes does not necessarily run equally on all systems. In most cases, it does not, so it is possible that work in a Resource Group consumes many CPU cycles on one system while it does not consume a lot on other systems:

- WLM will always try to balance the service consumption but there is no guarantee to it.
- In the end, each system must decide on the service that it gives to the work in the Resource Groups.
- For that reason, the systems exchange information about the demand and service consumption of the work in Resource Groups on other systems.
- Given the combination of the goals, the importance level, and the Resource Group capacity, some goals might not be achievable when capacity is restricted. The RMF Workload Activity Report shows you the service class delay because of resource capping.

Resource Group maximum management

The purpose of the capping function is to control the amount of CPU service rate that dispatchable units in a set of address spaces (or enclaves) in a Resource Group consume. Policy adjustment code forces this objective by measuring the CPU service rate consumed by the group (locally and in the sysplex). The CPU service rate values are accumulated first on local systems and then across the sysplex for a total. The total value of the consumed service rate is used to determine if it exceeds the Resource Group maximum service rate. This comparison then determines how much to throttle address spaces in the group. This throttling is done by limiting the dispatchability of the address spaces or enclaves in the Resource Group.

Capping pattern

The capping pattern determines the number of slices when service is given to work and the remaining slices when work is capped. There are 64 slices per second. These slices apply to elapsed time.

The control is done at the Resource Group level and not at the service class level. All address spaces or enclaves in all service classes in a Resource Group are controlled by the same number of cap slices. Meanwhile, the dispatching priority assigned to each service class period is still based on the goals. Therefore, work in the Resource Group with a more stringent goal will more likely run when the group is not capped.

Dispatchable units from address spaces or enclaves belonging to a Resource Group are made non-dispatchable during some time slices (cap slices) in order to reduce access to the CPU to enforce the Resource Group maximum. The time slice where they are set as dispatchable is an *awake slice*.

Because two groups can accumulate service units at different rates, each time slice for a group is set in proportion with a previously measured average CPU service rate that the Resource Group collects. Therefore, by knowing how much above the maximum a Resource Group is, it is simple to derive the number of cap slices that the address spaces in the group are going to get during every elapse of the 64 time slices.

Because the cap slices for different groups are evenly spread across the time slices and the time slices are of relatively short duration, the changes in dispatchability do not appear as abrupt service changes.

All address spaces or enclaves in a Resource Group on each system are made dispatchable for the same proportion of time and during the same intervals. Address spaces in the same group but on different systems can be made dispatchable for different proportions of time and during different intervals. The proportion of time that address spaces are made dispatchable on different systems is based on the importance and quantity of work in the group running on each system.

Work on the system that has more of the group's higher importance work can be made dispatchable for a larger proportion of time than lower importance work in the same group on another system. The intent is to equalize performance indexes for service class periods of the same importance within the Resource Group constraints across the sysplex.

Capping delay is treated as another form of processor delay when managing dispatching priorities to meet goals. Consequently, service class periods within a capped Resource Group can have their dispatch priorities increased to address capping delay as well as processor delay.

Every 10 seconds (the policy adjustment interval time), all Resource Groups are reappraised to determine if further adjustment is necessary. If so, the times that groups are to be set dispatchable or non-dispatchable are reevaluated. The 64 time slices and the cap slices are then reassigned.

Note: Discretionary goal management is a part of a Resource Group (a dynamic Resource Group) that uses only the MAXIMUM function of Resource Group capability.

Service ramps up

There can be a delay between the time service class starts working and the time Resource Group capping is enforced.

This example shows this phenomenon:

- Environment studied:
  - Single system z900-2C4 that can deliver 4 x 13082.58 = 52330.32 SU/s.
  - Some system workload.
  - Resource Group maximum set at 3000 SU/s.
  - No work was running during the last minute.
- Work arrives to consumes up to 10000 SU/s:
  - The workload consumes fewer SU/s in the beginning due to some I/O, paging, and other environmental setup actions.
  - When the workload starts, there is no historical data available, so the workload runs 10 seconds without any capping (the 10 seconds are the regular WLM policy adjustment time). So, the capping cannot be done at the correct point (initial start of the workload).
  - This “warm up” time can take several minutes before WLM caps with the right amount of cap slices. Figure 3-1 shows graphically what is happening. During the swing-in, service consumption can be as high as twice the maximum specified in the maximum value of the Resource Group definition.

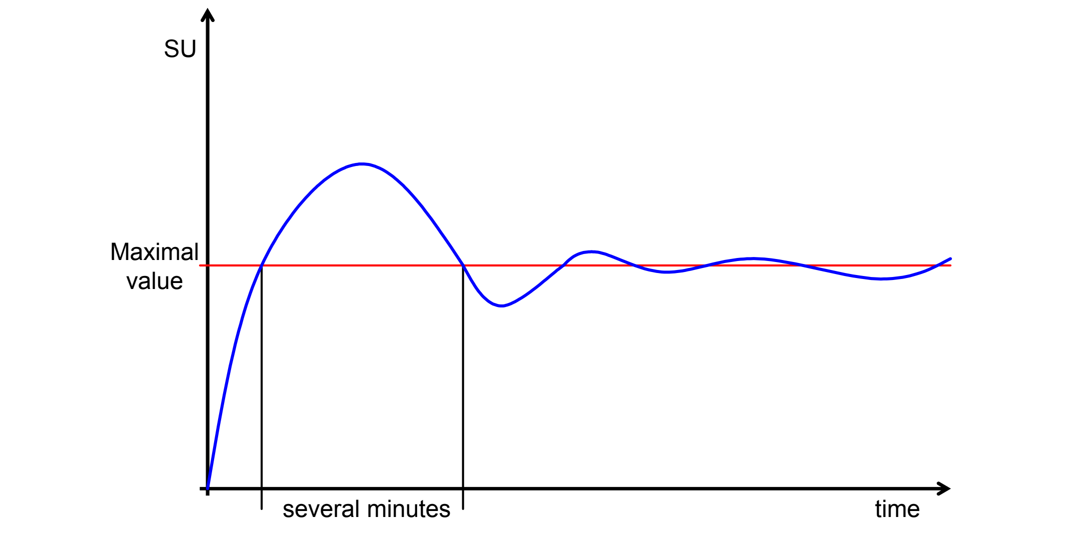


Figure 3-1 Resource Group service ramp up

Awake slice calculation

The capping is in effect only if the service class has consumed more than the maximum set in the Resource Group definition.

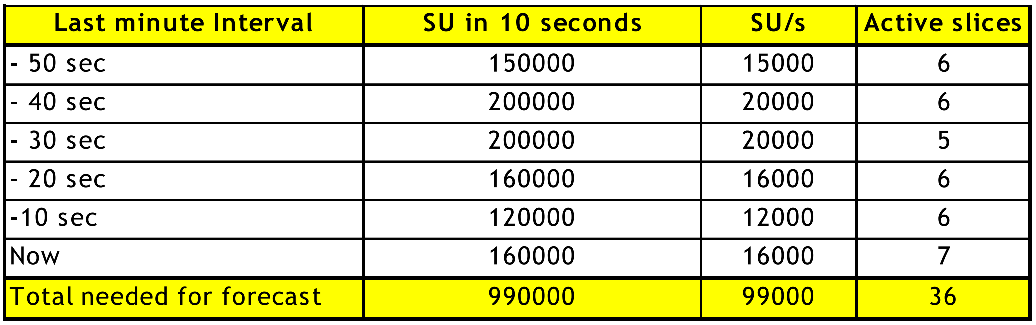
An *awake slice* is a slice where the address space will be marked as dispatchable.

The awake slice count calculation is done every 10 seconds.

When a Resource Group caps one or more service classes, the service units consumption (SU) and the amount of cap slices (CAP) in the last minute are used to forecast what will happen for the next 10 seconds:

- CAPawakeSlice = SU / CAP
- SLICES = MAX value / CAPawakeSlice

Figure 3-2 shows the SU/s consumption during the last minute in ranges of 10 seconds and is used in our calculation example.



Last minute Interval	SU in 10 seconds	SU/s	Active slices
- 50 sec	150000	15000	6
- 40 sec	200000	20000	6
- 30 sec	200000	20000	5
- 20 sec	160000	16000	6
-10 sec	120000	12000	6
Now	160000	16000	7
Total needed for forecast	990000	99000	36

Figure 3-2 Awake slice calculation

The MAX value contains SU/sec, so in *1 minute* you can consume *60 * MAX value*.

Suppose that you set a *MAX value of 16000 SU/s* in the Resource Group definition.

If in the last minute (as shown in Figure 3-2), you have consumed 990000 SU with 36 CAP, the awake slice is 99000(SU/s) / 36(CAP) = 2750 SU/s.

The number of slices for the next 10 seconds is 16000(MAX SU/s) / 2750(SU/s per awake slice) = 5.81 slices.

So SRM uses 5 slices in the next 10 seconds.

RMF reporting

Example 3-1 shows that service class BATPIER has a Resource Group RGPIER.

The capping accounts for 75.8% of the delay samples as reported in the CAPP field.

A TCB or SRB is marked non-dispatchable:

- Because a Resource Group maximum is being enforced.
- Or because of discretionary goal management. That is, if certain types of work are over achieving their goals, that work might be capped so that the resources can be diverted to run discretionary work.

Example 3-1 Resource Group capping in RMF report

```
REPORT BY: POLICY=WLMPOL     WORKLOAD=BAT_WKL    SERVICE CLASS=BATPIER    RESOURCE GROUP=RGPIER     PERIOD=1 IMPORTANCE=4
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
   AVG      8.41   ACTUAL           21.11.012   SSCHRT  81.1   IOC     24100    ABSRPTN      2059   SINGLE     0.0   AVG     3901.37
   MPL      8.41   EXECUTION         9.55.970   RESP     1.0   CPU      1056K   TRX SERV     2059   BLOCK      0.0   TOTAL   32795.2
   ENDED       6   QUEUED               3.078   CONN     0.6   MSO      9287K   TCB          48.3   SHARED     0.0   CENTRAL 32795.2
   END/S    0.01   R/S AFFINITY             0   DISC     0.0   SRB     18827    SRB           0.9   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE       11.11.962   Q+PEND   0.3   TOT     10386K   RCT           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION             349   IOSQ     0.1   /SEC    17310    IIT           0.4   EXP SNGL   0.0   SHARED     0.00
   AVG ENC  0.00   STD DEV                  0                                   HST           0.0   EXP BLK    0.0
   REM ENC  0.00                                                                APPL %        8.3   EXP SHR    0.0
   MS ENC   0.00
```

```
VELOCITY MIGRATION:   I/O MGMT   2.1%    INIT MGMT  2.1%

       ---RESPONSE TIME---  EX   PERF  AVG   --USING%- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
       HH.MM.SS.TTT        VEL  INDX ADRSP   CPU  I/O  TOT CAPP  CPU  I/O                      UNKN IDLE  USG  DLY  USG  DLY QUIE
GOAL                      30.0%
ACTUALS
SYS1                       2.1% 14.3   2.8   1.0  1.0 91.1 75.8 14.8  0.4                       6.9  0.0  0.0  0.0  0.0  0.0  0.0
```

Resource Group usage examples

Next we discuss examples of where your installation might benefit by using Resource Groups.

Protection of low goal service class from capping

If you have a service class which is overachieving a low goal, it can be subject to WLM capping to provide more service to discretionary work. To protect it from capping, and according to the WLM rules defined in “Selecting a receiver candidate” on page 45, you can assign a Resource Group without a minimum and maximum, and capping will not take place.

A low goal means:

- EXECUTION VELOCITY Goal is less than or equal to 30%.
- RESPONSE TIME Goal is longer than 1 minute.
- Any Importance level.

Example 3-2 shows service class BATPIER with a VELOCITY GOAL of 30%.

Example 3-2   Service class initial setting

```
REPORT BY: POLICY=WLMPOL     WORKLOAD=BAT_WKL    SERVICE CLASS=BATPIER    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=3
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--    ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
   AVG      2.97   ACTUAL           1.45.029   SSCHRT 899.2   IOC    223323   ABSRPTN     58154   SINGLE     0.0   AVG     3966.86
   MPL      2.97   EXECUTION        1.44.561   RESP     0.9   CPU      8820K  TRX SERV    58154   BLOCK      0.0   TOTAL  11780.8
   ENDED      12   QUEUED                468   CONN     0.6   MSO     77155K  TCB         425.0   SHARED     0.0   CENTRAL 11780.8
   END/S    0.02   R/S AFFINITY            0   DISC     0.0   SRB    153458   SRB           7.4   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE              0   Q+PEND   0.3   TOT     86352K  RCT           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION            262   IOSQ     0.0   /SEC   172707   IIT           3.1   EXP SNGL   0.0   SHARED     0.00
   AVG ENC  0.00   STD DEV            50.588                                  HST           0.0   EXP BLK    0.0
   REM ENC  0.00                                                              APPL %       87.1   EXP SHR    0.0
   MS ENC   0.00

   VELOCITY MIGRATION:   I/O MGMT  51.1%    INIT MGMT 51.0%

          ---RESPONSE TIME---  EX   PERF  AVG   --USING%- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
          HH.MM.SS.TTT        VEL  INDX ADRSP   CPU  I/O  TOT CAPP  CPU  I/O                      UNKN IDLE  USG  DLY  USG  DLY QUIE
   GOAL                      30.0%
   ACTUALS
   ITS1                      51.1%  0.6   1.0  20.1 21.7 40.0 18.0 15.4  6.6                      18.1  0.0  0.0  0.0  0.0  0.0  0.0
```

We see that service class BATPIER has 18% of capping delays due to its low VELOCITY goal of 30%. We do not want that, and we add a Resource Group RGPIER to fix it. This is shown in Example 3-3.

Example 3-3   Adding a Resource Group to a low goal service class

```
REPORT BY: POLICY=WLMPOL     WORKLOAD=BAT_WKL    SERVICE CLASS=BATPIER    RESOURCE GROUP=RGPIER     PERIOD=1 IMPORTANCE=3
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--    ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
   AVG      2.95   ACTUAL           1.32.499   SSCHRT  1109   IOC    267176   ABSRPTN     71780   SINGLE     0.0   AVG     3959.19
   MPL      2.95   EXECUTION        1.31.951   RESP     0.9   CPU     10507K  TRX SERV    71780   BLOCK      0.0   TOTAL  11696.3
   ENDED      15   QUEUED                547   CONN     0.6   MSO     91877K  TCB         506.3   SHARED     0.0   CENTRAL 11696.3
   END/S    0.03   R/S AFFINITY            0   DISC     0.0   SRB    189538   SRB           9.1   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE              0   Q+PEND   0.3   TOT    102840K  RCT           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION            139   IOSQ     0.0   /SEC   212328   IIT           3.7   EXP SNGL   0.0   SHARED     0.00
   AVG ENC  0.00   STD DEV            28.890                                  HST           0.0   EXP BLK    0.0
   REM ENC  0.00                                                              APPL %      107.0   EXP SHR    0.0
```

```
MS ENC   0.00

VELOCITY MIGRATION:   I/O MGMT  65.0%    INIT MGMT 59.6%

        ---RESPONSE TIME---  EX   PERF  AVG   --USING%- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
        HH.MM.SS.TTT         VEL  INDX ADRSP   CPU  I/O  TOT  CPU  I/O                          UNKN  IDLE USG  DLY  USG  DLY  QUIE
GOAL                        30.0%
ACTUALS
ITS1                        65.0%  0.5   1.0  27.3 29.2 30.5 21.5  9.0                          13.0  0.0  0.0  0.0  0.0  0.0  0.0
```

There is no more CAPP delay, and the service class will not be subject to being capped.

> **Note:** If the work needs to overachieve its goal to provide the required service, it would be better to adjust the goal.

Smoothing the effect of WLC capping

Workload Licence Charge (WLC) capping is in effect when your four-hour rolling average millions of service units per hour (MSU) consumption is greater than the defined capacity MSUs set in the Hardware Management Console (HMC).

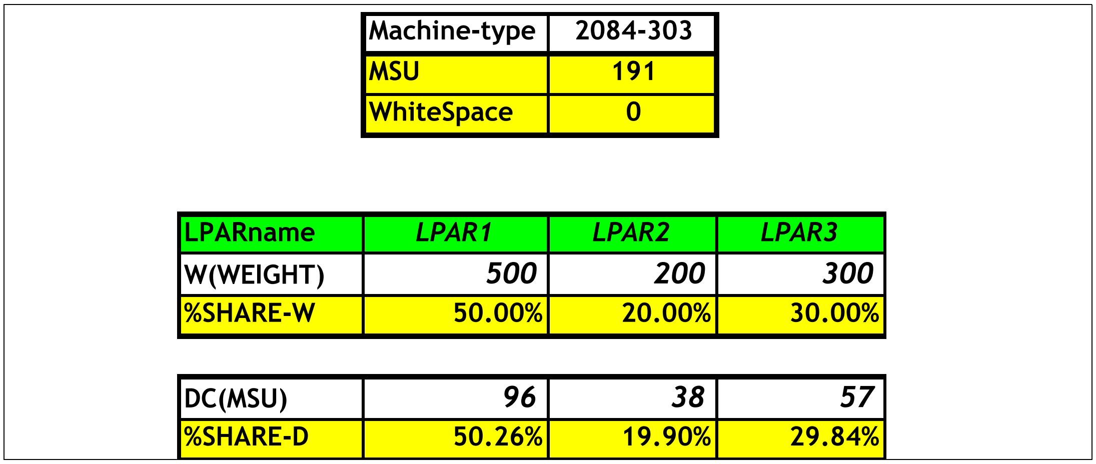
The MSU capacity of the LPAR can be burned with unimportant workloads (often during the night shift), and the LPAR can be capped then for a long time.

According to your particular settings, the capping effect is abrupt and can harm workloads such as TSO.

Resource Groups can be implemented to smooth this effect. This can lead to having a night and day shift service policy override that adds or removes the Resource Group. The Resource Groups can, for example, be added only in a period of the night shift and then removed.

Adding a Resource Group will probably elongate some CPU bound workloads during the night shift, but this will help you avoid reaching the four-hour rolling average MSU limit and will give consumption flexibility to the LPAR.

This is the implementation scenario. The machine is a z990 software model 303 rated at 191 MSU. Three LPARs are defined as shown in Figure 3-3: LPAR1, LPAR2, and LPAR3.



Machine-type	2084-303
MSU	191
WhiteSpace	0

LPARname	*LPAR1*	*LPAR2*	*LPAR3*
W(WEIGHT)	*500*	*200*	*300*
%SHARE-W	50.00%	20.00%	30.00%

DC(MSU)	*96*	*38*	*57*
%SHARE-D	50.26%	19.90%	29.84%

Figure 3-3 LPAR configuration

Note that the percentage share calculated from the LPAR weight and the percentage share calculated from the WLC definition are close. This will reduce the four-hour rolling average capping effect. We recommend setting up your configuration in this way. This will not be possible if your configuration has fewer MSU defined than the machine total number of MSU.

All of these LPARs have two logical processors.

The SU/s that can potentially be delivered in the physical machine is computed by selecting the physical machine model from the SU/s table and multiplying this number by the number of physical processors. In our example, the SU/s of a single engine of a z990-303 is 20075.2823. The total SU/s that can be delivered will be 60225.8469 SU/s.

LPAR2 is allowed to use 19.90% of this total, which is 11442.9109 SU/s.

We now have all of the information to make subsets of this potential SU/s usage and spread it into various Resource Groups.

For example, if we want to limit specific jobs to one third of the machine, then we create a Resource Group with 3814.30 SU/s. Policy override can help with setting on or setting off a Resource Group in a service class.

> **Note:** In this particular case, the SU/s calculation is done at the machine level and not at the LPAR level because the MSU is a machine level rating and a percentage of this MSU rating is required. This is different from calculating the SU/s at the LPAR level, which takes into account the number of logical processors.

Simulating a transaction rate goal for batch jobs

This section examines a particular usage of the MINIMUM SU/s setting in a Resource Group.

> **Note:** Here is a discussion about a particular usage of the MIN value in a Resource Group. It is not a general guideline nor a recommendation. It will be used at your own risk and only if you have a specific type of batch throughput that is described in the section.

This MINIMUM value can be used to simulate a transaction rate goal for batch jobs. Let us do some math:

```
Transaction Rate (TR) = Number of ended transactions / Elapsed Time
```

Also, TR = *N* / (*Tt*+*Tr*), where:

- *N* = average number of users logged on
- *Tt* = average user think time
- *Tr* = average transaction response time

WLM cannot manage TR because TR depends on Tt and N, which are beyond the scope of the current controls.

However, if you have *accurate information* about a particular set of batch jobs, you can emulate such a goal by setting a MINIMUM value (expressed in SU/s) in the Resource Group associated with a batch service class. Let us call this goal Target_TR.

The Target_TR formula is:

```
MIN CPU SU/s = Target_TR x CPU_SUs / (#_Ended_Jobs)
```

> **Note:** To be consistent with the units of the MINIMUM (SU per second), Target_TR is expressed in #Jobs / 60 seconds. For example, if you want 40 jobs per minute, you specify Target_Tr as 40/60.

To derive the MIN CPU SUs/sec, we need to capture how many CPU service units are consumed on average (along an RMF interval) per job. Calculate this by dividing CPU_SUs by the #_Ended_Jobs. If we multiply this ratio by our Target_TR, we derive the amount of batch CPU SUs/sec that we want to guarantee. Keep in mind that we are working with *average values*, and that these can be very different in other time periods. Also, WLM cannot guarantee such a minimum, if there is not a demand, such as few initiators and idle initiators. Also, recall that the Resource Group minimum is not enforced as is maximum capping value.

You can obtain CPU_SUs and #_Ended_JOBs from a representative period of time from the WKLD RMF report.

Recall that this MIN refers to raw CPU service units ((not multiplied by the service definition coefficient (SDC)); the CPU SDC is 1.0 (as recommended) or you need to adjust it.

Let us evaluate a scenario based on this reduced RMF report (Example 3-4).

Example 3-4 Batch Activity report

```
TRANSACTIONS    ---SERVICE----
AVG    129.69   IOC     12511K
MPL    129.66   CPU    207042K
ENDED    2891   MSO    196650K
END/S    1.61   SRB      4891K
#SWAPS    646   TOT    421094K
EXCTD       0   /SEC    234242
```

To match the formula, we have:

- CPU_SUs from the report in the column SERVICE at the CPU and SRB rows. Our value is 207042k. Note that we do not use the “k” unit in our calculation.
- #_Ended_JOBs from the report in the column TRANSACTIONS at the ENDED row. Our value is 2891.
- We want to evaluate what the MIN CPU SUs/sec for Target_TR = 40 jobs per minute should be.

The formula is:

- MIN CPU SU/s = Target_TR x CPU_SUs / (#_Ended_Jobs)
- Target_TR = 40/60
- CPU_SUs (TCB+SRB) = 211933k
- #_Ended_Jobs = 2891

The result is (40/60) x (211933 / 2891) = 48.8 K SU/s.

So, if your business needs 40 jobs/minute, the MINIMUM Resource Group value SU/s should be set at 48800.

3.1.2 CPU critical

In this section, we explain the feature available in WLM to guarantee the dispatching priority of selected service classes.

When you assign long-term CPU protection to critical work, you ensure that less important work will generally have a lower dispatch priority. (There are some rare exceptions, such as when other work is promoted because it is holding an enqueue for which there is contention.) This protection can be valuable for work that is extremely CPU-sensitive, such as certain CICS and IMS transactions.

It is possible that work in service classes with lower importance gets higher dispatch priority than work with a higher importance. Why? The CPU adjustment algorithm attempts to optimize the CPU usage by:

- Service consumption of the work
- Importance of the work
- Goal definition of the work

As a result, a service class with low importance, low goal definitions, and low service consumption can get the highest available dispatch priority in the system. This is not a problem as long as the work behaves well. But, it can hurt work of higher importance when the work starts to use large amounts of CPU in an unpredictable manner. This often happened in CICS when, after a low activity period (lunch time), people started to log on and experience response time elongation. After some policy adjustment, WLM gave CICS the proper dispatching priority and things ran fine again. The purpose of the CPU CRITICAL feature is to eliminate this problem.

CPU CRITICAL

The CPU CRITICAL feature protects work by not allowing work with lower importance to get a higher dispatch priority.

Work that needs protection is flagged. The result is that:

- All work with higher or equal importance can still compete freely for CPU resources with the protected service class.
- All work with lower Importance never gets above work with higher or equal Importance goals.

CPU CRITICAL can help in situations where work of lower importance might negatively impact work of high importance (for a short but visible period).

Do not overuse this feature; otherwise, you do not leave enough capability for WLM to efficiently balance the available CPU resources across the service classes in order to meet goals. WLM not only tries to meet the goals of the highest importance work, but also of lower importance work if the resource permits it.

CPU CRITICAL in action

Figure 3-4 on page 87 shows the effect of the feature: Service class X has been flagged as CPU CRITICAL (CC). All service classes having an IMPORTANCE higher than or equal to service class X will compete for CPU resource access (ServClas Z in Figure 3-4 on page 87). service classes Y1 and Y2 will never be allowed to have a DP higher than service class X, because their IMPORTANCE is lower.

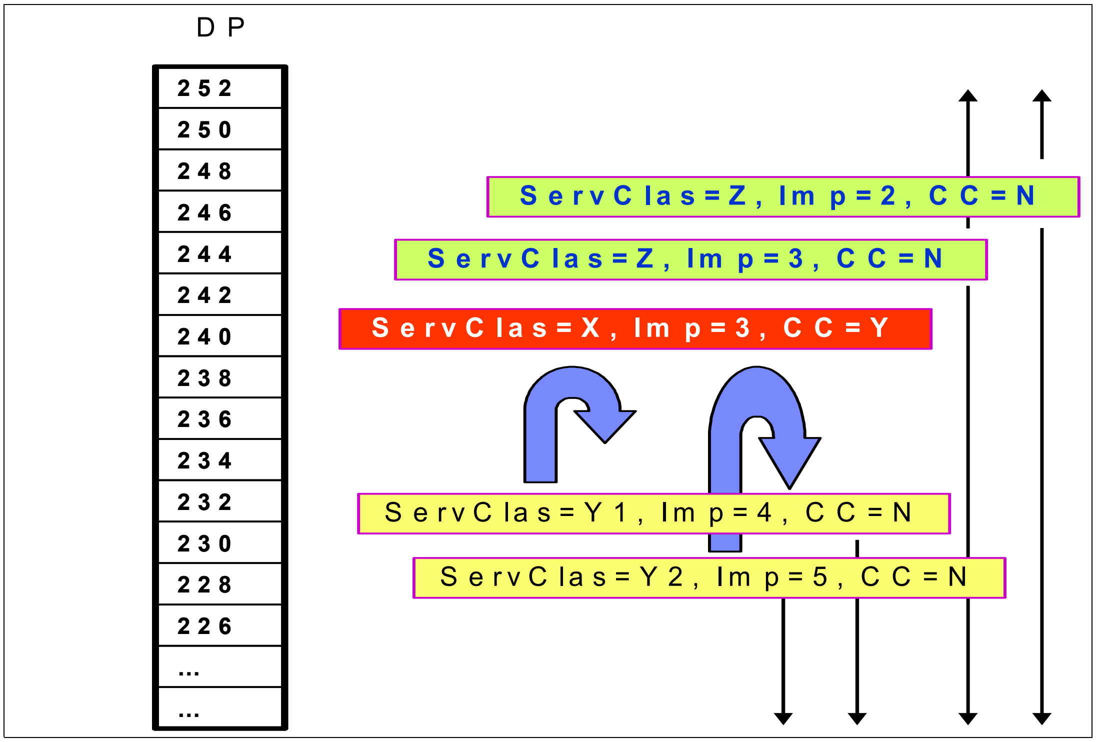


Figure 3-4 CPU CRITICAL in action

Who should use CPU CRITICAL

As we advise in the preceding section, use this feature for a workload of high business importance.

CICS production workloads and IMS production workloads are good candidates for this feature, because their activity can be subject to fluctuations.

Where to specify CPU CRITICAL

CPU CRITICAL is an attribute for a service class, so it is set in the service class definition, as shown in Example 3-5.

Example 3-5 CPU CRITICAL setting

```
                        Modify a Service Class
Command ===> ______________________________________________________

Service Class Name . . . . . : CICSW
Description  . . . . . . . . . WAS CICS transactions
Workload Name  . . . . . . . . WAS       (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . YES       (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal--------------------
Action  #  Duration    Imp.  Description
  __
  __    1              1     80% complete within 00:00:00.150
```

3.1.3 Storage critical

This is similar to CPU critical in the way that it protects service class storage from being stolen from lower importance work.

When you assign long-term storage protection to critical work, WLM restricts storage donations to other work. This option can be useful for work that needs to retain storage during long periods of inactivity, because it cannot afford paging delays when it becomes active again. With long-term storage protection assigned, this work loses storage only to other work of equal or greater importance that needs the storage to meet performance goals.

Storage of protected service classes is only taken when the *system* runs in conditions that are short on storage. See Figure 3-5.

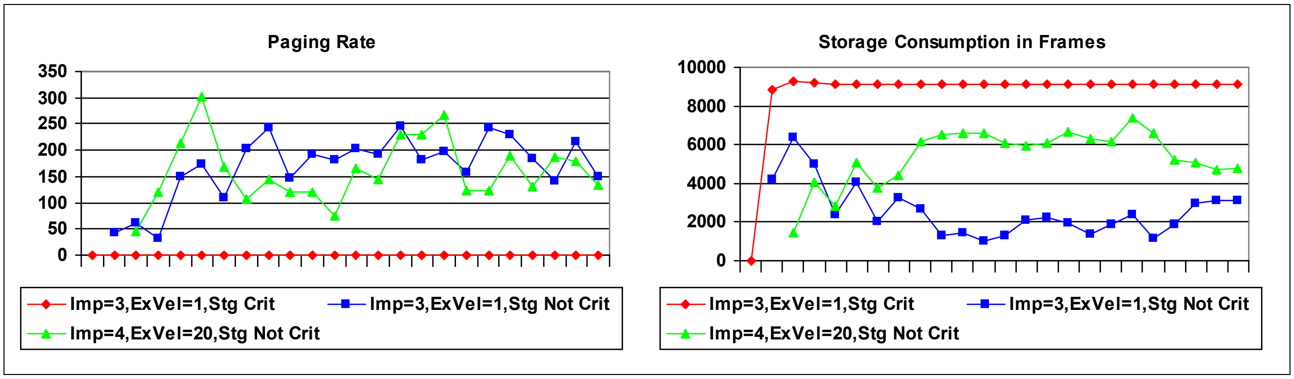


Figure 3-5 STORAGE CRITICAL in action

In Figure 3-5, the Imp=3 service class with the storage critical attribute is also protected against the Imp=3 service class without the storage critical attribute.

But if the system runs into serious storage constraints, this mechanism does not protect the work anymore.

Also, service classes with higher importance can use the frames, but not before the Imp=4 or Imp=3 service classes with no storage protection run out of frames.

Other characteristics are that storage remains protected even if no activity is occurring any longer. That is important for online regions (CICS and IMS) that do not show much activity overnight.

A safety valve exists: 25 pages per address space per hour go away.

Who should use STORAGE CRITICAL

Online workloads (CICS and IMS) that do not show much activity overnight should use STORAGE CRITICAL.

IRLM that has erratic activity and can be protected against page stealing should use STORAGE CRITICAL.

Where to specify STORAGE CRITICAL

STORAGE CRITICAL is an attribute for an address space or for the classification rule for CICS and IMS work.

- STORAGE CRITICAL for an address space

  Address spaces types, such as JES2 or STC, have explicit protection for their frames.

An address space must be in a service class that meets two requirements before it can be storage-protected:

– The service class must have a single period.

– The service class must have either a velocity goal, or a response time goal of over 20 seconds.

Note: These two requirements only apply to the address spaces classified under subsystem types: ASCH, JES, OMVS, STC, and TSO.

- STORAGE CRITICAL for the classification rule for CICS and IMS transactions

  Implicit protection of the address space that executes the CICS and IMS transaction

- STORAGE CRITICAL for non-managed service class

  You can specify STORAGE CRITICAL for an address space with the SYSTSC service class.

Example 3-6 shows the specification in JES.

Example 3-6 STORAGE CRITICAL in JES

```
Modify Rules for the Subsystem Type     Row 1 to 14 of 18
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : JES        Fold qualifier names?   Y  (Y or N)
 Description  . . . JES TYPE

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                         <=== More
           --------Qualifier--------           Storage    Manage Region
 Action    Type       Name     Start           Critical   Using Goals Of

  ____  1  TN         ITSOB*   ___             YES        TRANSACTION
        1  TN         ITSOR*                   NO         TRANSACTION
```

Example 3-7 shows the specification in CICS (subsystem classification rules).

Example 3-7 STORAGE CRITICAL in CICS

```
Modify Rules for the Subsystem Type       Row 1 to 9 of 9
Command ===> ____________________________________________ SCROLL ===> PAGE

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--------           Storage    Manage Region
Action    Type       Name     Start           Critical   Using Goals Of

 ____  1  SI         CICSPCA1 ___             YES        N/A
 ____  1  SI         SCSCERW1 ___             NO         N/A
 ____  2    TN         TRDL     ___           NO         N/A
```

Note: A service class can have storage-protected and unprotected address spaces.

3.1.4 Initiator dispatch priority

Initiator dispatch priority control provides the installation with the ability to set dispatch priority for initiators that have not been assigned to a service class below the dispatch priority for CPU-critical workloads.

This feature allows the control of the dispatch priority of initiators before job execution.

It applies to JES, APPC, and OMVS initiators and is implemented in a new parameter INITIMP in PARMLIB IEAOPT*xx* member.

Initiator behavior

Initiators run in SYSSTC at a dispatch priority of x'FE' and might use lots of CPU before a job is classified and executed in different areas such as:

- Installation-dependent tasks, such as exit routines for SAF, SMS ACS, or INITs.
- This can potentially prevent critical work of high importance from access to the CPU.
- This potential behavior becomes more prevalent with a decreasing number of CPUs.

The goal of this implementation is to make sure such work does not impact critical online work, such as CICS, IMS, or DB2. Importance-based initiator dispatching control is implemented through specification of the new parameter INITIMP in IEAOPT*xx* PARMLIB member. For further details on this parameter and how to use it, refer to 4.2.1, “IEAOPTxx usage” on page 114.

3.1.5 WLM CPU preemption

Think of *preemption* as the z/OS dispatcher interrupting a currently executing unit of work in order to dispatch higher priority work.

The z/OS dispatcher operates in a reduced preemption mode. The dispatcher uses this mode to interrupt an executing unit of work when a higher priority unit of work becomes ready to execute, but only after a brief delay period, the length of which is monitored and dynamically adjusted to achieve optimum performance. Reduced preemption can also be detrimental to system performance in situations where high priority work executes for only a short time. The amount of time that reduced preemption delays work can actually exceed the required execution time.

3.1.6 ENQ management

SRM provides an interface (Sysevent EnqHold/EnqRlse) that resource managers can use to signal that they are waiting for a resource that is held by someone else. This helps avoid critical work on your system, such as DB2, from being blocked waiting for resources, such as RACF® databases, IRLM locks, or latches, just because the blocker has low priority, is swapped out, or is not receiving services for any number of reasons.

Awareness of contention situations in authorized resource managers other than GRS, for example, is implicitly part of this solution. WLM enqueue management establishes the necessary WLM infrastructure, and we expect that authorized resource managers over time will support this effort.

An address space or enclave is promoted in terms of dispatch priority when it holds a resource that another address space or enclave wants. The resource manager indicates this situation to SRM through an ENQHOLD-sysevent. By promoting the address space or enclave for a limited amount of time, the system hopes that the holder gives up the resource faster than it usually does with a lower-dispatching priority. Also, while being promoted, the system ensures that the address space or address space associated with the enclave is not swapped out.

When a contention disappears, the resource manager notifies SRM through an ENQRLSE-sysevent.

You can set the enqueue promotion interval, shown in Figure 3-6, by the installation through the ERV-option in IEAOPT*xx*-parmlib member. The ERV-option specifies the CPU service units that an address space or enclave can absorb while it is promoted before the promotion is taken back.

In goal mode, the enqueue promotion dispatch priority is determined dynamically at every policy adjustment interval (10 sec). It can change based on available processing capacity and the amount of high dispatch work in the system. Address spaces are promoted to that priority if their current dispatch priority is not already higher than that.

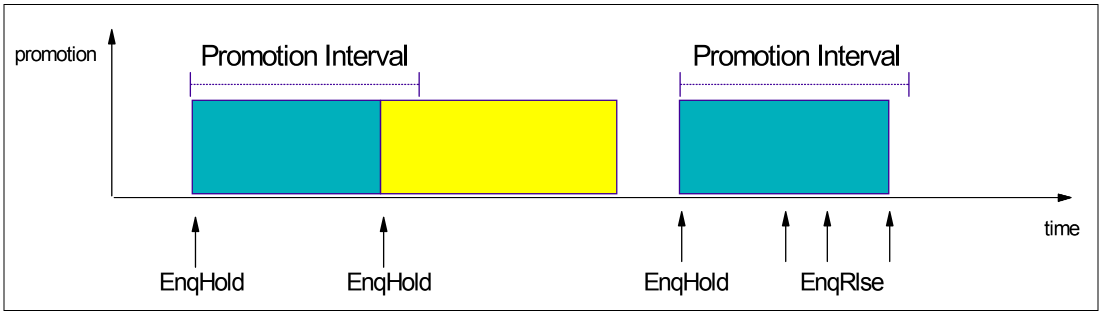


Figure 3-6 Enhanced enqueue management

SRM might promote an address space or enclave multiple times if SRM receives multiple notifications that work is waiting for the resource. Also, while there is an outstanding contention release notification, quiesced work is kept swapped in as long as there is ready work to run. This includes address spaces associated with enclaves.

SMF type 30 is expanded by a field in the processor accounting section that contains the CPU time consumed while enqueue is promoted. For more information, refer to *z/OS System Management Facilities (SMF)*, SA22-7630.

The ERV parameter in PARMLIB member IEAOPT*xx* controls the number of CPU service units that an address space or enclave can absorb in a situation of enqueue contention. For further details about this parameter, refer to 4.2.1, “IEAOPTxx usage” on page 114.

3.2 Server address space management

WLM provides a server address space management function. This function takes advantage of the new business unit of work (enclave) and allows WLM to manage the server address spaces of a work manager subsystem.

The design of server address space management addresses the following questions:

- If the goals are not met, can an additional server improve the performance index?

- If there is a resource constraint (CPU or storage) in the system, how do you reduce the activity of server address spaces?
- When should you decrease the number of server address spaces?
- Will the creation of a new server address space adversely impact the performance of other, more important goals?

The work manager subsystems that use the queuing manager services are getting the benefits of server address space management. Examples of IBM-supplied work managers that use these services are:

- DB2 for stored procedures
- Component Broker
- HTTP scalable Web server (IWEB)

These work managers use WLM services to permit workflow from their network-attached address spaces, through WLM, and into server address spaces for execution. We call the network-attached address spaces *queue managers* also and the server address spaces *queue servers*.

The following components participate in meeting the goal:

- Work manager

  A subsystem or work manager routes transactions to WLM, identifies the server JCL to WLM, and provides shell services for applications.

- WLM:
  - Creates or deletes server address spaces
  - Directs work into the server address spaces
  - Decides when to create a new server address space
  - Reports on goal achievement
  - Monitors the performance of transaction environments and gathers performance information

3.2.1 Application Environment

The server address management function is implemented through the Application Environment concept. An *Application Environment* is a grouping of transactions belonging to a work manager that have similar data definition and security requirements and therefore can be run in servers started by the same procedure. An Application Environment can have a system-wide or sysplex-wide scope, depending of the structure and capabilities of the work manager using the Application Environment. The WLM services, which the work manager uses, dictate the scope of the Application Environment. The queuing manager services provide system scope, and routing services provide sysplex scope for an Application Environment.

You must define your Application Environments and assign incoming requests to a specific Application Environment. JES2 and JES3 do not need special Application Environment definitions to exploit the server address space management functions.

WLM manages a separate queue for every Application Environment, and the server address spaces that are managed by WLM are created to serve work associated with a specific Application Environment queue. Defining multiple Application Environments allows you to separate your workload into different address spaces. In addition, within an Application

Environment, there is a separate queue for each service class, which means that work that is run with different goals runs in separate address spaces and can be managed independently.

The workload requirements for a given application can determine that multiple server address spaces should be activated. WLM decides to activate a new address space based on the following information:

- There is available capacity (CPU and storage) in the system and work in a service class is delayed waiting on the WLM queue.
- A service class is missing its goals, it is suffering significant queue delay, and other work (donor) in the system can afford to give up resources to the work that is missing its goals (receiver).

There is no mechanism provided for client intervention in this WLM process, other than the ability to limit the number of servers to one. WLM controls the activation and deactivation of an address space and also controls the number of server address spaces.

Application Environment queues

Various work requests (transactions and jobs) for a given Application Environment can have vastly different characteristics, resource consumption patterns, and requirements. For better control by WLM in goal mode, each Application Environment queue is logically divided into a set of subqueues. Each unique combination of Application Environment and service class defines a single Application Environment queue, and each server address space can process requests from only one queue.

Note: Remember that for every Application Environment queue, WLM starts at least one address space. Therefore, do not specify too many service classes in order to avoid having WLM start too many address spaces.

Transaction flow in an Application Environment

In this section, we give a short explanation about the transaction flow in an Application Environment:

1. A transaction reaches the work manager (WM), and the work manager checks the request against its definition. If there is no match against an Application Environment for that request, it is processed in the WM address space.
2. If the request matches an Application Environment, the work manager creates an enclave, using the WLM services, and sends the request to WLM. If an enclave already exists, that enclave is used for the request.
3. If a new enclave is created, WLM checks its classification rules and associates a service class with the request.
4. WLM enqueues the transaction to an Application Environment queue serving the transaction’s service class.
5. If there is no server address space active to serve the queue, WLM starts a new server address space. After the server address space initializes and connects to WLM, it selects the transaction from the queue and processes it.
6. If a server address space already exists, it selects the transaction and processes it.

Application Environment and transaction flow example

Figure 3-7 on page 94 shows an Application Environment example for multi-scalable HTTP Server (IWEB).

Queue Manager (QM) AS, started manually or by automation, receives the incoming transaction from TCP/IP: It either executes it if the request does not match against the defined AE or passes it to WLM.

WLM defines SC, creates an enclave, and places the transaction into the Application Environment Queue (AEQ). The transaction is executed next by the Queue Server (which receives sockets).

The Queue Server ASs (started and managed by WLM) are responsible for executing the transaction, which is taken from WLM AEQ queues.

Sysplex-wide resource managers, such as CICS, IMS, and DB2 can receive the transaction through fast CGI.

WLM is involved in:

- Enforcing goals and creating independent enclaves with TCBs
- Controlling the number of QS ASs, based on the goal and delay_for_server state

Any type of goals, Resource Group, and duration are allowed.

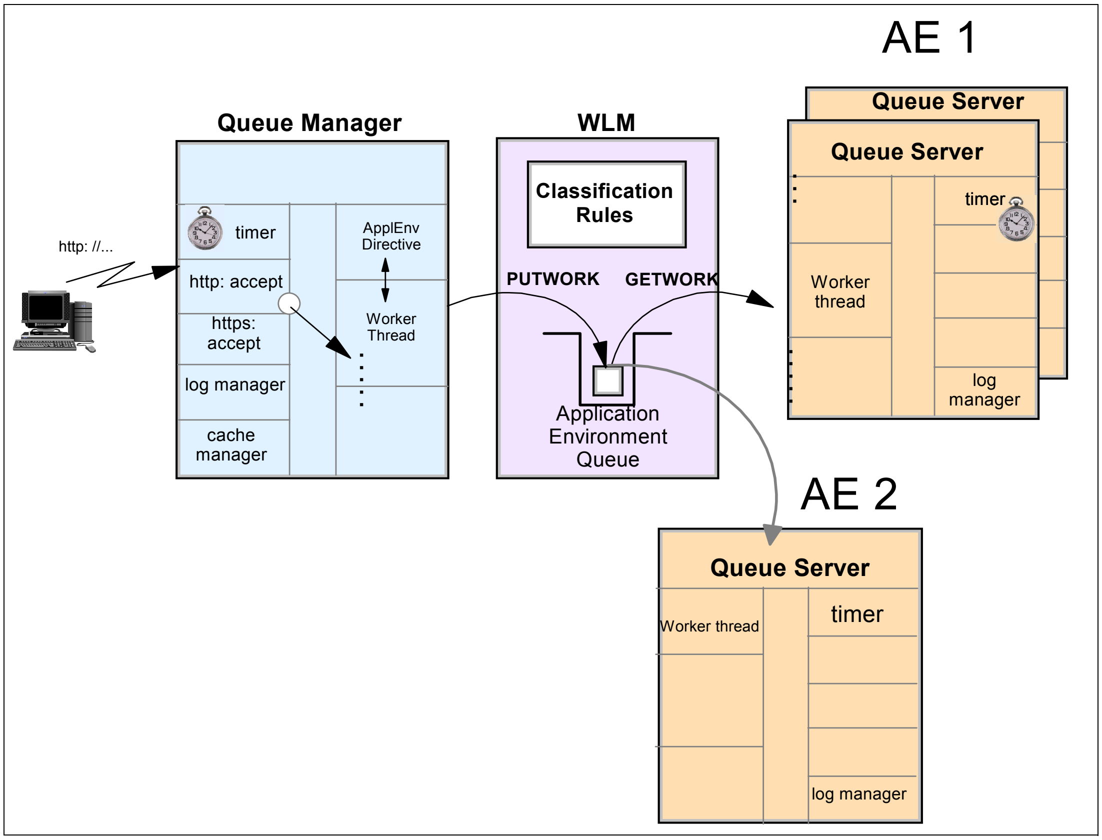


Figure 3-7 Application Environment and transaction flow example

Application Environment exploiters

Table 3-1 on page 95 shows the current Application Environment exploiters.

Table 3-1 Application Environments

Subsystem	**Subsystem type for classification**
DB2 Stored Procedures	► DDF
WebSphere ► HTTP Scalable Webserver ► MQSeries workflow	CB ► IWEB ► MQ

3.3 Workload routing support

This section explains the workload routing support available in WLM.

Routing support can provide multiple benefits:

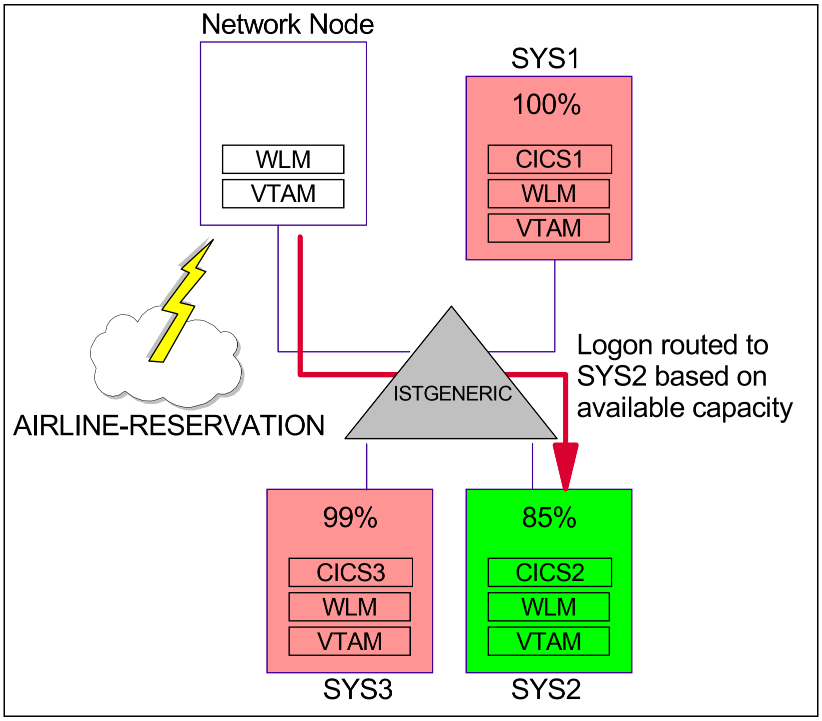
- High availability through redundancy.
- High performance through increased cluster efficiency, business goals, and importance considerations.
- Distribution of work across the Parallel Sysplex cluster according to the systems' load. This distribution avoids overloading systems with serious storage constraints in favor of:
  - Systems where similar work meets its goals
  - Underutilized systems
  - Systems where the least important work can be displaced

The implementation variations can be:

- Session placement with VTAM® generic resources
- Transaction routing using sysplex routing services

3.3.1 VTAM generic resources

A *generic resource* is a method of allowing multiple application programs to be known by a common name. The advantages are that users do not have to be aware of sysplex topology, and sessions are automatically balanced across the sysplex.

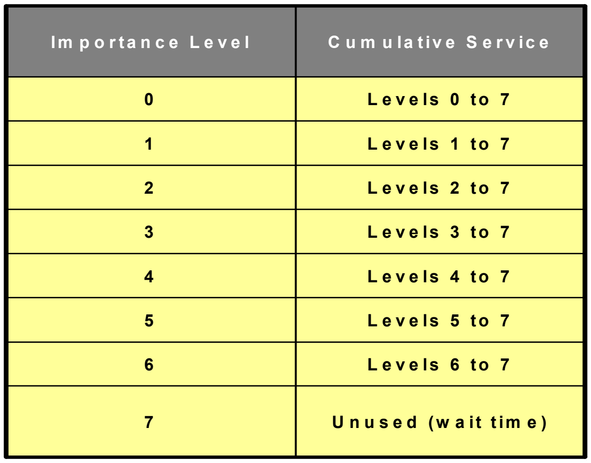


The major environments are CICS, IMS, Distributed DB2, TSO, and APPC.

A target system algorithm is based on a table that tracks service consumed by importance. The objective is to send work where there is available capacity, or if there is no available capacity, where the least important work will be displaced.

Target service is based on work type.

The routing decision for VTAM Generic Resources also takes into account the Performance Index (PI) of the target servers. This means that, beside the capacity, how well the target server achieves its goal is a factor for the routing decision.



Importance Level	Cumulative Service
0	Levels 0 to 7
1	Levels 1 to 7
2	Levels 2 to 7
3	Levels 3 to 7
4	Levels 4 to 7
5	Levels 5 to 7
6	Levels 6 to 7
7	Unused (wait time)

The table is built in importance level as follows:

- Level 0: System work in service classes SYSTEM and SYSSTC
- Level 1 to 5: Importance level from service definition
- Level 6: Discretionary work
- Level 7: Unused service

Each level includes service at its level and all lower levels.

Summaries contain 1, 3, and 10 minute averages.

VTAM exit customization

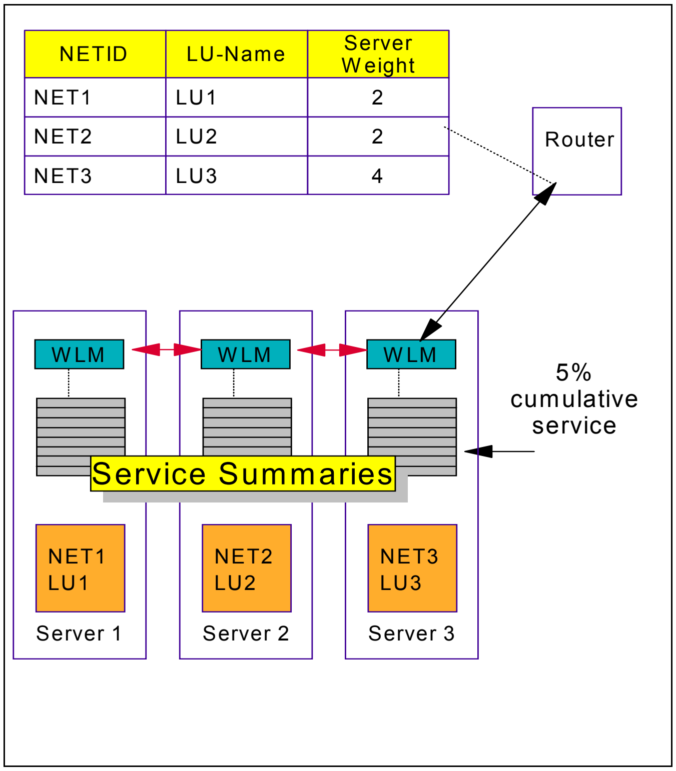
VTAM calls exit ISTEXCGR in SYS1.SAMPLIB during generic resource resolution. You must customize exit ISTEXCGR in order for it to have full capabilities. The coding example (Example 3-8 on page 97) is self-explanatory and tells you which OR IMMEDIATE (OI) instructions coding you should choose to enable all of the exit capabilities.

Example 3-8 ISTEXCGR exit

```
*
GENRSCRS DS    OH                      BEGIN RESOLUTION
         MVC   GRREXIT,NULL            SET EXIT CHOICE TO NULL
         OI    GRRFLAG1,GRRFWLMX       TURN ON CALL WORK LOAD MANAGER
*                                      BIT AND DEFAULT OFF CALL
*                                      GENERIC RESOURCE EXIT BIT @N1C
* /********************************************************************
* /*  The following is sample code to turn ON all GRRFLAG1 bits      *
* /*  It can be modified to set the preferred bits ON                 *
* /*  OI    GRRFLAG1,GRRFUVX+GRRFWLMX+GRRFNPLA+GRRFNPLL *
* /*                                                                           *
```

3.3.2 Sysplex routing

Sysplex routing allows a program, which routes work to multiple systems, to choose systems based on their existing workload.



NETID	LU-Name	Server Weight
NET1	LU1	2
NET2	LU2	2
NET3	LU3	4

A router asks WLM for advice such as where to route work in a sysplex, for example, to Sysplex Distributor at the frequency of every 1 to 3 minutes.

WLM returns a list of eligible servers with a weight assigned to each server.

Weight represents the relative volume of requests until WLM is asked for recommended servers the next time, for example:

- Servers 1 and 2 should receive 1/4 of the requests.
- Server 3 should receive 1/2 of the requests.

Recommendation criteria:

- Send work to systems with idle capacity.
- Find the system with the least important work that can be displaced, if no idle capacity exists.
- Do not recommend systems with serious storage constraints, unless all systems are constrained.

The TCP/IP domain name server (DNS), in conjunction with several IP servers, uses this function to allow the spread of IP requests across multiple systems.

The sysplex routing services allow work associated with a server to be distributed across a sysplex. They are intended for use by clients and servers.

In terms of the sysplex routing services, a *client* is any program routing work to a server. A *server* is any subsystem address space that provides a service on an MVS image.

The sysplex routing services enable distributed client/server environments to balance work among multiple servers. These services help the distributed programs make the routing decisions, rather than each installation make the routing decisions.

The sysplex routing services *provide information* for more intelligent routing. They do not route or distribute work requests. The server must use its existing routing mechanisms.

When a server is ready to receive work requests, it issues the IWMSRSRG macro to register itself to WLM. The server identifies itself by providing the following information:

- Location name
- Network ID
- LU name
- Stoken (the space token that identifies the server address space)
- Optionally, a health indication. This allows the server to indicate if there are constraints that might stop it from taking on new work, even when the system itself is not heavily loaded. The value is in the range of 0 (not acceptable) to 100 (acceptable).

WLM then makes this information available to all z/OS images in the sysplex.

When a client wants to route work to a server, the client issues the IWMSRSRS macro for a list of registered servers. To help the client decide where to route the request, this macro returns an associated weight for every eligible server. The weights represent the relative number of requests that each server should receive. The calculation of the weights uses various capacity considerations. The weights allow changes in system load and server availability to be factored into work distribution.

Note: A client should issue the macro on a regular basis to stay current with the changing conditions.

Calculation of server weight

The calculation of the server weight is based not only on the system CPU utilization but also the health and performance of the server.

WLM builds a table recording the CPU capacity for each system in the sysplex. The CPU consumption is recorded by Importance level. The Importance levels are listed in Table 3-2.

Table 3-2 WLM CPU capacity Importance levels

Level	Usage
0	Used for service consumed by the system
1-5	Are the externally defined Importance levels
6	Is used for discretionary work
7	Is unused service (equivalent number of service units)

Each row contains the following fields:

- Number of service units consumed in three minutes at the given importance level and below.
- Percentage of total capacity consumed in three minutes at given importance level and below. For example, level 4 in the table has the service units consumed by Importance 4, Importance 5, and discretionary work plus unused capacity.

The WLM algorithm occurs in the following order:

1. WLM scans the rows in the table in reverse starting with level 7 until it finds a level where one or more systems have at least 5% cumulative service units of capacity. WLM then uses this table level for all goal mode systems to assign relative weights.
2. A system weight is calculated for each system in the sysplex using the SU value in the selected row in the table.

   System weight = (SUs for this system at selected level * 64) / total SUs for all systems at the selected level.
3. Finally, it calculates a server weight: Server weight = system weight / # of servers on system.

If the *SPECIFIC* function is specified on the IWMSRSRS macro, additional factors are used when calculating the server weight:

- The WLM performance index of the server
- Delays due to the queue time of any independent enclaves owned by the server
- The health factor indicated by the server

Example 3-9 shows a output produced by the Assembler program.

Example 3-9 Server weight calculation

```
*WLMREG--+-------------------------------------------+
*WLMREG--|Location          |Netid   |Luname  |Weight|
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TN3270E           |        |wtsc48  |  06  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TN3270E           |        |wtsc69  |  05  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TN3270E           |        |wtsc55  |  05  |
*WLMREG--+------------------+--------+--------+------+
.
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TCPIP             |wtsc48  |TCPIP   |  03  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TCPIP             |wtsc48  |TCPIPA  |  03  |
*WLMREG--+------------------+--------+--------+------+
.
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|TEST_LOCATION     |NETSC69 |LUSC69  |  64  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|DB8F              |USIBMSC |SCPD8F1 |  41  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|DB8F              |USIBMSC |SCPD8F2 |  22  |
*WLMREG--+------------------+--------+--------+------+
*WLMREG--|DB7Q              |USIBMSC |SCPD7Q2 |  64  |
*WLMREG--+------------------+--------+--------+------+
```

Sysplex routing user: z/OS Load Balancing Advisor

The z/OS Load Balancing Advisor is a component of z/OS Communication Server, which communicates with outboard load balancers (LBs) and one or more z/OS Load Balancing Agents. The primary function of the z/OS Load Balancing Advisor is to provide external TCP/IP load balancing solutions, such as the CISCO Content Switching Module (CSM), with recommendations about which TCP/IP applications and target z/OS systems within a z/OS sysplex are best equipped to handle new TCP/IP workload requests. The load balancer can then use these recommendations to determine how to route new requests to the target applications and systems. (That is, how many requests should be routed to each target.) The recommendations provided by the Advisor are dynamic; they can change as the conditions of the target systems and applications change, and they include several key components:

- State of the target application and system

  This includes an indicator about whether the target application or target system is currently active. This allows the load balancer to exclude systems that are not active or do not have the desired application running.

- z/OS Workload Management (WLM) system-wide recommendations

  WLM recommendations provide a relative measure of a target system's ability to handle new workload (as compared to other target systems in the sysplex). The WLM recommendations are derived using several measures, including each system's available CPU capacity or capacity that can be displaced by higher importance workloads. The latter is important for scenarios where systems might be 100% utilized, but some systems might be running work that has a lower Importance level (as defined by the WLM policy), and the work can therefore be displaced by a higher Importance level workload. You retrieve this information by using the IWMSRSRS routing service of WLM.

- Application server health from a TCP/IP perspective

  TCP/IP statistics for target applications are monitored to determine if specific server applications are encountering problems keeping up with the current workload. For example, is a target TCP server application keeping up with TCP connection requests? Or, are requests rejected because the backlog queue is full? In scenarios where this occurs, the recommendations passed back to the load balancers will be adjusted appropriately so that the load balancer can direct fewer connections to any applications that experience these problems.

Sysplex routing user: DB2 Connect

DB2 Connect™ Enterprise Edition servers provide load balancing and fault-tolerance when routing connections to multiple sysplexes. When connected to a DB2 for OS/390 and z/OS database server running in a data sharing environment, DB2 Connect spreads the workload among the DB2 subsystems comprising the data sharing group, based on the system load information provided by the WLM. This support requires DB2 for OS/390 Version 6 or later. DB2 Connect receives a prioritized list of sysplex members from the WLM. Each sysplex returns weighted priority information for each connection address. DB2 Connect server then uses this list to handle the incoming CONNECT requests by distributing them among the sysplex members with the highest assigned priorities. For load balancing, the list of sysplex weighted priority information is obtained during each connection. If the DB2 Connect connection concentrator is enabled, this list is also used when determining where to send each transaction.

Note: You do not need to change the z/OS Distributed Data Facility (DDF) configuration in order to take advantage of the DB2 Connect sysplex exploitation.

With the addition of the concentrator, DB2 Connect now has the ability to balance the workload at transaction boundaries. You must enable the DB2 Connect concentrator in order for this to work. You achieve different granularity of load balancing depending on the version of DB2 on the host. If load balancing is running against DB2 for OS/390 Version 6.1 or later, DB2 Connect receives an updated status from the WLM on each transaction.

Connection concentrator introduces a concept of Logical Agent (LA), which handles user context while the coordinating agent (CA) continues to own the DB2 connection and thread. When a new application user connects, the new application user is assigned an LA. CA is needed to pass SQL to DB2, so one is assigned as soon as a new transaction is initiated. The key to this architecture is that CA is disassociated from the LA and is returned to the pool when the transaction completes (commit/rollback). Another key feature is the method of assigning CAs to new transactions in data sharing environments. DB2 Connect implements a sophisticated scheduling algorithm that uses z/OS WLM information to distribute workload across members of a data sharing group according to criteria set up in WLM. WLM is not only aware of the load on each member, but also each member's availability. This allows DB2 Connect to transparently relocate work away from failed or overloaded members to members that are up and underutilized. When you set the number of maximum logical agents higher than the number of coordinating agents, you activate DB2 Connect connection concentrator.

3.4 Soft capping

This section examines WLM management of *soft capping* to support Workload License Charges (WLC).

Soft capping refers to capping dynamically controlled by WLM as opposed to capping, which is applying a cap to a logical partition by selecting the "Initial Capping" field on the Hardware Management Console (HMC).

To support WLC, you can specify a system capacity in millions of service units per hour (MSUs). The CPU usage can spike above the set capacity temporarily, as long as the average usage remains below the limit. WLM keeps a four-hour rolling average of the CPU usage of the logical partition, and when the average exceeds the defined capacity limit, WLM will signal LPAR to cap the logical partition.

Soft capping only applies to shared CPs, it does not apply to zAAPs nor zIIPs since z/OS software licensing charges do not apply to these specialized processors.

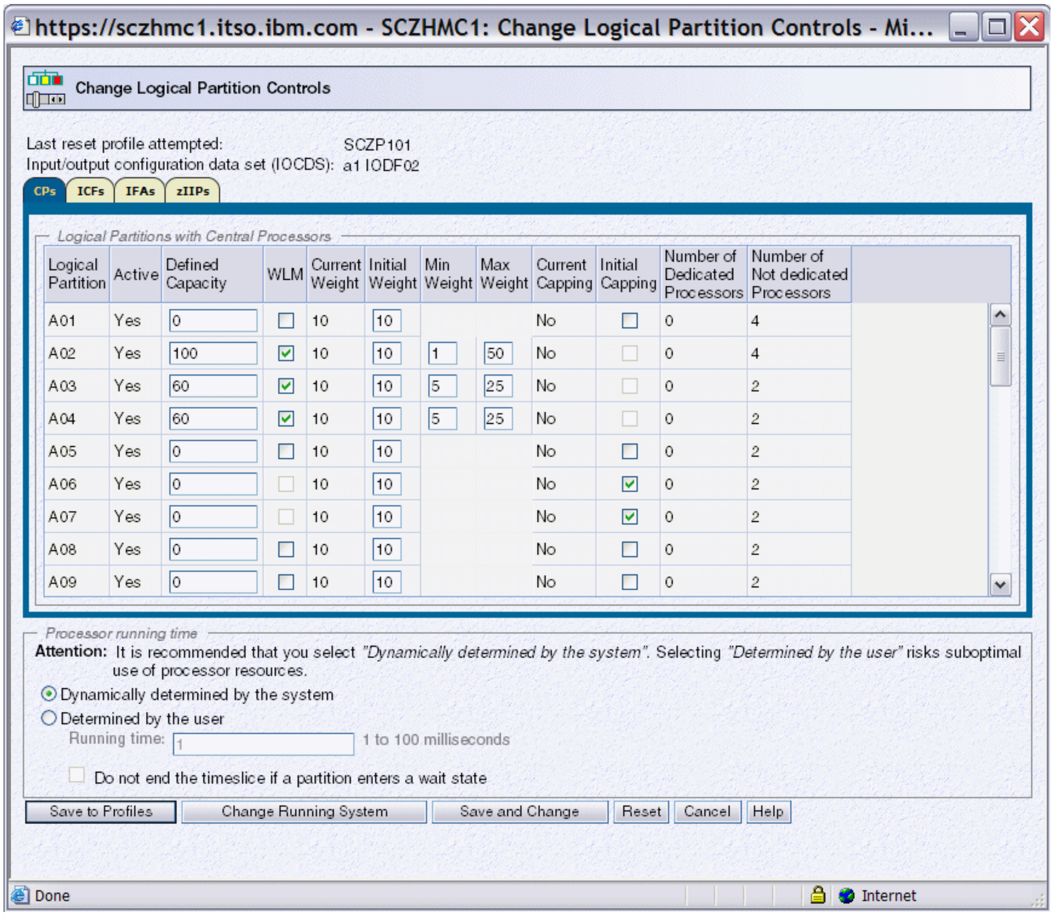
3.4.1 Defined capacity

Defined capacity sets the capacity of an individual partition when soft capping is selected. The Defined Capacity can be specified for each LP from the Hardware Management Console using the Change Logical Partition task.

Figure 3-8 shows an example:

- Logical partition A01 is "soft capped" because a defined capacity has been specified for the LP. WLM will manage the four-hour rolling average of CPU usage for the logical partition and limits the consumption to 120 MSUs.

  Note that the Defined Capacity field is enabled for CPs but disabled for IFAs or zIIPs tabs.
- Logical partition A02 is "hard capped" to a processing weight value of 10. The "initial capping" box is checked.


https://sczhmc1.itso.ibm.com - SCZHMC1: Change Logical Partition Controls - Mi...
Change Logical Partition Controls
CPs
ICFs
IFAs
zIIPs
Save to Profiles
Change Running System
Save and Change
Reset
Cancel
Help
Done
Internet

Last reset profile attempted: SCZP101
Input/output configuration data set (IOCDS): a1 IODF02

Logical Partitions with Central Processors

Logical Partition	Active	Defined Capacity	WLM	Current Weight	Initial Weight	Min Weight	Max Weight	Current Capping	Initial Capping	Number of Dedicated Processors	Number of Not dedicated Processors
A01	Yes	0	□	10	10			No	□	0	4
A02	Yes	100	☑	10	10	1	50	No	□	0	4
A03	Yes	60	☑	10	10	5	25	No	□	0	2
A04	Yes	60	☑	10	10	5	25	No	□	0	2
A05	Yes	0	□	10	10			No	□	0	2
A06	Yes	0	□	10	10			No	☑	0	2
A07	Yes	0	□	10	10			No	☑	0	2
A08	Yes	0	□	10	10			No	□	0	2
A09	Yes	0	□	10	10			No	□	0	2

Processor running time

Attention: It is recommended that you select *"Dynamically determined by the system"*. Selecting *"Determined by the user"* risks suboptimal use of processor resources.

● Dynamically determined by the system
○ Determined by the user
Running time: 1 1 to 100 milliseconds

□ Do not end the timeslice if a partition enters a wait state

Figure 3-8 Change Logical Partitions Control

The four-hour rolling average MSU consumption is calculated for each logical partition every 5 minutes.

When the four-hour rolling average MSU consumption is greater than the defined capacity MSUs, the logical partition must be capped at the HMC-defined weight (if possible) according to the terms and conditions of the WLC software pricing rules.

At that time, WLM asks PR/SM™ to cap the logical partition at the HMC-defined weight value.

When WLM caps the logical partition, there are three possible scenarios:

- The percentage share of the logical partition processing weight is equal to the percentage share of the defined capacity.

  In this case, the capping is done at the logical partition processing weight. This is the best and recommended situation.

- The percentage share of the logical partition processing weight is greater than the percentage share of the defined capacity.

  In this case, capping at the weight has no effect, because the logical partition has more MSU than allowed. To enforce the percentage share of WLC, PR/SM subtracts a certain number from the logical partition processing weight to match the desired percentage share and calculates a “phantom” logical partition that receives the remaining unused weight. This guarantees that other logical partitions are not affected by the weight management, because the total logical partition processing weight stays the same.

- The percentage share of the logical partition processing weight is lower than the percentage share of the defined capacity.

  WLM defines a cap pattern that repeatedly applies and removes the cap at the logical partition processing weight. Over time, this looks as though the partition is constantly capped at its defined capacity limit.

  The cap pattern depends on the difference between the capacity based on the weight and the defined capacity limit. If the weight is small compared to the defined capacity, the capacity of the partition can be reduced drastically for short periods of time. This can cause performance to suffer. Therefore, we recommend to keep both definitions as close as possible.

3.4.2 Group capacity

Group capacity is similar to defined capacity, but it allows you to define a limit for LPARs as a group instead of individually. Group capacity can work together with defined capacity: An LPAR receives either its defined capacity or its share of the group capacity, whichever is less.

IBM introduces group capacity in z/OS 1.8. Partitions running earlier releases of z/OS are not considered part of the group. Because capacities are defined in the HMC, hardware support, available in the near future, will also be required.

3.5 Intelligent Resource Director

Intelligent Resource Director (IRD) is a set of functions implemented by actions on:

- z/OS
- WLM
- I/O supervisor (IOS)
- SAP

The functions are:

- LPAR CPU Management
- LPAR weight management
- Dynamic Channel Path Management (ESCON/FICON channels with Directors only)
- Channel subsystem priority queuing

Let us explain the purpose of these functions.

Originally, a CEC was designed to host just one copy of an operating system code (single image). However, several modern business needs, such as the need for isolation, for ease of installing new workloads, and for continuous operation, create lots of LPAR images under LPAR in the same CEC.

Because each system is unique in itself with its own data, and program problems start to arise with many logical partitions (LPs), such as complexity, unbalancing, and lack of availability, the solution is to allow data sharing among the LPs through the implementation of coupling facilities. Now, any transaction can run anywhere, and with dynamic load balancing algorithms, we can balance the load in LPs in the same and different CECs. All the systems are clones, because they can reach the same data and programs.

However, it is difficult to implement data sharing for certain types of data. Then, the unbalanced behavior is not completely resolved. IRD introduces the idea of dynamically taking the hardware (CPUs and channels) to the LPAR where it is most needed, therefore fixing the problem. In other words, let us move the power to where it is necessary.

This chapter does not cover IRD itself, but gives advice about how to have a WLM policy that works properly with IRD.

z/OS Intelligent Resource Director, SG24-5952, explains all IRD functions and setup.

Policy considerations for IRD

This section addresses important considerations before you implement IRD.

Importance and goal of a service class

You must understand the importance of a service class as LPAR cluster-wide. The WLM policy must not be done at a specific LPAR level, for example, IMPORTANCE 3 service classes must serve the same types of work in the LPAR cluster.

This can be difficult when you have different types of LPARs in the LPAR cluster. For example, you share CPU resources among production, pre-production, and development LPARs, or you share CPU resources among production LPARs of different companies.

These examples highlight that you must review and synchronize the IMPORTANCE before you start IRD.

Minimum number of logical processors

Certain software can suffer because of single engine LPAR configuration. This is software that is often time-dependent and cannot wait for system actions such as SVC dump to complete. A new IEAOPT*xx* parameter VARYCPUMIN available with z/OS 1.6 is designed to limit the scope of WLM CPU management. The number of logical processors specified for a specific LPAR cannot exceed the number of physical processors of the CPC.

3.5.1 Linux for System z LPAR management

This section explains how to enable non-z/OS CPU management.

If you have logical partitions running Linux or other non-z/OS systems, you can manage CPU resources across these logical partitions in accordance with workload goals.

> **Note:** Non-z/OS partition CPU management does not support the management of partitions running OS/390 systems.

Non-z/OS partition CPU management allows WLM to shift weight from either z/OS or non-z/OS partitions to another z/OS or non-z/OS partition in the same LPAR cluster.

> **Note:** Linux® images must have shared CPs and must not be capped. Only standard general purpose CPUs are supported; Integrated Facility for Linux (IFL) CPUs are not supported.

Setting the cluster name

Before activating the Linux partition, specify the CP management cluster name under the Options tab of the activation profile. This causes LPAR to group the Linux partition with the appropriate sysplex.

Setting the system name

To set the system name:

- For the Linux partition

  For the Linux partition, the system name must be set in order to activate LPAR CPU management. WLM needs the Linux system name in order to handle the Linux partition. This also applies if you only use the PX qualifier. The sysplex name specified in the service element panel must be the same as the z/OS sysplex name. The system name must be unique in relation to the other system names in the same sysplex.

  Set the system name by running the **`insmod hwc_cpi.o system_name=sysname`** command. hwc_cpi.o currently ships in the directory /lib/modules/*x.y.z*/kernel/drivers/s390/char/, where *x.y.z* is the kernel version.

  If you change the Linux system name, then you must reIPL to make it known to LPAR CPU management.

- For VSE/ESA™ and z/VM

  VSE/ESA and z/VM set the system name during IPL.

Setting the goal

A new subsystem type SYSH is introduced for Linux CPU management. The following restrictions apply to subsystem SYSH:

- SY and PX are the qualifiers that are valid for SYSH.
- No Resource Groups can be associated with service classes for SYSH.
- CPU protection cannot be assigned.
- SYSH supports velocity goals with single periods, but no discretionary goals.

3.6 Specialty processors

IBM currently supplies four processor types called *specialty processors*. See Figure 3-9 on page 106. The Integrated for Linux (IFL) and the Internal Coupling Facility (ICF) processors, the zAAP (z890, z990, *and* z9-109) and ZIIP (z9-109 *only*) are all specialty processors. Use these processor types to offload certain types of work from the z/OS general purpose processor:

- IFL can only run the Linux operating system, or z/VM in support of Linux.
- ICF can only run Coupling Facility Control Code (CFCC).
- zAAP is used by the IBM Java Virtual Machine (JVM) to execute Java code.
- zIIP is designed for processing certain type of database workloads.

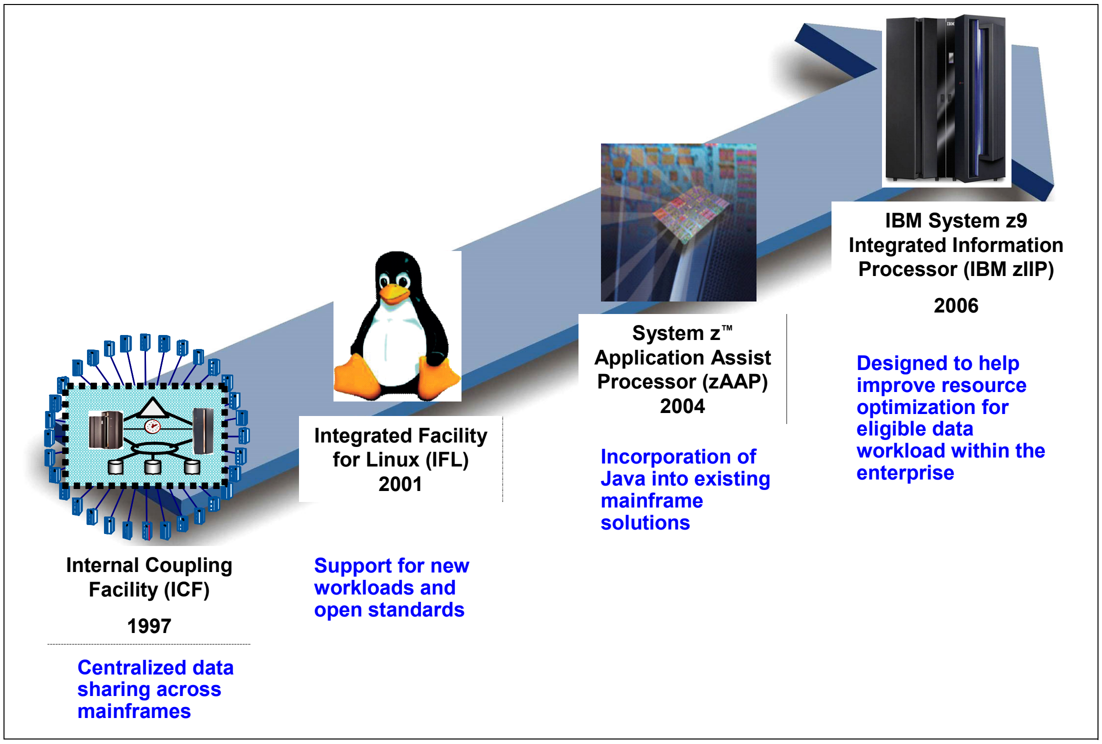

Internal Coupling Facility (ICF)
1997
Centralized data sharing across mainframes
Integrated Facility for Linux (IFL)
2001
Support for new workloads and open standards
System z™ Application Assist Processor (zAAP)
2004
Incorporation of Java into existing mainframe solutions
IBM System z9 Integrated Information Processor (IBM zIIP)
2006
Designed to help improve resource optimization for eligible data workload within the enterprise

Figure 3-9 Specialty processors

The specialty processor consists of the same hardware as the general purpose processor. Specific to specialty processors are:

- Cannot use them to IPL the z/OS operating system
- Do not handle I/O interrupts
- Do not add to WLC pricing charges
- Are not considered in IRD weight management
- Are not considered in WLM routing decisions
- Are not included in defined capacity computations and Resource Group management

With the z890 and z990, all the specialty processors belong to the ICF pool (zIIP, zAAP, IFL, and ICF) and inherit their weight from the CP pool. The z9-109 processors belong to a separate pool and their weight can be specified via the HMC. Specialty processors can be shared across LPARs like standard processors.

Note: For planning and implementation of specialty processors, use the zSeries Processor Capacity Reference Tool (zPCR). The zPCR tool and the documentation are available at:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1381

3.6.1 Internal Coupling Facility Processor (ICF)

These processors can only run Coupling Facility Control Code (CFCC). They are not visible to the z/OS operating system or applications. You use these processors to create a Coupling Facility for use by the exploiters of Parallel Sysplex functions.

3.6.2 Integrated Facility for Linux Processor (IFL)

This processor can only run the Linux Operating System or z/VM in support of Linux.

3.6.3 zSeries Application Assist Processor (zAAP)

zSeries Application Assist Processor (zAAP) brings Java to the mainframe in a new characterizable physical unit (PU). The zAAP processor is a characterizable PU (as IFL, ICF, and SAP) in the IBM System z machine models 990, 890, and z9-109, capable of only executing Java instructions under the control of a JVM in z/OS 1.6.

Execution of the Java applications on zAAPs, within the same z/OS SMP LPAR as their associated database subsystems, can also help simplify the server infrastructures and improve operational efficiencies. For example, use of zAAPs could reduce the number of TCP/IP programming stacks, firewalls, and physical interconnections (and their associated processing latencies) that might otherwise be required when the application servers and their database servers are deployed on separate physical server platforms.

zAAP can also help saving you IBM software charges in WLC, because you can have additional Java capacity without affecting the total MSU/hour of the CEC. There is no rewriting of the Java code to use the zAAP processor.

zAAP processors usually do not run z/OS code and cannot be used in IPLs. The maximum number of zAAPs is equal to the number of CPs. A zAAP processor is defined in the LPAR's image profile.

CPU using and delay samples for zAAPs are accumulated by WLM and are considered in policy adjustment decisions. Up to z/OS 1.7, WLM manages zAAPs as a extension of the general purpose CP; starting with z/OS 1.8, work on zAAPs is managed independently. zAAPs now contribute to using, delay samples, and service times. We call this functionality *zAAP processing enhancement*.

Note: You must install APARs OA14131 and OA13953 on z/OS 1.6 and z/OS 1.7 to enable the zAAP processing enhancement.

The JVM is the only authorized zAAP user. The JVM requests zAAP switch authorization on the initial entry. Other programs attempting to use zAAP authorization are rejected. The instruction Store System Information (STSI) does not report zAAP capacity.

zSeries Application Assist Processor and Java code flow

The Java code task passes control to the JVM task, which interprets the application Java code and sometimes compiles it with the Just in Time compiler (JIT). This flow is also valid for WebSphere for z/OS, where the majority of the product code is written in Java. Both tasks run in the same address space. Figure 3-10 on page 108 shows the flow.

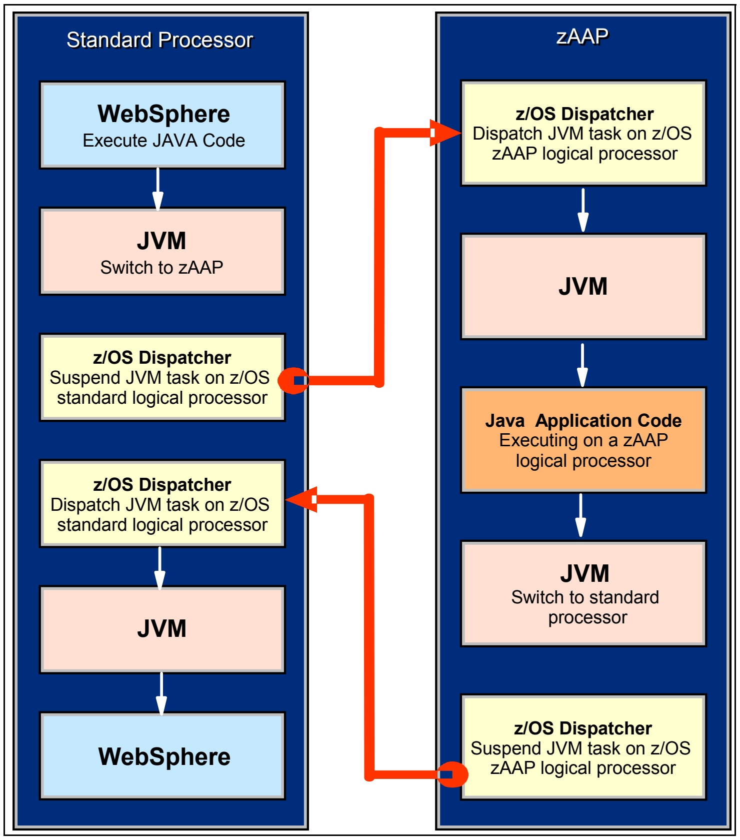

Standard Processor
WebSphere
Execute JAVA Code
JVM
Switch to zAAP
z/OS Dispatcher
Suspend JVM task on z/OS standard logical processor
z/OS Dispatcher
Dispatch JVM task on z/OS standard logical processor
JVM
WebSphere
zAAP
z/OS Dispatcher
Dispatch JVM task on z/OS zAAP logical processor
JVM
Java Application Code
Executing on a zAAP logical processor
JVM
Switch to standard processor
z/OS Dispatcher
Suspend JVM task on z/OS zAAP logical processor

Figure 3-10 zAAP processor and WebSphere

For WebSphere for z/OS, the minimum exploitation level is WebSphere 5.1 with Software Developer Kit (SDK) 1.4 and z/OS 1.6.

3.6.4 zSeries Application Assist Processor positioning

This section describes the positioning of zAAP compared to normal CPU and dispatchable unit management. For enabling JAVA to run on a zAAP, you must be at least at z/OS 1.6 and JVM 1.4.

You can optionally decide which processors are eligible to run Java code at each installation. For the z/OS Operating System, you do this in PARMLIB member IEAOPT*xx*. For the Java code to run on zAAP, there are three parameters set in the startup options of the JVM.

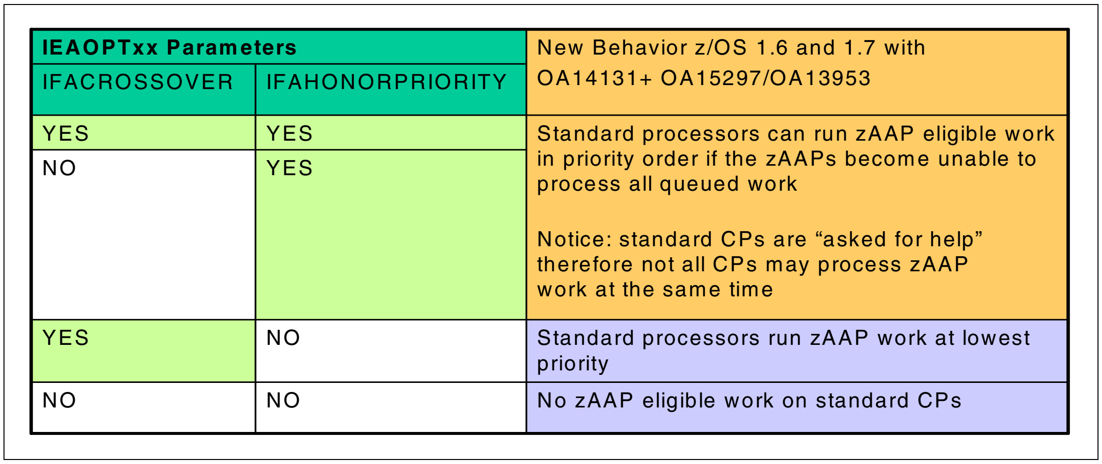
The JVM parameters are:

- Ixifa -off: Java code is not allowed to run on zAAP.
- xifa - force: Projection of zAAP usage is turned on even if there is no physical zAAP, SMF records are written, and Java code is allowed to run on zAAP if available.
- xifa - on: Java code is allowed to run on zAAP (default since JVM 1.4 with APAR PQ86689).

The z/OS parameters in IEAOPT*xx* are (Figure 3-11 on page 109):

- IFACrossover = Yes (default): Eligible JVM code can run both in the logical CPUs’ zAAP or in logical CPUs on standard general purpose CPs.

- IFACrossover = No: Eligible JVM code only runs in the logical CPUs' zAAP in the LPAR, unless the zAAP's CPs are not operational.
- IFAHonorPriority = Yes: WLM instructs the z/OS dispatcher to honor the Dispatchable Unit (DU) dispatching priority independently of their zAAP affinity.
- IFAHonorPriority = No: WLM instructs the z/OS dispatcher to place DUs with zAAP affinity with the lowest dispatching priority on the standard general purpose logical CPU. The dispatching priority of the zAAP affinity DU is honored when dispatching in zAAP.
- ZAAPAWTM: This parameter controls the alternate weight management of SRM. Omit this parameter to let it default, unless you want to disable alternate weight management, which we do not recommend. If alternate weight management is turned off, standard CPs do not help zAAP work.



IEAOPTxx Parameters		New Behavior z/OS 1.6 and 1.7 with OA14131+ OA15297/OA13953
IFACROSSOVER	IFAHONORPRIORITY	
YES	YES	Standard processors can run zAAP eligible work in priority order if the zAAPs become unable to process all queued work Notice: standard CPs are "asked for help" therefore not all CPs may process zAAP work at the same time
NO	YES	
YES	NO	Standard processors run zAAP work at lowest priority
NO	NO	No zAAP eligible work on standard CPs

Figure 3-11 IFACROSSOVER and IFAHONORPRIORITY combinations

In the case of WLC capping, WLM turns off honor priority before capping the LPAR. This event is communicated to the z/OS dispatcher. In this situation, the regular logical CPs process the zAAP DU work with a dispatching priority lower than discretionary. WLM turns on honor priority again when the four-hour rolling average drops below the defined capacity limit.

Note: With IFAHonorPriority=YES specified, work is dispatched on the general purpose logical CPUs even if the zAAPs are not fully loaded. Specifying IFACrossover=NO *and* IFAHonorPriority=NO disables Java eligible work on the general purpose logical CPUs.

3.6.5 WLM zAAP DUs states

Each 250 milliseconds, WLM samples the AS/enclave states. The introduction of zAAP DUs generates changes in the counters:

- CPU Using: DU (from AS/enclave) is found executing on a regular CP (old).
- zAAP Using: DU (from AS/enclave) is found executing on a zAAP (new).
- zAAP_on_CPU Using: DU (from AS/enclave) that is eligible running on a zAAP is executed on a regular CPU (new). This counter is key mainly if the dispatcher honors global priorities for selecting DUs from the zAAP ready DU.
- CPU Delay: DU (from AS/enclave) is delayed for a regular CPU (old).
- zAAP Delay: DU (from AS/enclave) is delayed for a zAAP (new). Depends on the number of zAAPs, the JVM workload, and the parameter HonorPriority.

The RMF CPU activity report tells you how busy zAAPs are.

3.6.6 z9 Integrated Information Processor (zIIP)

zSeries Integrated Information Processor (zIIP) is a characterizable PU (as IFL, ICF, SAP, and zAAP) in the IBM System z machine model z9-109 that is capable to execute eligible database workloads.

Currently, the only product from IBM that supports redirecting some of its work to a zIIP is DB2 V8; however, the API to use the zIIP is available to other Independent Software Vendors (ISVs). The benefits of using a zIIP processor to execute DB2 code are the CPU cycles that are saved on the general purpose processor. It also saves IBM software charges in WLC, because you can have additional capacity without affecting the total MSU/hour of the server.

Work which is enabled to run on a zIIP includes:

- Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) application serving: For applications running on z/OS, UNIX®, Intel®, or Linux on System z™ that access DB2 for z/OS V8 on a System z9™ via DRDA® over a TCP/IP connection, DB2 gives z/OS the necessary information to have portions of these SQL requests directed to the zIIP.
- Data Warehousing applications: Requests that utilize DB2 for z/OS V8 for long-running parallel queries, including complex star schema parallel queries, might have portions of these SQL requests directed to the zIIP when DB2 gives z/OS the necessary information. These queries are typical in data warehousing implementations. The addition of select long-running parallel queries can provide more opportunity for DB2 clients to optimize their environment for Data Warehousing while leveraging the unique qualities of service provided by System z9 and DB2.
- DB2 for z/OS V8 utilities: You can redirect a portion of DB2 utility functions that are used to maintain index structures (LOAD, REORG, and REBUILD INDEX) that typically run during batch d to the zIIP.

> **Note:** zIIP is designed in a way that a program can work with z/OS to have a portion of its enclave Service Request Block (SRB) work directed to the zIIP. The above types of DB2 V8 work execute in enclave SRBs where portions can be sent to the zIIP. Stored procedures and user-defined functions do not use SRBs and, therefore, are not eligible.

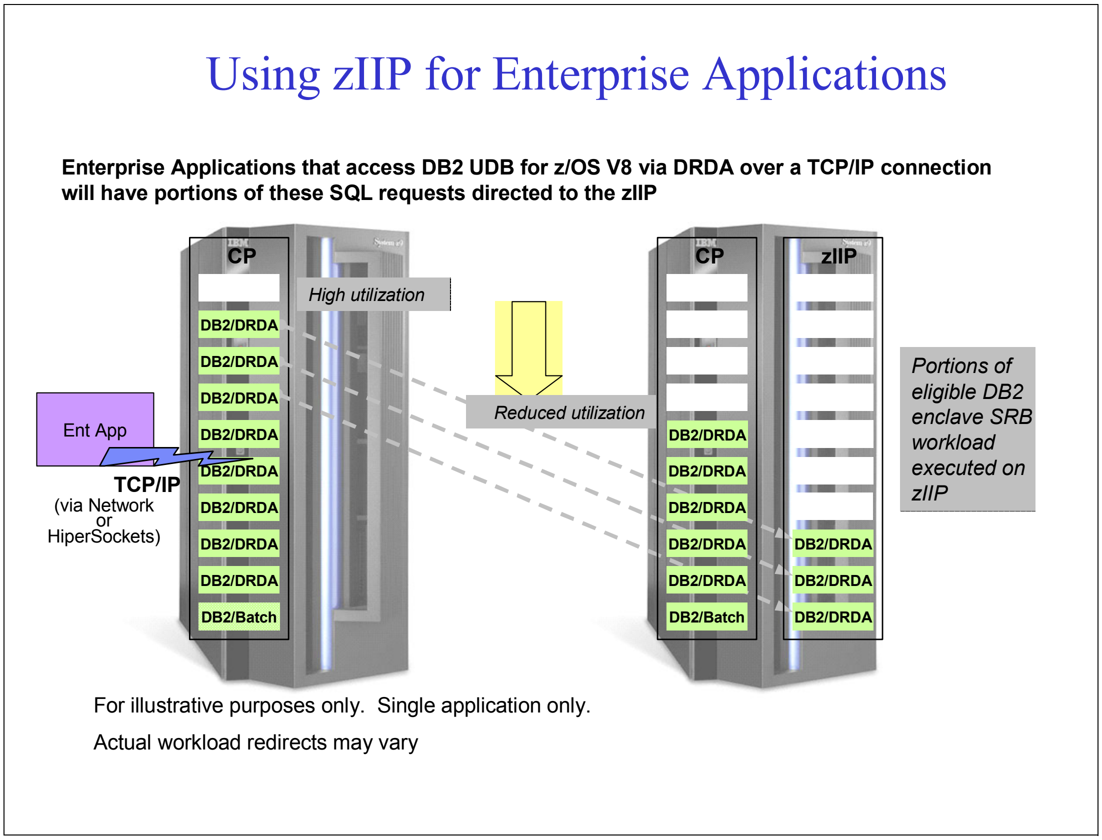
The support for zIIPs starts with DB2 Version 8 and z/OS 1.6.

In many respects, WLM support of zIIPs is equivalent to the zAAP support, but zIIP work is still managed as an extension of general purpose processor work and zIIP work flows over to general purpose processors, because there is no external control.

As with the zAAP processors, zIIP processors do not run z/OS code and cannot be used to IPL. The maximum number of zIIPs is equal to the number of general purpose CPs. You define a zIIP processor in the LPAR's image profile.

z9 Integrated Information Processor and DB2 code flow

In the sample shown in Figure 3-12 on page 111, DB2 requests coming into the system via DRDA over a TCP/IP network are dispatched on z/OS within enclave SRBs. DB2 marks portions of this work as zIIP eligible and the z/OS dispatcher directs this portion to the zIIP.



Using zIIP for Enterprise Applications

Enterprise Applications that access DB2 UDB for z/OS V8 via DRDA over a TCP/IP connection will have portions of these SQL requests directed to the zIIP

For illustrative purposes only. Single application only.

Actual workload redirects may vary

Figure 3-12 zIIP processor and DB2

3.6.7 z9 Integrated Information Processor positioning

You can set the following options for zIIP in IEAOPT*xx*:

- PROJECTPU = YES: This parameter enables the projection of zIIP usage, and SMF records are written even if there are no zIIP processors available. You can analyze these records to evaluate the implementation of a zIIP.
- ZIIPAWTM: This parameter controls the SRM alternate weight management for zIIPs. Omitting this parameter takes the default, unless you want to disable alternate weight management, which we do not recommend. If alternate weight management is turned off, standard CPs do not help zIIP work.

Note: There is no external control for zIIP execution, similar to zAAP.

3.6.8 WLM zIIP DU states

Each 250 milliseconds, WLM samples the enclave states. The introduction of zIIP DUs generates changes in the counters:

- CPU Using: DU (from enclave SRB) is found executing on a regular CP.
- zIIP Using: DU (from enclave SRB) is found executing on a zIIP.
- zIIP_on_CPU Using: DU (from enclave SRB) that is eligible running on a zIIP is executed on a regular CPU.
- CPU Delay: DU (from enclave SRB) is delayed for a regular CPU.

- zIIP Delay: DU (from enclave SRB) is delayed for a zIIP. Depends on the number of zIIPs and the DB2 workload.

The RMF CPU activity report tells you how busy zIIPs are.

Note: You must install APAR OA16005 on z/OS 1.6 and z/OS 1.7 to enable zIIP reporting support.

4

Implementation and workload classifications

This chapter contains general steps and considerations when you implement Workload Manager (WLM) in z/OS. This chapter discusses the steps needed to customize WLM for your installation, for example, the service definitions needed to define your workload and goals.

4.1 Overview

WLM goal mode requires that you are in a single system sysplex (monoplex) or a multisystem sysplex. We assume prior implementation of required sysplex components has been completed. WLM uses a specially formatted WLM couple data set where the service definition will be written. This data set must be accessible by all systems in the sysplex. A WLM ISPF application is used to define the service definitions, such as policy, workload, service classes, classification rules, report classes, and so forth. These definitions represent the performance objectives of the various workloads running in your system.

The following are the steps to implement goal mode:

1. Set up performance objectives.
2. Set up a service definition from the performance objectives.
3. Implement the WLM ISPF administrative application.
4. Allocate the WLM couple data set.
5. Make available the WLM couple data set to the sysplex.
6. Install your service definition on the WLM couple data set.
7. Adjust SMF recording.
8. Activate your service policy.

4.2 WLM implementation

The following sections describe in more detail the steps to implement WLM.

4.2.1 IEAOPT*xx* usage

The following IEAOPT*xx* parameters are used in goal mode:

- Special options:
  - CNTCLIST: TSO/E clist transaction control
  - DVIO: Directed VIO usage
- Adjusting constants:
  - ERV: Enqueue residence constants
  - RMPTTOM: SRM invocation interval constant
- CPU management constants:
  - CCCAWMT: Alternate Wait Management
- Dispatching priority for JES, APPC, and OMVS initiators:
  - INITIMP: Initial dispatching priority
- Pageable storage shortage constants:
  - MCCFXEPR: First 16 megabytes fixed storage%
  - MCCFXTPR: Online storage fixed %
- Central and expanded storage threshold constants:
  - MCCAFCTH: Central storage low and OK threshold
  - MCCAECTH: Expanded storage low and OK threshold

- Selective enablement for I/O constants:
  - CPENABLE: IO interruption threshold
- Swap rate scaling factor:
  - SWAPRSF: Weight cost of doing exchange swap
- zAAP workflow:
  - IFACROSSOVER: zAAP eligible work running on standard CP or zAAP CP only.
  - IFAHONORPRIORITY: zAAP work in priority order or after discretionary work on standard CP (see APARs OA14131, OA15297, and OA13953).
- zIIP Reporting:
  - PROJECTCPU: Enable zIIP usage projection without real zIIP.
- LPAR Vary CPU management:
  - VARYCPU: Availability of LPAR Vary CPU Management
  - VARYCPUMIN: Minimum number of CPs to keep online during WLM LPAR management (available from z/OS 1.6)

The recommended settings for some of the IEAOPT*xx* parameters are:

- MCCAFCTH (avqlow, avqokay)

  Specifies the low and the OK threshold values for central storage. The *lowvalue* indicates the number of frames on the available frame queue when stealing begins. The *okvalue* indicates the number of frames on the available frame queue when stealing ends. With APAR OA144409 installed, customers should take the default values for MCCAFCTH:

  - *avqlow* is the maximum of 400 and .2% of pageable storage
  - *avqokay* is the maximum of 600 and .4% of pageable storage

  Refer to OW55729 and OW55902 for more information about MCCAFCTH settings.

- ERV

  The enqueue residence value (ERV) specifies the absolute number of CPU service units that an address space or enclave is allowed to absorb when it is possibly causing enqueue promotion. This value is independent of the values of the service coefficients. During this enqueue residency time, the address space (including the address space associated with an enclave) is not considered for swap out based on recommendation value analysis. The address space or enclave runs with a high enough priority to guarantee the needed CPU time.

  ERV is in effect for an address space or enclave that meets one of the following criteria:

  - The address space or enclave is enqueued on a system resource needed by another address space.
  - An authorized program in the address space or enclave obtains control of the resource (even if another address space does not need that resource) as a result of issuing a reserve for a DASD device that is SHARED.

  Until WLM enqueue management is fully operational in your environment, the following recommendation might help you avoid certain enqueue-related contention situations in your environment.

  The default value for ERV is 500, which indicates that if an address space is swapped out holding an enqueue, it should be swapped back in until it has either released the enqueue, or has used 500 CPU service units. However, empirical analysis shows that this might not be enough in today’s environments.

Therefore, you should ensure that work, on average, has enough CPU service time to release the enqueue before being swapped back out again. You should verify the correct value by looking at the report and evaluating whether the default value needs any further change. Today, most installations run with an ERV of 50000.

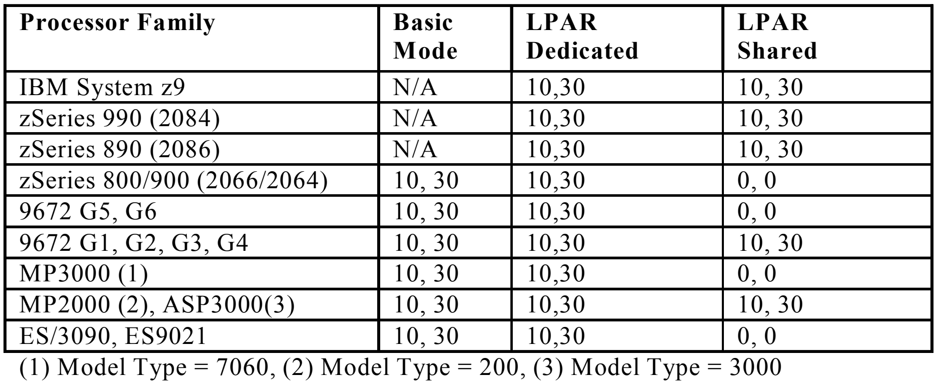
By coding ERV=50000, you can avoid enqueue lockout situations where low priority or discretionary workload can be swapped out while holding an enqueue.

This process is sometimes referred to as *enqueue promotion*. Enqueue promotion is limited by the promotion interval, which is governed by your setting of the ERV value. This limitation is enforced in order to prevent abuse of this facility by users who could intentionally create contention in order to receive better service.

- CPENABLE

  Specifies the low (ICCTPILO) and high (ICCTPIHI) threshold values for the percentage of I/O interruptions to be processed through the test pending interrupt (TPI) instruction path in the I/O supervisor (IOS). SRM uses these thresholds to control the number of processors enabled for I/O interruptions.

  Recommended CPENABLE settings are Figure 4-1.



Processor Family	Basic Mode	LPAR Dedicated	LPAR Shared
IBM System z9	N/A	10,30	10, 30
zSeries 990 (2084)	N/A	10,30	10, 30
zSeries 890 (2086)	N/A	10,30	10, 30
zSeries 800/900 (2066/2064)	10, 30	10,30	0, 0
9672 G5, G6	10, 30	10,30	0, 0
9672 G1, G2, G3, G4	10, 30	10,30	10, 30
MP3000 (1)	10, 30	10,30	0, 0
MP2000 (2), ASP3000(3)	10, 30	10,30	10, 30
ES/3090, ES9021	10, 30	10,30	0, 0

(1) Model Type = 7060, (2) Model Type = 200, (3) Model Type = 3000

Figure 4-1 Recommended CPENABLE settings

  Refer to Flash10337. The link is:

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10337

- CNTCLIST

  This specifies if the individual commands in a TSO/E CLIST are treated as separate commands for transaction control. CNTCLIST=NO means that the CLIST is treated as a single transaction. CNTCLIST=YES means that each command is to be treated as an individual transaction. By specifying CNTCLIST=YES, SRM control of a TSO/E command becomes the same whether the command is executed explicitly or as part of a CLIST.

  Setting this to YES might help you improve the time it takes to log on if you have a long TSO logon CLIST. A huge CLIST results in each CLIST command executing always in the first period of the service class assigned to the user ID. None of the CLIST is executed in the second or third period of the service class.

  Setting this to NO helps you manage your TSO workload with a long CLIST. A CLIST that consumes a lot of CPU is moved to the second performance period. This helps the short TSO transactions that execute in the first period.

- INITIMP

This option is used to set the priority of initiators. In goal mode, the initiator is run in service class SYSSTC with a dispatching priority of 254. The only way to change this dispatching priority is through the INITIMP option. The INITMP value affects only the system on which it has been set. The reason that you might want to lower the importance of the initiators is that some installation exits might use a large amount of CPU if they are invoked at Job selection/initialization.

The valid parameter settings for INITIMP are:

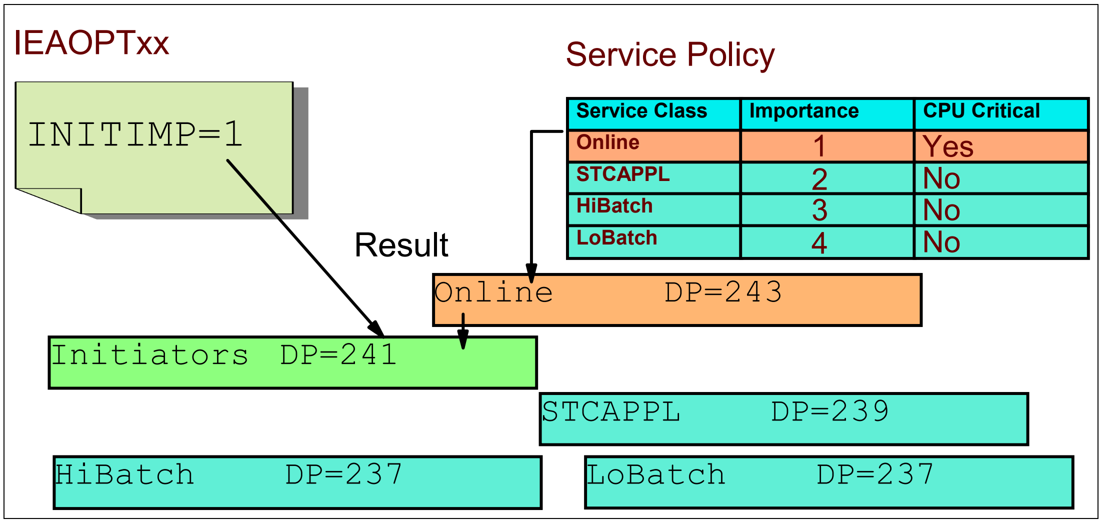
- INITIMP=0: The dispatch priority is set to the dispatch priority of service class SYSSTC, which is 254. This is the default and recommended setting. This setting is functionally equivalent to what took place prior to z/OS V1R5.
- INITIMP=1: The dispatch priority is set lower than the lowest dispatch priority for any CPU-critical service class with an Importance of one.
- INITIMP=2: The dispatch priority is set lower than the lowest dispatch priority for any CPU-critical service class with an Importance of two.
- INITIMP=3: The dispatch priority is set lower than the lowest dispatch priority for any CPU-critical service class with an Importance of three.
- INITIMP=E: The dispatch priority is set to the enqueue promotion dispatch priority, which is calculated dynamically and ensures access to the processor. This does not guarantee that CPU-critical work always has a higher dispatch priority, but it should not be negatively impacted.

Note: In the event that INITIMP is specified as 1, 2, or 3 and there is no service class in the active WLM policy that is defined as CPU-critical with an Importance of 1, 2, or 3, dispatch priority is calculated as if INITIMP=E was specified.

The following are examples to show the interrelationship of these parameters and the dispatching priority.

Example 1: The example shown in Figure 4-2 on page 118 has the following definitions in SYS1.PARMLIB and the WLM service policy:

- The ONLINE service class in the service policy is defined with CPU-critical, Importance 1.
- The INITIMP parameter (INITIMP=1) in the IEAOPT*xx* parmlib member specifies that the initiator should have a dispatching priority (DP) that always has to be lower than CPU-critical and Importance 2 work.
- As a result, all work for ONLINE always has a higher DP than initiators before job execution.
- All other work also has a lower DP than the online service class work.


IEAOPTxx

INITIMP=1

Service Policy

Service Class	Importance	CPU Critical
Online	1	Yes
STCAPPL	2	No
HiBatch	3	No
LoBatch	4	No

Result

Online DP=243

Initiators DP=241

STCAPPL DP=239

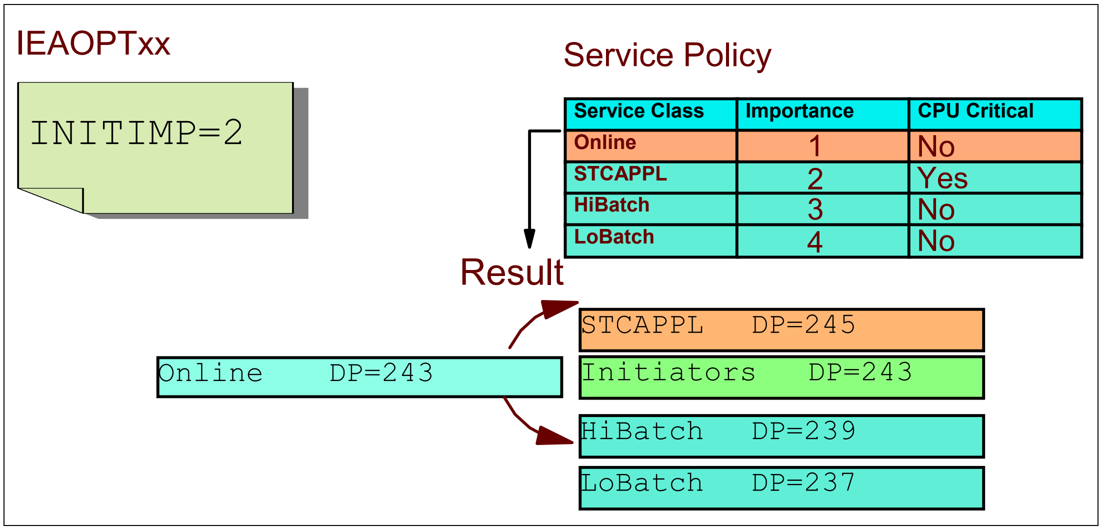
HiBatch DP=237

LoBatch DP=237

Figure 4-2 Dispatching priority example 1

Example 2: The example shown in Figure 4-3 has the following definitions in SYS1.PARMLIB and the WLM service policy:

- The STCAPPL service class in the service policy is defined with CPU-critical and Importance 2.
- The INITIMP specification, (INITIMP=2), in the IEAOPT*xx* parmlib member specifies that the initiator should have a dispatching priority that always has to be lower than CPU-critical and Importance 2 work.
- As a result, the initiators are always below STCAPPL.
- Online work can have any dispatching priority below and above STCAPPL and the other initiators.


IEAOPTxx

INITIMP=2

Service Policy

Service Class	Importance	CPU Critical
Online	1	No
STCAPPL	2	Yes
HiBatch	3	No
LoBatch	4	No

Result

STCAPPL DP=245

Online DP=243

Initiators DP=243

HiBatch DP=239

LoBatch DP=237

Figure 4-3 Dispatching priority example 2

Refer to *z/OS MVS Initialization and Tuning Reference*, SA22-7592, for more information about the specification of the INITIMP IEAOPT*xx* parameter.

4.2.2 WLM ISPF application

You use the WLM ISPF application to define your service definition. These definitions are stored in ISPF tables. You can save these ISPF tables in a PDS. After you have completed your service definitions, you need to install these definitions into the WLM couple data set (CDS) before you can activate it. Refer to Figure 4-4.

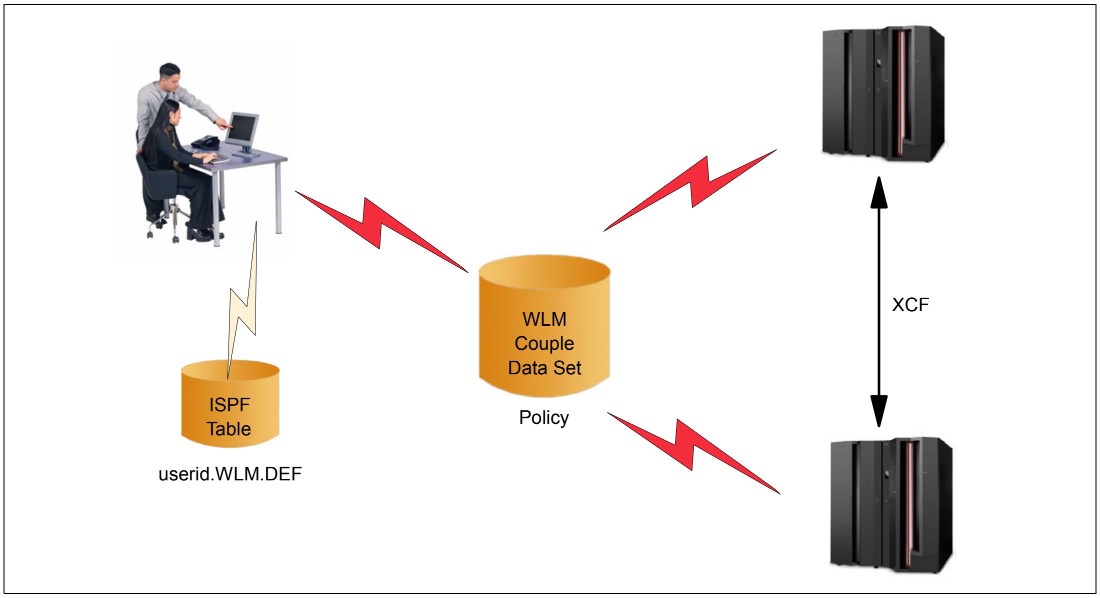


Figure 4-4 WLM ISPF application

Note that the WLM ISPF application uses the data set name *userid.*WLM.SAVE*xx* (where *userid* is the TSO user ID accessing the application and *xx* is any two-digit number). WLM ISPF allocates this data set for recovery purposes. WLM deletes this data set when the ISPF application is ended. Do not use this naming convention to prevent conflict with WLM.

Security access

You can control access to the WLM ISPF application by defining the facility class MVSADMIN.WLM.POLICY in RACF. When a user tries to access the WLM ISPF application, it checks whether that user has access to this facility class. You must determine who needs access to it. Access levels can be either:

- READ

  Recommended access for system programmers, system operators, help desk support personnel, and production control staff. With READ access, the user can:

  – Define, display, and modify a service definition stored in an MVS partitioned data set.
  – Extract a service definition from a WLM couple data set.
  – Display the service definition.
  – Print the service definition.

- UPDATE

  Recommended access for the service administrator, some system programmers, and possibly the performance administrator. With UPDATE access, the user can:

  – Perform all the functions available for READ access.
  – Allocate a WLM couple data set to the sysplex (use the SETXCF command).
  – Install a service definition to a WLM couple data set.
  – Activate a service policy.

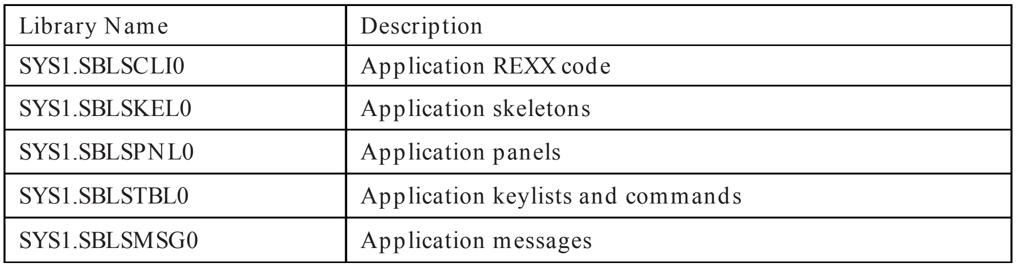
You can use RDEFINE to add a profile to the RACF database. Then, use PERMIT to permit or deny access to the RACF profile. Do not forget to issue the SETROPTS REFRESH command after the PERMIT command to refresh the RACF database and activate the changes that you have made.

Example 4-1   RACF definition sample

```
Example of using RDEFINE to define the facility class for WLM ISPF application:
RDEFINE FACILITY MVSADMIN.WLM.POLICY UACC(NONE) NOTIFY(user)
Where:
user Indicates the user that should be notified of unauthorized access
attempts to the database.

Example of using PERMIT to allow access to the WLM ISPF application:
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user) ACCESS(READ)
PERMIT MVSADMIN.WLM.POLICY CLASS(FACILITY) ID(user) ACCESS(UPDATE)

Where:
user Indicates the user or user group that needs access to the WLM ISPF
application.
ACCESS Indicates the type of access, either READ or UPDATE
```

Starting your WLM ISPF application

To start the WLM application, use the TSO/E REXX exec IWMARIN0. The exec concatenates (via LIBDEF and ALTLIB) the following libraries necessary to run the application (Figure 4-5).



Library Name	Description
SYS1.SBLSCLI0	Application REXX code
SYS1.SBLSKEL0	Application skeletons
SYS1.SBLSPNL0	Application panels
SYS1.SBLSTBL0	Application keylists and commands
SYS1.SBLSMSG0	Application messages

Figure 4-5   WLM ISPF libraries

The exec also allocates some MVS partitioned data sets for the service definition using TSO ALLOCATE, and then invokes the WLM panels. If you have different data set naming conventions for your IPCS/WLM libraries, or if you use storage-managed data sets, you should use the WLM application exits IWMAREX1 and IWMAREX2. For more information about how to code the exits, see Appendix A, “Customizing the WLM ISPF Application” in *z/OS MVS Planning: Workload Management,* SA22-7602.

To start the application, specify:

```
ex ’SYS1.SBLSCLI0(IWMARIN0)’
```

To add the WLM application to the ISPF primary panel, add the WLM option to display as part of the menu, then specify:

```
WLM,’CMD(%IWMARIN0) NEWAPPL(IWMP) PASSLIB’
```

For more information about IWMARIN0 and for examples about how to start the application specifying the WLM exits, see Appendix A, “Customizing the WLM ISPF Application” in *z/OS MVS Planning: Workload Management,* SA22-7602.

4.2.3 WLM couple data set management

In this section, we discuss WLM couple data set management.

Allocating the WLM couple data set

You need to define a WLM couple data set for storing the service definition information. If you are running a sysplex with mixed release levels, you should format the WLM couple data set from the highest level system. This allows you to use the current level of the WLM ISPF application. You can continue to use the downlevel WLM ISPF application on downlevel systems provided that you do not attempt to exploit the new function.

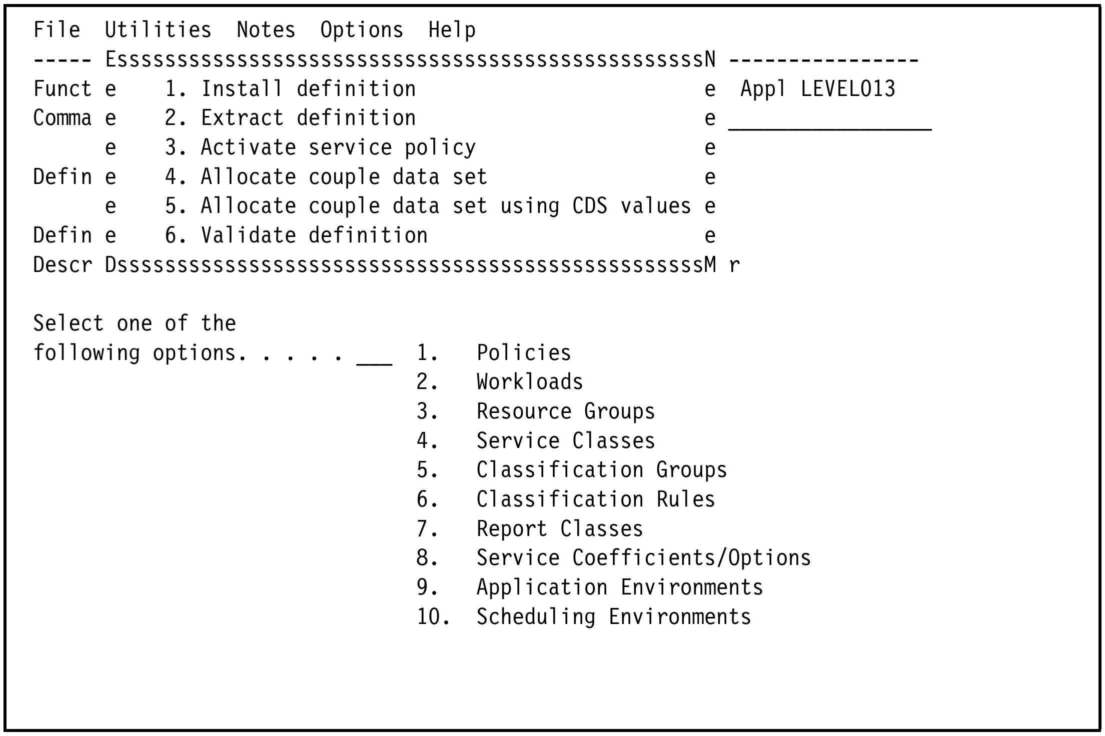
There is no need to allocate the WLM CDS in high performance DASD devices, because there is not much I/O to this data set. The only time this data set is accessed is during IPL and when the service definition is installed into or extracted from the WLM CDS.

To allocate the WLM couple data set, you can either use the facility provided in the WLM ISPF application, or you can run an XCF utility in batch. For each case, you need to estimate how many workload management objects you are storing on the WLM couple data set. You must provide an approximate number of:

- Policies in your service definition
- Workloads in your service definition
- Service classes in your service definition

The values that you define are converted to space requirements for the WLM couple data set that is allocated. The total space is not strictly partitioned according to these values. For most of these values, you can consider the total space to be one large pool.

To use the WLM ISPF application to allocate the WLM Couple data set (CDS), go to the **Utilities** option on the menu bar (PF10). See Figure 4-6.



```
File  Utilities  Notes  Options  Help
----- EsssssssssssssssssssssssssssssssssssssssssssssssssN ----------------
Funct e    1. Install definition                        e  Appl LEVEL013
Comma e    2. Extract definition                        e _________________
      e    3. Activate service policy                   e
Defin e    4. Allocate couple data set                  e
      e    5. Allocate couple data set using CDS values e
Defin e    6. Validate definition                       e
Descr DsssssssssssssssssssssssssssssssssssssssssssssssssM r

Select one of the
following options. . . . . ___  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
```

Figure 4-6 WLM ISPF Utilities option (PF10)

You can also use the XCF utility to allocate the WLM CDS. You can use the JCL provided in the IWMFTCDS member of SYS1.SAMPLIB. See Figure 4-7.

```
//FMTCDS JOB MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IXCL1DSU
//STEPLIB  DD   DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
   DEFINEDS SYSPLEX(PLEX1)
            DSN(SYS1.WLMCDS01) VOLSER(TEMPAK)
            MAXSYSTEM(32)
            CATALOG
        DATA TYPE(WLM)
             ITEM NAME(POLICY) NUMBER(10)
             ITEM NAME(WORKLOAD) NUMBER(35)
             ITEM NAME(SRVCLASS) NUMBER(30)
             ITEM NAME(APPLENV) NUMBER(100)
             ITEM NAME(SCHENV) NUMBER(100)
/*
```

Figure 4-7 Sample JCL to allocate WLM CDS

For details about allocating your WLM CDS, refer to *z/OS MVS Planning: Workload Management,* SA22-7602.

Managing your WLM CDS

In the future, you might need to reallocate your WLM CDS, because the current size is already too small or maybe you are upgrading to a new z/OS version and you need to migrate your old CDS to the new z/OS CDS.

You can use a series of SETXCF commands to switch the current primary and alternate CDS to the newly allocated CDS. After you do this, all the systems now use the newly allocated CDS. See Example 4-2.

Example 4-2 Example of making reallocated couple data sets available

```
1. Allocate two new couple data sets as described in previous section.
For this example, it is assumed you want a primary and an alternate couple data
set, and that the names of the new data sets are SYS1.WLMP residing on volume
SYS001 and SYS1.WLMA residing on volume SYS002.

2. Make SYS1.WLMP the alternate using the command:
SETXCF COUPLE,TYPE=WLM,ACOUPLE=(SYS1.WLMP,SYS001)
As part of this processing, SETXCF copies the contents of the current primary WLM
couple data set to SYS1.WLMP, which now is the new alternate.

3. Switch SYS1.WLMP to primary using the command:
SETXCF COUPLE,TYPE=WLM,PSWITCH

4. Now make SYS1.WLMA the new alternate using the command:
SETXCF COUPLE,TYPE=WLM,ACOUPLE=(SYS1.WLMA,SYS002)
As in Step 1 above, this causes the contents of the new primary WLM couple data
set SYS1.WLMP to be copied to the new alternate SYS1.WLMA.
```

If you are making newly formatted WLM couple data sets available to the sysplex, you can continue to use your old WLM ISPF application to modify, install, and activate your service definition (as long as you do not exploit new functions), or you can switch to the new WLM ISPF application that came with your new version of z/OS.

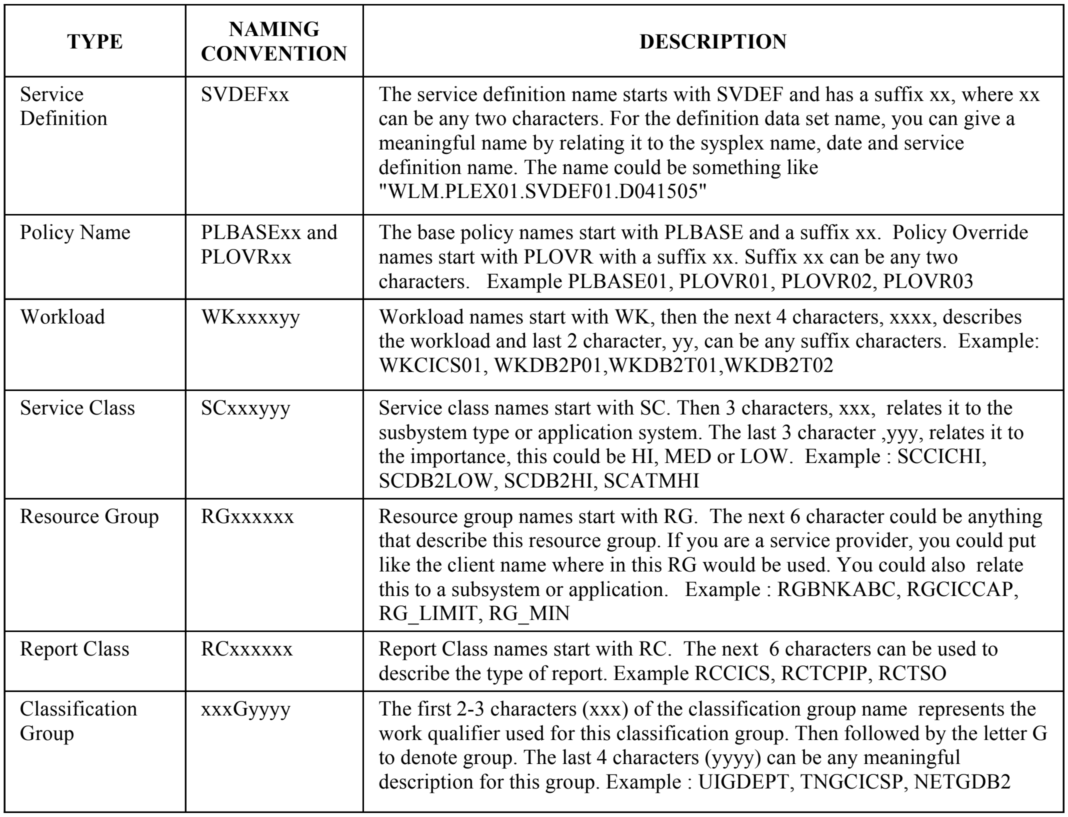
4.2.4 Specifying your service definition

This section discusses how to set up your service definition. A service definition represents the business goal and performance objectives that WLM uses to manage the workload of the system. It consists of one or more service policies, workloads, service classes, and classification rules. Optionally, it might also include Resource Groups, report classes, scheduling, and Application Environments.

Before you actually start defining the different constructs used in the WLM policy, it is important to plan first the naming convention that you want to use. There is no such thing as a best practice naming convention. It is up to your installation to decide what is right and what makes sense to you. We are not here to tell you what is the “right” way to do this, but rather to get you to start thinking about this. Keep this in mind that your naming convention should help you identify different elements, such as service classes, Resource Group, resource classes, classification group, workload, and so forth. It is a good idea to get a group of people together to discuss this.

In goal mode, the names that you choose will be viewed by more people now. It is no longer hidden in the IPS/ICS parmlib. Always give meaningful names, but also be sensitive to your clients. If you are running a service bureau, some of your client's executives might not be too happy to see that their job is always running on a service class called BATSLOW. If you like to use your goals as part of the name, for example BATVEL30, this might be a meaningful name, but remember that every time you change your goals, you also need to change your name.

Figure 4-8 on page 124 provides a sample naming convention to get you started.

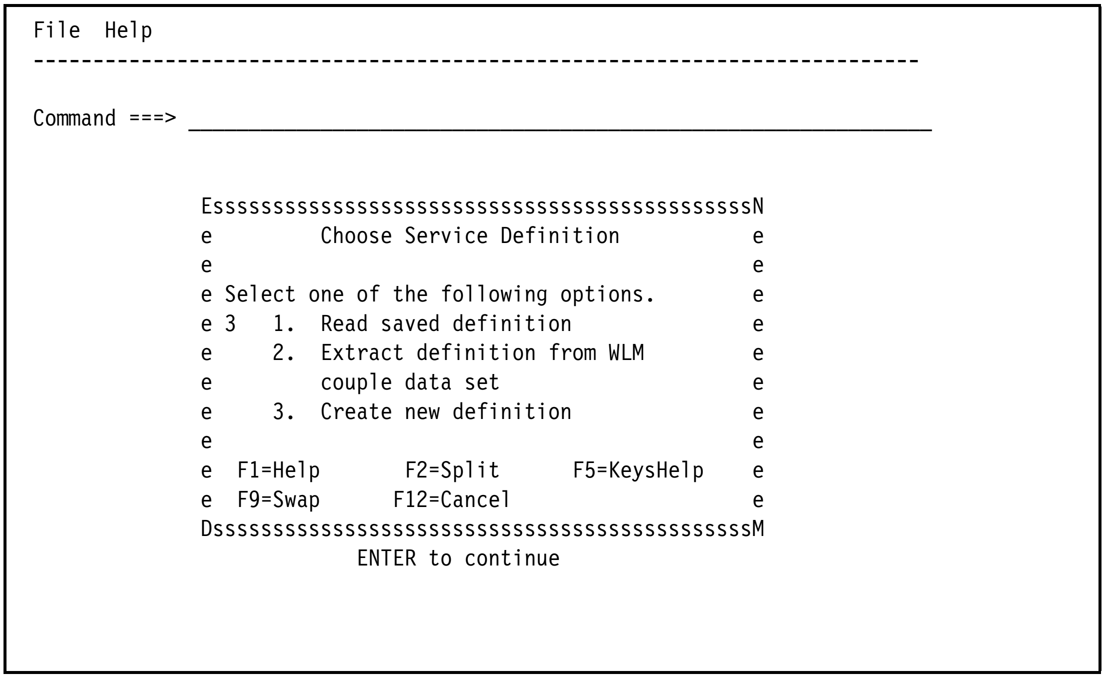


TYPE	NAMING CONVENTION	DESCRIPTION
Service Definition	SVDEFxx	The service definition name starts with SVDEF and has a suffix xx, where xx can be any two characters. For the definition data set name, you can give a meaningful name by relating it to the sysplex name, date and service definition name. The name could be something like "WLM.PLEX01.SVDEF01.D041505"
Policy Name	PLBASExx and PLOVRxx	The base policy names start with PLBASE and a suffix xx. Policy Override names start with PLOVR with a suffix xx. Suffix xx can be any two characters. Example PLBASE01, PLOVR01, PLOVR02, PLOVR03
Workload	WKxxxxyy	Workload names start with WK, then the next 4 characters, xxxx, describes the workload and last 2 character, yy, can be any suffix characters. Example: WKCICS01, WKDB2P01,WKDB2T01,WKDB2T02
Service Class	SCxxxyyy	Service class names start with SC. Then 3 characters, xxx, relates it to the susbystem type or application system. The last 3 character ,yyy, relates it to the importance, this could be HI, MED or LOW. Example : SCCICHI, SCDB2LOW, SCDB2HI, SCATMHI
Resource Group	RGxxxxxx	Resource group names start with RG. The next 6 character could be anything that describe this resource group. If you are a service provider, you could put like the client name where in this RG would be used. You could also relate this to a subsystem or application. Example : RGBNKABC, RGCICCAP, RG_LIMIT, RG_MIN
Report Class	RCxxxxxx	Report Class names start with RC. The next 6 characters can be used to describe the type of report. Example RCCICS, RCTCPIP, RCTSO
Classification Group	xxxGyyyy	The first 2-3 characters (xxx) of the classification group name represents the work qualifier used for this classification group. Then followed by the letter G to denote group. The last 4 characters (yyyy) can be any meaningful description for this group. Example : UIGDEPT, TNGCICSP, NETGDB2

Figure 4-8 Sample naming convention for the different WLM constructs

After you decide on your installation naming convention, you need to plan for your service definition. You need to understand your business goals. Are there any current service level agreements with your users? Which applications are the business critical applications? What are their relative importances to one another? What is the current performance of my system and sysplex?

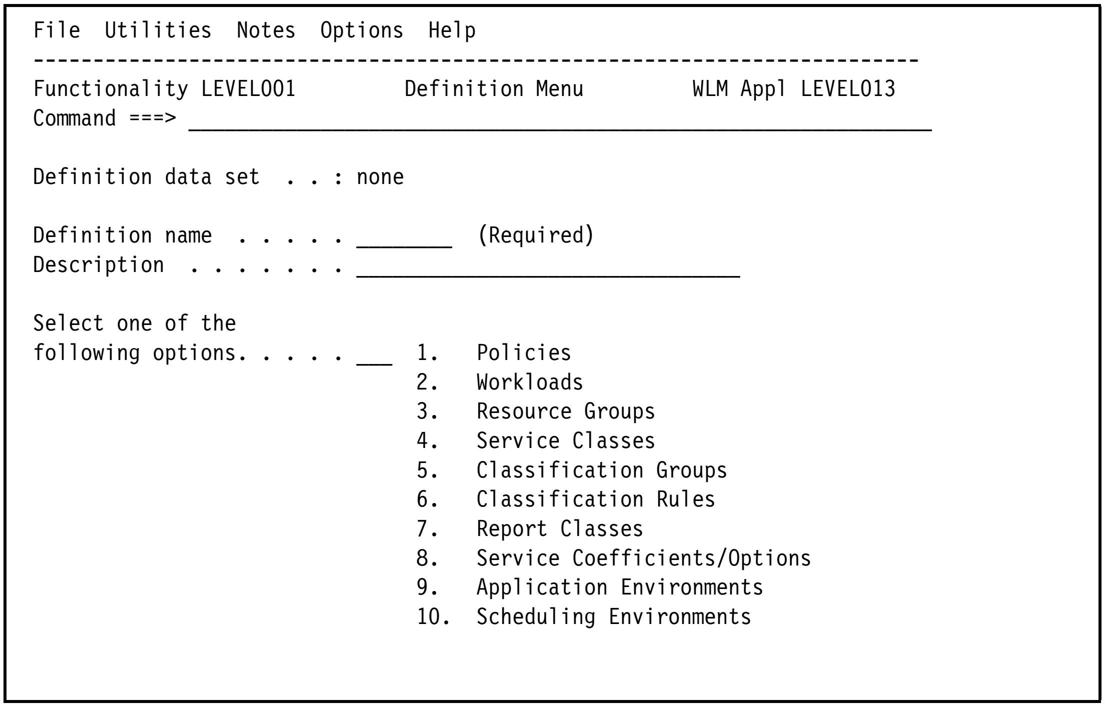
When you start the WLM ISPF application, the first panel asks you to choose your service definition. You can either create a new one, load the service definition from an existing PDS, or extract the service definition from the current WLM CDS. See Figure 4-9 on page 125.



```
File  Help
--------------------------------------------------------------------------

Command ===> ______________________________________________________________


              EssssssssssssssssssssssssssssssssssssssssssssssN
              e         Choose Service Definition            e
              e                                              e
              e Select one of the following options.         e
              e 3   1.  Read saved definition                e
              e     2.  Extract definition from WLM          e
              e         couple data set                      e
              e     3.  Create new definition                e
              e                                              e
              e  F1=Help       F2=Split      F5=KeysHelp     e
              e  F9=Swap      F12=Cancel                     e
              DssssssssssssssssssssssssssssssssssssssssssssssM
                           ENTER to continue
```

Figure 4-9 WLM ISPF application: Choosing your Service Definition

Choose option **3** if you want to create a new definition. After you press Enter, you go to the main Definition Menu (Figure 4-10).



```
File  Utilities  Notes  Options  Help
--------------------------------------------------------------------------
Functionality LEVEL001        Definition Menu        WLM Appl LEVEL013
Command ===> ______________________________________________________________

Definition data set  . . : none

Definition name  . . . . . ________  (Required)
Description  . . . . . . . ________________________________

Select one of the
following options. . . . . ___  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
```

Figure 4-10 WLM Definition Menu

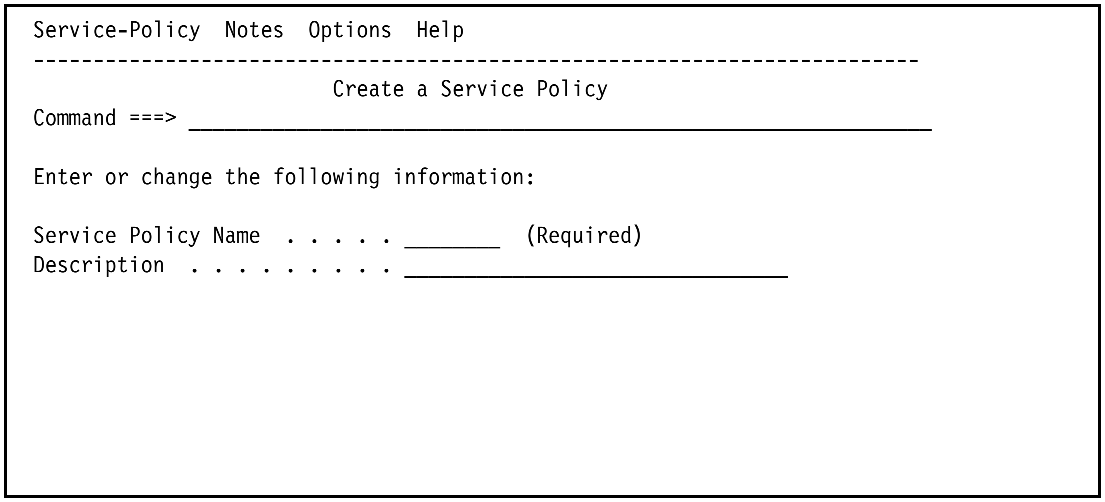
You need to give your service definition a name and a short description. From this menu, you can start defining the policy, service classes, classification rules, workloads, and so forth. On the menu bar, you can use the Notes options to document the changes made to your service definition. There is a notepad where you can edit and track your changes to the definitions.

You can work with one service definition at a time. If you have multiple service definitions, you can save each one in a separate PDS. To use the service definition, it needs to be installed into the WLM CDS, and the service policy that is defined within the service definition needs to be activated. can be used by WLM, it has to install onto the WLM CDS and the service policy within the service definition activated.

4.2.5 Service policy definition

A *service policy* is a named collection of overrides to the goals set in the base service definition. You can only have one active policy at any one time. A policy applies to all the work running in a sysplex. At certain times, there might be a need to change performance goals. For example, your prime shift goals might not be the same as during off-shift. In this case, you can set up another policy that is more appropriate to your off-shift hours.

When you create your service definition, you might choose to define one empty default policy with no overrides at all. Next, create your workloads and service classes. Then, determine how and when your service classes can have different goals at different times. Define additional policies with the appropriate overrides for these time periods. When you activate your override policy, the overrides are merged with the service class and Resource Group specifications in the service definition. Figure 4-11 shows the WLM ISPF panel to define your policy.

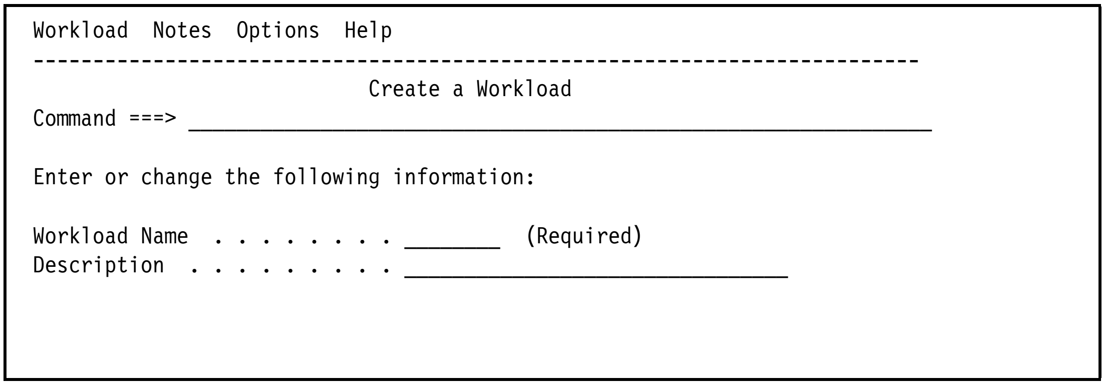


```
Service-Policy  Notes  Options  Help
---------------------------------------------------------------------------
                         Create a Service Policy
Command ===> ______________________________________________________________

Enter or change the following information:

Service Policy Name  . . . . . ________  (Required)
Description  . . . . . . . . . ________________________________
```

Figure 4-11 WLM ISPF service policy definition

4.2.6 Workload specification

Workload is a way to group work that is meaningful for the installation to monitor. Logically, a workload is a group of one or more service classes. You associate a service class to a workload by using the WLM ISPF service class menu. Figure 4-12 on page 127 shows the Create a Workload panel.



```
 Workload  Notes  Options  Help
 ---------------------------------------------------------------------------
                             Create a Workload
 Command ===> ______________________________________________________________

 Enter or change the following information:

 Workload Name  . . . . . . . . ________  (Required)
 Description  . . . . . . . . . ________________________________
```

Figure 4-12   Create a Workload

You only use a workload to group service classes for reporting purposes. It does not affect workload management. You can arrange workloads by subsystem, such as CICS or IMS, by major application, such as Production, Batch, or Office, or by line of business, such as ATM, Inventory, or department.

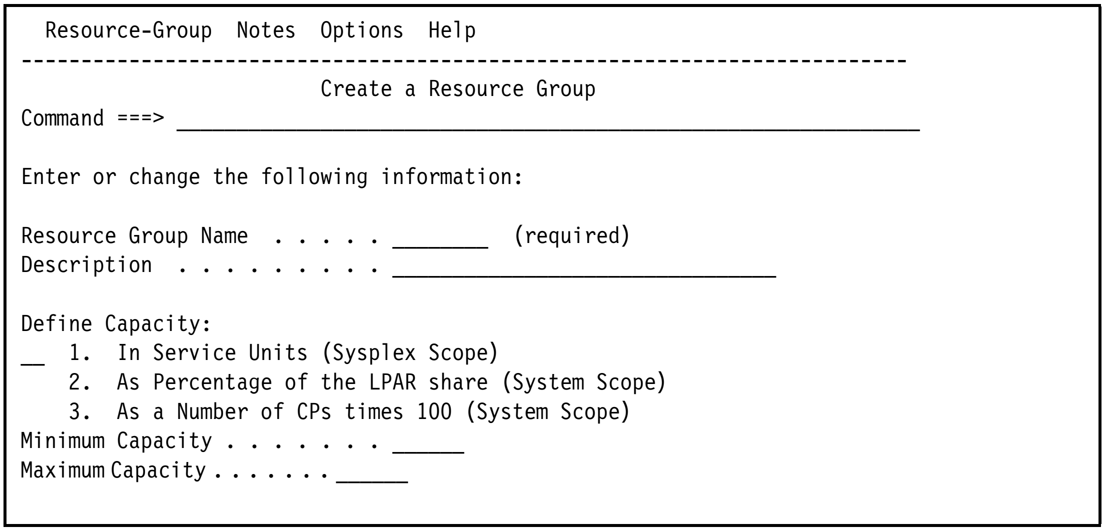
4.2.7 Resource Group specification

A *Resource Group* defines the processor capacity available to groups of service classes in the sysplex. Resource Groups are not required; however, they are used to modify normal WLM management. You can use a Resource Group to:

- Limit the amount of processing capacity available to one or more service classes.
- Set a minimum processing capacity for one or more service classes in the event that the work does not achieve its goals.

You can specify the minimum and maximum capacity of a Resource Group. The minimum and maximum apply to all systems in the sysplex. You can assign only one Resource Group to a service class, but you can have multiple service classes associated with one Resource Group. You can define up to 32 Resource Groups per service definition.

Figure 4-13 shows the Create a Resource Group panel.



```
  Resource-Group  Notes  Options  Help
---------------------------------------------------------------------------
                          Create a Resource Group
Command ===> ______________________________________________________________

Enter or change the following information:

Resource Group Name  . . . . . ________  (required)
Description  . . . . . . . . . ________________________________

Define Capacity:
__  1.  In Service Units (Sysplex Scope)
    2.  As Percentage of the LPAR share (System Scope)
    3.  As a Number of CPs times 100 (System Scope)
Minimum Capacity . . . . . . . ______
Maximum Capacity . . . . . . ______
```

Figure 4-13   Create a Resource Group

For more detail about Resource Group parameters, see 3.1.1, “Resource Groups” on page 76.

4.2.8 Working with service classes

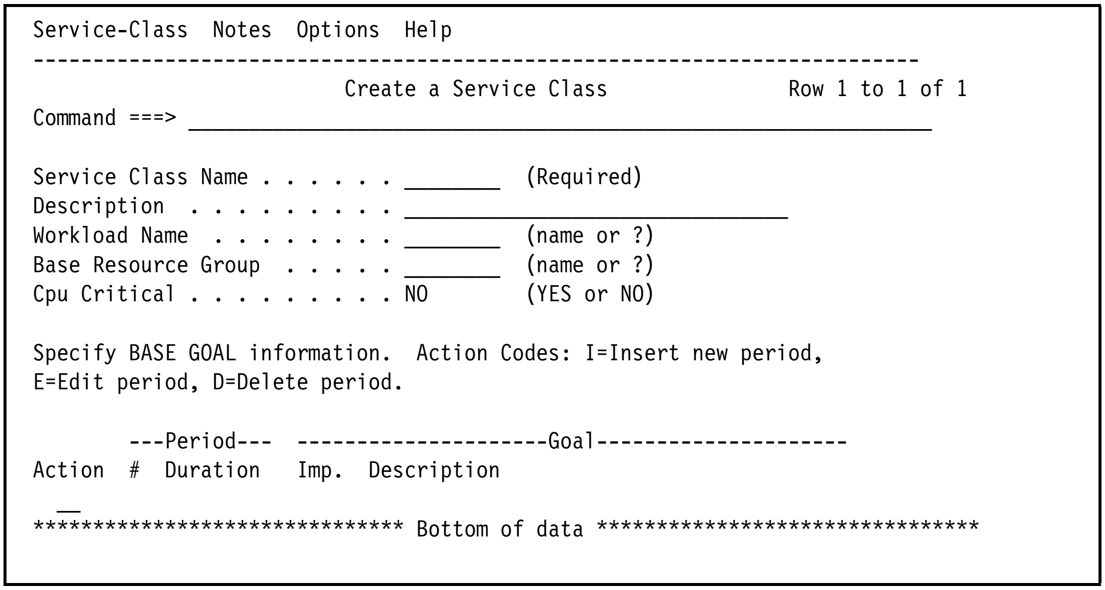
A *service class* is a named group of work with similar performance goals, resource requirements, or business importance. In the service class, you assign the goals of the work and their relative importance. You can choose from four types of goals:

- Average response time
- Response time with percentile
- Execution velocity
- Discretionary

After you select the type of goal that is most appropriate to the work, specify a goal, an importance, and a duration for a performance period. *Performance periods* are available for work that has variable resource requirements and for which your goals change as the work uses more resources. *Duration* is the amount of service units that a period should consume before going on to the next goal. You can specify up to eight performance periods. Multiple performance periods are allowed, except for CICS subsystems and IMS subsystems.

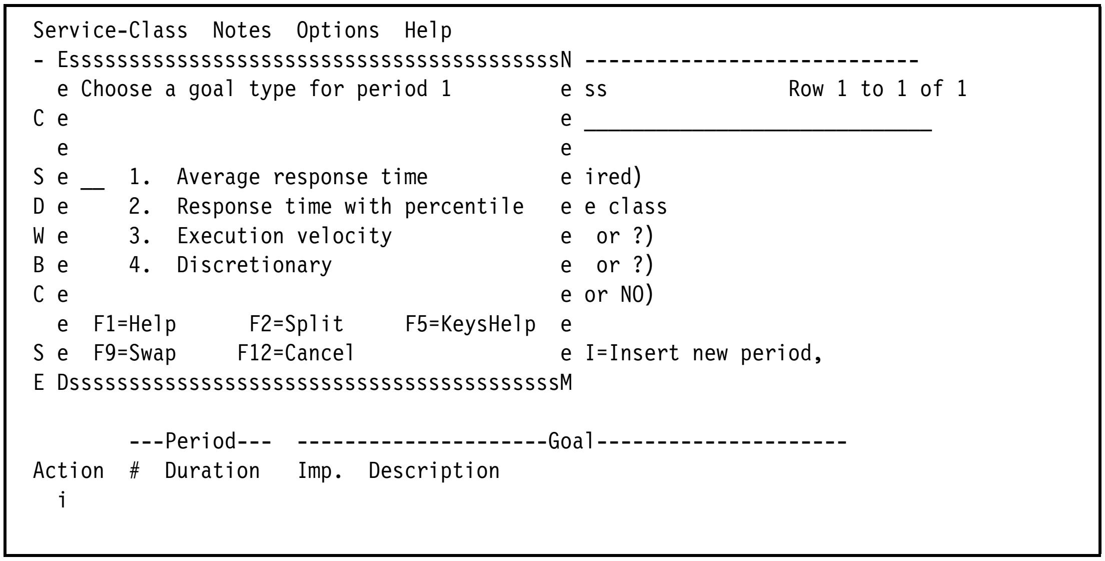
In the service class panel, you relate the service class to a particular workload and Resource Group. To associate an incoming work with a particular service class, you must set up your classification rules. Make sure you create your service classes before creating classification rules. You can create your service classes using the main definition panel, option 4. This is a relatively minor task (compared to the planning and preparation).

Figure 4-14 shows the Create a Service Class panel. A pop-up window appears when you choose the Insert New Period (I) or Edit Period (E) action code. Refer to Figure 4-15 on page 129.



```
 Service-Class  Notes  Options  Help
 --------------------------------------------------------------------------
                           Create a Service Class            Row 1 to 1 of 1
 Command ===> ______________________________________________________________

 Service Class Name . . . . . . ________  (Required)
 Description  . . . . . . . . . ________________________________
 Workload Name  . . . . . . . . ________  (name or ?)
 Base Resource Group  . . . . . ________  (name or ?)
 Cpu Critical . . . . . . . . . NO        (YES or NO)

 Specify BASE GOAL information.  Action Codes: I=Insert new period,
 E=Edit period, D=Delete period.

         ---Period---  ---------------------Goal---------------------
 Action  #  Duration   Imp.  Description
   __
 ******************************* Bottom of data ********************************
```

Figure 4-14   Create a Service Class



```
Service-Class  Notes  Options  Help
- EsssssssssssssssssssssssssssssssssssssssN ----------------------------
  e Choose a goal type for period 1        e ss             Row 1 to 1 of 1
C e                                        e ____________________________
  e                                        e
S e __  1.  Average response time          e ired)
D e     2.  Response time with percentile  e e class
W e     3.  Execution velocity             e  or ?)
B e     4.  Discretionary                  e  or ?)
C e                                        e or NO)
  e  F1=Help      F2=Split     F5=KeysHelp  e
S e  F9=Swap     F12=Cancel                e I=Insert new period,
E DsssssssssssssssssssssssssssssssssssssssM

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  i
```

Figure 4-15 Setting your performance goals

After you complete the definition of the various service classes and their goals, you can now proceed to classify your workload to service classes.

4.2.9 Classification rules

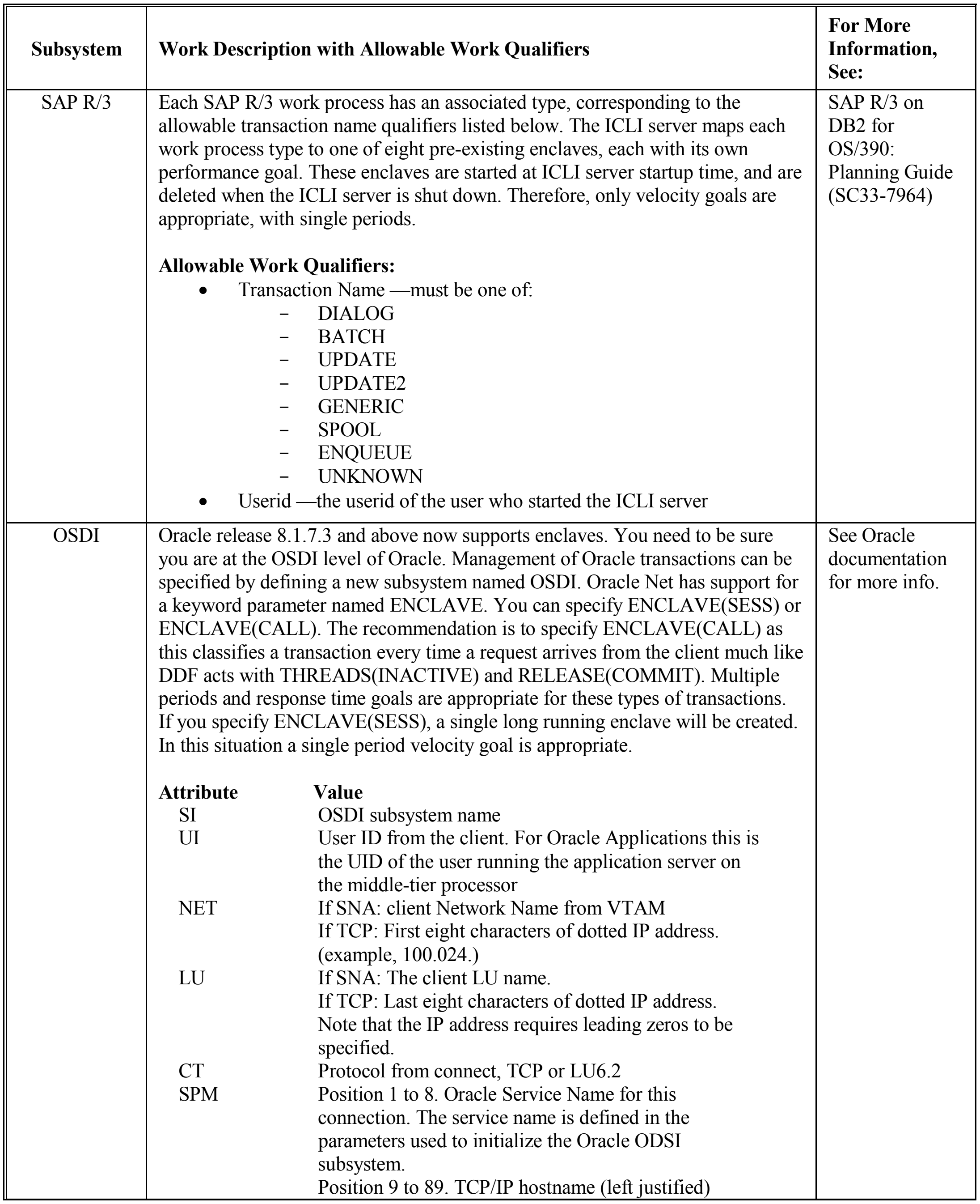
WLM uses classification rules to map work coming into the system to specific service class and report class. This classification is based on work qualifiers. A *work qualifier* identifies a work request to the system. The first qualifier is the subsystem type that receives the work request. When you define the rules, start with the service classes that you have defined, and look at the type of work that they represent. Determine which subsystem types process the work in each service class. Table 4-1 shows the IBM-supplied subsystem type and Figure 4-16 on page 131 shows the non-IBM subsystem type.

Table 4-1 Subsystem type supplied by IBM

Subsystem type	Work description	Enclave, address space, or LPAR
ASCH	The work requests include all APPC transaction programs scheduled by the IBM-supplied APPC/MVS transaction scheduler.	Address space
CB	The work requests include all Component Broker client object method requests.	Enclave
CICS	The work requests include all transactions processed by CICS Version 4 and above.	See footnote [a]
DB2	The work requests include only the queries that DB2 has created by splitting a single, larger query and distributed to remote systems in a sysplex. The local piece of a split query, and any other DB2 work, is classified according to the subsystem type of the originator of the request (for example, DDF, TSO, or JES).	Enclave
DDF	The work requests include all DB2 distributed data facility (DB2 Version 4 and above) work requests.	Enclave

Subsystem type	Work description	Enclave, address space, or LPAR
EWLM	Allows you to assign EWLM transaction and service classes by name to WLM service classes.	Enclave
IMS	The work requests include all messages processed by IMS Version 5 and above.	See footnote a
IWEB	The work requests include all requests from the Web that are serviced by the Internet Connection Server (ICS), Domino® Go Webserver, or IBM HTTP Server for z/OS. These requests also include those handled by the Secure Sockets Layer (SSL). This also includes transactions handled by the Fast Response Cache Accelerator.	Enclave
JES	The work requests include all jobs that JES2 or JES3 initiates.	Address space
LSFM	The work requests include all work from LAN Server for MVS.	Enclave
MQ	The work requests include MQSeries Workflow work, such as new client server requests, activity executions, activity responses, and subprocess requests.	Enclave
NETV	The work requests include NetView network management subtasks and system automation (SA) subtasks created by Tivoli® NetView for z/OS.	Enclave
OMVS	The work requests include work processed in z/OS UNIX System Services forked children address spaces. (Work that comes from an enclave is managed to the goals of the originating subsystem.)	Address space
SOM	The work requests include all SOM client object class binding requests.	Enclave
STC	The work requests include all work initiated by the START and MOUNT commands. STC also includes system component address spaces, such as the TRACE and PC/AUTH address spaces.	Address space
TSO	The work requests include all commands issued from foreground TSO sessions.	Address space
SYSH	Identifies non-z/OS partitions (that is, the Linux partition) in the LPAR cluster, which need to be managed by WLM according to business goals set for the partition.	LPAR

a. CICS and IMS do not use enclaves, but use a different set of WLM services to provide transaction management. CICS and IMS transactions can be assigned only response time goals (either percentile or average) within single period service classes. You can choose whether WLM will manage the CICS and IMS regions according to the response time goals of the transaction or according to the goals of the service class assigned to the region.



Subsystem	Work Description with Allowable Work Qualifiers	For More Information, See:
SAP R/3	Each SAP R/3 work process has an associated type, corresponding to the allowable transaction name qualifiers listed below. The ICLI server maps each work process type to one of eight pre-existing enclaves, each with its own performance goal. These enclaves are started at ICLI server startup time, and are deleted when the ICLI server is shut down. Therefore, only velocity goals are appropriate, with single periods. **Allowable Work Qualifiers:** • Transaction Name —must be one of: – DIALOG – BATCH – UPDATE – UPDATE2 – GENERIC – SPOOL – ENQUEUE – UNKNOWN • Userid —the userid of the user who started the ICLI server	SAP R/3 on DB2 for OS/390: Planning Guide (SC33-7964)
OSDI	Oracle release 8.1.7.3 and above now supports enclaves. You need to be sure you are at the OSDI level of Oracle. Management of Oracle transactions can be specified by defining a new subsystem named OSDI. Oracle Net has support for a keyword parameter named ENCLAVE. You can specify ENCLAVE(SESS) or ENCLAVE(CALL). The recommendation is to specify ENCLAVE(CALL) as this classifies a transaction every time a request arrives from the client much like DDF acts with THREADS(INACTIVE) and RELEASE(COMMIT). Multiple periods and response time goals are appropriate for these types of transactions. If you specify ENCLAVE(SESS), a single long running enclave will be created. In this situation a single period velocity goal is appropriate. **Attribute** **Value** SI OSDI subsystem name UI User ID from the client. For Oracle Applications this is the UID of the user running the application server on the middle-tier processor NET If SNA: client Network Name from VTAM If TCP: First eight characters of dotted IP address. (example, 100.024.) LU If SNA: The client LU name. If TCP: Last eight characters of dotted IP address. Note that the IP address requires leading zeros to be specified. CT Protocol from connect, TCP or LU6.2 SPM Position 1 to 8. Oracle Service Name for this connection. The service name is defined in the parameters used to initialize the Oracle ODSI subsystem. Position 9 to 89. TCP/IP hostname (left justified)	See Oracle documentation for more info.

Figure 4-16 Non-IBM subsystem type

Then, understand which work qualifiers your installation can use for setting up the rules. Work qualifiers are used to identify the incoming work request. Your installation might follow certain naming conventions for these qualifiers. These naming conventions can help you to filter work into service classes. Also, understand which work qualifiers are available for each subsystem type. You can then decide which qualifiers you can use for each service class. Table 4-2 on page 132 shows the qualifiers supported by each IBM-defined subsystem type.

Table 4-2 Qualifiers supported by each IBM-defined subsystem type

	ASCH	CB	CICS	DB2	DDF	EWLM	IMS	IWEB	JES	LSMF	MQ	NETV	OMVS	SOM	STC	TSO	SYSH
Accounting Information	*			*	*				*						*	*	
Collection Name		*		*	*									*			
Connection Type				*	*												
Correlation Information				*	*												
EWLM Service Class						*											
EWLM Transaction Class						*											
LU Name			*	*	*		*					*					
Netid				*	*		*										
Package Name				*	*												
Perform				*					*						*	*	
Plan Name				*	*												
Priority				*					*		*	*					
Procedure Name				*	*												
Process Name				*	*						*						
Scheduling Environment Name				*					*								
Subsystem Collection Name				*	*				*								
Subsystem Instance		*	*	*	*		*	*	*	*	*	*					
Subsystem Parameter				*	*			*			*			*	*		
Sysplex Name	*	*	*	*	*		*	*	*	*	*		*	*	*	*	*
System Name	*												*		*	*	*
Transaction Class/Job Class	*	*		*			*	*	*		*	*					
Transaction Name/Job Name	*	*	*	*			*	*	*	*	*	*	*		*		
Userid	*	*	*	*	*		*	*	*		*	*	*	*	*	*	

The following tables provide details about the work qualifiers that you can use to classify SAP requests into the DB2 environment.

Table 4-3 WLM qualifiers for DB2 DDF classification for SAP requests

DB2 client identifier	IFI field	WLM qualifier
Correlation ID	QWHCCV	CI
Accounting string suffix	QMDASUFX	AI (positions 56–143)
Primary authorization ID	QWHCAID	UI
Plan name	QWHCPLAN	PN
Transaction name	QWHCEUTX	PC
Workstation name	QWHCEUWN	SPM (positions 17–34)
Client user ID	QWHCEUID	SPM (positions 1–16)

The following tables show how the SAP attributes map to DB2 Client Identifiers. (Table 4-4 applies to ABAP™ while Table 4-5 applies to SAP Java Workload.)

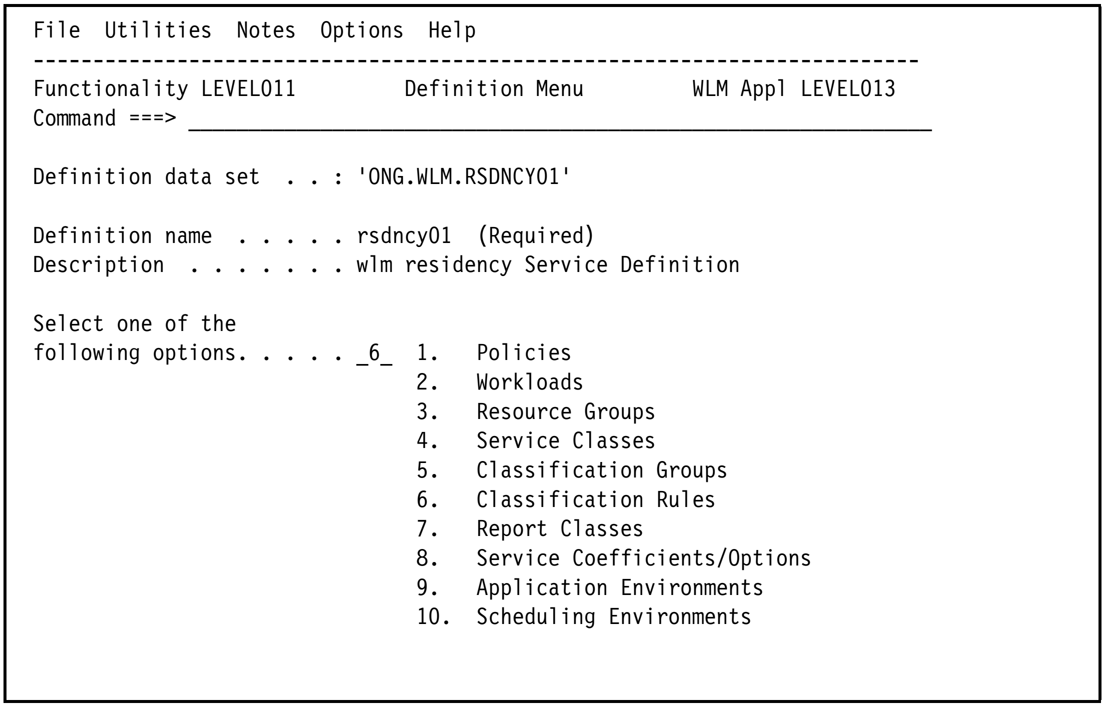
Table 4-4 ABAP attributes

SAP attribute	DB2 client identifier	Positions
SAP system ID (SAPSID)	Correlation ID	1–3
Work process type	Correlation ID	4–6
Work process number	Correlation ID	7–9
Host name of application server	Accounting string suffix	1–32
SAP system number (SAPSYSTEM)	Accounting string suffix	33–34
Work process ID	Accounting string suffix	35–44
Database connection name	Accounting string suffix	45–74
User ID used to connect to DB2	Primary authorization ID	All (1–8)
Plan name of SAP	Plan name	All (1–8)

Table 4-5 SAP Java workload attributes

SAP Java attribute	DB2 client identifier	Positions
SAP system ID (SAPSID)	Primary authorization ID	4–6
Host name of application server	Workstation name	All (1–18)
Indicator for Java workload (constant db2jcc)	Correlation ID	1–6

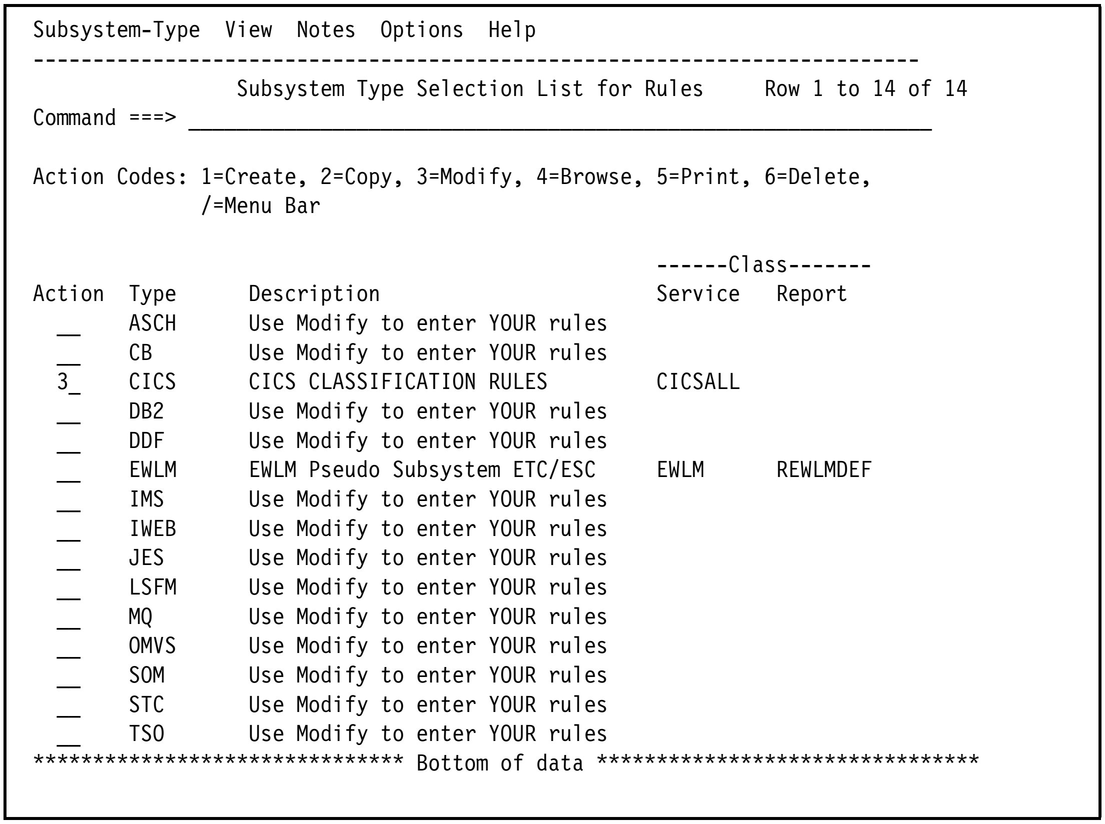
To define your classification rules in the WLM ISPF application. Choose option **6** from the Definition Menu. See Figure 4-17 on page 134.



```
File  Utilities  Notes  Options  Help
--------------------------------------------------------------------------
Functionality LEVEL011        Definition Menu        WLM Appl LEVEL013
Command ===> ______________________________________________________________

Definition data set  . . : 'ONG.WLM.RSDNCY01'

Definition name  . . . . . rsdncy01  (Required)
Description  . . . . . . . wlm residency Service Definition

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
```

Figure 4-17   Definition Menu

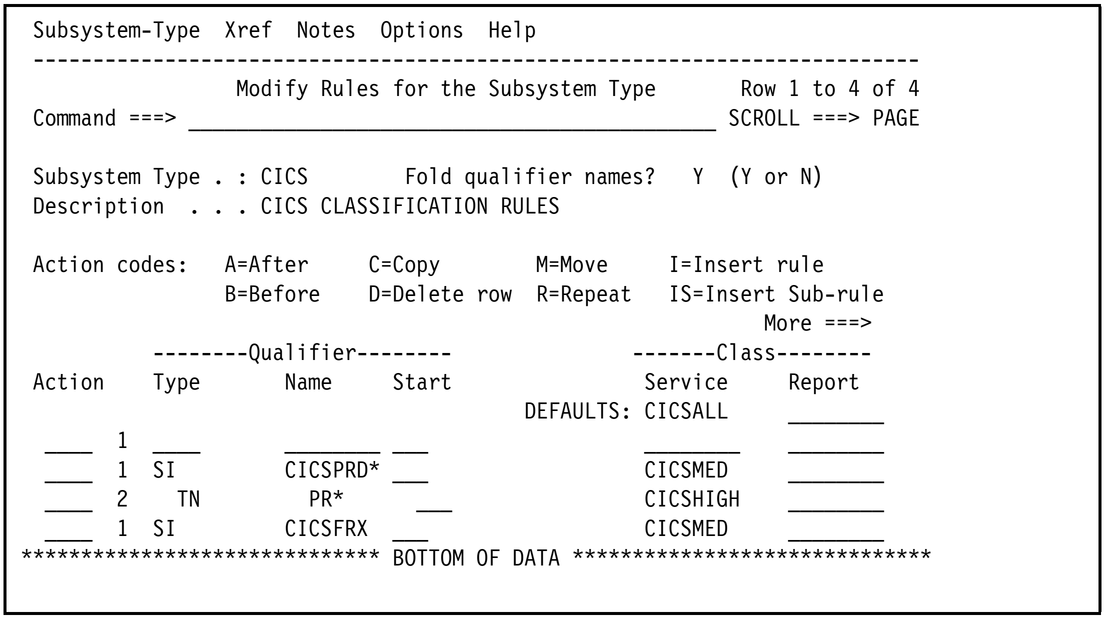
You then need to choose the subsystem type for which you want to define the classification rules. Figure 4-18 shows the panel for the Subsystem Type Selection List for Rules.



```
Subsystem-Type  View  Notes  Options  Help
--------------------------------------------------------------------------
                 Subsystem Type Selection List for Rules     Row 1 to 14 of 14
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
  __    CB        Use Modify to enter YOUR rules
  3_    CICS      CICS CLASSIFICATION RULES        CICSALL
  __    DB2       Use Modify to enter YOUR rules
  __    DDF       Use Modify to enter YOUR rules
  __    EWLM      EWLM Pseudo Subsystem ETC/ESC    EWLM      REWLMDEF
  __    IMS       Use Modify to enter YOUR rules
  __    IWEB      Use Modify to enter YOUR rules
  __    JES       Use Modify to enter YOUR rules
  __    LSFM      Use Modify to enter YOUR rules
  __    MQ        Use Modify to enter YOUR rules
  __    OMVS      Use Modify to enter YOUR rules
  __    SOM       Use Modify to enter YOUR rules
  __    STC       Use Modify to enter YOUR rules
  __    TSO       Use Modify to enter YOUR rules
******************************* Bottom of data ********************************
```

Figure 4-18   Subsystem Type Selection List for Rules

After you have chosen the subsystem type that you want to classify, you need to enter the classification rules. Figure 4-19 is an example of a classification rule for the CICS subsystem. In this example, there are two classification rules with level 1 qualifiers: Subsystem instance CICSPRD* and CICSFRX. There is a level 2 qualifier with transaction name PR* under CICSPRD*. Wildcard notation was used for this work qualifier. The PR* qualifier applies only to transactions associated with the level 1 qualifier of CICSPRD*.



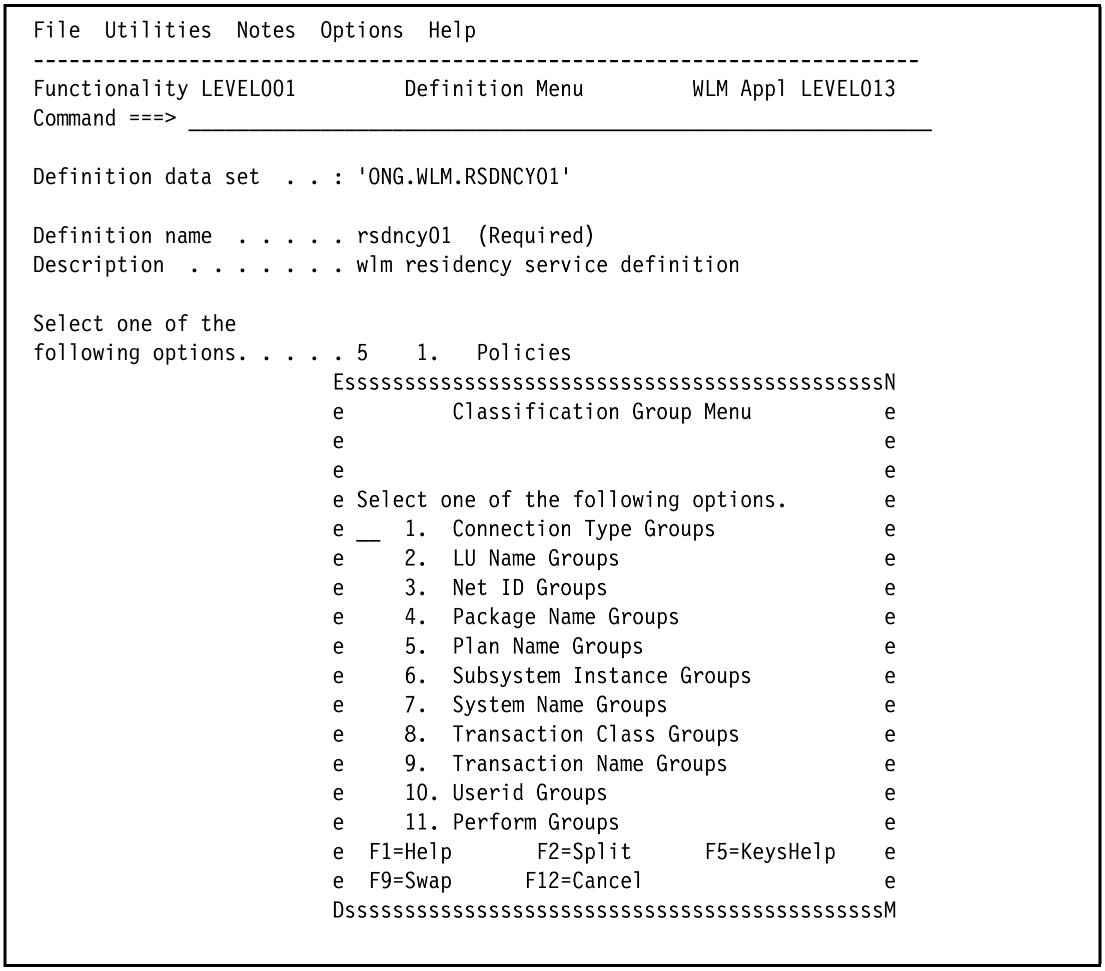
```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                  Modify Rules for the Subsystem Type       Row 1 to 4 of 4
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : CICS        Fold qualifier names?   Y  (Y or N)
 Description  . . . CICS CLASSIFICATION RULES

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                              More ===>
           --------Qualifier--------               -------Class--------
 Action    Type       Name     Start                Service     Report
                                          DEFAULTS: CICSALL     ________
  ____  1  ____       ________ ___                  ________    ________
  ____  1  SI         CICSPRD* ___                  CICSMED     ________
  ____  2    TN         PR*      ___                CICSHIGH    ________
  ____  1  SI         CICSFRX  ___                  CICSMED     ________
***************************** BOTTOM OF DATA *****************************
```

Figure 4-19 Modify Rules for the Subsystem Type

If a work request comes in from a CICS VTAM applid other than CICSPRD* or CICSFRX, then it is assigned the CICSALL default service class. If a work request comes in from CICSPRD1, and the transaction name is PR01, then this classifies as CICSHIGH. If the transaction coming from CICSPRD1 has a name that starts with anything other than PR, then it is classified as CICSMED. If another work request comes in from CICSFRX, then it is assigned the CICSMED service class.

The order of the nesting and the order of the level 1 qualifiers determine the hierarchy of the classification rules. Keep in mind that the system sequentially checks for a match against each level one rule. When it finds a match, it continues just in that family through the level two rules for the first match. Similarly, if a match is found at any given level, then its sub-rules are searched for further qualifier matches. The last matching rule is the one that determines the service class and report class to be assigned.

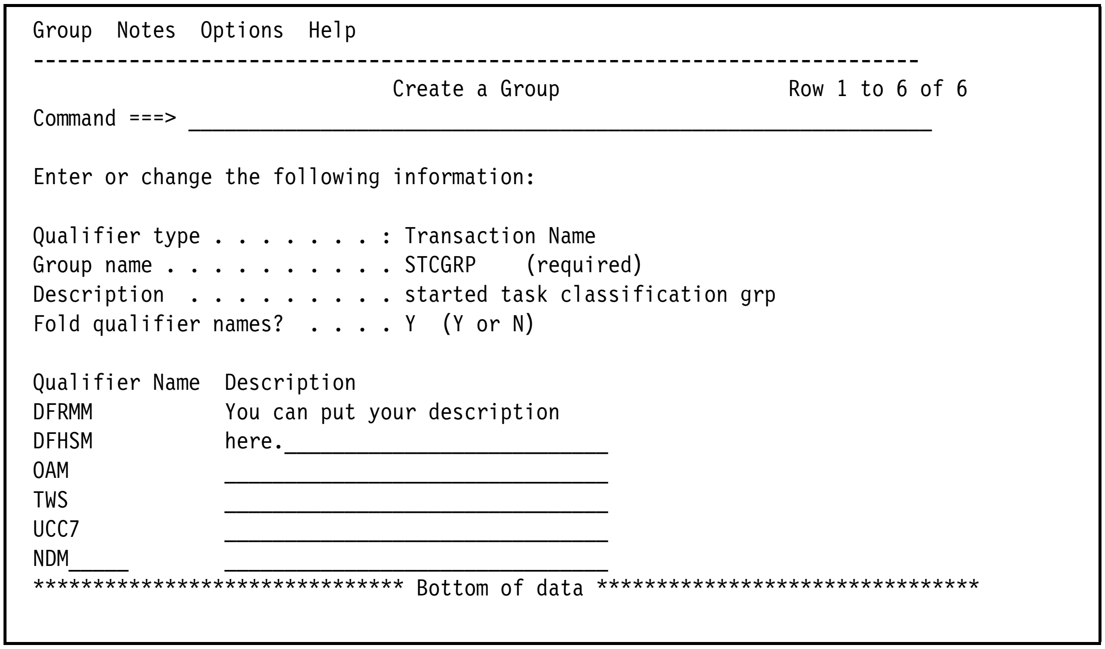
If you have more than five instances of a work qualifier that you need to classify under one subsystem type, you can use the classification group to make it simpler and more efficient. A *classification group* is a collection of the same work qualifier. A classification group of more than five members is quicker to check than single instances in the classification rules. You can create a classification group by choosing option 5 in the Definition Menu (Figure 4-10 on page 125). You receive a pop-up panel requiring that you choose a work qualifier that you want to group. See Figure 4-20 on page 136.



```
 File  Utilities  Notes  Options  Help
 --------------------------------------------------------------------------
 Functionality LEVEL001        Definition Menu        WLM Appl LEVEL013
 Command ===> ______________________________________________________________

 Definition data set  . . : 'ONG.WLM.RSDNCY01'

 Definition name  . . . . . rsdncy01  (Required)
 Description  . . . . . . . wlm residency service definition

 Select one of the
 following options. . . . . 5    1.   Policies
                          EssssssssssssssssssssssssssssssssssssssssssssssN
                          e         Classification Group Menu           e
                          e                                             e
                          e                                             e
                          e Select one of the following options.        e
                          e __  1.  Connection Type Groups              e
                          e     2.  LU Name Groups                      e
                          e     3.  Net ID Groups                       e
                          e     4.  Package Name Groups                 e
                          e     5.  Plan Name Groups                    e
                          e     6.  Subsystem Instance Groups           e
                          e     7.  System Name Groups                  e
                          e     8.  Transaction Class Groups            e
                          e     9.  Transaction Name Groups             e
                          e     10. Userid Groups                       e
                          e     11. Perform Groups                      e
                          e  F1=Help       F2=Split      F5=KeysHelp    e
                          e  F9=Swap      F12=Cancel                    e
                          DssssssssssssssssssssssssssssssssssssssssssssssM
```

Figure 4-20   Classification Group Menu

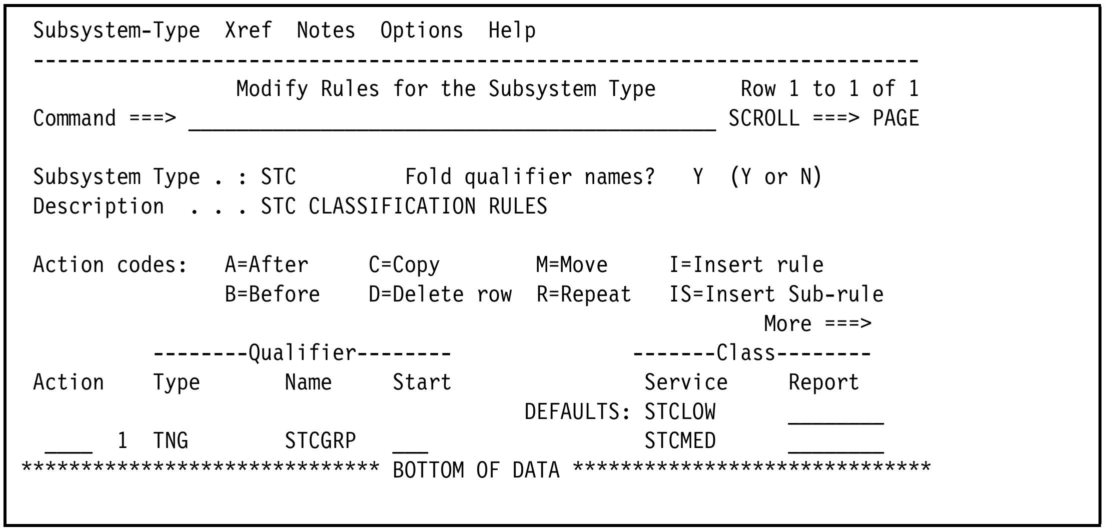
Figure 4-21 on page 137 is an example of a STCGRP that we used to group STC.



```
 Group  Notes  Options  Help
 --------------------------------------------------------------------------
                              Create a Group                Row 1 to 6 of 6
 Command ===> ______________________________________________________________

 Enter or change the following information:

 Qualifier type . . . . . . . : Transaction Name
 Group name . . . . . . . . . . STCGRP    (required)
 Description  . . . . . . . . . started task classification grp
 Fold qualifier names?  . . . . Y  (Y or N)

 Qualifier Name  Description
 DFRMM           You can put your description
 DFHSM           here.__________________________
 OAM             ________________________________
 TWS             ________________________________
 UCC7            ________________________________
 NDM_____        ________________________________
 ******************************* Bottom of data ********************************
```

Figure 4-21 Create a Group

After you define your group, go to the classification rules for your subsystem type. Use the group type and group name on the qualifier type and name fields. The group types are specified by adding `G` to the qualifier type abbreviation. For example, a transaction name group is indicated as `TNG`. Figure 4-22 shows the panel for modifying rules using classification groups.



```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                  Modify Rules for the Subsystem Type      Row 1 to 1 of 1
 Command ===> ____________________________________________ SCROLL ===> PAGE

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

 Action codes:   A=After     C=Copy       M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
           --------Qualifier--------             -------Class--------
 Action    Type      Name     Start                Service     Report
                                         DEFAULTS: STCLOW      ________
  ____  1  TNG       STCGRP   ___                  STCMED      ________
****************************** BOTTOM OF DATA ******************************
```

Figure 4-22 Modify Rules for the Subsystem Type using classification group

Note: For a detailed description and samples about how to classify DB2 Stored Procedures, refer to *DB2 for z/OS Stored Procedures: Through the Call and Beyond*, SG24-7083.

4.2.10 Service definition coefficient

The amount of system resources that an address space or enclave consumes is measured in *service units*. Service units are calculated based on the CPU, SRB, I/O, and storage (MSO) service that an address space consumes.

Service units are the basis for period switching within a service class that has multiple periods. The duration of a service class period is specified in terms of service units. When an address space or enclave running in the service class period has consumed the amount of service specified by the duration, workload management moves it to the next period. The work is managed to the goal and importance of the new period.

Because not all kinds of service are equal in every installation, you can assign additional weight to one kind of service over another. This weight is called a *service coefficient*.

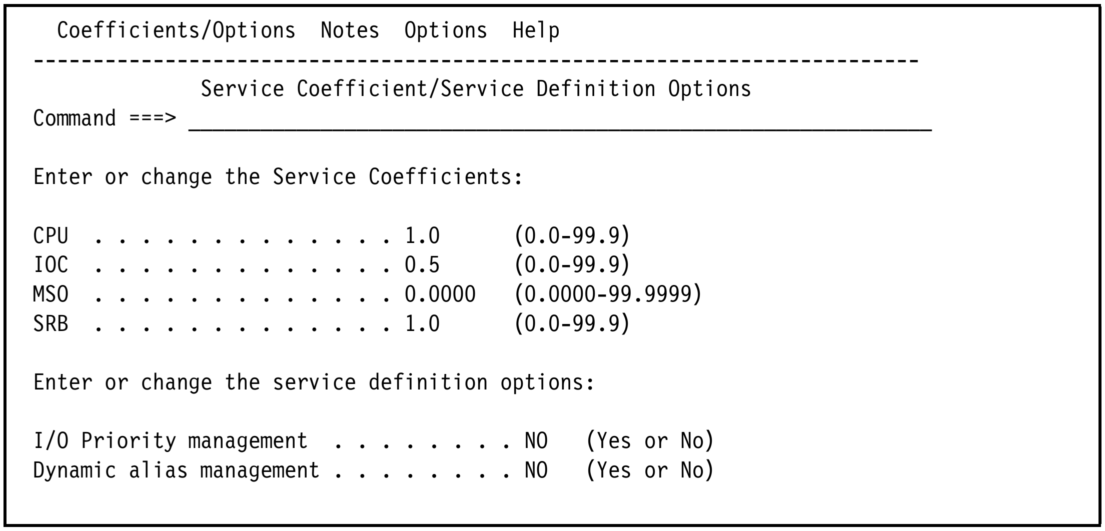
With the increased size in processor capability in our processors today, the default values for these coefficients have become inflated. Processors can consume much higher amounts of service, and as a result, service unit consumption numbers are very high. These high numbers can cause problems if they reach the point where they wrap in the SMF fields. If they wrap, you can see abnormally large transaction counts, and last period work might be restarted in the first period. It is possible for you to make them smaller, yet still maintain the same relationship between the coefficient values. Consider changing your definitions to these values:

```
CPU    1
SRB    1
IOC    0.5
MSO    0
```

If you decide to change the coefficients, you must recalculate your durations and accounting procedures.

> **Tip:** If you want to gather storage service information by service class, but do not want it affecting your durations or accounting procedures, use an MSO coefficient of 0.0001. This results in very low MSO service unit numbers, but still allows you to obtain storage service information through RMF.

To set your service definition coefficient, choose option **8** on the Definition Menu panel (see Figure 4-10 on page 125). This brings you to the Service Coefficient/Service Definition Options panel. See Figure 4-23 on page 139.



```
  Coefficients/Options  Notes  Options  Help
--------------------------------------------------------------------------
              Service Coefficient/Service Definition Options
Command ===> ______________________________________________________________

Enter or change the Service Coefficients:

CPU  . . . . . . . . . . . . . 1.0      (0.0-99.9)
IOC  . . . . . . . . . . . . . 0.5      (0.0-99.9)
MSO  . . . . . . . . . . . . . 0.0000   (0.0000-99.9999)
SRB  . . . . . . . . . . . . . 1.0      (0.0-99.9)

Enter or change the service definition options:

I/O Priority management  . . . . . . . . NO   (Yes or No)
Dynamic alias management . . . . . . . . NO   (Yes or No)
```

Figure 4-23 Service Coefficient/Service Definition Options

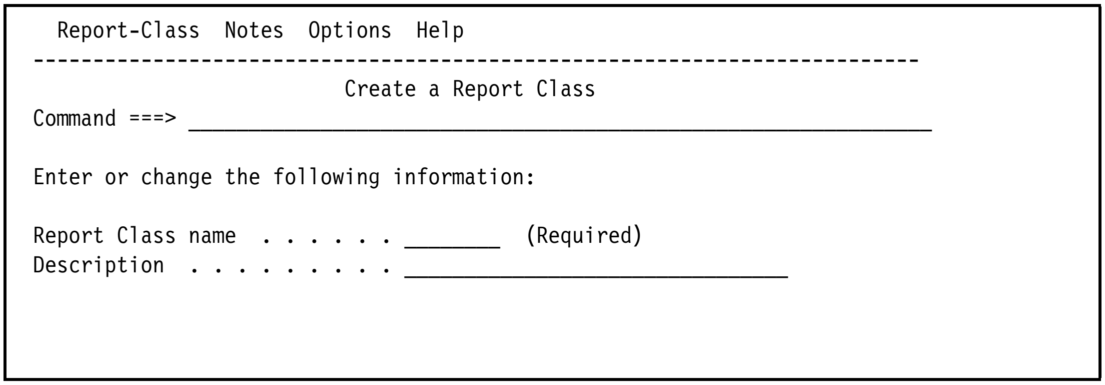
4.2.11 Report class

WLM allows you to specify report classes to report on a subset of transactions running in a single service class (homogeneous report) but also to combine transactions running in different service classes within one report class (heterogeneous report). Heterogeneous report classes can cause incorrect performance data, because the data collected is based on different goals, importance, or duration.

To create a report class, you can either:

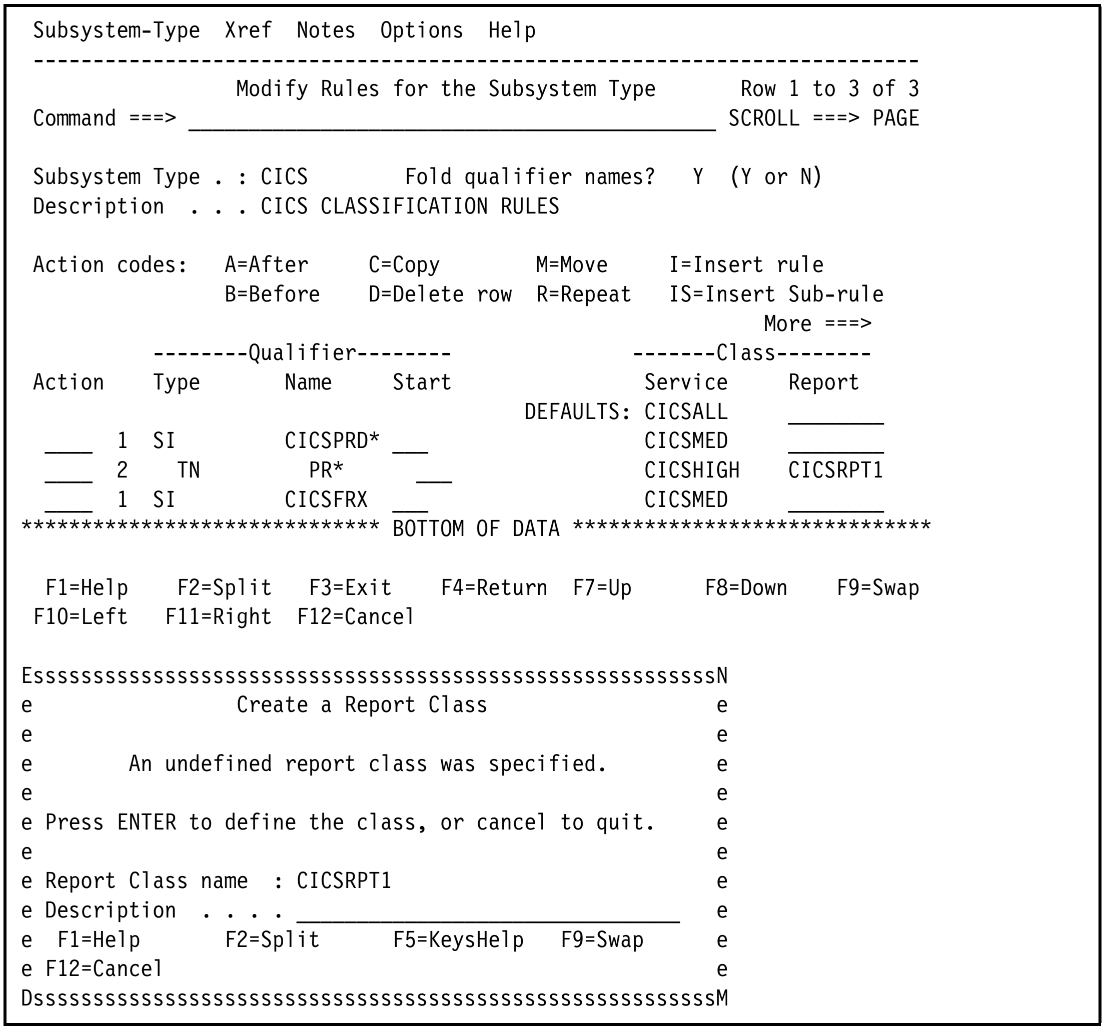
- Select option 7 from the main Definition Menu. See Figure 4-24.
- Use the Modify Rules for the Subsystem Type panel to specify the report class. When you specify a new report class, a pop-up panel prompts you enter the report class description. Figure 4-25 on page 140 shows an example of specifying a report class on the Modify Rules for Subsystem type panel. The name of the report class is `CICSRPT1`.

You can have up to a maximum of 999 report classes.



```
  Report-Class  Notes  Options  Help
--------------------------------------------------------------------------
                            Create a Report Class
Command ===> ______________________________________________________________

Enter or change the following information:

Report Class name  . . . . . . ________  (Required)
Description  . . . . . . . . . ________________________________
```

Figure 4-24 Create a Report Class using option 7 of the Definition Menu



```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                  Modify Rules for the Subsystem Type       Row 1 to 3 of 3
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : CICS       Fold qualifier names?   Y  (Y or N)
 Description  . . . CICS CLASSIFICATION RULES

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                              More ===>
           --------Qualifier--------               -------Class--------
 Action    Type       Name     Start                Service     Report
                                          DEFAULTS: CICSALL     ________
  ____  1  SI         CICSPRD* ___                  CICSMED     ________
  ____  2    TN         PR*      ___                CICSHIGH    CICSRPT1
  ____  1  SI         CICSFRX  ___                  CICSMED     ________
****************************** BOTTOM OF DATA ******************************

  F1=Help    F2=Split   F3=Exit    F4=Return  F7=Up      F8=Down    F9=Swap
 F10=Left   F11=Right  F12=Cancel

EssssssssssssssssssssssssssssssssssssssssssssssssssssssssssN
e                 Create a Report Class                    e
e                                                          e
e        An undefined report class was specified.          e
e                                                          e
e Press ENTER to define the class, or cancel to quit.      e
e                                                          e
e Report Class name  : CICSRPT1                            e
e Description  . . . . ________________________________   e
e  F1=Help       F2=Split      F5=KeysHelp   F9=Swap       e
e F12=Cancel                                               e
DssssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
```

Figure 4-25 Create a Report Class using the Modify Rules for the Subsystem Type panel

4.2.12 Application Environment

An *Application Environment* is a group of application functions requested by a client to be executed in server address spaces. When incoming work associated with an Application Environment is received, WLM can dynamically start the server address space to handle the work request. It dynamically manages the number of server address spaces depending on the performance objectives.

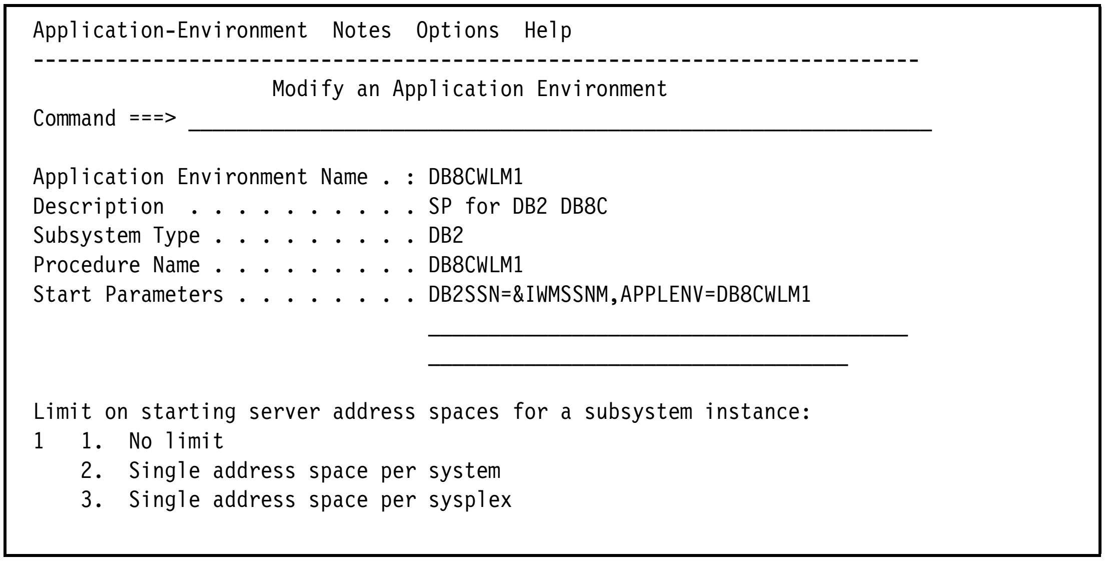
Currently, the IBM-supplied work managers that use Application Environments are:

- DB2 (subsystem type DB2)
- SOMobjects® (subsystem type SOM)
- Component Broker (subsystem CB)
- Internet Connection Server, IBM HTTP Server for OS/390 (subsystem type IWEB)
- MQSeries Workflow (subsystem type MQ)

To define an Application Environment, you need to specify:

- Application Environment name
- Subsystem Type associated with the Application Environment
- JCL Procedure Name to start the server address space
- Start Parameters to use with the JCL procedures
- The limitation, if there is one, on starting the server address spaces

Figure 4-26 shows an example of a DB2 Application Environment.



```
Application-Environment  Notes  Options  Help
--------------------------------------------------------------------------
                    Modify an Application Environment
Command ===> ______________________________________________________________

Application Environment Name . : DB8CWLM1
Description  . . . . . . . . . . SP for DB2 DB8C
Subsystem Type . . . . . . . . . DB2
Procedure Name . . . . . . . . . DB8CWLM1
Start Parameters . . . . . . . . DB2SSN=&IWMSSNM,APPLENV=DB8CWLM1
                                 ________________________________________
                                 ___________________________________


Limit on starting server address spaces for a subsystem instance:
1   1.  No limit
    2.  Single address space per system
    3.  Single address space per sysplex
```

Figure 4-26 Application Environment definition

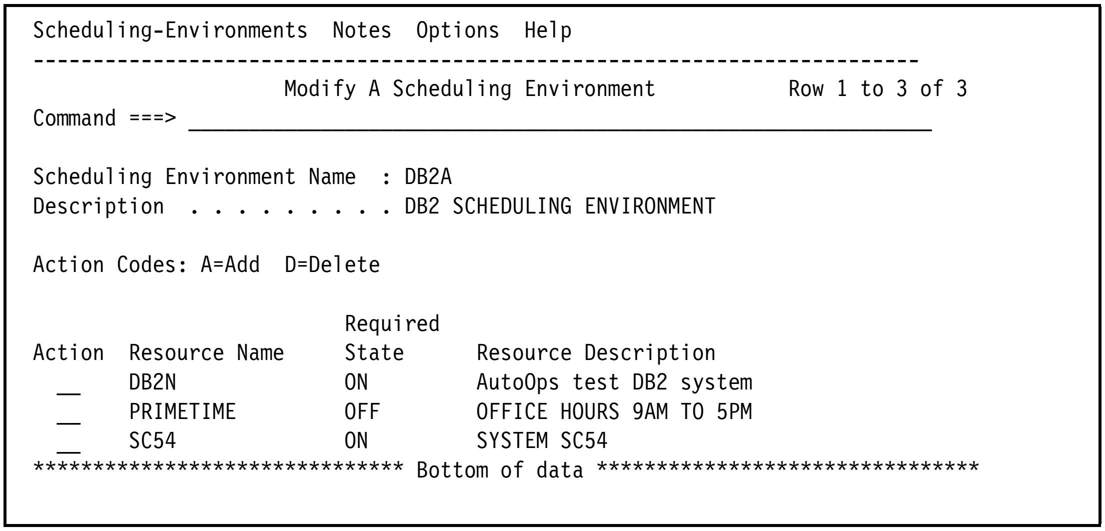
4.2.13 Scheduling environment

A *Scheduling environment* is a list of resources along with their required states. WLM uses a scheduling environment to identify which systems in a sysplex can handle the work that is requested. You define the resources required and their corresponding states in the WLM service definition. If a system has the necessary match for the resource required, then work will be allowed to execute in that system.

To implement a scheduling environment, you need to:

1. Create at least one scheduling environment in your WLM service definition. Define the resource names and the required state (`on` or `off`) for those resources.
2. For each system in the sysplex, you must set the state of the resource. Use **`F WLM,RESOURCE=resource_name,setting`** to set the resource state.
3. Associate your incoming work with a particular scheduling environment. Currently, only JES2, JES3, and Intelligent Data Miner can use the scheduling environment. Check with your subsystem documentation to see if your subsystem can use a scheduling environment.

Figure 4-27 on page 142 shows a sample scheduling environment with its resources.



```
Scheduling-Environments  Notes  Options  Help
---------------------------------------------------------------------------
                     Modify A Scheduling Environment          Row 1 to 3 of 3
Command ===> ______________________________________________________________

Scheduling Environment Name  : DB2A
Description  . . . . . . . . . DB2 SCHEDULING ENVIRONMENT

Action Codes: A=Add  D=Delete

                          Required
Action  Resource Name     State      Resource Description
  __    DB2N              ON         AutoOps test DB2 system
  __    PRIMETIME         OFF        OFFICE HOURS 9AM TO 5PM
  __    SC54              ON         SYSTEM SC54
******************************* Bottom of data ********************************
```

Figure 4-27 Scheduling environment

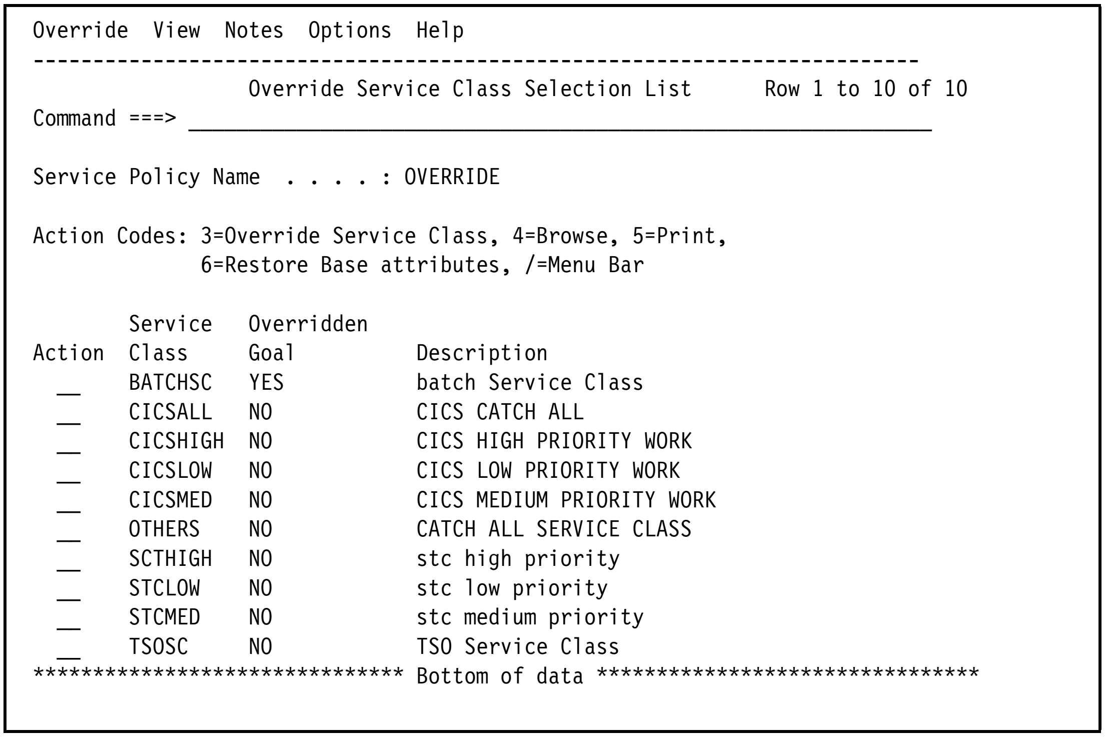
4.2.14 Policy overrides

Your business needs might require that you have different performance goals at different times. In order to do this, you can define a policy that overrides your base policy. You can change these values using policy overrides:

- Service class goals
- Service class CPU protection
- Service class Resource Group assignment
- Resource Group attributes

You cannot change classification rules, an Application Environment, or a scheduling environment through policy overrides. Also, you cannot add or delete service classes through a policy override.

To define a policy override, choose option **1**, Policies from the Definition Menu (Figure 4-10 on page 125), and create your override policy. After you create this policy, use Action Codes 7 (Override Service Classes) or 8 (Override Resource Group) to select what you want to override. If you choose option 7, a list of service classes displays. Option 8 displays a list of Resource Groups. Figure 4-28 on page 143 is a sample Override Service Class Selection List panel.



```
Override  View  Notes  Options  Help
--------------------------------------------------------------------------
                  Override Service Class Selection List      Row 1 to 10 of 10
Command ===> ______________________________________________________________

Service Policy Name  . . . . : OVERRIDE

Action Codes: 3=Override Service Class, 4=Browse, 5=Print,
              6=Restore Base attributes, /=Menu Bar

        Service   Overridden
Action  Class     Goal          Description
  __    BATCHSC   YES           batch Service Class
  __    CICSALL   NO            CICS CATCH ALL
  __    CICSHIGH  NO            CICS HIGH PRIORITY WORK
  __    CICSLOW   NO            CICS LOW PRIORITY WORK
  __    CICSMED   NO            CICS MEDIUM PRIORITY WORK
  __    OTHERS    NO            CATCH ALL SERVICE CLASS
  __    SCTHIGH   NO            stc high priority
  __    STCLOW    NO            stc low priority
  __    STCMED    NO            stc medium priority
  __    TSOSC     NO            TSO Service Class
****************************** Bottom of data ********************************
```

Figure 4-28   Override Service Class Selection List

Chose Action code **3** to override the settings. After you complete the changes, you will see a `YES` on the Overridden Goal column.

To activate your policy, you can either use the WLM ISPF application or use the operator command VARY WLM.

For an example of using policy overrides for Resource Group, refer to 6.5, “Resource Group usage” on page 210.

4.2.15 How to use policy overrides

Generally, a unique service policy definition fits all of the performance requirements of an installation. There are situations where some modifications of the actual service policy are necessary. Using override capability prevents you from having to reinstall a new service definition by allowing you to apply changes to the running service policy.

In this section, we explain how to perform the override of the policy to dynamically add and then remove a Resource Group and to modify the definitions of this Resource Group.

Adding and removing a Resource Group

Example 4-3 on page 144 shows the Resource Group OVRG that we want to add to the BATMED service class.

Example 4-3 Resource Group definition

```
  Resource-Group  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                         Modify a Resource Group
Command ===> ______________________________________________________________

Enter or change the following information:

Resource Group Name  . . . . : OVRG
Description  . . . . . . . . . ITSO Resource Group override

Define Capacity:
1   1.  In Service Units (Sysplex Scope)
    2.  As Percentage of the LPAR share (System Scope)
    3.  As a Number of CPs times 100 (System Scope)
Minimum Capacity . . . . . . . ______
Maximum Capacity . . . . . . . 10000
```

First, we create a new service policy, named OVERPOL, as described in Example 4-4.

Example 4-4 Service policy override creation

```
  Service-Policy  Notes  Options  Help
--------------------------------------------------------------------------
                         Create a Service Policy
Command ===> ______________________________________________________________

Enter or change the following information:

Service Policy Name  . . . . . overpol   (Required)
Description  . . . . . . . . . ITSO policy override____________
```

Then, as shown in Example 4-5, we use option **7** to specify that we want to override service policy OVERPOL.

Example 4-5 Specifying override service class for a service policy

```
  Service-Policy  View  Notes  Options  Help
--------------------------------------------------------------------------
                      Service Policy Selection List           Row 1 to 2 of 2
Command ===> ______________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              7=Override Service Classes, 8=Override Resource Groups,
              /=Menu Bar
                                                    ----Last Change-----
Action  Name      Description                       User      Date
  7_    OVERPOL   ITSO policy override              CASSIER   2005/04/02
  __    WLMPOL    Sample WLM policy                 CASSIER   2005/04/02
```

Example 4-6 on page 145 shows how to specify that service class BATMED is the target of the override.

Example 4-6 Specifying the Service Class to override

```
Override  View  Notes  Options  Help
--------------------------------------------------------------------------
                Override Service Class Selection List      Row 1 to 18 of 40
Command ===> ______________________________________________________________

Service Policy Name  . . . . : OVERPOL

Action Codes: 3=Override Service Class, 4=Browse, 5=Print,
              6=Restore Base attributes, /=Menu Bar

        Service   Overridden
Action  Class     Goal          Description
  __    ASCHDEF   NO            ASCH/APPC default
  __    ASCHHI    NO            APPC high priority
  __    ASCHLO    NO            APPC low priority
  __    BATHI     NO            high priority batch
  __    BATLO     NO            low priority batch
  3_    BATMED    NO            medium priority batch
  __    BATPIER   NO            test   priority batch
  __    BATSUB    NO            test   priority batch
  __    CBDEF     NO            Component Broker Default
  __    CBHI      NO            Component Broker high priority
  __    CBLO      NO            Component Broker low priority
  __    CICSCONV  NO            CICS conversational trans
  __    CICSDEF   NO            CICS Default trans
  __    CICSHI    NO            CICS high priority trans
```

At this point, we specify that the override will add Resource Group OVRG to BATMED service class. Example 4-7 shows this task.

Example 4-7 Override of BATMED service class

```
Service-Class  Xref  Notes  Options  Help
--------------------------------------------------------------------------
               Override attributes for a Service Class       Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Policy Name  . . . . : OVERPOL
Service Class Name . . . . . : BATMED

Override the following information:
Resource Group . . . . . . . . OVRG      (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Action Codes: I=Insert new period, E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __
  __    1              5    Execution velocity of 20
```

After completing these steps, you need to install the service definition. Now, we need to activate the override with the command shown in Example 4-8 on page 146. This command can be automatically scheduled and issued by the automatic operator of your choice.

Example 4-8 Command to activate override policy

```
V WLM,POLICY=OVERPOL
IWM001I WORKLOAD MANAGEMENT POLICY OVERPOL NOW IN EFFECT
```

In the SDSF display activity panel, you can check that the new definition is active, as shown in Example 4-9. BATMED service class now has Resource Group OVRG assigned.

Example 4-9 SDSF view of Resource Group

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------
SDSF DA #@$1  (ALL)    PAG    0 SIO    40 CPU  19/ 18
COMMAND INPUT ===>
NP   JOBNAME  U% Workload SrvClass SP ResGroup Server
     JESJ029  18 BAT_WKL  BATMED    1 OVRG     NO
     JESJ031  18 BAT_WKL  BATMED    1 OVRG     NO
     JESJ023  18 BAT_WKL  BATMED    1 OVRG     NO
     JESJ039  21 BAT_WKL  BATMED    1 OVRG     NO
     JESSOP6  19 BAT_WKL  BATSUB    1          NO
```

When you do not need this variation any longer, you can reset to the initial policy definition (without Resource Group OVRG assigned to service class BATMED) by issuing the command shown in Example 4-10.

Example 4-10 Command to restore initial policy

```
V WLM,POLICY=WLMPOL
IWM001I WORKLOAD MANAGEMENT POLICY WLMPOL NOW IN EFFECT
```

Example 4-11 shows the service class that no longer has a Resource Group assigned.

Example 4-11 SDSF view of original setting of BATMED service class

```
Display  Filter  View  Print  Options  Help
---------------------------------------------------------------
SDSF DA #@$1  (ALL)    PAG    0 SIO   190 CPU  67/ 61  LINE 1-2
COMMAND INPUT ===>                                            S
NP   JOBNAME  PU% Workload SrvClass SP ResGroup Server   Quiesc
     JESJ029   61 BAT_WKL  BATMED    1          NO
     JESJ026   61 BAT_WKL  BATMED    1          NO
     JESJ032   61 BAT_WKL  BATMED    1          NO
     JESJ027   69 BAT_WKL  BATMED    1          NO
     JESJ036   69 BAT_WKL  BATMED    1          NO
```

Overriding a Resource Group definition

In this example, we describe how to override the maximum capacity setting of an existing Resource Group definition.

Example 4-12 on page 147 shows the actual setting.

Example 4-12 Actual setting of Resource Group RGMED

```
  Resource-Group  Notes  Options  Help
--------------------------------------------------------------------------
                         Create a Resource Group
Command ===> ______________________________________________________________

Enter or change the following information:

Resource Group Name  . . . . . RGMED     (required)
Description  . . . . . . . . . ITSO Resource Group

Define Capacity:
1   1.  In Service Units (Sysplex Scope)
    2.  As Percentage of the LPAR share (System Scope)
    3.  As a Number of CPs times 100 (System Scope)
Minimum Capacity . . . . . . . ______
Maximum Capacity . . . . . . . 15000
```

We create the service policy override OVERRG.

Example 4-13 shows that we now have three service policies:

- WLMPOL is the base policy.
- OVERPOL is the override policy for adding and removing the Resource Group.
- OVERRG is the service policy override to modify the Resource Group capacity definition.

Example 4-13 OVERRG service policy override creation

```
Service-Policy  View  Notes  Options  Help
--------------------------------------------------------------------------
                      Service Policy Selection List           Row 1 to 3 of 3
Command ===> ______________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              7=Override Service Classes, 8=Override Resource Groups,
              /=Menu Bar
                                                   ----Last Change-----
Action  Name      Description                      User      Date
  __    OVERPOL   ITSO policy override             CASSIER   2005/04/02
  __    OVERRG    ITSO policy override resource gr CASSIER   2005/04/02
  __    WLMPOL    Sample WLM policy                CASSIER   2005/04/02
```

Example 4-14 on page 148 shows that we use option **8** on policy OVERRG to override the Resource Group.

Example 4-14 Specifying Resource Group override

```
  Service-Policy  View  Notes  Options  Help
--------------------------------------------------------------------------
                      Service Policy Selection List            Row 1 to 3 of 3
Command ===> ______________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              7=Override Service Classes, 8=Override Resource Groups,
              /=Menu Bar
                                                    ----Last Change-----
Action  Name      Description                       User      Date
  __    OVERPOL   ITSO policy override              CASSIER   2005/04/02
  8_    OVERRG    ITSO policy override resource gr  CASSIER   2005/04/02
  __    WLMPOL    Sample WLM policy                 CASSIER   2005/04/02
```

Example 4-15 shows how to select which Resource Group we want to override.

Example 4-15 Specifying the Resource Group to override

```
  Resource-Group  View  Notes  Options  Help
--------------------------------------------------------------------------
                  Override Resource Group Selection List       Row 1 to 4 of 4
Command ===> ______________________________________________________________

Service Policy Name  . . . . : OVERRG

Action Codes: 3=Override Resource Group, 4=Browse, 5=Print,
              6=Restore Base attributes, /=Menu Bar

        Resource  Overridden
Action  Group     attributes    Description
  __    LIMITRG   NO            control quiesced nonswappables
  __    OVRG      NO            ITSO Resource Group override
  3_    RGMED     NO            ITSO Resource Group
  __    RGPIER    NO
```

Example 4-16 shows the change for the maximum capacity value.

Example 4-16 Setting new maximum capacity value for Resource Group OVERRG

```
Override  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                 Override Attributes for a Resource Group
Command ===> ______________________________________________________________

Service Policy Name  . . . . : OVERRG
Resource Group Name  . . . . : RGMED

Enter or change the following:

New Minimum Capacity . . . . . ______
New Maximum Capacity . . . . . 08000
```

Example 4-17 on page 149 shows a RMF Monitor III Display report for RGMED. The maximum capacity value is still 15000 SU/s.

Example 4-17   Resource Group RGMED display before override activation

```
Service Definition: WLMDEF                Installed at: 04/02/05, 11.26.03
     Active Policy: WLMPOL                Activated at: 04/02/05, 11.26.17

               Exec Vel  Resp. Time            Resource  ----- Capacity -----
Name     T  I    Goal       Goal     Duration    Group     Min    Max    Act


BAT_WKL  W
BATMED   S  5     20                           RGMED        N/A  15000  14079
BATSUB   S  2     30                                        N/A    N/A 5.8083
STC_WKL  W
OPSDEF   S  3     40                                        N/A    N/A 1579.3
```

Example 4-18 shows policy OVERRG activation by the VARY WLM command.

Example 4-18   Command to activate OVERRG service policy

```
V WLM,POLICY=OVERRG
IWM001I WORKLOAD MANAGEMENT POLICY OVERRG NOW IN EFFECT
```

Example 4-19 shows a RMF Monitor III Display report for service class BATMED. You can see that Resource Group RGMED now has a maximum capacity of 8000 SU/s.

Example 4-19   Resource Group RGMED display after override activation

```
Service Definition: WLMDEF                Installed at: 04/02/05, 11.26.03
     Active Policy: OVERRG                Activated at: 04/02/05, 11.37.09

               Exec Vel  Resp. Time            Resource  ----- Capacity -----
Name     T  I    Goal       Goal     Duration    Group     Min    Max    Act


BAT_WKL  W
BATMED   S  5     20                           RGMED        N/A   8000   7960
BATSUB   S  2     30                                        N/A    N/A 6.0833
STC_WKL  W
```

4.2.16 Evaluating your SMF records

When you implement goal mode, make sure that you turn off the collection of SMF type 99 records. These records trace the decisions that WLM makes while in goal mode, and the actions taken. SMF type 99 records are written frequently and are required for detailed diagnosis only. With APAR OW28820 installed, SMF99 records are buffered in WLM private storage. A dump of the WLM address space contains the last 15 minutes of SMF99 records. This should be sufficient for many diagnostic situations.

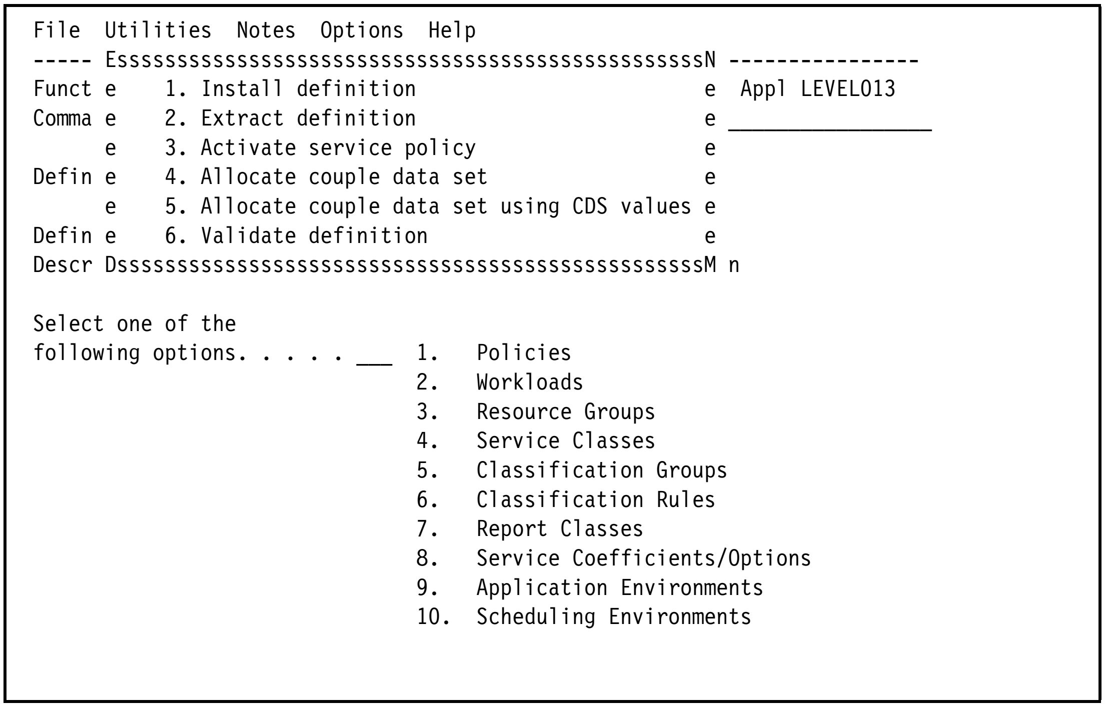
Review your accounting based on SMF record type 30 or record type 72 records; you might need to update your accounting package.

For more information about SMF record changes for goal mode, see *z/OS MVS System Management Facilities*, SA22-7630.

4.2.17 Installing and activating your service definition

To install your service definition, you can either use the WLM ISPF application or use the install definition utility.

To use the WLM ISPF application, go into it specifying the name of the PDS containing your service definition. From the Definition Menu, go to **Utilities** on the action bar. Then select the pull-down option **Install definition**. See Figure 4-29 for the Install service definition panel.



```
File  Utilities  Notes  Options  Help
----- EssssssssssssssssssssssssssssssssssssssssssssssssN ----------------
Funct e    1. Install definition                        e  Appl LEVEL013
Comma e    2. Extract definition                        e ________________
      e    3. Activate service policy                   e
Defin e    4. Allocate couple data set                  e
      e    5. Allocate couple data set using CDS values e
Defin e    6. Validate definition                       e
Descr DssssssssssssssssssssssssssssssssssssssssssssssssM n

Select one of the
following options. . . . . ___  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
```

Figure 4-29   Install service definition panel

To use the install definition utility, you need to customize the sample JCL IWMINSTL, which ships in SYS1.SAMPLIB. Follow the instructions in the prolog of the JCL. After you prepare the JCL, start it from the command console or submit it as a batch job. Figure 4-30 on page 151 shows the sample JCL for the install definition utility.

```
//IWMINSTL   JOB MSGLEVEL=(1,1),REGION=0M
//*
//SETPARM1  SET   SVDEFPDS='your.definition.pds'
//SETPARM2  SET   FORCE='N'
//SETPARM3  SET   POLNAME=''
//*
//STEP1    EXEC  PGM=IKJEFT1B,
//         PARM='IWMARIDU F=&FORCE P=&POLNAME'
//SYSPROC  DD DSN=SYS1.SBLSCLI0,
//         DISP=SHR
//*
//ISPLLIB  DD DSN=ISP.SISPLPA,
//         DISP=SHR
//         DD DSN=ISP.SISPLOAD,
//         DISP=SHR
//ISPMLIB  DD DSN=ISP.SISPMENU,
//         DISP=SHR
//ISPPLIB  DD DSN=ISP.SISPPENU,
//         DISP=SHR
//ISPTLIB  DD DSN=ISP.SISPTENU,
//         DISP=SHR
//ISPSLIB  DD DSN=ISP.SISPSENU,
//         DISP=SHR
//ISPTABL  DD DISP=NEW,UNIT=SYSALLDA,SPACE=(CYL,(1,1,5)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//ISPPROF  DD DISP=NEW,UNIT=SYSALLDA,SPACE=(CYL,(1,1,5)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//ISPLOG   DD DISP=NEW,UNIT=SYSALLDA,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//ISPCTL1  DD DISP=NEW,UNIT=SYSALLDA,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//ISPLST1  DD DISP=NEW,UNIT=SYSALLDA,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=FBA,LRECL=121,BLKSIZE=1210)
//SVDEF    DD DSN=&SVDEFPDS,
//         DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
```

Figure 4-30 Install definition utility sample JCL

4.3 Current WLM users upgrading to a new OS

This section discusses several considerations when you migrate to a new release of the operating system (OS). It discusses WLM compatibility and coexistence issues between OS releases.

4.3.1 WLM compatibility and coexistence

If you are running with mixed z/OS releases on a sysplex, consider whether you need to apply compatibility APARs to enable different levels of WLM to coexist until you can upgrade the entire sysplex to the new release.

WLM running in OS/390 2.10 up to z/OS 1.6 have no compatibility issues. These releases can coexist with each other without any PTF needed. If your operating system is lower than

OS/390 2.10, then you need to refer to *z/OS MVS Planning: Workload Management,* SA22-7602, for the corresponding compatibility PTFs.

Note: The WLM service definition is stored in ISPF tables. When a new release adds certain specifications to the service definition, structural changes to the ISPF tables are required. When you open a service definition whose table structure is different (older) than the one currently used by the WLM ISPF application, the WLM ISPF application automatically updates the service definition structure. After this occurs, older levels of the WLM ISPF application cannot read the service definition unless you install the compatibility APARs.

From OS/390 2.10 up to z/OS 1.6, there is no change in the ISPF table structures. If you are running on an operating system lower than OS/390 2.10, then refer to *z/OS MVS Planning: Workload Management,* SA22-7602, for more information.

4.3.2 WLM enhancements matrix

Table 4-6 lists all the new functions that were added to WLM from z/OS 1.3 through 1.7, either as part of the z/OS release (indicated with an asterisk (*)) or as an APAR.

Table 4-6 WLM enhancements matrix

Function	z/OS 1.8	z/OS 1.7	z/OS 1.6	z/OS 1.5	z/OS 1.4
Lightweight EWLM instrumentation through WLM enclave services	*				
Enhanced initiator balancing	*	*	*	*	*
Controlling initiator dispatch priority in goal mode	*	*	*	*	OW55344
WLM-managed initiator start considers system affinities	*	OA10814	OA10814	OA10814	OA10814
Dynamic Application Environments	*	*	*	*	OW54622
Stateful session placement	*	*	*	OA04699	OA04699
Enhancements for WLM-established stored procedures	*	*	*	OA04555	OA04555
Sub-capacity pricing enhancements	*	*	*	*	OW55509
SCRT extensions for dedicated processors	*	*			
Enhanced enqueue promotion	*	*	*	*	*
DB2 latch contention	*	*	OA08949	OA08949	
zAAP reporting support	*	*	*		
Dynamically handle speed changes on z890, z990, or later	*	OA12155	OA07510 OA12155	OA07510	OA07510
Sysplex routing enhancements	*	*			

Function	z/OS 1.8	z/OS 1.7	z/OS 1.6	z/OS 1.5	z/OS 1.4
Enhancements in computation of free capacity in LPAR environments	*	*	OA10006	OA10006	OA10006
EWLM monitoring support 1	*	*	OA07196		
WLM support of multiple subchannel sets (z9-109)	*	*			
EWLM policy correlation	*	OA12784	OA12784		
zIIP reporting support	*	OA16005	OA16005		
zAAP processing enhancement	*	OA14131 OA13953	OA14131 OA13953		
z/OS 1.8 coexistence	*	OA13837	OA13837	OA13837	OA13837
RSM/SRM support for > 128 GB real storage	*				
IEAOPT MCCAFCTH enhancement	*	OA14409	OA14409		
Routing (“Storm Drain”) enhancements	*	OA14310			

4.4 WLM tools

This section is about the WLM tools available from IBM. You can download these tools through the Internet. We have added information about these tools here for easy reference. To download the tools, go to the following Web site:

http://www.ibm.com/servers/eserver/zseries/zos/wlm/tools/

4.4.1 Service Definition Formatter

The Service Definition Formatter is a small tool that assists you to display your WLM service definition. To use the tool, you only need to download the WLM service definition to your workstation and load it into the spreadsheet. Then, you can use the various worksheets to display parts of your service definition to get a better overview of your workload management definitions.

The tool is *not* a service definition editor. You must enter all modifications to the WLM service definition through the WLM Administrative Application.

To install the tool, download SetUpSvDef.exe (228 KB) to your workstation and execute the installation program. For more information, refer to the readme.pdf in the installation path of the tool.

4.4.2 WLM Work Queue Viewer

The WLM Work Queue Viewer (WLMQUE) is a small ISPF-based tool that can assist you to display the Application Environments that are currently in use on your z/OS system. This can be helpful for using WebSphere Application Servers when you specify minimum and maximum limits for the number of server address spaces that should be started. You can view the number of started and active server address spaces, and the service classes used as work queues for the Application Environments with the help of the REXX command list. Use this tool for any kind of Application Environment from WebSphere, DB2, or user-specified types and applications.

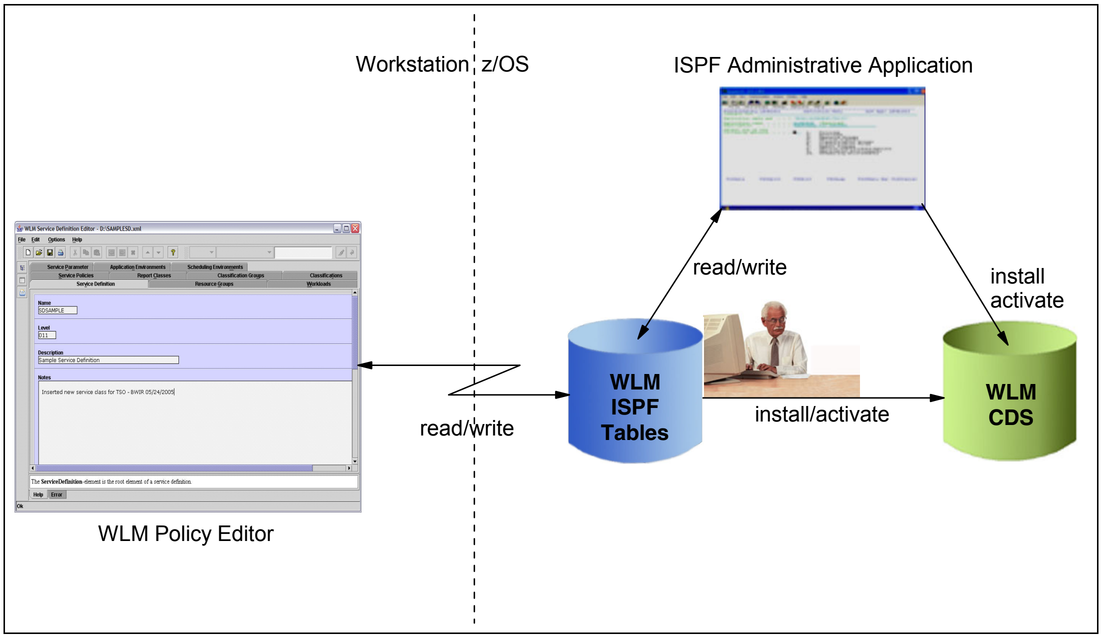
To install the tool, download wlmque.zip (version 1.8.001, 90 KB) to your workstation. Unzip the file and follow the installation instructions in readme.txt. The tool was written on a z/OS 1.8 system and should run from z/OS 1.6 and above.

4.4.3 WLM Service Definition Editor

As an alternative to the WLM ISPF application, a user friendly graphical interface is available. This WLM policy editor is a workstation-based application that is able to read data from WLM ISPF data sets. The data is converted to xml format on the workstation. In this format, you can create advanced reports and modify the Service Policy. The WLM Service Definition Editor can convert the xml format back to ISPF format and saves the changed data (service definition) in a WLM ISPF data set on the host.

The WLM Service Definition Editor does not provide support to extract, install, and activate z/OS WLM service definitions and to allocate WLM couple data sets. This means that the WLM Service Definition Editor does not interact with the WLM component of the z/OS operating system; it solely operates on WLM service definitions in ISPF format.

Install the service definition using either the existing ISPF WLM Administrative Application or the WLM operator commands. Figure 4-31 depicts the process.



Name
SDSAMPLE

Level
011

Description
Sample Service Definition

Notes
Inserted new service class for TSO - BWIR 05/24/2005

The **ServiceDefinition**-element is the root element of a service definition.

Figure 4-31 Policy editor process

This is a list of the advanced functionalities available with the new application compared to the WLM ISPF interface:

- Easy creation and editing of WLM service definitions:
  - Clearly arranged representation of service definition elements via table or tree views
  - Direct manipulation of service definition elements (no pop-up windows)
  - Identification of doubtful specifications
  - Error list

  - Context-sensitive help information
  - Automatic upload and download of service definitions via FTP and FTPS
- Support for analysis and optimization of WLM service definitions:
  - Different views (tree view, table view, or print view)
  - Display of service definition element relationships
  - Sorting and rearranging capability of service definition elements in table view
  - Searching capability of entire service definition
- Printout of WLM service definitions in HTML format
- National language support

Getting started

The instructions to install and use this application are:

1. Download the WLM Service Definition Editor.

   The new WLM Administrative Application is available from the IBM Web site:

   http://www.ibm.com/server/eserver/zseries/zos/wlm/tools/sdeditor.html

   In our configuration, we downloaded the WLM Service Definition Editor Version 1.0.1 installation program and installed it on a Windows® XP Workstation. After starting the setup, the installation wizard guides you through the process.

2. Start the Service Definition Editor.

   To start the WLM service definition on your workstation, select the **IBM Workload Management** program and click **IBM WLM Service Definition Editor**.

   Figure 4-34 shows an example of the workstation’s window.

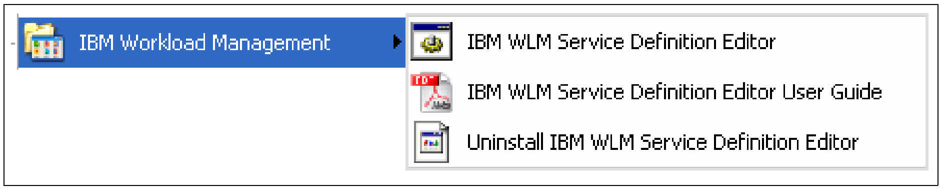
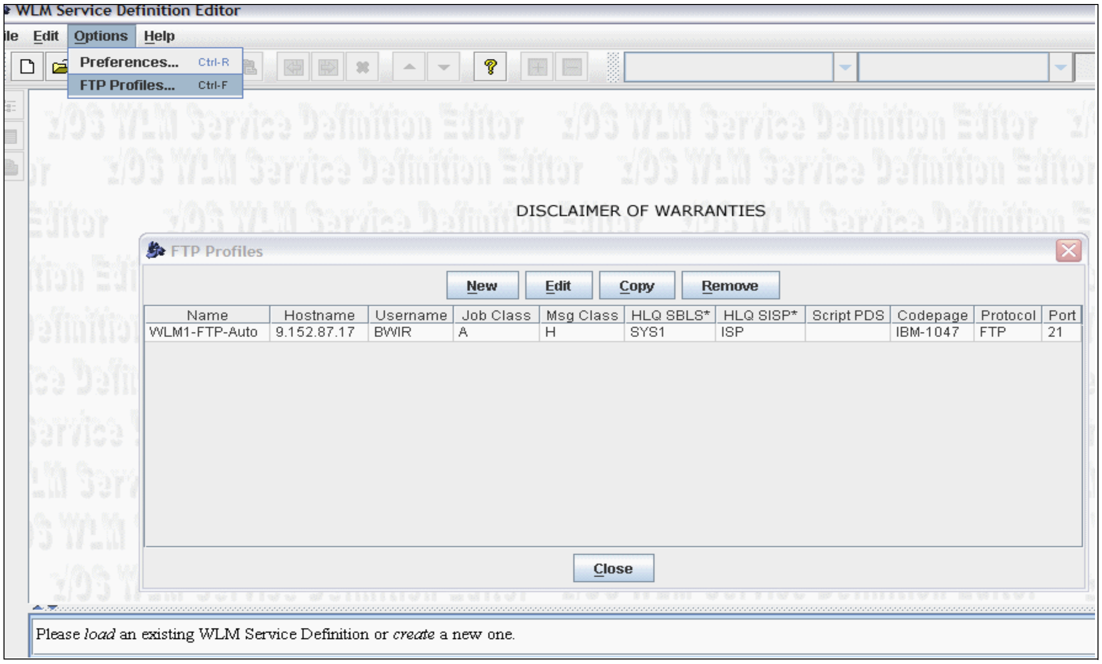

IBM Workload Management
IBM WLM Service Definition Editor
IBM WLM Service Definition Editor User Guide
Uninstall IBM WLM Service Definition Editor

Figure 4-32 Starting the IBM WLM Service Definition Editor

> **Note:** Only one instance of the WLM Service Definition Editor can run at a time. Do not close the DOS window which comes up with the Java application, because if you do, you close the Java application without cleaning up the WSELog file. If you try to start the application again, then you get an error message that the application is already running. In this case, delete the file WSELog in the directory where you installed the WLM Service Definition Editor. You can then restart the application.

3. Download your service definition.

   Although you can create a new service definition with the WLM service definition, you might want to start with your existing active service definition on your z/OS system. To do so, the ISPF format service definition has to be transferred to your workstation. Check that FTP from the mainframe to the workstation is enabled and get the IP address of your mainframe where the service definition exists. In our case, we select **Options** → **FTP Profile**. Figure 4-33 on page 156 shows an example of the workstation window.

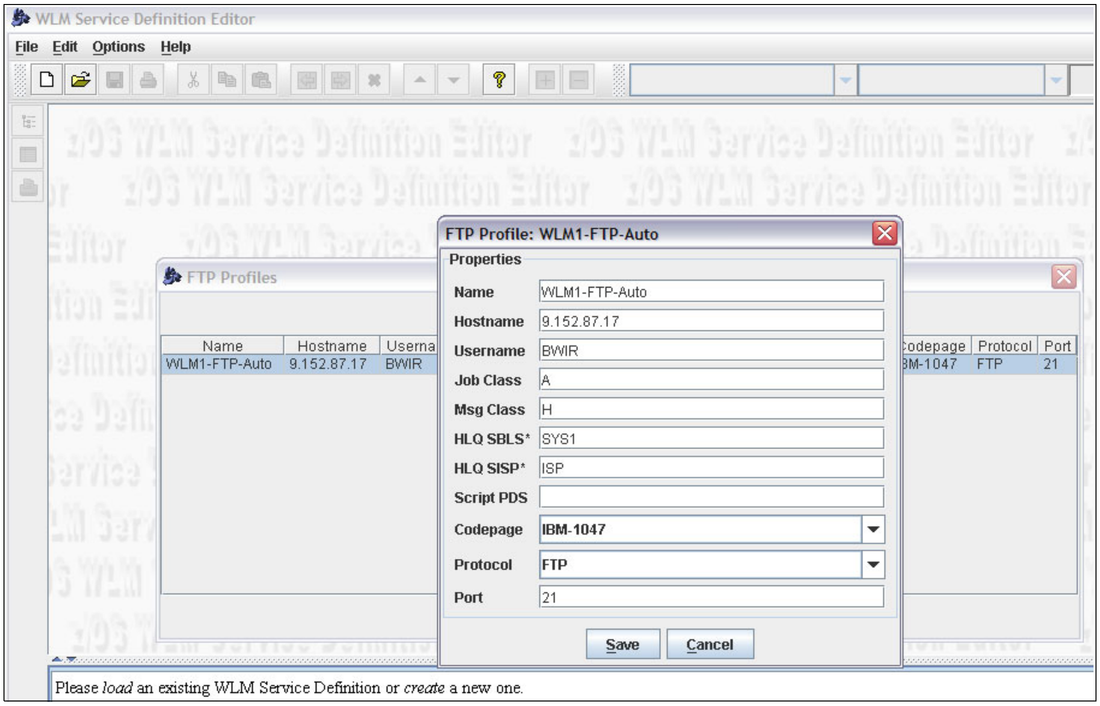


Name	Hostname	Username	Job Class	Msg Class	HLQ SBLS*	HLQ SISP*	Script PDS	Codepage	Protocol	Port
WLM1-FTP-Auto	9.152.87.17	BWIR	A	H	SYS1	ISP		IBM-1047	FTP	21

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-33 Select Options → FTP Profiles

Now, click the name to highlight the profile and select **Edit** to change the data with your system setup. When finished, select **Save** to save your FTP profile and then **Close**. Figure 4-34 shows an example of the workstation window.



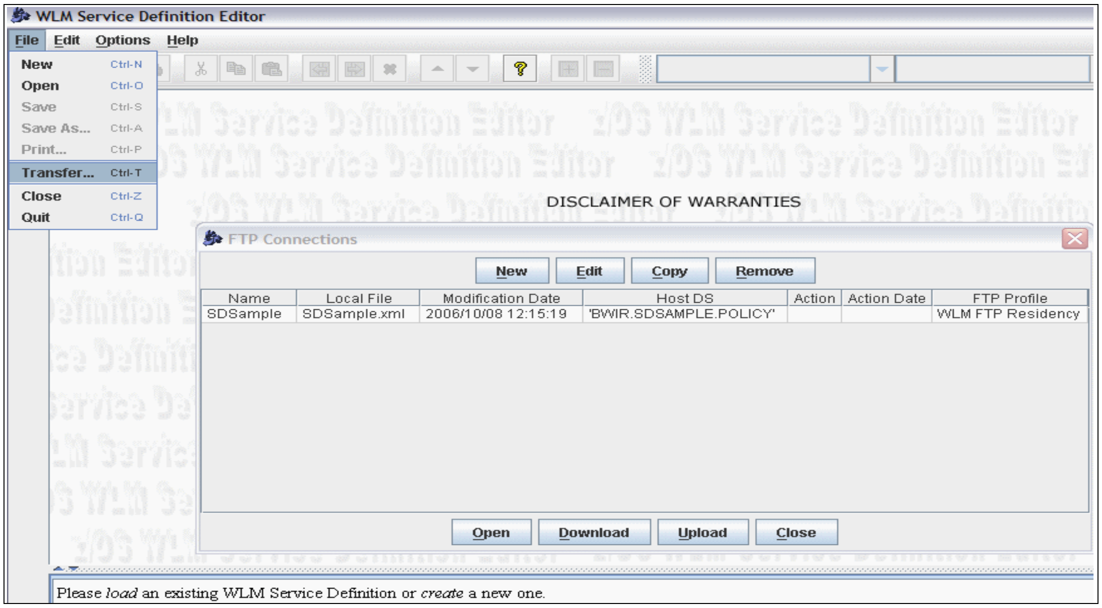
Name	Hostname	Userna		Codepage	Protocol	Port
WLM1-FTP-Auto	9.152.87.17	BWIR		3M-1047	FTP	21

Name	WLM1-FTP-Auto
Hostname	9.152.87.17
Username	BWIR
Job Class	A
Msg Class	H
HLQ SBLS*	SYS1
HLQ SISP*	ISP
Script PDS	
Codepage	IBM-1047
Protocol	FTP
Port	21

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-34 Edit and save your FTP Profile

Upload and download of a service definition is triggered through the Transfer option of the file menu. To work with your service definition, first you have to download it to your workstation. On your workstation window, select **File** → **Transfer.** Figure 4-35 shows an example of the workstation window.



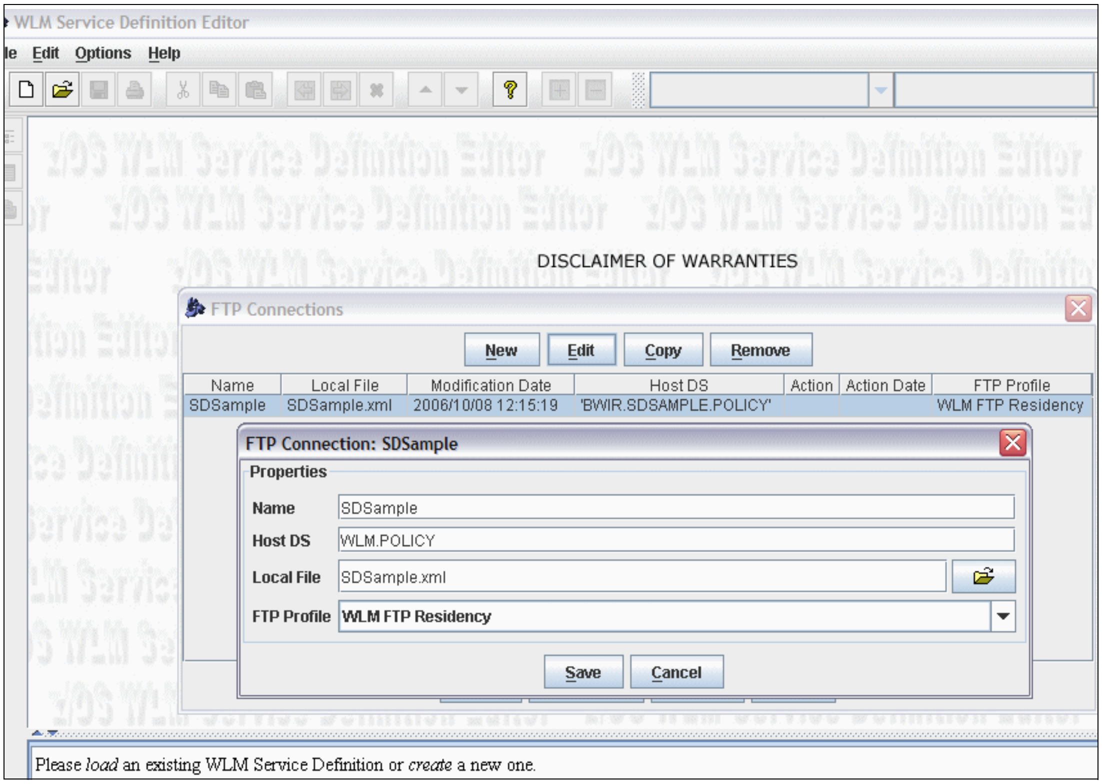
DISCLAIMER OF WARRANTIES

Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
SDSample	SDSample.xml	2006/10/08 12:15:19	'BWIR.SDSAMPLE.POLICY'			WLM FTP Residency

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-35 Set up File Transfer

Click the name to highlight the Policy name and select **Edit** to change the name to the Host DS to the data set name that reflects the ISPF data set where you saved the WLM service definition on the host. Change the Name and Local File fields to new values unless you want to override the sample files delivered with WLM Service Definition Editor. Figure 4-36 on page 158 shows an example of the workstation window.



Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
SDSample	SDSample.xml	2006/10/08 12:15:19	'BWIR.SDSAMPLE.POLICY'			WLM FTP Residency

Name: SDSample

Host DS: WLM.POLICY

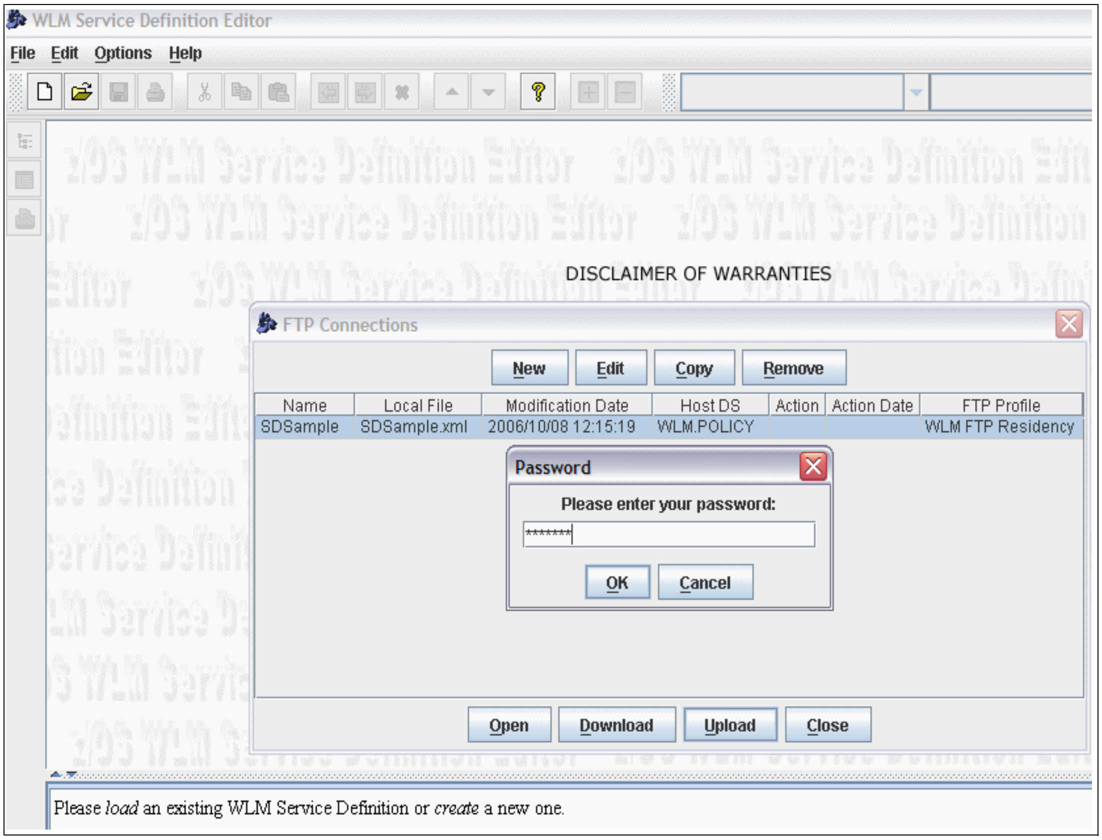
Local File: SDSample.xml

FTP Profile: WLM FTP Residency

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-36 Update WLM ISPF data set and workstation names

Select **Save** to save the changes. Download and upload of a service definition is triggered by selecting **File** → **Transfer**. To work with your WLM ISPF service definition, you have to download it first to your workstation. On your workstation window, click the name of your service definition to highlight it, then select **Download** and enter your password to access the host. Click **OK**. Figure 4-37 on page 159 shows an example of the workstation window.



DISCLAIMER OF WARRANTIES

Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
SDSample	SDSample.xml	2006/10/08 12:15:19	WLM.POLICY			WLM FTP Residency

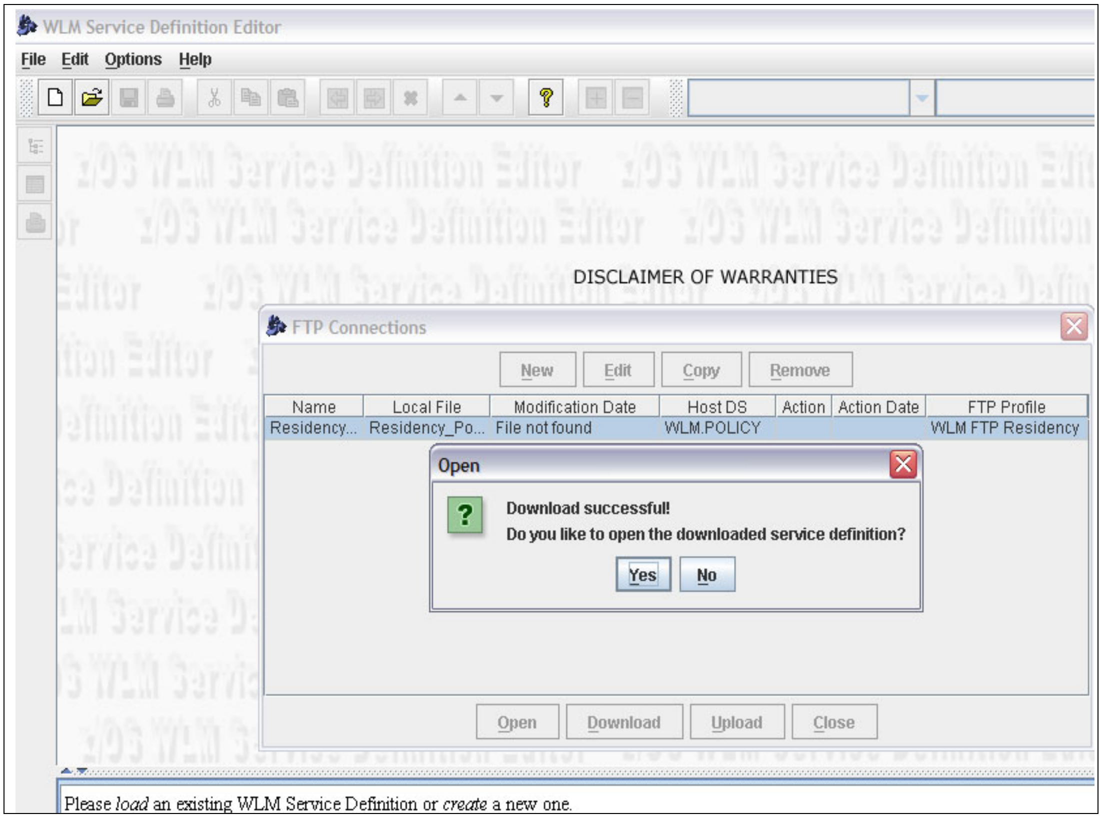
Please enter your password:

****

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-37 Downloading the service definition

In the DOS window that opens together with the Java application at startup time, you see the progress of your FTP. After the FTP finishes successfully, your service definition policy is now stored on your workstation. Select **Yes** to start working with your WLM service definition on your workstation. Figure 4-38 on page 160 shows an example of the workstation window.



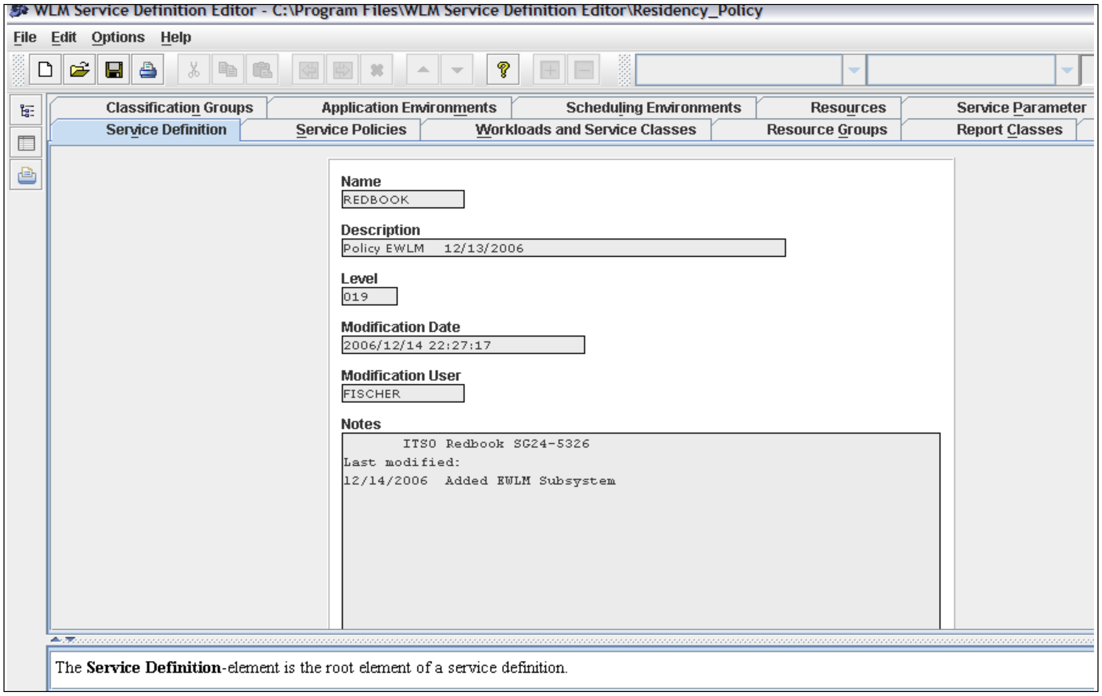
Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
Residency...	Residency_Po...	File not found	WLM.POLICY			WLM FTP Residency

Download successful!
Do you like to open the downloaded service definition?

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-38 Download WLM service definition

Now you are ready to work with your WLM service definition. To display the types of elements in the service definition, just click one of the tabs. Figure 4-39 shows the service definition window, which is the default window when you start.



Name
REDBOOK

Description
Policy EWLM 12/13/2006

Level
019

Modification Date
2006/12/14 22:27:17

Modification User
FISCHER

Notes

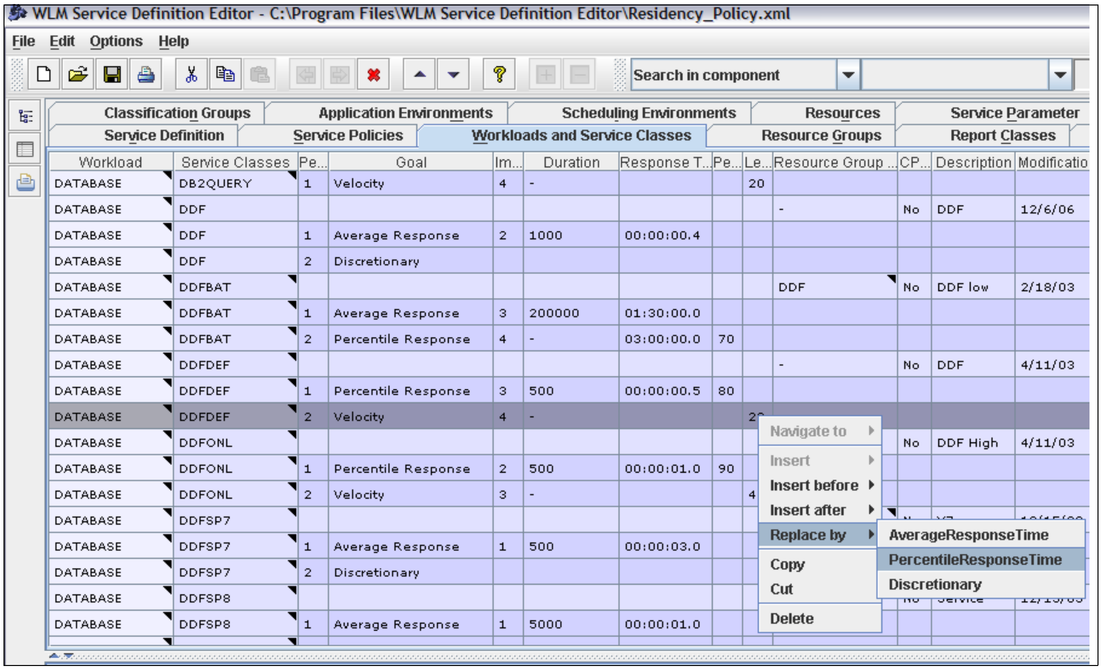
```
       ITSO Redbook SG24-5326
Last modified:
12/14/2006  Added EWLM Subsystem
```

The **Service Definition**-element is the root element of a service definition.

Figure 4-39 Display of the Service Definition

4. Modify your service definition.

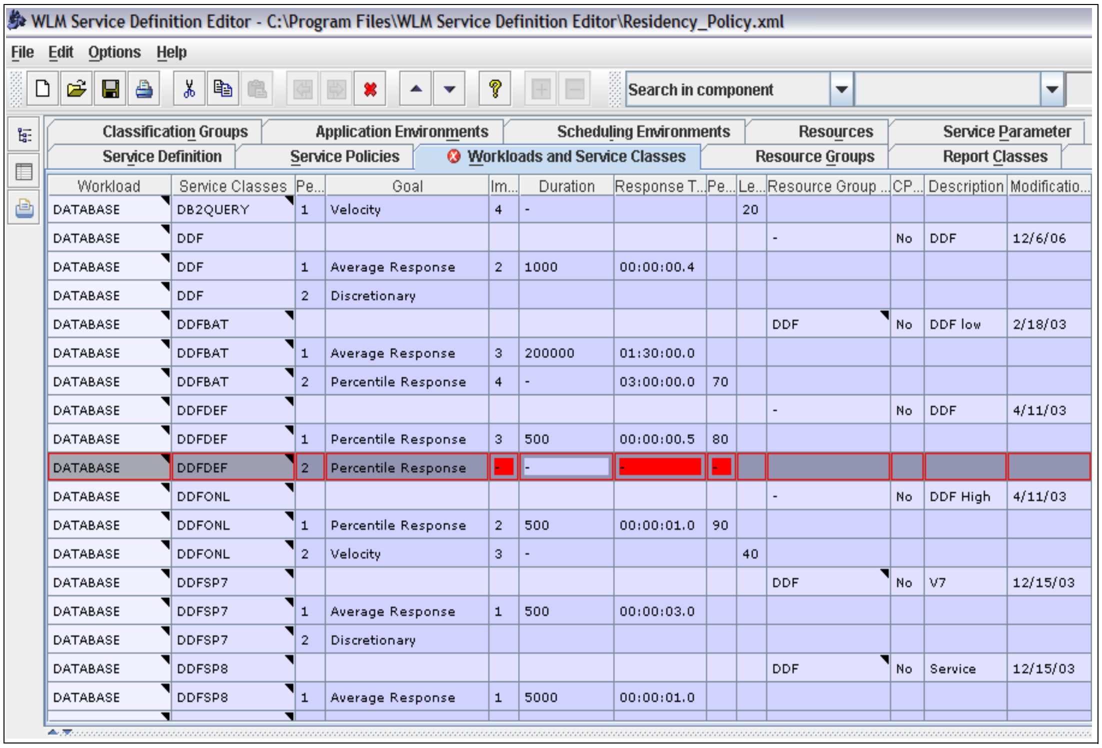
   To change a service class, select the **Workloads and Service Classes** tab, click the line of the service class that you want to change in order to select (highlight) that service class, and then right-click. Select **Replace by** and the type of goal you want. In our case, we want **PercentileResponse Time**. Figure 4-40 shows an example of the workstation window.



Workload	Service Classes	Pe...	Goal	Im...	Duration	Response T...	Pe...	Le...	Resource Group ...	CP...	Description	Modificatio
DATABASE	DB2QUERY	1	Velocity	4	-			20				
DATABASE	DDF								-	No	DDF	12/6/06
DATABASE	DDF	1	Average Response	2	1000	00:00:00.4						
DATABASE	DDF	2	Discretionary									
DATABASE	DDFBAT								DDF	No	DDF low	2/18/03
DATABASE	DDFBAT	1	Average Response	3	200000	01:30:00.0						
DATABASE	DDFBAT	2	Percentile Response	4	-	03:00:00.0	70					
DATABASE	DDFDEF								-	No	DDF	4/11/03
DATABASE	DDFDEF	1	Percentile Response	3	500	00:00:00.5	80					
DATABASE	DDFDEF	2	Velocity	4	-			2[illegible]				
DATABASE	DDFONL									No	DDF High	4/11/03
DATABASE	DDFONL	1	Percentile Response	2	500	00:00:01.0	90					
DATABASE	DDFONL	2	Velocity	3	-			4				
DATABASE	DDFSP7									[illegible]	[illegible]	[illegible]
DATABASE	DDFSP7	1	Average Response	1	500	00:00:03.0						
DATABASE	DDFSP7	2	Discretionary									
DATABASE	DDFSP8									[illegible]	[illegible]	[illegible]
DATABASE	DDFSP8	1	Average Response	1	5000	00:00:01.0						

Figure 4-40 Insert a new service class

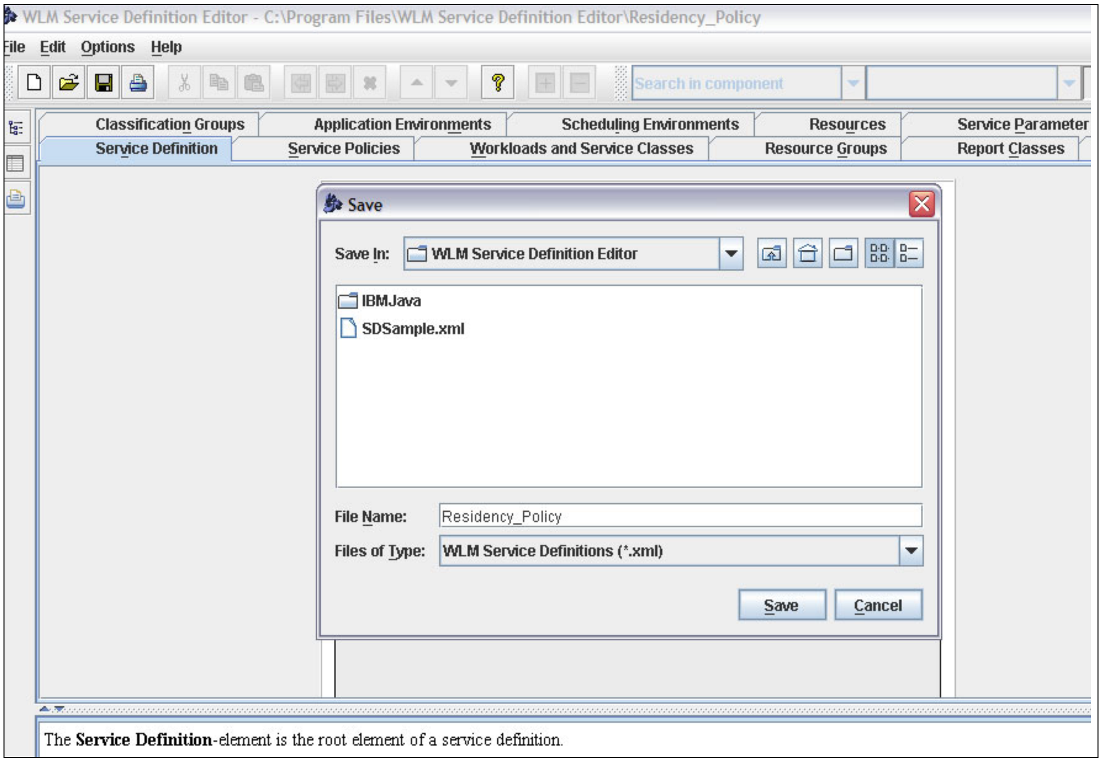
   Figure 4-41 on page 162 guides you through the definition process. The (red) highlighted areas are the minimum required areas that you have to fill with data.



Workload	Service Classes	Pe...	Goal	Im...	Duration	Response T...	Pe...	Le...	Resource Group ...	CP...	Description	Modificatio...
DATABASE	DB2QUERY	1	Velocity	4	-			20				
DATABASE	DDF								-	No	DDF	12/6/06
DATABASE	DDF	1	Average Response	2	1000	00:00:00.4						
DATABASE	DDF	2	Discretionary									
DATABASE	DDFBAT								DDF	No	DDF low	2/18/03
DATABASE	DDFBAT	1	Average Response	3	200000	01:30:00.0						
DATABASE	DDFBAT	2	Percentile Response	4	-	03:00:00.0	70					
DATABASE	DDFDEF								-	No	DDF	4/11/03
DATABASE	DDFDEF	1	Percentile Response	3	500	00:00:00.5	80					
DATABASE	DDFDEF	2	Percentile Response	-	-	-	-					
DATABASE	DDFONL								-	No	DDF High	4/11/03
DATABASE	DDFONL	1	Percentile Response	2	500	00:00:01.0	90					
DATABASE	DDFONL	2	Velocity	3	-			40				
DATABASE	DDFSP7								DDF	No	V7	12/15/03
DATABASE	DDFSP7	1	Average Response	1	500	00:00:03.0						
DATABASE	DDFSP7	2	Discretionary									
DATABASE	DDFSP8								DDF	No	Service	12/15/03
DATABASE	DDFSP8	1	Average Response	1	5000	00:00:01.0						

Figure 4-41 Change service class DDFDEF Period 2

You might want to save your downloaded and modified service definition now. To save your service definition, select **File** → **Save as** and enter the name where you want the application to save your Service Policy. Figure 4-42 on page 163 shows an example of the workstation window.



File Name: Residency_Policy

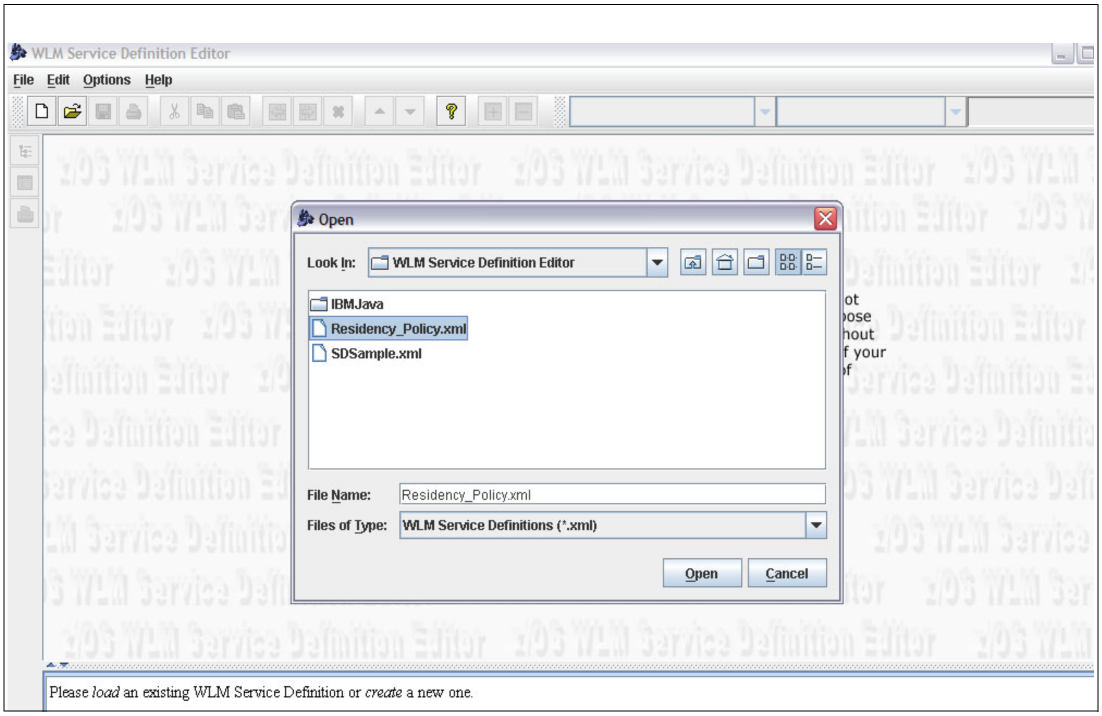
Files of Type: WLM Service Definitions (*.xml)

The **Service Definition**-element is the root element of a service definition.

Figure 4-42   Save the service definition

After specifying a name for your service definition, click **Save** to save the service definition.

The next time that you start the WLM service definition Editor and select **File** → **Open,** you see the Policy that you saved on the selection menu. Click the Policy that you want to open in order to select it (highlight it) and select **Open**. Figure 4-43 on page 164 shows an example of the workstation window.



Look In: WLM Service Definition Editor

IBMJava
Residency_Policy.xml
SDSample.xml

File Name: Residency_Policy.xml

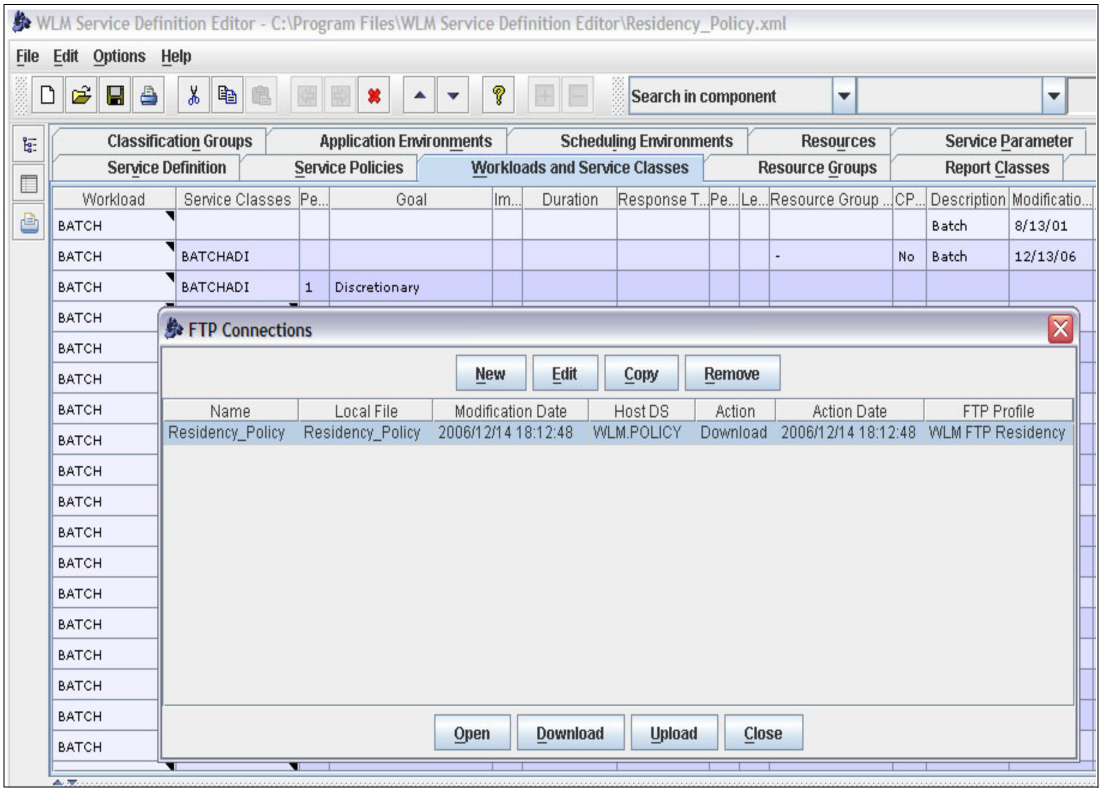
Files of Type: WLM Service Definitions (*.xml)

Please *load* an existing WLM Service Definition or *create* a new one.

Figure 4-43 Open saved service definition

5. Upload your service definition.

   After you finish with modifications, you can upload the changed service definition to the Host. Select **File** → **Transfer**, click the Name to select (highlight) your service definition, and select **Upload** on your workstation window. Figure 4-44 on page 165 shows an example of the workstation window.

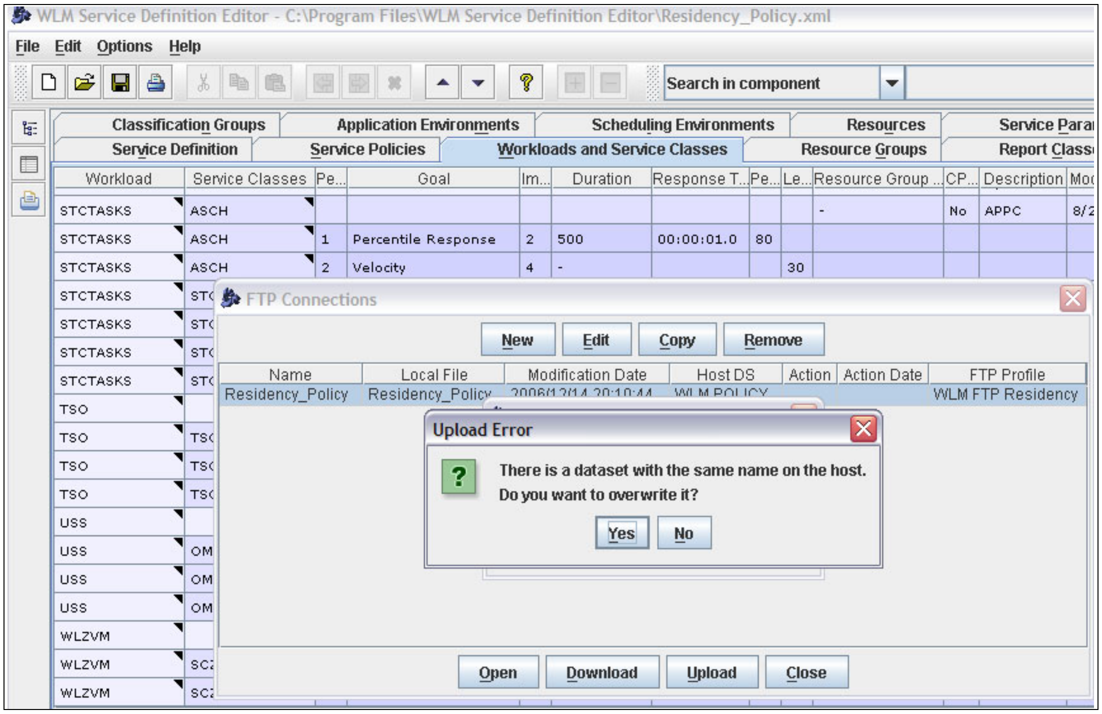


Workload	Service Classes	Pe...	Goal	Im...	Duration	Response T...	Pe...	Le...	Resource Group ...	CP...	Description	Modificatio...
BATCH											Batch	8/13/01
BATCH	BATCHADI								-	No	Batch	12/13/06
BATCH	BATCHADI	1	Discretionary									
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												
BATCH												

Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
Residency_Policy	Residency_Policy	2006/12/14 18:12:48	WLM.POLICY	Download	2006/12/14 18:12:48	WLM FTP Residency

Figure 4-44 Upload your service definition

The WLM service definition Editor application now initiates an FTP connection to the host. If the specified ISPF data set already exists on the host, a window appears asking you for permission to overwrite the existing data set. Select **Yes** to save your modified service definition to the ISPF data set. Figure 4-45 on page 166 shows an example of the workstation window.



Workload	Service Classes	Pe...	Goal	Im...	Duration	Response T...	Pe...	Le...	Resource Group ...	CP...	Description	Mo
STCTASKS	ASCH								-	No	APPC	8/2
STCTASKS	ASCH	1	Percentile Response	2	500	00:00:01.0	80					
STCTASKS	ASCH	2	Velocity	4	-			30				
STCTASKS	STC											
STCTASKS	STC											
STCTASKS	STC											
STCTASKS	STC											
TSO												
TSO	TSC											
TSO	TSC											
TSO	TSC											
USS												
USS	OM											
USS	OM											
USS	OM											
WLZVM												
WLZVM	SC											
WLZVM	SC											

Name	Local File	Modification Date	Host DS	Action	Action Date	FTP Profile
Residency_Policy	Residency_Policy	2006/12/14 20:10:44	WLM.POLICY			WLM FTP Residency

There is a dataset with the same name on the host.
Do you want to overwrite it?

Figure 4-45 Saving the new service definition

For further details, refer to the online help available on the application panel or to the documentation supplied in PDF format. You downloaded this documentation PDF file together with the other files into the folder where you installed the application. To open the PDF file, either select it when you start the Application or open WSEUserGuide.pdf from the folder.

5

Setting goals

This chapter gives you generic recommendations for setting up Workload Manager (WLM).

This chapter discusses tips to:

- Manage your service definition.
- Use a service class.
- Select the type of goals to use.
- Use Resource Groups.

5.1 Using service classes

In setting up your service class, there are two objectives that you want to achieve:

- Distinguish different work and define service classes so that different groups of users get the service that they need from the operating system.
- Keep your service classes to a minimum. Every 10 seconds, WLM examines all service classes in order to determine which service classes need help. If there are too many service classes, then it can take a long time for WLM to help lower importance service classes.

  Additionally, the more service classes you have, the more likely it is that some of the lower importance service classes receive no service if the system is CPU-constrained.

We should define service classes to distinguish all types of work and users from each other; however, we should minimize the number of service classes in order to help WLM to work efficiently. Here are suggestions to help satisfy both requirements:

- Define a default service class with a specific name for each major subsystem even if it is not used today in the environment. With this approach, new work is always classified by the subsystem that brings it into the system. Otherwise, new work is classified to the SYSOTHER service class, and it is a good practice to leave this service class empty. It is also a good practice to define a default service class for the STC subsystem. Otherwise, address spaces that do not match any classification attribute are classified in the SYSSTC service class.
- Service classes should only be defined for work when sufficient demand exists for them in the system. You can measure the demand either by the service consumption or by the amount of ending transactions during high utilization periods of this work. It probably does not make sense to define service classes with a demand of less than 1% of measured service units at any period or less than 1 transaction ending per minute. It is usually better to combine this work with other similar work.
- It is not meaningful to create service classes for the same type of work with identical service objectives. For example, if you define started tasks for different applications with exactly the same goals and importance, you can combine them in the same service class and use report classes to differentiate them for reporting purposes.
- Do not combine diverse types of work into one service class. For example, do not combine transactions and address spaces into one service class. SRM sampling data, plots, and projections can become distorted when you do this.
- Do not define extra service classes for reporting purposes. WLM supports up to 999 report classes, which should satisfy all reporting requirements. See 5.2.9, “Using report classes” on page 183 for instructions to set up report classes.
- Use the SPM qualifier to correctly classify system address spaces in the SYSTEM or SYSSTC service classes. For a detailed discussion of SYSTEM and SYSSTC, refer to “General considerations for STC” on page 224. System tasks are tasks that are given the privileged and system task attribute in the IBM-supplied program properties table or in the SCHED*xx* parmlib member. SPM rules assign these tasks to either SYSTEM or SYSSTC. Classifying system address spaces into service classes with low service or low importance can cause severe system problems. Example 7-4 on page 227 shows a classification method and explains the usage of the SPM parameter and what you should manually classify in SYSSTC.
- Limit the number of started task service classes to two or three in addition to SYSSTC and SYSTEM.

- Identify other critical system-started tasks, and if CPU requirements are reasonable and there is sufficient storage, assign them to SYSSTC.
- Monitor the SYSSTC service class's CPU usage. If high importance and online systems experience delay caused by SYSSTC, adjust SYSSTC by moving heavy CPU users to a different service class.
- Evaluate the need for a high velocity service class which can be used to push emergency batch jobs through the system during periods of heavy CPU utilization. Only use it for special cases; otherwise, it can be counterproductive. We recommend that you monitor this service class usage.
- Consider creating a special high velocity, high importance TSO service class for emergencies. If the system hangs, you can use the RESET command to assign your TSO user ID to this class to allow you to investigate. Note that you might still need to wait for WLM to identify that this service class is delayed and adjust its priority. Assigning the TSO user ID to SYSSTC is an alternative.
- To handle runaway jobs or transactions, consider creating a "sleeper" service class. Associate the service class with a Resource Group and specify a maximum service unit capacity of 1. "Quiescing a batch job" on page 208 shows you how to use this Resource Group setting for batch and you can use this example in other cases.
- Ensure that you have properly classified enclave transactions in the WLM service definition in order to prevent them from being managed as discretionary work.
- Avoid the common error of not classifying Distributed DB2 Transactions ((running under the Distributed Data Facility (DDF) subsystem type)), and then they end up managed as discretionary work.
- Consider the following method to handle unclassified work. Create a service class (for example, service class = UNCLASS) that you can use as the default service class in the classification rules for subsystems that have not been specifically assigned a service class. Assign this service class an appropriate goal and give it a unique report class to identify work that has fallen through the rules. Regularly check for work in SYSOTHER or your installation's defined default service class (UNCLASS) and classify it to the correct service class.

Using these guidelines, you should be able to stay in a range of 25 to 30 service class periods with nondiscretionary goals, even in environments with a high diversity of work on the systems. It is not a problem to define more service class periods as long as not all of them are active at the same time. Thirty is not a hard boundary. It is possible to run a sysplex environment with more active service classes. We recommend 30 to reduce efficiency problems. Note that service classes with discretionary goals do not count, because they are not managed to goals.

5.2 Defining and maintaining service goals

Use the following guidelines when defining your service goals for the first time or in the process of reevaluating your goal definitions.

Always define goals based on measurement intervals of high system utilization or high system contention periods. This gives you the best indication of whether the goals can be achieved.

You should base goals on what can actually be achieved, not what you want to happen. The goals you set are how WLM knows whether your system is running OK, or whether it needs to make changes. If you set goals higher than necessary, WLM moves resources from lower importance work to higher importance work which might not actually need the resources.

Avoid using goals to prioritize work. Use the degree of importance to indicate the importance of the work, and set the goal to what is actually needed. Do not expect work of higher importance to always get a higher dispatching priority than lesser important work. Dispatching priorities depend on the goal and the characteristics of the work. It is common to see low velocity goal work with a dispatching priority higher than high velocity work due to differences in the work; commonly, high priority work also demands more CPU. If you really need to manage the dispatching priorities, use the CPU Critical function.

The test of whether the WLM policy works is when the system is heavily loaded. At quiet times, just about any policy performs OK. When the system is busy, the policy determines where limited resources go.

5.2.1 Using historical data to set goals

In Figure 5-1, we want to determine a goal for an IMS service class. Figure 5-1 shows the CEC, system, and workload utilization for IMS. You can observe four periods of high system utilization in this trend report, which shows a two-day period at the end of a quarter in 2004. For all four periods, the utilization of the CEC and the utilization of the LPAR or system are high, but only the first and the third periods show a high utilization for the IMS workload. These are the periods on which we base our decisions.

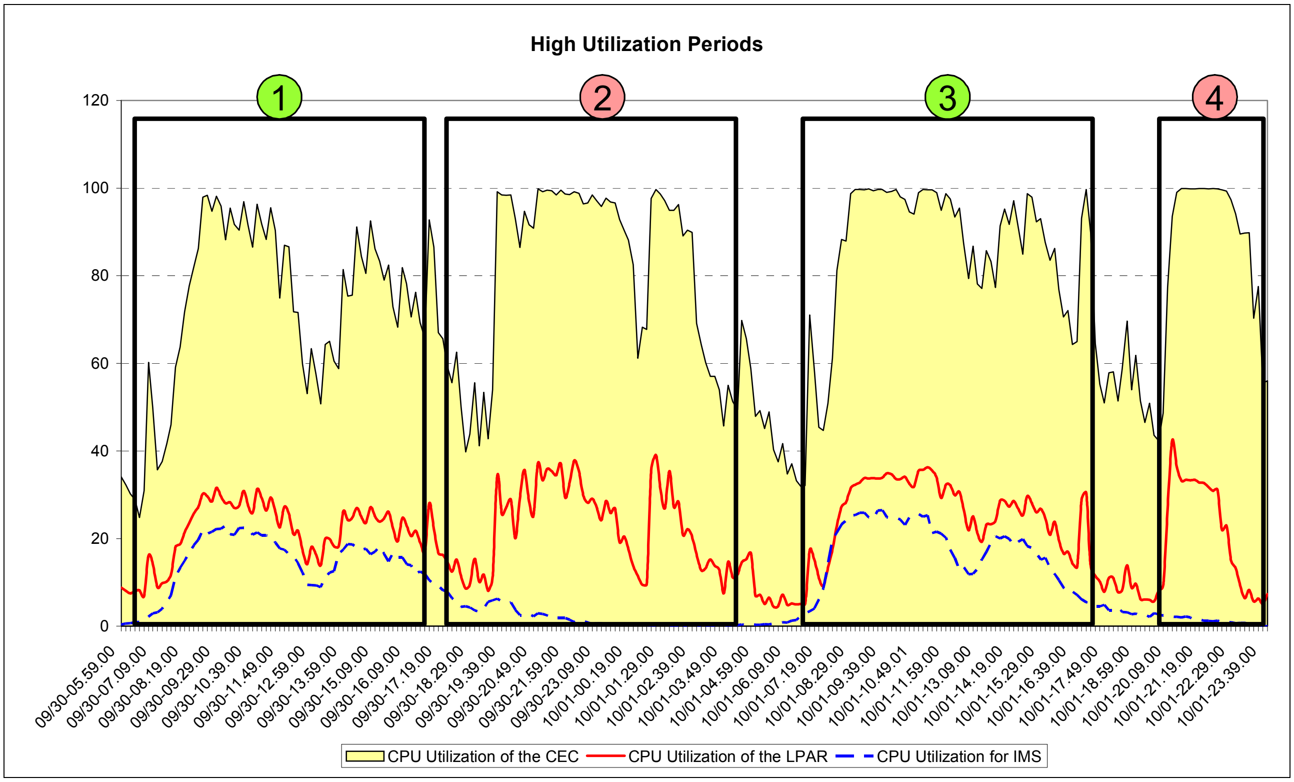


Figure 5-1 IMS service class utilization periods

The graphic in Figure 5-2 on page 171 shows the addition of the achieved execution velocity. Observe that the value is extremely variable during low utilization periods, and that the lowest values for a longer time period happen during the identified high utilization periods for IMS. The two squares mark the range for both periods in which the achieved execution velocity varies. For the first period, it is in the range between 30 and 55, and in the second period, it is in the range from below 20 to 40.

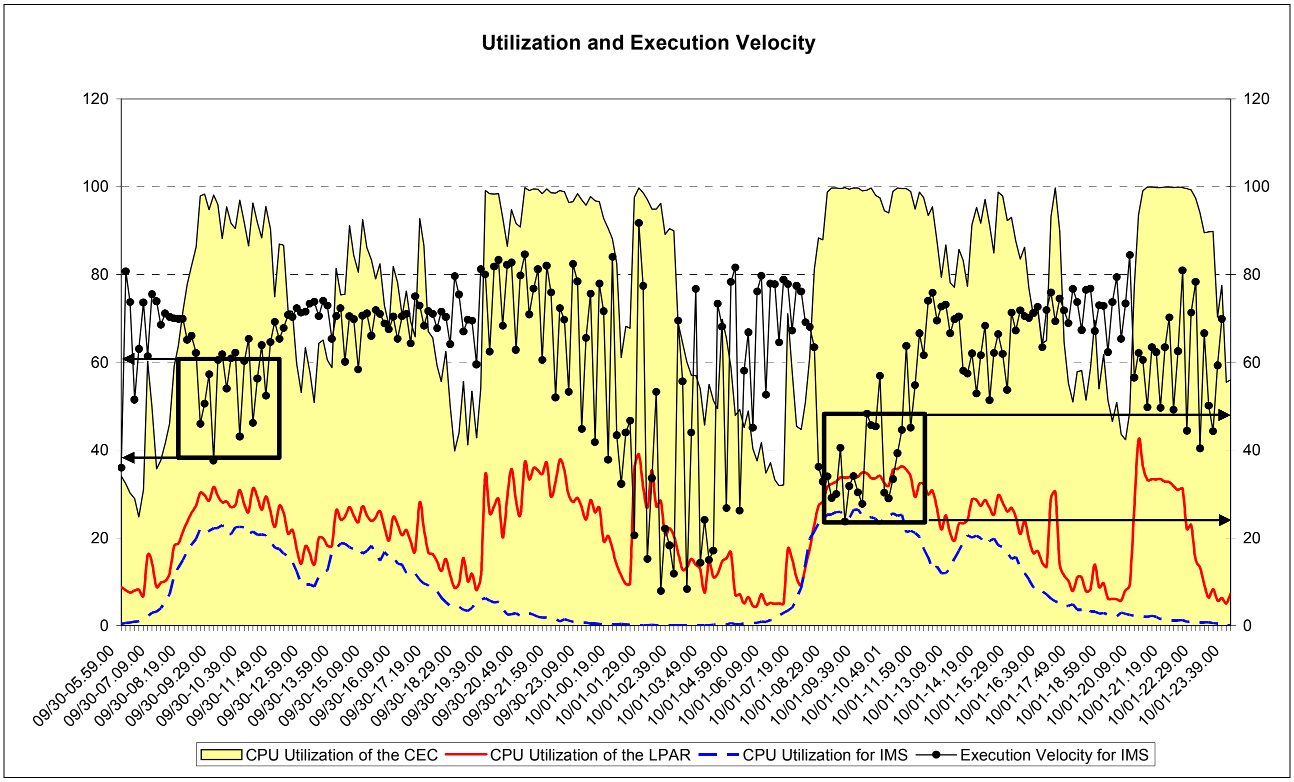


Figure 5-2 Utilization and execution velocity

This is now the base for defining the execution velocity goal. We can also observe that a value seems to be the lower margin during normal contention periods. As a result, a value of 40 is probably a meaningful value for this work on this system. During very high periods, the goal value is a little too high, so WLM puts more attention to this work; while during normal periods, this value can still be achieved. An alternative is to specify a value of 50. This is now already a stringent value during normal contention periods, and it is only meaningful if the period of 10/01 is really exceptional. If this system is part of a sysplex, you must verify whether this value is usable for the work on the other systems, too.

This example shows an approach to determine an execution velocity goal for the work. The approach for determining a response time goal is similar. Figure 5-3 on page 172 shows the same time period for the same IMS workload. Now, we plot the IMS ended transaction rate (ETRX) and the ended transaction time (TTM) also. Note that the ended transaction rate follows the curve of the IMS workload utilization from the chart in Figure 5-2. The average transaction ended time also shows a similar pattern to the execution velocity on the chart in Figure 5-2. While it also shows high fluctuation during periods with few transactions running, it is quite stable during periods where many transactions are in the system. Again, on 10/1, the average response time is slightly worse than during other high utilization periods.

Nevertheless, notice that with all similarities, the change in response time seems to have a less noticeable effect than the change in execution velocity even if the observed values are doubled in both cases. The execution velocity shows a range from 30 to nearly 60 during a higher contention period, and the average response time is within a range between 0.1 seconds and 0.2 seconds.

In order to completely assess the response time goal, you must examine the response time distribution as well. If this is the first step to introduce the response time goal, the distribution might not be available, but you need to consider it in a later step. As a starting point, we suggest that you define an initial percentile between 85% and 90%, but you need to adjust the response time value, too.

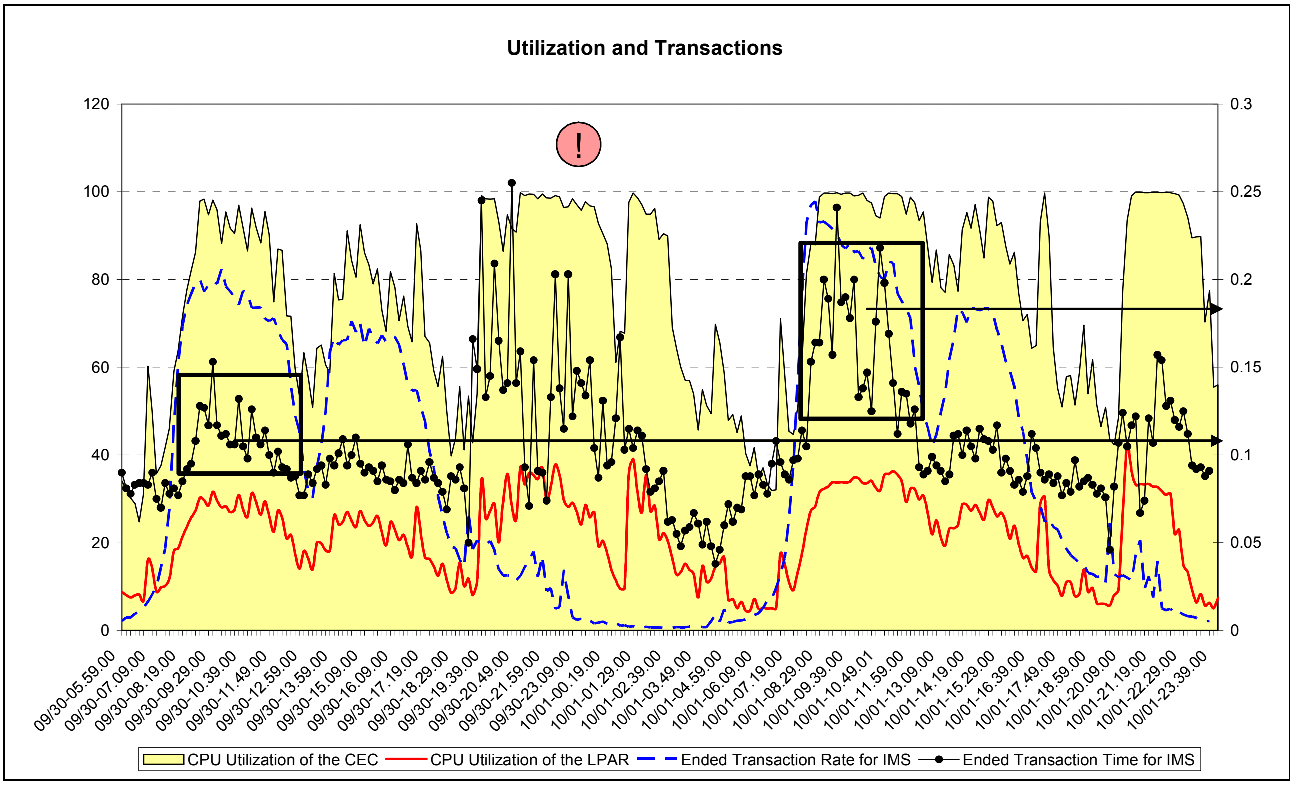


Figure 5-3 Utilization and transactions

In these examples, we demonstrate how to select a time period to derive an execution velocity or response time goal for an online workload. Now, we need to determine which periods to use when we try to define the goals for the batch work. Referring back to Figure 5-1 on page 170, we see that there were two periods when the online workload was executed and two high utilization periods where other work, in this case, batch, was executed. In order to determine which periods to use, consider:

- If the batch work really runs only in the periods when no online work is running, we can use the same approach as above to determine the time period for the goal definition for batch.
- If the batch work also runs in parallel to the online work, it is better for you to use the same periods you use for the online work, even if the batch CPU consumption is much smaller during these periods. This is simply because, in most cases, we do not want batch to impact the online work.

5.2.2 Using response time goals

There are two types of response time goal: *average response time* and *percentile response time*. Work must have at least 10 completions in 20 minutes to use a response time goal.

The response time represents the time that WLM is aware of the unit of work. For an enclave, this is the time between enclave create and enclave delete. For CICS and IMS, it is the time when CICS and IMS start the monitoring environment for the transaction until the WLM API report is issued. For a batch job, response time is the time when the job has finished the conversion step until it is completed and the result is placed on the output queue. For TSO, response time is the time from when the user presses Enter until the result is returned to the terminal. Note that the response time measures different things depending on the subsystem that is measured, and it is not an end-to-end response time.

In percentile response time goals, you set a response time target for a given percentage of transactions (for example, 90% of transactions within 1 second). WLM tries to ensure that the selected percentage of transactions finishes within the specified time. If there are a few transactions that last much longer than most, they do not affect the management of the majority.

You might used average response time goals if the transaction response times of a workload are extremely homogeneous. Unfortunately, this is often not the case. You can also use average response time goals during goal mode migration. It can be used as a starting point for setting your response time goals, if you only have the existing average response time. Initially, set an average response time goal. After running for a few days, you can collect the response time distribution from RMF and use this distribution to determine whether the workload is better suited for an average response time goal or a percentile response time goal.

A few transactions with very long response times can greatly influence average response time goals. Figure 5-4 on page 174 shows a response time distribution that is measured internally by WLM. The example shows how only a few long-running transactions can influence an average response time goal.

In Figure 5-4 on page 174, the average response time goal was set to 1 second. For the observation period, there were 435 transactions running or ending. WLM internally creates buckets around the response time goal in order to measure how many transactions are ending or running within response time ranges. WLM does this for any response time goal. The blue bars show the amount of transactions ending or running in a certain bucket. For example, 141 of the 435 transactions were ending or running within half of the goal. The red curve shows the accumulated number of transactions for all buckets up to the current bucket. Around 70% of all transactions were found in buckets within the goal and 90% of all transactions were within 1.25 seconds.

However, the average response time was higher than ten seconds. This was caused by Three very long-running transactions, which appear in the right-most bucket, caused this situation. The difficulty is that it is nearly impossible for WLM to find a positive resource adjustment for the work when the goal is missed by a factor of 10. If the goal had been defined as a percentile response time of 90% of the transactions to be completed within one second, then the goal would have been missed by only a small number of transactions. WLM has a much better chance of rearranging access to the resources to help the work running in this service class.

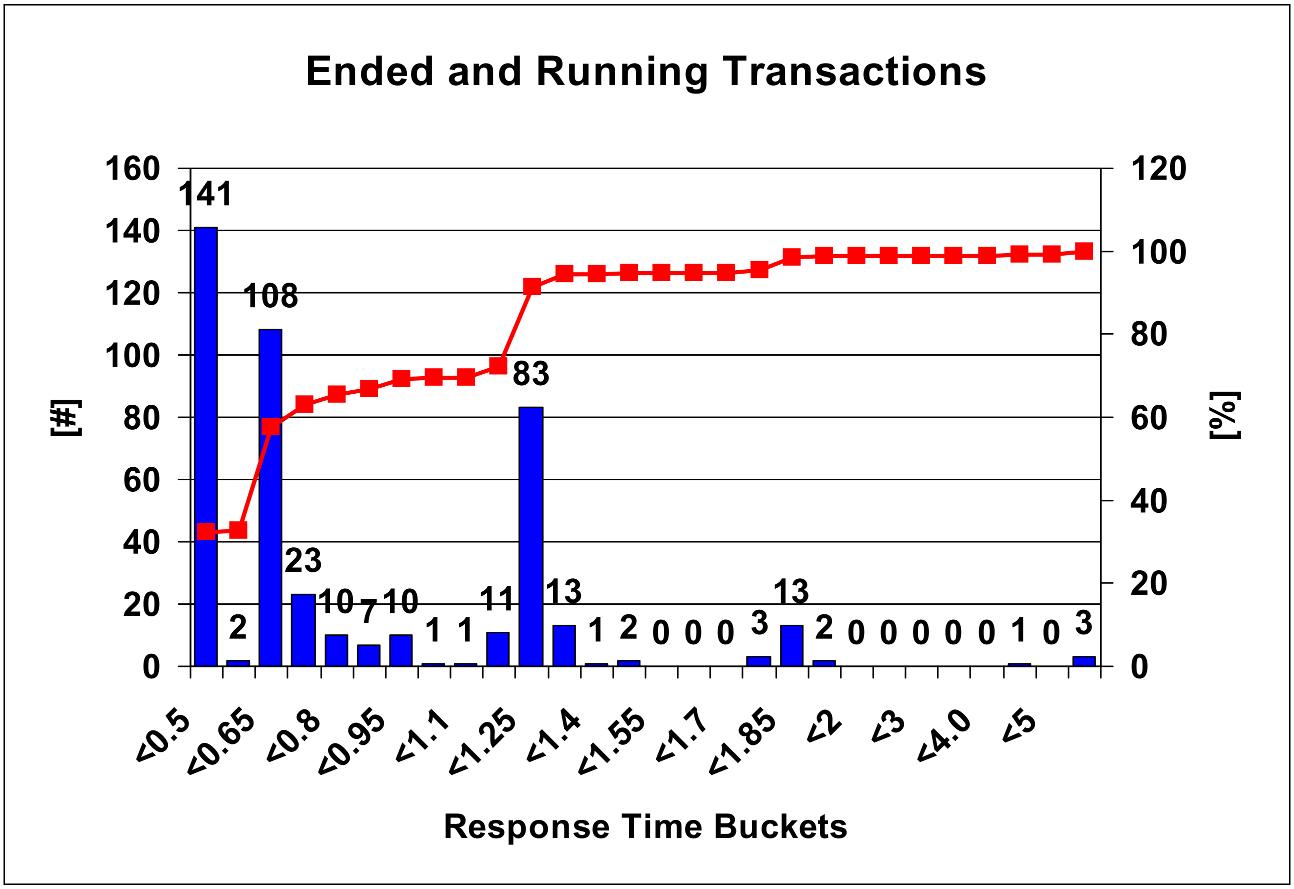


Figure 5-4 Response time buckets

The example in Figure 5-4 also shows that too stringent of a goal is not always helpful. The goal must always allow WLM to make adjustments. This is most likely when the performance index is around 1 or does not exceed 1 by too high of a margin.

For storage-constrained systems, the usage of response time goals might provide an important advantage for interactive workloads.

5.2.3 Using response time goals for CICS and IMS

WLM allows you to classify the work of the CICS and IMS subsystems to service classes with a response time goal. The advantage of doing this is that you get much better and more granular reporting feedback of the work and that you use a more stable goal definition for the CICS and IMS work as opposed to using execution velocity goals for the CICS and IMS regions. Since OS/390 Release 10, you can also exempt certain regions from response time management and thus exclude work, which cannot be managed very well toward response time goals, from the management.

For using response time goals, you need to decide how much granularity you really want for CICS and IMS work, or is it possible to distinguish a small amount of potentially long-running transactions from the majority of short-running transactions. In order to answer this question, we have to review how WLM manages transactions. As we see in this chapter, WLM creates internal service classes for all address spaces that execute the same set of transactions. If we now classify work to two different service classes (SCSHORT with a high importance and a short response time goal and SCLONG with a low importance and a long response time goal), WLM creates up to three internal service classes based on possible combinations of transactions that are executed by the server regions. The three internal service classes are:

- INTA = {only transactions for SCSHORT}
- INTB = {only transactions for SCLONG}
- INTC = {transactions for SCLONG and SCSHORT}

As the system runs, we see that all transactions were potentially executed by all available server regions of the CICS environment. A deeper analysis showed that 65 CICS server regions were in the system, and that one of these CICS server regions executed transactions of SCLONG and SCSHORT. By examining the contribution of the work running in each region and service class, we can draw the following picture (Figure 5-5).

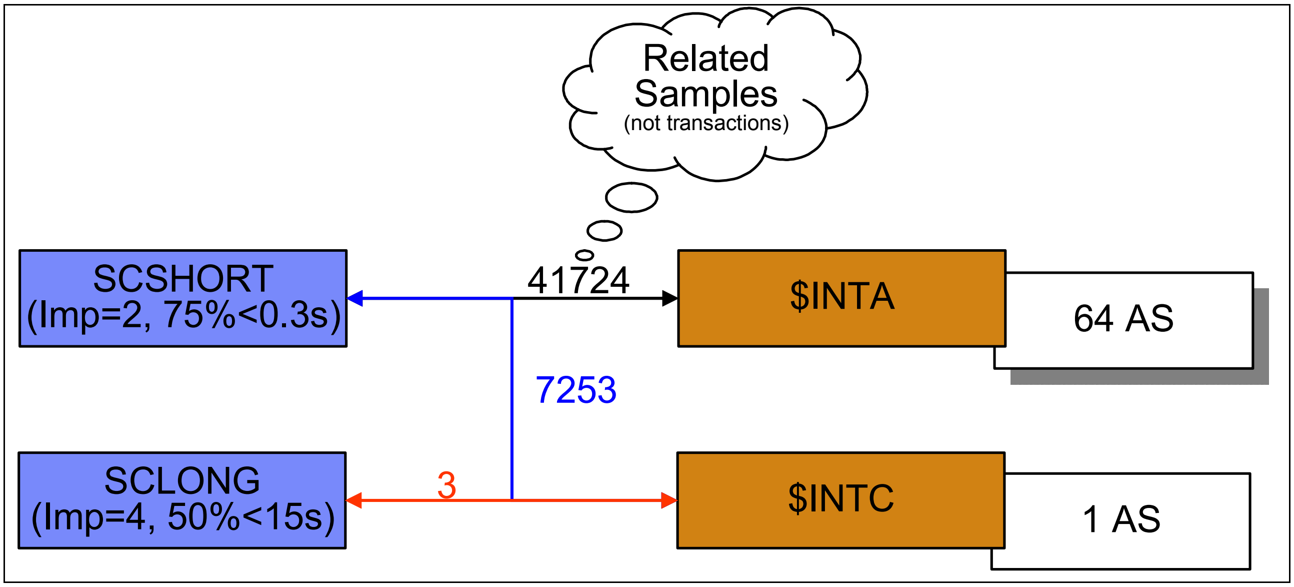


Figure 5-5 CICS service classes contribution of work

Figure 5-5 shows that only a few samplings were executed against the address space executing the transactions associated with the SCLONG service class, while transactions for the SCSHORT service class were executed across all the address spaces.

Figure 5-6 on page 176 now shows a surprising result. It plots the dispatch priorities for both internal service class periods, and we can observe that the dispatch priority for the internal service class $INTC is usually higher than the dispatch priority for $INTA. This is certainly not the result we want, because the external service class SCLONG was created to give the long-running transactions less access to the system resources than the short-running transactions.

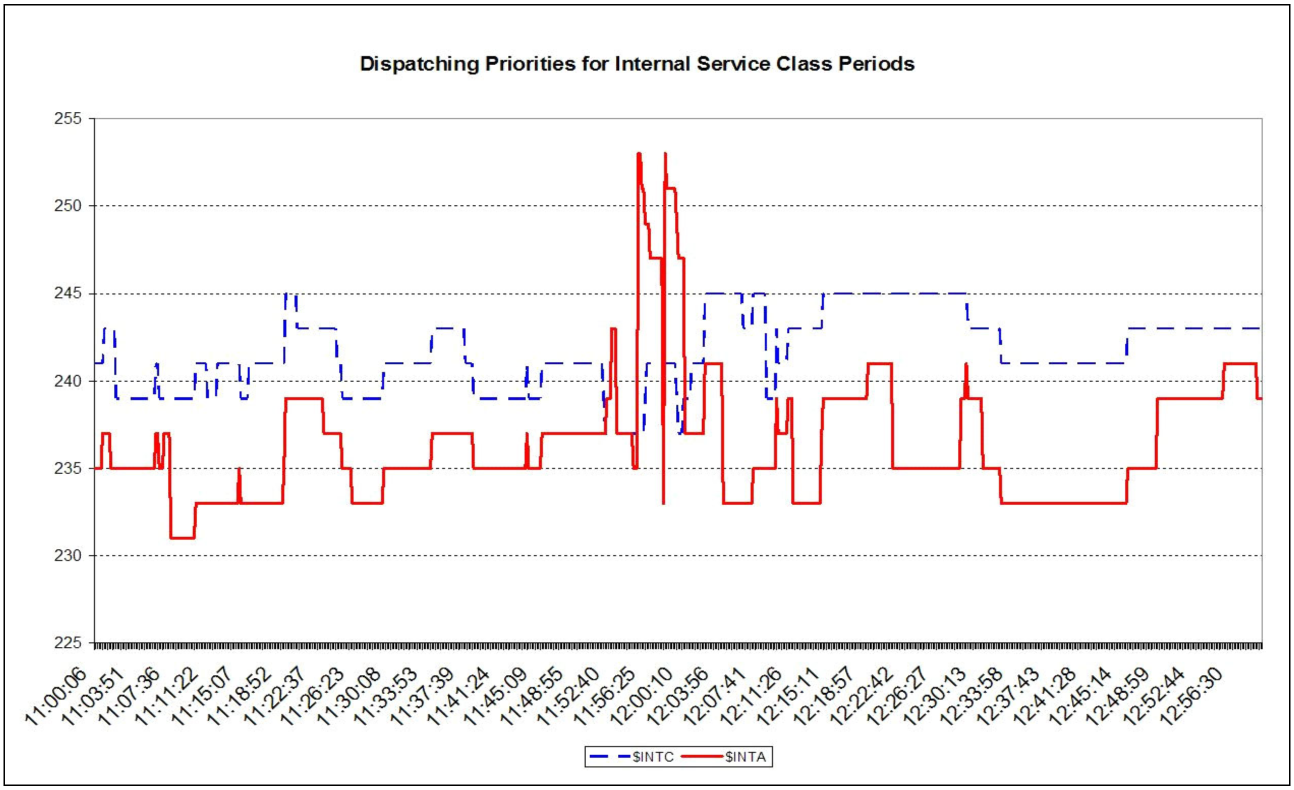


Figure 5-6 Dispatching priorities for internal service classes

So, why is this the case? The answer is simple. The internal service class with only one address space requires much lower CPU than the service class with 64 address spaces, and WLM always tries to optimize the exploitation of the existing system resources. We can also observe that in cases where the service goals for SCSHORT are not met, WLM reverses this decision (around 11:57), because the 64 address spaces have a higher contribution to the overall goal achievement for SCSHORT.

Nevertheless, this situation is unexpected, and also shows that distinction is not always beneficial. In case you do not know whether it is meaningful to distinguish different types of transactions, use report classes and monitor the amount of ending transactions for each report class. This gives you an idea as to whether a separation is meaningful. In this case, we should merge the two service class definitions into one to let the percentile response time process discard the outlying transactions (long-running transactions). However, we recommend that you define two report classes (one for the short transactions and one for the long-running transactions) to get and monitor the average response times of both types of transactions.

5.2.4 Using execution velocity goals

Execution velocity goals define the amount of delay that is acceptable for the work, or, in other words, how long we use a resource compared to the amount of time that we wait to use it.

Use velocity goals when other types of goals are inappropriate, such as:

- When WLM cannot obtain the transaction start and stop time, for example, there are many started tasks.
- Workloads that do not have enough transactions completing for you to use a response time goal. WLM needs to see at least three transactions complete within a 20-minute interval to get useful statistics for a response time goal.
- Work that includes user think time, for example, when a program sends data back to the requester and waits on a response before continuing.

A velocity goal disadvantage is that velocity goals are difficult to translate into real world terms. It is reasonably easy to understand what a response time of 90% of transactions completing in 0.75 seconds means. Users can typically tell the difference between that and 90% complete in 1.5 seconds. Can users tell the difference between a velocity of 60% and 30%? If they are equivalent to response times of 0.1 second and 0.2 second, probably not. If they mean response times of 0.75 or 1.5 seconds, then yes. At times, it is difficult to judge whether a velocity goal is acceptable, too high, or too low.

The achievable velocity for a workload depends greatly on the number of CPUs on the system. Section 6.6, “Impact of the number of engines on velocity” on page 212 shows examples of the effect.

On the 5 CPU machine with DB2 running in SYSSTC so it is at the highest priority, DB2 achieved a velocity of 74.9%. The same workload on the single CPU system only achieved 13.3%. On the 5-way machine, a goal of 60-70% might be appropriate. On the single processor, that goal is much too high and 10% is more reasonable.

Considerations for using velocity goals

When using velocity goals:

- You usually cannot achieve an execution velocity goal above 90% because of reduced preemption. *Reduced preemption* refers to the fact that a low-priority address space or enclave can remain dispatched for a short amount of time even after a higher priority address space or enclave becomes ready to execute. Another reason is that all the SYSTEM and SYSSTC address spaces have a higher dispatching priority than any velocity-type goal.
- Execution velocity depends on how many samples are collected for a service class and, therefore, also on the amount of work running in a service class. Consider this example. For simplicity, we use only CPU using and delay samples and assume that no other samples have been collected:
  - You have a service class with one unit of work. In this case, it is only possible to collect 40 samples in a 10-second period, because WLM samples the state of the work every 250 ms. Let us assume that we found the work ready and waiting for the CPU 20 times, using the CPU 10 times and in an idle state 10 times. The idle samples are not counted, so the achieved execution velocity for the work is 33.
  - Now let us take a service class with 10 units of work. WLM can now collect 10 times more samples in a 10-second period. For an execution velocity of 33, WLM now needs 200 delay samples and 100 using samples.

  We can see that a small change for the service class with only 1 unit of work makes a big difference to WLM. To accommodate these situations, WLM uses a minimum number of samples on which to base its decisions. In order to obtain a minimum, WLM keeps a history of samples and accumulates the number of history times until the minimum amount is reached. This means that WLM needs to base its decisions for the service class with only one unit of work on a much longer time period than for the service class with 10 units of work.

This also means that WLM can respond more quickly when there are multiple address spaces in a service class. With a single address space, SRM needs several minutes to collect enough samples. This is another good reason to combine little used service classes with others.

- Certain work, such as VTAM and IRLM, spend the majority of time in the unknown state and the idle state. Because of this, there might be few samples for using and delay (the ones used for execution velocity calculation). As a consequence, the velocity can fluctuate sharply. So, do not be surprised if you see an RMF report showing the execution velocity of an address space, such as VTAM, with low figures. Velocity goals are normally unsuitable for VTAM and IRLM.
- Many clients think in terms of high/medium/low velocities. Execution velocities depend on various aspects and they can vary for many reasons. Therefore, we recommend that you distinguish execution velocity goals of different service classes by at least a value of 10. Everything lower than 10 does not make a big difference and cannot really be distinguished by WLM from a measurement aspect.
- The same work often achieves different velocities if it is run multiple times. Many factors affect the velocity that is achieved.
- When WLM calculates the execution velocity, it does not take into account the unknown state. This includes the delays not tracked by SRM, such as locks or enqueues. Contrast this with response time goals, where the time that corresponds to unknown samples is included in the response time. In a resource-constrained system, service classes with large unknown times with velocity goals exhibit performance indexes (PIs) that vary more than other goal types, because the work is effectively managed on a subset of its total delays. Try to use response time goals if possible, instead of execution velocity goals.
- Consider using low velocities, low importance levels, and Resource Groups for work with the potential to dominate system resources.
- Reevaluate velocity goals when you turn on I/O priority queuing or batch initiator management. These functions change the velocity calculation. RMF reports the projected velocity values (migration velocities) to help you determine new goals before you migrate to these functions.
- Reevaluate velocity goals when you change your hardware. In particular, moving to fewer, faster processors requires changes to velocity goals.

5.2.5 Using the SYSSTC service class

The SYSSTC service class does not have a goal, but it runs with a fixed dispatching priority above all of the WLM-managed work.

Use SYSSTC for work that is of high importance but uses little CPU. Address spaces, such as VTAM, TCP/IP, and IRLM are good examples. System monitor address spaces might go in SYSSTC as long as they are well behaved; that is, they only use CPU for short periods of time. Multiple CPU systems might be able to tolerate occasional high CPU users in SYSSTC, but this becomes critical with only one or two CPUs. On a single CPU system, all regular work stops while a task in SYSSTC is using the CPU. WLM cannot adjust priorities to give regular work more CPU at the expense of SYSSTC.

5.2.6 Using service class periods

Service class periods allow you to age work while it is running in the system. This is extremely useful in cases where it is unpredictable how long work really runs in the system. It is a mechanism to protect your system from long-running work that consumes too many resources because it runs at an importance level that is too high.

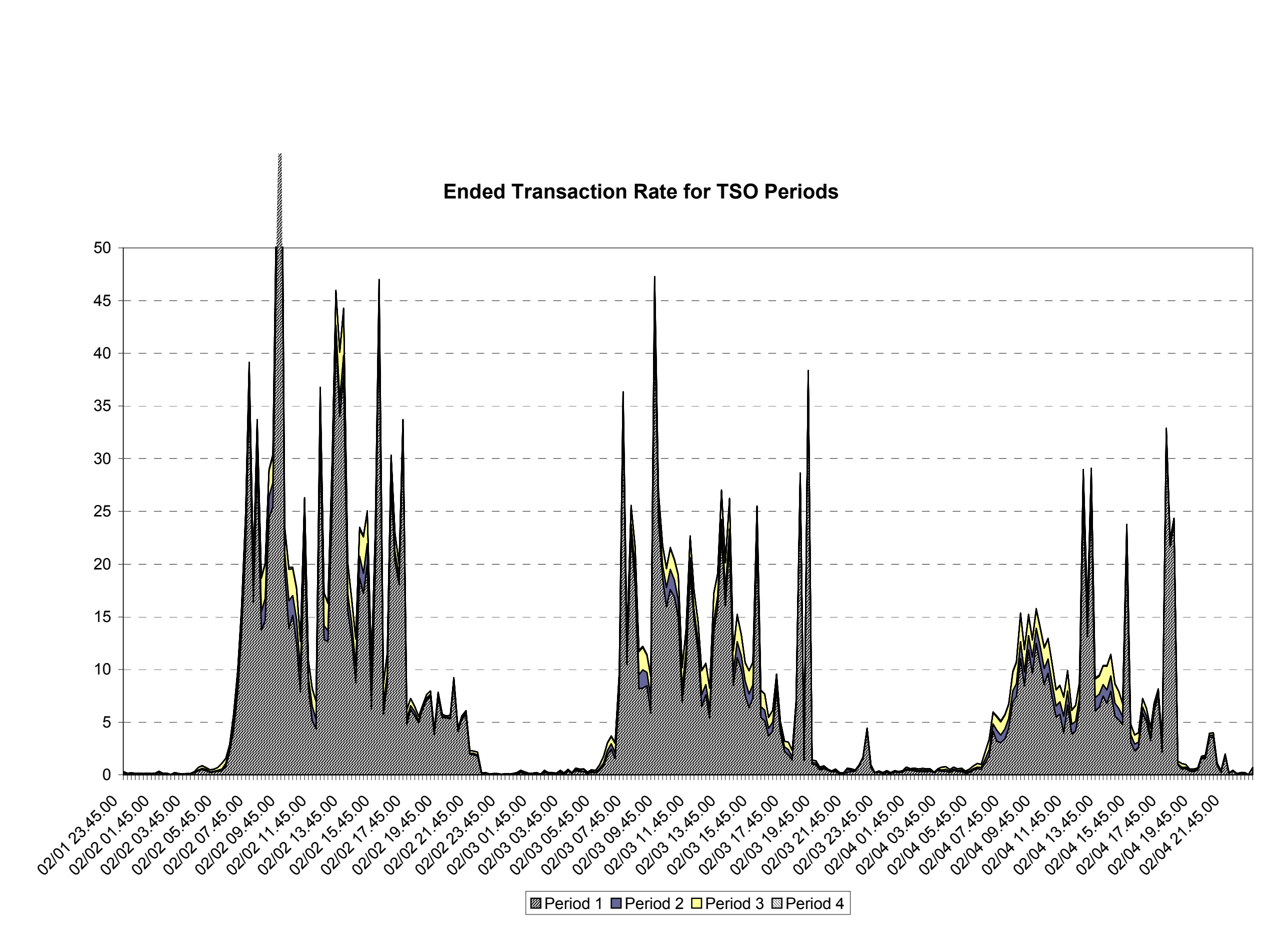


Figure 5-7 Ended transaction rate for TSO periods

To use service class periods efficiently, examine how many transactions end in each period and how much CPU each period consumes, as shown in Figure 5-8 on page 180. Ideally, the majority of all transactions end in the first period with only a small amount in the last period.

Figure 5-8 on page 180 shows a TSO service class with four periods. Observe that the bulk of all transactions end in the first period, and only a small amount of transactions end in the other three periods. Figure 5-8 on page 180 shows the CPU consumption in each period. Now, we can observe the inverse picture. The first period consumes only a small amount of CPU, while the last period, which runs at the lowest importance level, consumes a high amount of CPU.

This example also shows an undesirable setup for the second and third periods. The second period only consumes a small amount of CPU, and there are not many transactions ending in this period. You can remove the second period and put it together with the third period. Many examples show that two or three periods are sufficient in most cases to implement a meaningful workload aging algorithm.

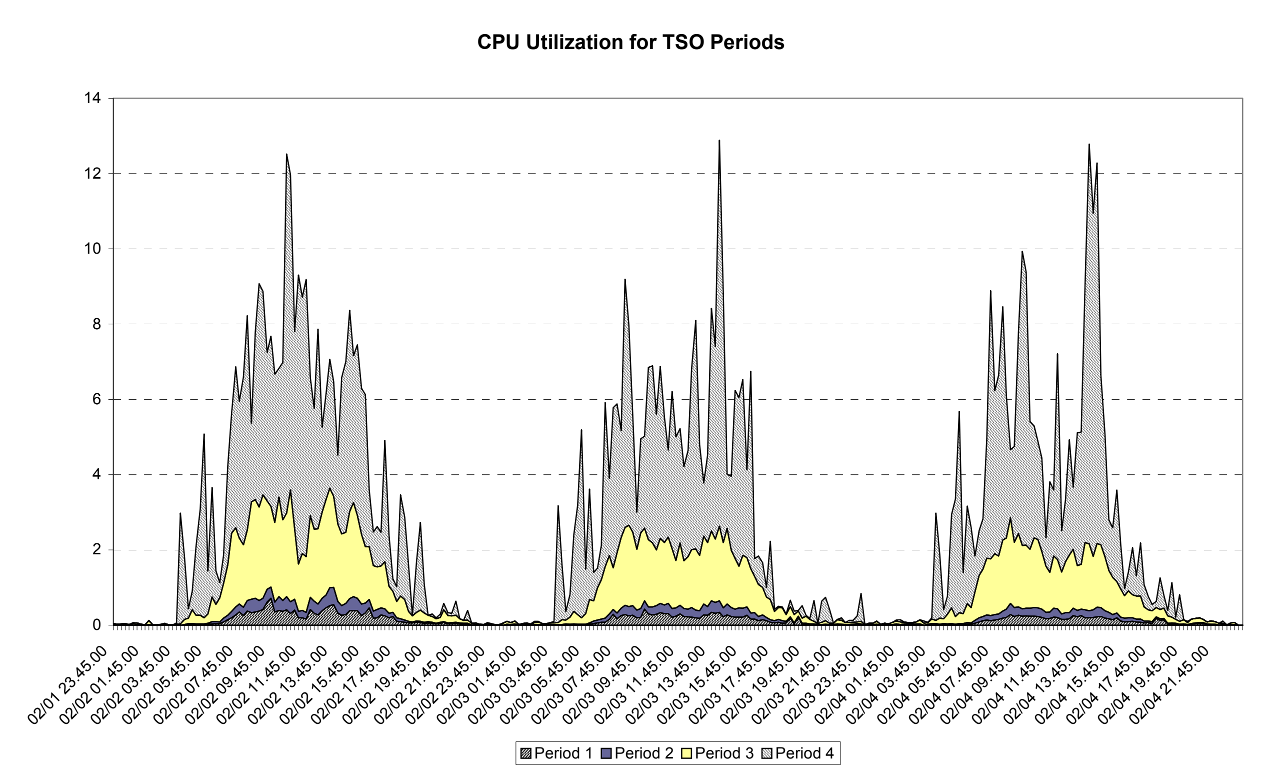


Figure 5-8 CPU utilization for TSO periods

A simple way to check that you have set the proper durations into the service class definition period is to analyze a RMF report class report (SGPER), as shown in Example 5-1.

Example 5-1 SGPER RMF Report

```
 PER IMPORTANCE  PERF     --TRANSACTIONS--     -------------RESPONSE TIME--------------   -EX VEL%-   -CPU--  -EXE--
                 INDX     -NUMBER-      -%-     ------GOAL------  ---ACTUAL---     TOTAL   GOAL  ACT   USING%  DELAY%
  1    1          0.5        11292       94     00.00.00.150  90%    97.1%           91.7%        57.1      0.1     0.0
  2    2          0.6          564        5     00.00.00.750  90%    95.6%                        76.1     19.6     9.4
  3    2          0.3           96        1                                                    25 73.7     36.5    18.6
TOTAL                        11952      100
```

We use the TRANSACTIONS NUMBER (or percentage) column to analyze the transactions' distribution of the different periods.

In Example 5-1, we see that 94% of the total amount of ended transactions were executed in the first period, 5% in the second period, and only 1% in the third period.

Furthermore, for each period, we can correlate these numbers with the APPL % field in the SCPER RMF report to check the amount of CPU per transaction period, which ideally is small in the first period and higher in the other periods.

5.2.7 Using importance levels

The *importance* of a service class tells WLM which work to prefer and from which work to take resources if resources become tight on the system. Ideally, your installation exploits all existing importance levels, but in many cases, the work running on the systems only runs in one or two importance levels. See Figure 5-9 on page 181.

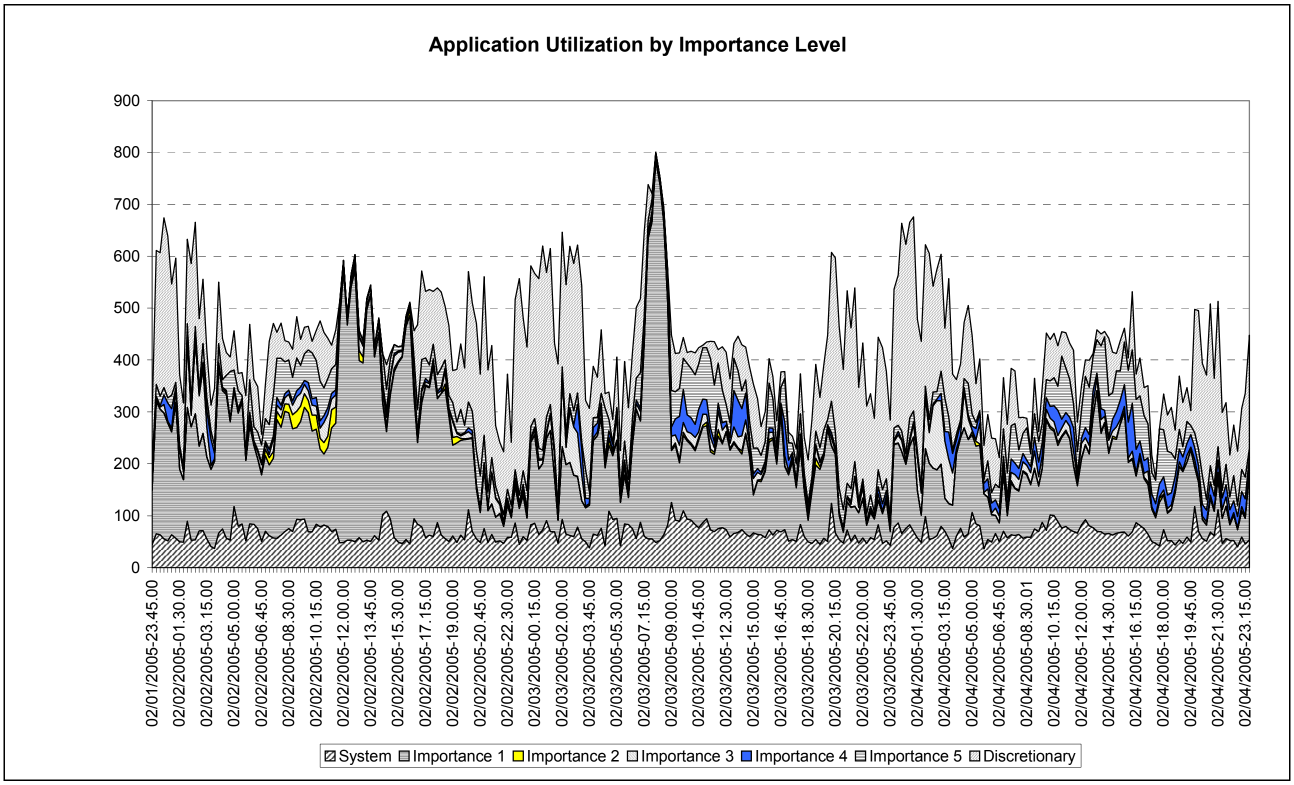


Figure 5-9 Application utilization showing importance 1 overutilization

Figure 5-9 shows a typical picture of the work running in different importance levels on a system. The majority of the work runs at importance 1 while there is some batch work running at importance 6.

Figure 5-10 on page 182 shows a better usage of the available WLM importance levels. Nevertheless, noticed that in this example a lot of work is running at the system level in service class SYSSTC.

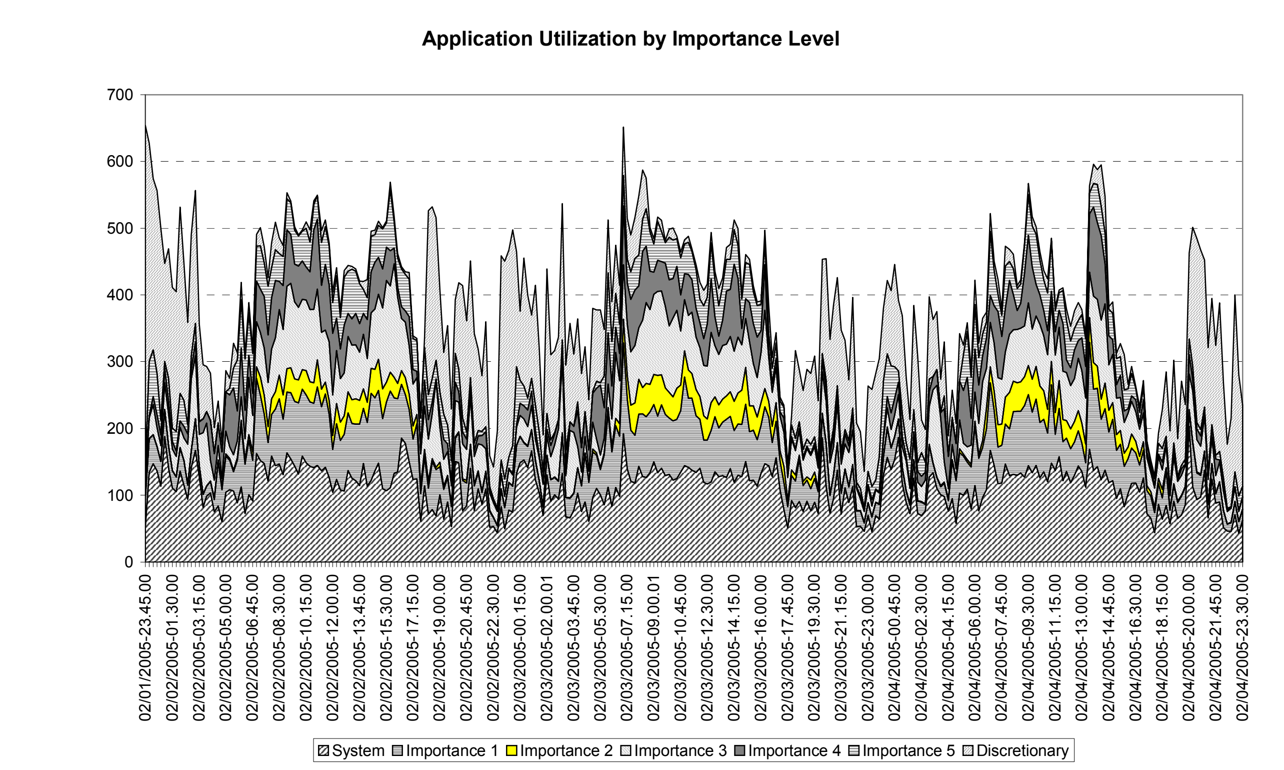


Figure 5-10 Application utilization showing a better use of importance levels

5.2.8 Using discretionary goals

Discretionary goals are for work that is processed using resources that other work does not require to meet the other work's goals. Discretionary work has no specific business goals attached to it.

Discretionary work can make it easier to manage an extremely busy system. A z/OS system can run at 100% CPU busy without problems, as long as there is work that can wait during the peaks of more important work. Defining discretionary work allows WLM to know immediately which work must donate resources when an important workload spikes without having to wait for a WLM interval and going through the normal donor/receiver logic. In fact, WLM does not have to do anything, because SRM and the normal dispatch priorities handle discretionary work.

For discretionary work to function properly, the percentage of discretionary work needs to be high enough to buffer the normal fluctuations of the nondiscretionary work. If there is sufficient nondiscretionary work to drive the system to 100% busy for long periods of time, then discretionary receives very poor service, and you do not get the benefit of having work that can immediately donate resources.

If you have work in the system that should always be the donor when more important work needs CPU immediately, then that work is a good candidate for discretionary work. Long running, resource intensive batch jobs are a good example.

When discretionary work receives too little service

If discretionary work receives too little service, there are two possible solutions:

- Use a service class with a low importance goal instead of discretionary. However, depending on your other workloads, this service class still might not receive adequate service.
- Move *more* work into discretionary. If only a small percentage of your work is discretionary, it is likely to experience problems when the system is busy. By moving more low importance work from other service classes into discretionary, you might be able to reduce the amount of nondiscretionary work enough that discretionary as a whole receives better service. This also allows your more important work to benefit from the flexibility that discretionary provides.

Capping overachieving work

When nondiscretionary work is overachieving its goals, WLM can cap it to give more service to discretionary work.

WLM only steals resources from other work that meets the following criteria:

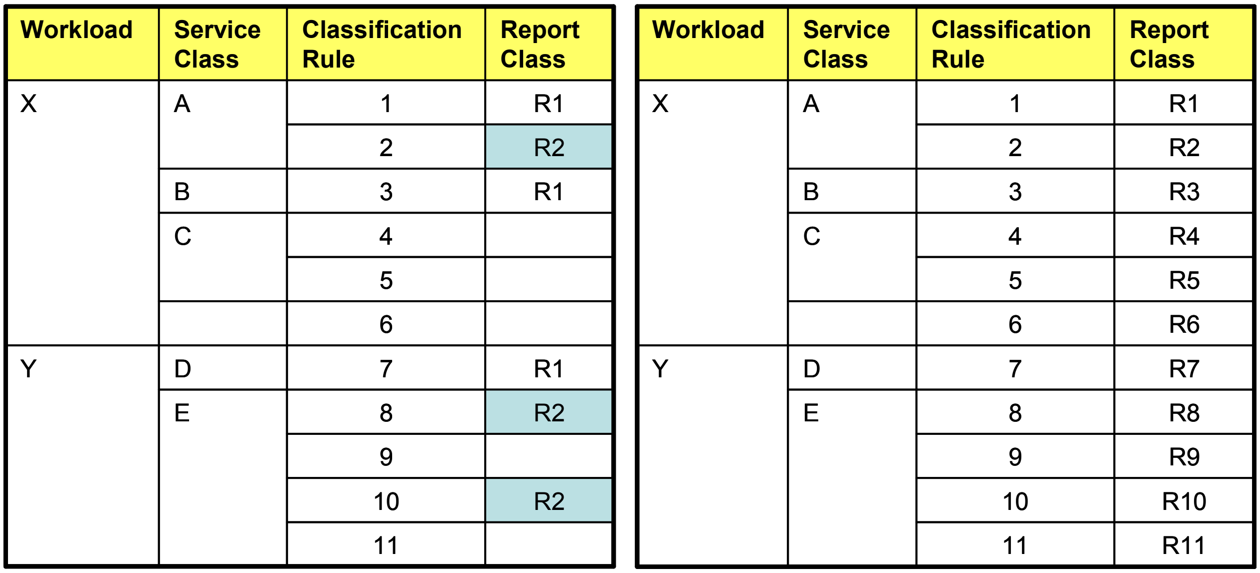
- It is not part of a Resource Group.
- It has a velocity goal of not more than 30 or a response time goal of over one minute.
- The PI of the service class is less than 0.71.

When the conditions are met, this overachieving work is capped and the freed resource is used to execute discretionary work. The amount of resource stolen is very small, because WLM keeps the PI of the capped work between 0.71 to 0.81.

It is up to your installation whether you want to define discretionary goals. If you have work that fits into this criteria, then there is no harm in using discretionary goals. If you do not like the capping algorithm that is used with discretionary goals and you want to protect your low-priority work from being capped, then you can easily negate this by defining a Resource Group with null minimum and maximum values. You must assign this Resource Group to the service classes that you want to protect from capping.

5.2.9 Using report classes

One important activity is defining your report classes. *Report classes* allow the definition of sufficient granularity to obtain more detailed information for analysis. See Figure 5-11 on page 184.



Workload	Service Class	Classification Rule	Report Class
X	A	1	R1
		2	R2
	B	3	R1
	C	4	
		5	
		6	
Y	D	7	R1
	E	8	R2
		9	
		10	R2
		11	

Workload	Service Class	Classification Rule	Report Class
X	A	1	R1
		2	R2
	B	3	R3
	C	4	R4
		5	R5
		6	R6
Y	D	7	R7
	E	8	R8
		9	R9
		10	R10
		11	R11

Figure 5-11 Report Classes definition

Figure 5-11 shows two typical ways to define report classes relative to workloads, service classes, and classification rules in a service definition. The table on the left uses distinct report classes to group work together that is related because of similar business or installation characteristics. The table on the right uses report classes to distinguish all classification rules and to provide more granular characteristics of the workload. Many clients define report classes like the example on the left; however, the example on the right provides you a better understanding of how different classifications are attributed to a service class. The example on the right also provides you a better understanding of which parts of a workload might grow and which might shrink. You can then combine report classes for your reporting according to your business requirements.

We advise you to use report classes extensively. They are the base for a deeper understanding of the work running in service classes. Also, starting with z/OS 1.3, report classes provide the same granularity as service classes. Finally, one of the greatest strengths of WLM is its ability to monitor the work running in the system, and report classes are the way to make this visible for your installation.

5.2.10 Using Resource Groups

A *Resource Group* is an amount of CPU capacity across the sysplex that you can assign to one or more service classes. You can use a Resource Group to:

- Limit the amount of processing capacity available to one or more service classes.
- Set a minimum amount of processing for one or more service classes in the event that the work is not achieving its goals.

You specify the minimum and maximum amount of capacity in unweighted CPU service units, LPAR share, or percentage of a single CPU. The minimum and maximum capacities apply to all systems in a sysplex. You can assign only one Resource Group to a service class. You can assign multiple service classes to one Resource Group. You can define up to 32 Resource Groups in a service definition.

When you use a Resource Group, remember that:

- If a Resource Group is not meeting its minimum capacity and work in that Resource Group is missing its goals, WLM attempts to give CPU resources to that work, even if the action causes more important work (outside the Resource Group) to miss its goal.
- If work in a Resource Group is consuming resources above the specified maximum capacity, the system throttles the associated work to slow down the rate of resource consumption. The system can use a number of ways to do this. It can swap the address space out, change its dispatching priority, or cap the amount of service units that can be consumed.

If you use Resource Groups, it might be advantageous to change the CPU and SRB user coefficients to 1. If you do not use Resource Groups, continue using the current user coefficients. Migrate to the final set of coefficients prior to migrating to goal mode.

For a more detailed discussion of Resource Groups, refer to 4.2.7, “Resource Group specification” on page 127.

A Resource Group is an optional function. Unless you have a special need to limit or protect processor capacity for a group of work, skip defining a Resource Group and just let WLM manage all of the processor capacity to meet performance goals.

5.2.11 Using CPU and storage critical

If you use CPU or storage critical, be aware that this only protects your work for the specific resource. You still need a meaningful goal for your work for other cases that can happen in the system. The following example demonstrates what can happen if you do not have a meaningful goal for other work in the system.

In Figure 5-12 on page 186, we defined a service class with importance 1 and CPU critical. We set the goal to an execution velocity goal of 1, and this was the only importance 1 work in the system. Now during normal operations, the CPU critical option ensures that the dispatch priority of the work is always above all other work in the system, but there are situations where this is insufficient. At a certain point in time, it became necessary to restart the work in this service class due to an application failure (1 on chart). In addition, the system ran in a stress situation and the CPU resource became constrained. WLM now looked for a way to help the system, and one of the main controls in such cases is to swap out work in order to regain system resources. Because of the restart, the address spaces of the work running in the importance 1 service class became swappable, and WLM chose this address space for swap out (2 on chart). It took a long time for the address space to get into the system again (3 on chart), because the goal was so low and the performance index of the service class was always below 0.15. Only over time, WLM increased the multiprogramming level (MPL) target to let the address spaces back.

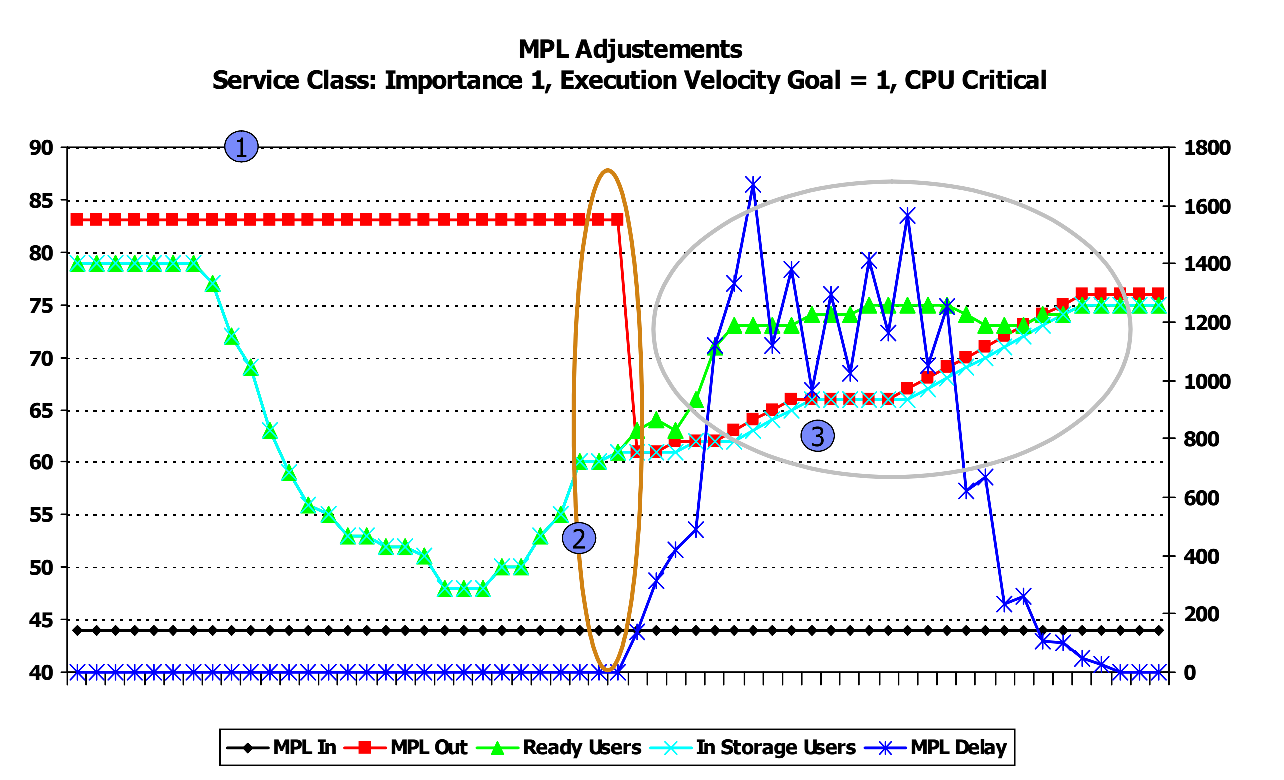


Figure 5-12 CPU critical scenario

To explain Figure 5-12:

- MPL delays are in the right Y-axis and are expressed in number of samples.
- MPL In is the number of address spaces that must always stay in storage (strg).
- MPL Out is the maximum number of address spaces that can stay in strg.

When the Ready Users number is greater than the In Storage Users number, you can then assume that the delta is SWAPPED OUT USERS. This situation can be the cause for the MPL Delay.

5.3 Managing service definitions

It is important to develop your own operational procedures when it comes to updating and installing your service definitions. A service definition can be extracted to a partitioned data set (PDS) and can be updated by multiple people at the same time. Also, if you saved to PDS, edited this service definition, and then did an install of the service definition, you overlaid the data in the WLM CDS. The previous service definition no longer exists. If you ever need to retrieve the old service definition, you must extract the current definition, undo the changes made, and then reinstall the definition.

To ensure data integrity, be sure that the service definition changes are always made to the latest service definition. The best way to do this is to always extract the service definition from the WLM CDS. We also suggest that you have one data set naming convention for your service definition. You also need to create your backup procedures to ensure that you can revert back to the original service definitions if needed.

Here is a sample procedure to manage your service definition:

1. Define your naming convention for your service definition, for example, HLQ.WLMPOL*xx*.V*mmddyy*.
2. Extract your service definition from the WLM CDS. This PDS serves as your current backup. Give this PDS a name such as `HLQ.WLMPOL00.V032105.`
3. When you need to update your service definition, extract the definition to another PDS. All changes in the service definition will be stored in this new PDS. So, at this stage, there are two definition PDSs in the system. For example, the new PDS has a name of `HLQ.WLMPOL01.V040305` and the original is `HLQ.WLMPOL00.V032105`.
4. Update your service definition. Use the notepad facility in the menu bar to add a change record for the service definition.
5. Install the service definition into the WLM CDS.
6. Exit the WLM ISPF application.

This is just a sample procedure. You can tailor one according to your local procedures.

5.3.1 Revisiting service definitions

Revisiting a service definition comprises the following steps:

1. Determine whether your classification rules still work: whether all of your service classes are still used and whether the usage of your service classes has changed.
2. Determine whether the achievement of your goals still falls in the range you decided for your installation. Do this regularly in order to avoid problems for your installation.
3. Determine whether Application Environment and scheduling environment definitions are still appropriate.

5.3.2 Reevaluating service definition

View this task as a consequence of revisiting the service definition:

1. If classification rules do not match any longer or if the workload has changed, it might be necessary to change the rules, remove service classes, and potentially add new service classes.
2. If goals are not in the expected range or workloads do not perform as expected, you might need to tune the goal definitions.
3. If any definition in the service definition becomes obsolete or if you determine that the service definition does not provide enough information, adjust the definitions.

5.3.3 When things are not working

The objective of setting up a WLM policy is to specify how resources should be allocated when they are in short supply. When things work properly, and you have a mix of different work in the system, the system should be able to run smoothly even at 100% busy for periods of time. The key is to know which work can be delayed when things get busy.

Setting up a single CPU system is more difficult than multiple CPU systems. On a single CPU system, the CPU is 100% busy for short periods of time even when the average utilization is low, which can cause problems such as erratic response times. A single CPU system is more sensitive to problems with the WLM policy.

You need to look at these areas when you start having problems when the system is busy:

- Try to ensure that even your lowest priority workloads get at least some service. When work receives no service at all, you start to see additional problems. For example, a job needs some CPU service to respond to the cancel command. If a job receives no CPU at all, you will probably have problems cancelling the job.

  Too many low importance service classes can cause this problem. The classes at the end of the queue might never get service. The best solution is to combine service classes so that there is a reasonable amount of work running in each service class. For example, if you have five low importance service classes, you might find the fourth and fifth do not get any service. If you combine them all into one service class (or make them discretionary), they all share the service that was received by the first three service classes.

- Make sure that your goals are reasonable. If your goals are too high, WLM might waste time on high importance workloads that are really acceptable, instead of helping lower importance work that needs attention.

  Goals that you set too high can result in erratic performance, because WLM makes changes to try to meet the goals. Ideally, you want WLM to find a setting that meets all your goals and only changes things when the workload changes. This results in the most consistent and predictable performance.

- Remember that the system is finite. It can seem wrong to take resources from more important work and give them to less important work. However, that is often necessary to allow the less important work to perform adequately. As long as the more important work achieves its business objectives, it makes sense to do it. This is what WLM is designed to do.

- Do not be afraid to change multiple things at once if many service classes are not meeting their goals. System programmers tend to be cautious, and they like to change one thing at a time and observe the results. However, the performance of all of the service classes in a system is interrelated, and if you miss many goals, you might not be able to detect a result from changing a single goal. You cannot tell whether the change was beneficial or not.

  It is often better to make relatively large adjustments until you get close to what you need. When the WLM policy is mostly working, you can make smaller adjustments to fine-tune the system.

- There is no single perfect policy. The WLM policy does not directly change the priorities of work. WLM uses the policy to measure how well the work is running and work out whether changes are necessary. In many cases, different policies can give much the same result. It often does not make sense to obsess over small details.

  Conversely, sometimes a small change can make a big difference. If you have a service class with period 1 duration of 100000, and a lot of work in the service class was completing in just over 100000 service units, a small increase in the duration might make a dramatic difference to the performance by allowing the work to finish in period 1.

6

Batch considerations

This chapter discusses practical recommendations to manage batch workload with Workload Manager (WLM). It covers the following subjects:

- WLM-managed initiators: How they work and the latest enhancements
- Types of goals that you need to use for batch jobs
- How to classify the batch environment
- Usage of policy overrides and Resource Groups

6.1 General considerations

Usage of WLM-managed initiators is a major part of your considerations for batch workload. Classification and type of goal for a batch job are dependent on the type of initiator you use. So, before giving recommendations about service class goals and classification rules, we describe in detail what these new initiators are and how they work.

This chapter also discusses the scheduling environment and the latest Tivoli Workload Scheduler for the z/OS (TWS) enhancement.

6.1.1 WLM-managed initiators

WLM manages initiators based on the goals specified in their service class period. In z/OS 1.8, IBM has further enhanced this functionality by extending the scope of batch management to be able to optimize the distribution of the batch workload across a sysplex. This functionality is called *improved batch initiator balancing*.

JES2 attempts to use approximately the same *percentage* of active WLM-managed initiators in each service class on each system. JES2 does not try to use the same *number* of initiators on each system; the number used is in proportion to the number started on each system. If WLM has started 20 initiators on system A and 10 on system B, and 15 jobs are submitted, 10 should run on system A and 5 on system B.

This balancing only comes into effect when there are idle initiators; most commonly when other jobs have finished but the initiator is still active. If there are no free initiators, jobs run wherever another job finishes, or WLM starts new initiators.

Initiator balancing does not necessarily keep the number of initiators the same, or keep all systems equally busy. It does stop one system from always selecting the work when there are free initiators elsewhere, which gives WLM a better picture of the workload when it is deciding whether to start or stop initiators.

You can display JES2 targets using the $D SRVCLASS command. Example 6-1 shows the output of the command. We can see how many initiators are started and in use for the service class and the JES2 targets for the number of jobs on each system. We can see that the targets are in proportion to the number of initiators that WLM has started on each system.

Example 6-1 $D SRVCLASS command

```
$DSRVCLASS(BATCHLOW),LONG
$HASP889 SRVCLASS(BATCHLOW)
$HASP889 SRVCLASS(BATCHLOW)  QAFF=(ANY)TYPE=DYNAMIC,
$HASP889                     COUNTS(SC63)=(INITS=24,
$HASP889                     ACTIVE=16,TARGET=17),
$HASP889                     COUNTS(SC64)=(INITS=16,
$HASP889                     ACTIVE=11,TARGET=10),
$HASP889                     MASCOUNT=(INITS=40,
$HASP889                     ACTIVE=27)
```

Note: The improved batch initiator balancing requires all systems at z/OS 1.8 or higher. If you have pre-z/OS 1.8 systems, JES2 preferably starts jobs on the *submitting* system.

How WLM batch initiator management works

On the job class basis, you can decide if the initiators should be managed by JES or by WLM. If JES-managed, there is a fixed number of initiators available, each one assigned to one or more job classes. Incoming jobs can be started as long as there are idle initiators available for that specific job class. In the WLM-managed environment, the number of initiators is controlled by WLM, and the initiators are assigned to service classes. WLM has several criteria for starting and stopping initiators:

- New initiators are started for a service class if there is a need to achieve the goals of that service class. To determine this, WLM takes the queue delay time of jobs waiting for execution while checking during the 10-second policy adjustment cycle, and it considers which resources are required for the workload belonging to a service class to be able to achieve the goal. If WLM determines that the reduction of queue delay time has the best effect, WLM starts new initiators for that service class. And, WLM starts those initiators first on systems with the lower load. For a system that runs on 100% CPU usage, new initiators are only started if there are jobs on the queue in a service class with an importance that is high enough to displace work with a lower importance that is already running on the system.
- Another criterion is independent of the goals. If there are jobs waiting in the queue to be started and a system has enough unused capacity in terms of available CPU and memory to take at least one of those jobs, an additional initiator is started on that system.
- There are also additional criteria for starting initiators. For example, if jobs are waiting that have system affinity to one or more systems, but no initiators are available for them, one initiator is started for those jobs.
- WLM also stops initiators if one of these situations occurs:
  - If there are, over time, significantly more initiators available for a service class than needed (not used).
  - If a system is so heavily loaded that there is a shortage of available CPU or memory capacity. WLM keeps at least one initiator per service class on each system for at least one hour.

If there is a constrained system that has initiators for the service class of the waiting batch work, and there is at least one other system that has enough available CPU and memory to take over a waiting job of the average size of that service class, initiators are moved from the constrained system to the unconstrained one.

At every 10-second interval, WLM performs the following check with the assumption that there are jobs waiting in the queue:

- WLM performs a check to verify whether the system is constrained. WLM considers a system constrained when the CPU consumption is more than 95%, based on a one-minute average. If a system is constrained, the service class with the lowest importance that has waiting jobs and more than one initiator assigned is identified as a candidate for stopping an initiator.
- If this happens, WLM checks the other systems. If WLM finds at least one system that has enough memory and low enough CPU usage that the system remains unconstrained after taking over the job of the identified service class, one initiator on the constrained system is stopped.
- If more than one system is over 95% CPU-utilized, only the most constrained system is a candidate for reducing one initiator per interval.
- On unconstrained systems, up to five new initiators can be started per 10-second interval if there are enough CPU and memory available.
- WLM takes special care if there are jobs waiting that have system affinity for constrained systems and, therefore, WLM cannot direct them to unconstrained systems. In this case, the reduction of initiators is limited in order to maintain enough initiators for those types of jobs.

The balancing mechanism shows that there is no one-to-one correlation between initiators stopped on one system and initiators started on another one. It is an asynchronous process during which initiators are reduced on constrained systems and more initiators are started on unconstrained systems. The action of stopping an initiator is not instantaneous: Stopping an initiator is pending until the job that is running finishes.

Batch initiator management example

We performed a simple test to demonstrate the management of initiators.

We ran a steady stream of jobs in the system. The jobs were short-running jobs so there were always jobs ending and new jobs starting. The jobs were all submitted from the same system, SC63. After a short time, WLM started enough initiators for all the jobs to run. The jobs were evenly spread between SC63 and SC64.

After the jobs were running, we submitted enough jobs with heavy CPU utilization to drive the CPU usage on SC63 to 100%. Over the next few minutes, we saw the number of jobs running on SC63 decrease until there were none of the short-running sample jobs running there, and they were all running on SC64.

We then cancelled the large jobs running on SC63. A few jobs started running there immediately using some of the initiators that were freed by the large jobs, but most jobs kept running on SC64, which still had most of the initiators. The system kept running like this for some time, because SC64 was not busy enough for WLM to actively move initiators. We found that the number of jobs on each system did even out over time, but this is probably dependent on the characteristics of the work.

JES2 service class management

z/OS 1.8 added the capability to control JES2 job selection by service class. For example, you might want to stop work in a service class for long-running work from running on a particular system if you are planning to shut that system down.

You can stop work from running in a service class using the $P SRVCLASS or $T SRVCLASS commands. Use the commands $S SRVCLASS or $T SRVCLASS to restart work.

The service class names are the same as the WLM service classes.

Types of service classes

JES2 tracks two types of service classes: *dynamic service classes* and *permanent service classes.*

Dynamic service classes are created when a job enters the system with a service class that is not currently known. Dynamic service classes are deleted after a period of time when there are no pre-execution jobs for that service class.

Permanent service classes are created using the $ADD SRVCLASS or $T SRVCLASS commands. Permanent service classes are no different than dynamic service classes, except that they are not deleted after a period of inactivity. This avoids the problem where you want to stop future work from running in a service class, but JES2 has deleted the service class because it has had no recent work.

A permanent service class can be converted to a dynamic service class using the $T SRVCLASS command when the class is no longer required. JES2 will delete it after a period of inactivity.

Service class example

Examples of these types of service classes are:

1. Display a service class that is currently unknown to JES2:

   ```
   $DSRVCLASS(BATCHLOW)
   $HASP003 RC=(52),D
   $HASP003 RC=(52),D SRVCLASS(BATCHLOW)  - NO SELECTABLE
   $HASP003           ENTRIES FOUND MATCHING SPECIFICATION
   ```

2. Add the service class:

   ```
   $ADD SRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)  QAFF=(ANY)TYPE=PERMANENT,
   $HASP889                     MASCOUNT=(INITS=0,ACTIVE=0)
   ```

   The service class name must be an existing WLM service class.

3. Stop work in the service class from running on this system:

   ```
   $PSRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)  QAFF=(SC64,SC65,SC70)
   $HASP889                     TYPE=PERMANENT
   ```

   This command is equivalent to using the $T SRVCLASS command to remove the current system from the QAFF list.

4. Allow work to run on this system again:

   ```
   $SSRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)  QAFF=(ANY)TYPE=PERMANENT
   ```

5. Set the service class to dynamic so JES2 can delete it if it is not needed:

   ```
   $T SRVCLASS(BATCHLOW),TYPE=DYNAMIC
   $HASP889 SRVCLASS(BATCHLOW)
   $HASP889 SRVCLASS(BATCHLOW)  QAFF=(ANY)TYPE=DYNAMIC,
   $HASP889                     MASCOUNT=(INITS=0,ACTIVE=0)
   ```

Jobs with system affinity

APAR OA10814 solves the problem occurring with WLM-managed initiators and jobs with system affinity. Before this APAR, WLM, before starting new initiators, performed a check to verify if enough initiators were already available in the sysplex for the service class. The check was performed on the overall configuration, not just on each image to verify that in case of jobs with affinity, this particular image had initiators to satisfy the batch work. Jobs with system affinity could wait a long time on the queue before getting an initiator, because the sum of idle initiators over the sysplex was big enough, but these initiators were idle on other systems. With this APAR, WLM now checks that idle initiators can be used by jobs with system affinity and, if not, it starts new initiators on the right image.

Considerations for multiple JES2 MAS complexes in a sysplex

There is no requirement for all the systems in the sysplex to be in the same MAS complex. However, the WLM policy is sysplex-wide and independent of the JES2 member or MAS complex.

- Jobs with the same classification attributes (job class and so on) are assigned the same service class, no matter which MAS they are in unless the subsystem collection name is included in the classification attributes.
- The service class determines the goal, regardless of which MAS they are in.

- Limiting the number of jobs in execution by class through the JOBCLASS(c) XEQCOUNT parameter is only MAS-wide, not sysplex-wide.

Tip: If you are using WLM-managed initiators, do not classify the batch work from more than one MAS into the same WLM service class. This can lead WLM to start new initiators for a service class in the wrong MAS.

Operational changes

While WLM-managed initiators reduce the need to actively manage job initiation, there are still constraints within which WLM must operate. Because operators lose the ability to control the maximum number of WLM-managed initiators or to force jobs into immediate execution by starting more initiators, WLM provides mechanisms to do so that apply to WLM-managed initiators. Likewise, the need for orderly shutdown remains even when you use WLM-managed initiators.

Limiting the number of jobs

Most often the need to restrict the number of concurrently executing jobs derives from either a physical limitation such as the number of tape drives or from a service level agreement (SLA) that specifically limits the number of jobs that an organization can run concurrently. Typically, this is accomplished by limiting the number of initiators started with the restricted job class in their selection list. This model is now implemented for WLM-managed and JES-managed initiators by specification of JOBCLASS(*x*) XEQCOUNT=(MAXIMUM=5) in the JES2 initialization deck or via the $TJOBCLASS(*x*) command. Starting with z/OS 1.8, you specify this count at the member level JOBCLASS(Z),XEQMEMBER(SYSA)=MAX=3.

Forcing immediate initiation

Because you cannot use the $SI command to start an initiator for a WLM-managed job class, the $SJ command enables the operator to force immediate initiation of a specific WLM-managed job regardless of goals or the job's position on the job queue. WLM starts an initiator on the system that is most likely to have enough CPU capacity to support additional work. This initiator selects the specific job targeted by the $SJ command and initiates that job. When the job completes, the initiator terminates. Because the initiator processes only the targeted job, it does not give the operator the capability to manually add long-term initiator capacity to a job scheduling environment.

Stopping initiation

In order to prepare for hardware or software maintenance, you might need to quiesce a system. Because WLM-managed initiators cannot be stopped via $PI, a new command, $P XEQ, provides this purpose. $P XEQ announces your intent to perform an orderly shutdown of JES2. JES-managed initiators cease selecting new work; WLM-managed initiators drain and then terminate. $S XEQ allows selection of new work, and if running in goal mode with WLM-managed initiators, it allows WLM to create initiators again. Both commands are single-system in scope; when WLM-managed initiators are used, WLM will likely start initiators elsewhere to compensate for the loss of capability on the system that is quiesced.

Stopping a service class

You can use the $P SRVCLASS or $T SRVCLASS to prevent work in a particular service class from starting on a particular system. This can be useful to avoid starting long-running jobs when planning a system shutdown. See “JES2 service class management” on page 194 for an example.

6.1.2 Scheduling environment

Similar to, but independent from member affinity, a job might also be assigned to a scheduling environment to ensure that it executes on the members in the MAS that have the required resources or specific environmental states. The states can be associated with hardware resources such as a vector facility, to an address space such as DB2, or to an abstract state such as time of day (shift) or day of the week. The resources and their required states are defined when adding a scheduling environment to the WLM policy. The required state of the resource can be either `on` or `off`. The scheduling environment is enabled on a System when the resources belonging to it are in the required state defined in the WLM policy. You can change the state of a resource with the modify WLM command.

You can then specify the environment with the SCHENV= keyword parameter on the JOB statement, or through an installation exit. Environments can also be assigned via a JOBCLASS default or using the $T Job JES2 command. Scheduling environments are installation-defined sixteen-character names that can be available on any of the systems in the sysplex or none of the systems.

The scheduling environment is validity-checked by the converter and must be defined to workload management, or else the jobs fail with a JCL error. JES2 detects the availability of scheduling environments on each member and allows an initiator to select jobs only if the specified scheduling environment is available. This is true for both JES2-managed initiators and WLM-managed initiators. In addition, any member affinity specified through the SYSAFF= parameter as well as the job class and initiator class must match.

6.1.3 TWS integration with WLM

With PK08998, PK08999, and PK09001 APARs, TWS enhances its integration with WLM with the final objective of reducing the cost of JCL management and job monitoring associated with WLM scheduling environments in your installation. In fact, with this maintenance level:

- There is no longer a need to manually tailor each JCL any time that you need to associate a scheduling environment with a job or change an existing association:
  - Granularity of association is possible due to definition at the operation level (via ISPF panel or Programming Interface Facility ((PIF)).
  - Global changes for groups of jobs are possible due to Mass Update and Batch Loader support.
- Fast detection of scheduling environment problems related to TWS submitted jobs is available:
  - The new extended status, *Waiting for scheduling environment,* allows you to track problems with scheduling environments via ISPF dialogs or PIF-based Current Plan queries.
  - Filtering on job status by using the scheduling environment definition allows you to easily identify all operations waiting for a specific scheduling environment availability.
- The automatic resubmitting of jobs waiting on scheduling environment availability is implemented.

  The new event tracking on scheduling environment status generated by Trackers allows the Controller scheduler to automatically retry the submission of operations ready but waiting for scheduling environment availability.

- Multiple sysplex support and considerations:

  In a multiple sysplex environment within WLM, you can define the same scheduling environment name for more than one sysplex (that is, scheduling environment SE1 can be defined both in SysplexA and in SysplexB, however, they are separate objects, each with a different status).

  You are not required to provide all scheduling environment definitions and sysplex associations with the TWS product, because these are detected automatically after you have coded the OPCOPTS SYSPLEXID parameter in the Controller and Trackers.

  – The monitoring status check of a job scheduling environment is done at Tracker level and using the WLM macros with a scope of sysplex-wide, allowing the support of the sysplex to be handled by one Controller. (A similar check done at Controller level would have been limited to the Controller sysplex scope only.)

  – Monitoring the status of the scheduling environment and the events that are sent to the Controller also carries the indication from which sysplex they are coming according to the Tracker that generates them.

    For multiple sysplex support, the OPCOPTS SYSPLEXID parameter has been added by this maintenance level:

    SYSPLEXID (*nn*)

    where *nn* is an integer number from 1 to 65000 and the default is 1.

    This parameter applies both to Trackers and Controllers.

    The following scenario, related to the configuration illustrated in Figure 6-1 on page 199, provides an example of this function.

    Suppose the same scheduling environment, `DB2UP`*,* is defined on both Sysplex1 and Sysplex2, and initially is not available on both sysplexes:

    - A job is submitted with SCHENV=DB2UP on Tracker T3 on system D, Sysplex1.
    - Tracker T3 checks whether the scheduling environment `DB2UP` is available, and sends the event *submit failed* to the Controller, providing the sysplex name (Sysplex1) and the scheduling environment name.
    - The Controller puts the operation in ready status, waiting for the scheduling environment to become available, and stores the scheduling environment name, `DB2UP`, and the sysplex ID of 1.
    - `DB2UP` becomes available on Sysplex2, and Tracker T5 of Sysplex2 sends an event to the Controller to inform it.
    - The Controller checks the operation ready waiting for scheduling environment against the scheduling environment name `DB2UP` and Sysplex ID 2, and finds no match.
    - `DB2UP` became available on the image B in Sysplex1 and Tracker T4 of Sysplex1 sends an event to the Controller to inform it.
    - The Controller checks the operation ready waiting for the scheduling environment against the scheduling environment name `DB2UP` and sysplex ID 1 and finds a match. The operation is resubmitted.

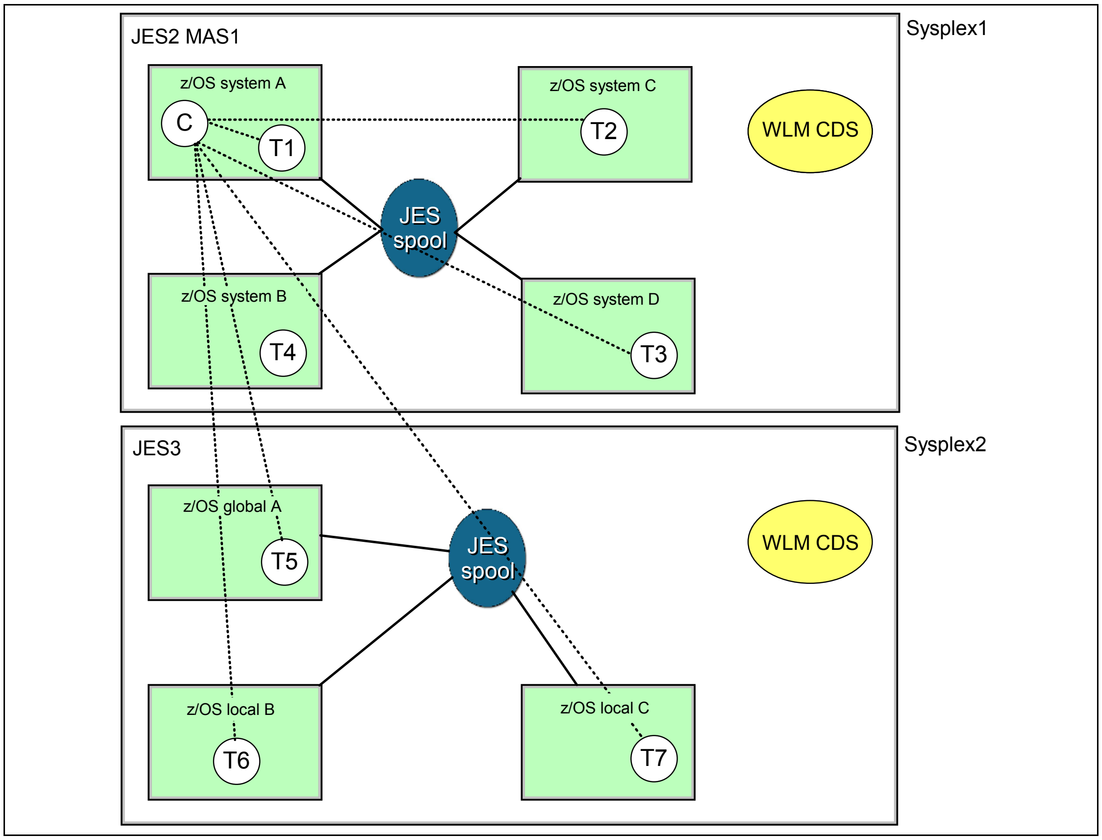


Figure 6-1 Multiple sysplex configuration with JESplex matching the sysplex boundaries

TWS is also maximizing the usage of system resources and improving the definition of workload management for better load balancing and service level handling. In fact:

- All jobs that need to meet a predefined service goal or that require a specific set of resources to be available can have an associated scheduling environment predefined as part of the workload planning process.

  When using WLM-controlled initiators, WLM determines in which logical partition (LPAR) or system the job executes. This occurs depending upon the availability of the WLM scheduling environment. This environment is specified in the JOB card with the SCHENV parameter and is inserted by the TWS product automatically, based on the operation definition.

- Automatic load balancing and routing based on scheduling environment specifications:

  - WLM resources can be used in a JESplex, and the scope is sysplex-wide. Whenever a scheduling environment is defined on more than one system belonging to the sysplex, WLM checks workload status to choose the system where the job has to execute, depending on other specifications and policies defined in WLM. With TWS scheduling environment specifications at the operation level, you can then plan workload balancing sysplex-wide.

  - Another advantage of working with WLM scheduling environments is that you have the capability to switch workload to another system in case of problems or in case of scheduled system stops, either planned or unplanned, without having to edit the JCL, but by just leveraging the specifications provided at the TWS level.

Implementation tasks

The following new elements are supported to achieve the functionality that we describe.

Scheduling environment definition in the Application Description and in the Current Plan with automatic JCL tailoring

Now, you have the capability to:

- Using the ISPF dialog or PIF, you now can take advantage of new functions:
  - Add, delete, and modify the scheduling environment name for a specific operation in the Application Description.
  - Add, delete, and modify the scheduling environment name for a specific operation in the Current Plan.

    These changes are allowed only if the operation status is different than `Started` in order to maintain the information about the used scheduling environment name until the job ends.
  - The current user exit, EQQUX001, is modified to be able to INSERT/REPLACE the scheduling environment of a related job at submit time. The returned name value is stored in the Current Plan.
  - A new user exit, EQQDPX01, is provided as part of the batch process used to generate the Current Plan. The exit is invoked for each Operation of type Computer Automation, not E2E, and the returned value of scheduling environment is stored in the Current Plan.
  - The SCHENV=SE-*name* parameter is automatically inserted in the JCL JOB card according to the scheduling environment name defined in the Current Plan.

Preexistent definition in the JCL JOB card

To avoid extra processing for the Tracker submitting the job or to selectively allow the activation of the new functionality, a new parameter in the OPCOPTS initialization statement is supported:

SECHECK (ALL/OPERONLY/NO)

This parameter applies to the Tracker or to the Controller having the Submit function active (use the destination blank to submit local jobs).

To activate this function, you must specify a different value than `NO`.

If you activate this function, you must use it for all the Trackers and Controllers within the same sysplex. This is necessary to correctly implement the status monitoring mechanism, because each Tracker activates the listening exit only on the related z/OS image:

`ALL`

The JCL is processed to check the scheduling environment availability in both of the following cases:

- The user specified a scheduling environment name at the operation level or either the EQQUX001 or EQQDPX01 user exit provided a value for the operation. If so, the SCHENV=SE-*name* keyword is inserted or updated in the JCL JOB card. Any preexisting value in JCL is overwritten.
- The user did not specify a scheduling environment name at the operation level and no user exit provided a scheduling environment value for it, but a SCHENV=SE-*name* keyword exists in the JCL JOB card:

OPERONLY

The JCL is processed to check the scheduling environment availability only if a scheduling environment name is specified at the operation level within the Current Plan, either via a user definition or due to the involvement of the EQQUX001 or EQQDPX01 user exit. The scheduling environment name, provided per the described methods, always overrides any preexisting statement specification in the JCL JOB card.

NO

This is the DEFAULT. No type of check is done for any job.

The Tracker executes the scheduling environment availability check according to the SECHECK parameter specification.

- If the scheduling environment check is required and if the scheduling environment is available:
  - JCLs are tailored (together with the SPIN tailoring, if this is also requested), inserting, or updating the SCHENV=SE-*name* keyword in the JOB card.
  - The Tracker informs the Controller about the successful submission (or about problems that are not due to scheduling environment) with the already existing IJ1 event.
- If the scheduling environment check is required and if the scheduling environment is not available on any system in the sysplex (or at JESplex level):
  - Jobs are not submitted.
  - The Tracker informs the Controller about this missing submission (the new submit event, IJ4, will contain information about this).
  - The Tracker manages it, implementing a listening mechanism on scheduling environment availability. As soon as the scheduling environment becomes available, the Tracker informs the Controller, generating the new event VS, and the Controller retries the Job submission.
- If the scheduling environment does not exist at all (it is not defined to WLM):
  - Jobs are not submitted.
  - The Tracker informs the Controller about this missing submission. The new submit event, IJ4, will contain this information.

Check on SE availability

The Tracker submitting the job is going to check via WLM Query Service if the scheduling environment is available:

- If the scheduling environment is available on any system in the sysplex, the job is submitted, delegating WLM to identify the system where the job will be executed.

  In order to have full workload balancing support, configure one JESplex (JES2 MAS or JES3 Complex) in the sysplex, but your installation might have reasons to deviate from this configuration.

  In order to support configurations when there are more than one JESplex within a sysplex, a new parameter has been added, named OPCOPTS JESPLEX (List of System Names) to represent the associated list of system names used as an additional filter in the query to obtain the availability of the scheduling environment at the JESplex level.

  In addition, you can also have a mixture of multiple JES complexes (JES2 or JES3) within a sysplex. In this situation, a job submitted in a JES complex cannot be routed by WLM to another JES complex.

  You can see an example of this situation described in Figure 6-2 on page 202.

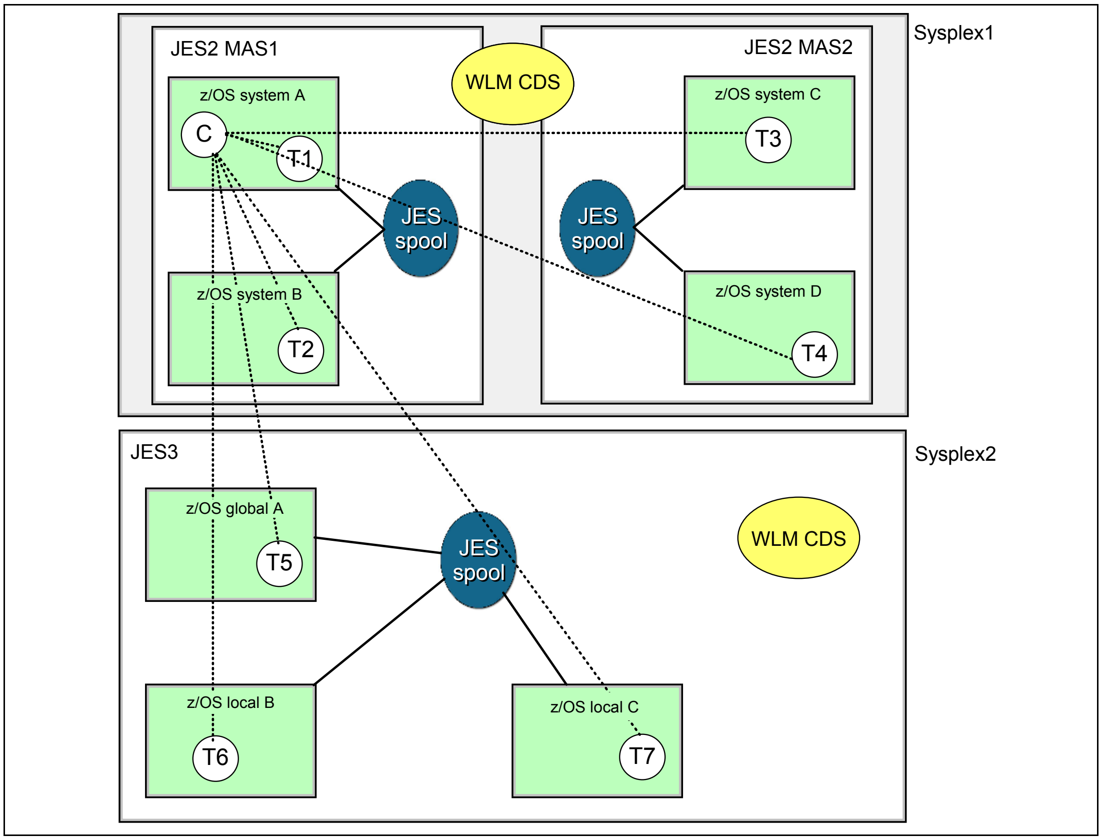


Figure 6-2 Multiple sysplex configuration with multiple JESplexes within the sysplex boundaries

A scheduling environment, DB2UP, is available on system D only. When system A is queried, the returned output indicates that DB2UP is available in the sysplex, ignoring the JES2 MAS 1 boundary.

If a job, requiring the scheduling environment DB2UP, is submitted on system A, it remains on the JES input queue.

You can avoid this situation by making Trackers aware of the JESplex configuration via the new OPCOPTS JESPLEX (List of System Names) initialization parameter statement, where the List of System Names represents the JESplex to which the Tracker belongs. Table 6-1 illustrates the definition required to map the configuration described in Figure 6-2, where T1, T2, T3, and T4 are the Trackers installed on the z/OS images A, B, C, and D. They should have the following OPCOPTS JESPLEX settings.

Table 6-1 Tracker definitions

Tracker name	OPCOPTS settings
T1	OPCOPTS JESPLEX (A,B)
T2	OPCOPTS JESPLEX (A,B)
T3	OPCOPTS JESPLEX (C,D)
T4	OPCOPTS JESPLEX (C,D)

When the job is submitted on system A, Tracker T1 checks the results of WLM query for DB2UP availability within its JESPLEX list (including A and B). In this case, Tracker T1 understands that `DB2UP` is not available on its JESplex, and Tracker T1 does not submit the job.

This is a static definition. To change the configuration, you need to stop the Tracker, update the parameter, and restart the Tracker again.

It is always possible to support more JESplexes in the same sysplex, and, therefore, to avoid the use of the OPCOPTS JESPLEX (System Name List) parameter. Supporting more JESplexes follows the best practices approach in the scheduling environment definition and usage. WLM documentation provides more recommendations, too.

- If the SE is not available on any system in the sysplex (or at Tracker JESplex level, if the OPCOPTS JESPLEX parameter is specified), the operation is set in *Ready* status with extended status *waiting for SE*:
  - This is not done if the operation comes from a Restart and Cleanup path. In this case, the operation is set in error status SERC (scheduling environment not available for R&C operation).
  - Operations ready but waiting for scheduling environment are considered not eligible for submission by the Controller.
- If the scheduling environment is not defined in the sysplex, the operation status is set in *Error* status with a new error code SEUN (SE undefined).

Retry of operations waiting for scheduling environment

Trackers activate an ENFREQ macro listening mechanism for event code 57, which is the event that signals the status change of an scheduling environment.

Trackers generate the new event VS to signal to the Controller that a scheduling environment is again available on a specific image. At this point, the Controller resets the extended status so that the operation is again eligible for submission.

Monitoring operations in ready status waiting for scheduling environment

Any list displaying the operation status and extended status supports the new *Waiting for SE* extended status. Any command currently supported for ready operation is available.

6.2 Types of goals

WLM goal types deserve special considerations for batch jobs. With a good understanding of your batch environment, you can begin to evaluate your goals and determine which service classes might be better served by WLM initiators. Keep in mind that when using WLM-managed initiators, JES queue time is now a part of the velocity calculation as it is with the response time calculation.

6.2.1 Response time goal

A response time goal is best for short, homogeneous batch jobs. The response time goal can be average or with percentile, although percentile goals are often preferable because they are not affected by a few very long response times.

If you choose JES2 initiators, then remember that the queued time due to a lack of initiators is part of the response time. Although WLM cannot do anything to help a job waiting for an initiator, the performance index of the service class will grow. If you have large job execution queues, the queue delay can dominate the response time or velocity. This can cause the system to not address other delays because it would not significantly affect the performance index (PI). Response time does not include the TYPRUN=HOLD delay or JCLHOLD delay, but it does include:

- Operational delays (jobs or job class held by operator command)
- System or resource affinity delay
- Scheduling delay due to class limits or duplicate jobnames
- Time waiting for an initiator

Good candidates for this type of goal

In order to assign work assigned to a response time goal, you need to have enough completions within a period of time; for example, at least ten completions in twenty minutes are necessary for a response time to be effective.

The following types of batch can be good candidates for this type of goal:

- Compile jobs
- Short printing jobs
- Jobs with steady arrival rates (not bulk submission by schedulers)
- Short-running jobs
- Jobs with uniform runtimes
- Jobs where the resource delay impact on response times is minimal
- Jobs that repetitively treat a constant amount of data

6.2.2 Velocity goal

If you are using WLM-managed initiator, queued delay is now part of percentage velocity calculation. With JES2-managed initiators, queue time is not part of the velocity evaluation. This goal is more appropriate for most of the batch jobs, with the following considerations:

- Good value for velocity is difficult to determine.
- When determined, you need to periodically revisit this value (in the case of a change of the profile of the batch or a hardware modification). Speed and the number of logical processors can affect CPU components of the velocity; technology of the disk subsystem can affect the I/O component of the velocity. In a sysplex, the achieved velocity can be differ if a job executes in one system or in another system.

Good candidates for this type of goal

Good candidates are:

- Long-running jobs where queue time is insignificant
- Highly variable execution times
- Jobs with few completions (not enough historical data to establish a response time)

6.3 Classification rules

In your installation, be careful not to mix JES-managed and WLM-managed job classes in the same service class. WLM allows you to do so rather than failing the jobs, but in this environment, the management of the initiators can be unpredictable. When both initiation schemes are used for the same service class, the correlation between queue delay and the number of initiators that WLM recognizes as servicing the service class is weakened. Wrong information can theoretically lead to bad decisions, therefore, our recommendation against doing so.

No checks are available between JOBCLASS definition (mode=JES or WLM) and the rules in the service definition; assigning a service class to a job through its jobclass is the only way.

There is no way to assign a specific service class to a job through the MASDEF definition for the specific system where the job is going to be executed, because the service class is assigned during the conversion phase and the executing system is not yet determined.

The starting of new WLM initiators is predicated on the batch service class missing its goal due to JES queue delay, and WLM making the determination that starting new initiators reduces this delay and thereby contributes to the batch service class meeting the installation's defined goals. To be able to determine which jobs are the best candidates for the WLM batch initiator, you need to have a clear understanding of your batch environment:

- Look at the SMF 30 records to find the job distribution by class, by shift, and so forth.
- Look at the SMF 72 records to find the multiple periods, DPRTY, queue delays, velocity, response times, and the performance index if reported.
- Look at your current job class rules and how they are assigned and enforced.

Then you should analyze your initiator structure and understand the following characteristics:

- Check if there are any asymmetric configurations.
- Verify which job classes are restricted to specific systems.
- Understand if there are any special limits by job class or by system.
- Understand if there are any limits on work packs, tape drives, storage, and so forth.
- Check if the installation makes changes via operator procedures or automation.

Classification

The following qualifiers are supported by the JES subsystem type:

- *Accounting information* passed on the JOB statement (not the EXEC statement). This is ignored if ACCTFLD=OPTIONAL is specified in the JOBDEF statement of the JES2 init deck.
- *Perform* identifies the performance group number specified using the PERFORM keyword on the JCL statement.
- *Priority* is a value between 0 and 15 identifying the priority associated with the batch job submitted through JES2.
- *Scheduling Environment Name* associated with the job.
- *Subsystem Collection Name* to identify the XCF group name (XCFGRPNM of the MASDEF statement of the JES2 init deck).
- *Subsystem Instance* is the subsystem name from IEFSSN*xx* (JES2 or JES3).
- *Sysplex Name* is a qualifier that you can use if you have the same service definition for multiple sysplexes or monoplexes, then you need to assign different service classes based on the specific sysplex or monoplex in which the work is running.
- *Job class* from CLASS keyword of the JOB statement.
- *Jobname* from the JOB statement.
- *Userid* specified on the USER keyword of the job statement.

Restriction: System name cannot be used as a qualifier because the classification occurs during the conversion phase. At this moment, the system in which the job will run is still unknown.

Tip: You can use Subsystem Collection Name (SSC) to assign a different service class to a job in a sysplex with several MASPlexes, even if these MASPlexes have only one member.

As a basic rule, classify the following jobs into two categories:

- Jobs for WLM-managed initiators
- Jobs for JES2-managed initiators

Two sets of service classes are necessary to handle these jobs. Due to the different velocity calculation, the same service class cannot be used for a job running in an JES2 initiator and another job running in a WLM-managed initiator.

In each category, you can define two or three service classes. There is no significant difference between two service classes with a close velocity goal. Slow, medium, and fast are good starting points.

We highly recommend that you use groups (transaction name) if you have a lot of jobs to classify.

6.4 Best practices

In this section, we suggest best practices for you to use in planning your batch environment.

6.4.1 WLM-managed initiators

First of all, WLM-managed initiators do not replace JES2-managed initiators. Depending on your needs and your environment, you should still use each type of initiator.

For example, WLM-managed initiators are not the best solution for CICS or IMS regions. When submitted, this type of job should start immediately, even if the system is heavily constrained. You can measure the queuing time to decide if this task is not reasonable for your online environment. For example, if you have to restart a CICS region during the online window, you expect it to start as quickly as possible. Most likely, these jobs often have affinity to a particular system.

Non-candidates (keep these with JES-managed initiators) are:

- Jobs that need to run immediately or emergency jobs
- Just-in-time jobs managed by a job scheduler
- Critical path job streams
- Jobs with long resource delays
- Jobs with system affinity (until OA10814 is applied)
- Operator-held jobs
- Tape-constrained environments or jobs subject to waiting on data set enqueues
- Jobs used to support IMS and CICS regions

This list is not exhaustive and, depending your environment and your needs, our recommendations might not apply.

Fragmentation

WLM initiators might suffer from the same storage fragmentation problems as JES initiators. WLM does not detect *dirty initiators* and replace them with new initiators. You need to perform this task manually using the P INIT,A=*asid* command; WLM then replaces this initiator with a new initiator.

If you cannot determine the ASID, or if you want to recycle all initiators as part of a regular cleanup process, you can use the $P XEQ and $S XEQ commands. The $P XEQ command flags all WLM initiators on that system to terminate. The $S XEQ command enables WLM to start new initiators (without needing to wait for the old initiators to end). Beware that the $P XEQ command purges WLM’s history that you can use to understand how many initiators are needed for each service class. It might take some time for WLM to create the same number of initiators that existed before.

You can also use the SDSF Y action character, which generates the MVS P (STOP) command.

Note: Starting from z/OS 1.5, the VSM™ CHECKREGIONLOSS(*bbb,aaa*) parameter in the DIAG*xx* member of the PARMLIB can be used to automatically terminate and restart initiators when the maximum available region size decreases the amount specified by this parameter.

Balancing

Before z/OS 1.8, initiators were mostly started on the submitting member of the sysplex (where the scheduler product runs).

The enhancement provided by the Batch Initiator Balancing function is to improve the performance and throughput of batch workload over the sysplex. Its intention is not to reach an equally balanced distribution of batch jobs over the LPARs of a sysplex. That is why initiator balancing only takes place when at least one of the systems of the sysplex has a CPU utilization of more than 95%, while other systems have idle capacity. There is not any improvement to the total batch throughput by removing initiators from the system with the highest load if this system has enough idle capacity to run batch jobs without CPU constraints. This is true even if the other systems have more idle CPU capacity.

JES2 parameter

In the JESPARM configuration, do not to define a JOBCLASS with the MODE=WLM if this class is already defined to a JES2 initiator. See Example 6-2.

Example 6-2 JES2 parameter

```
JOBCLASS(A,B) MODE=WLM,
              ......
JOBCLASS(*)   MODE=JES,
              ......
INITDEF PARTNUM=50
I1 CLASS=ABCDEFG12345,START=NO
I2 CLASS=AB,START
I3 CLASS=ABC,START
I4 CLASS=ABC,START
..........
```

Example 6-2 shows an extract of the JES2 parameters. This setup is prone for the potential problem where Init I2 will not be able to handle any jobs, because classes A and B are WLM-managed.

Using the JES2 display command, you obtain the result shown in Example 6-3 on page 208.

Example 6-3 JES2 Display Initiator command

```
$DI
$HASP892 INIT(1) 152
$HASP892 INIT(1)   STATUS=DRAINED,CLASS=CDEFG12345,
$HASP892           INELIGIBLE_CLASS=(A-WLM,B-WLM),NAME=1
$HASP892 INIT(2) 153
$HASP892 INIT(2)   STATUS=INACTIVE,CLASS=,
$HASP892           INELIGIBLE_CLASS=(A-WLM,B-WLM),NAME=2,
$HASP892           ASID=002B
$HASP892 INIT(3) 154
$HASP892 INIT(3)   STATUS=INACTIVE,CLASS=C,
$HASP892           INELIGIBLE_CLASS=(A-WLM,B-WLM),NAME=3,
$HASP892           ASID=002C
```

Example 6-3 shows that INIT(2) is not serving either class A or B, because they are WLM-managed.

6.4.2 Quiescing a batch job

If you need to temporarily quiesce a batch job, for example, because this job is looping or consuming too many resources, you can use the RESET command:

```
E jobname,QUIESCE
```

In some cases, this command might not be successful because the address space is marked as non-swappable, and even with the lowest dispatching priority (FB), it continues to run.

It is possible to circumvent this problem by creating a special service class with a discretionary goal and a Resource Group with a maximum of 1. You can assign this service class to the job, using the RESET command:

```
E jobname,SRVCLASS=SCQUIESC
```

In this way, you do not completely stop the job, but you can reduce its CPU consumption to a minimum.

Note: In a CPU-constrained environment, you might not be able to cancel a quiesced job. Resuming this job first by using the E *jobname*,RESUME command might be necessary to make the cancel effective.

6.4.3 Choosing the right velocity

Each workload has its own velocity, depending on the type of work and the hardware configuration. You can verify in the RMF reports or Monitor III Display which velocity can be achieved while the system is not constrained. There is a bigger difference between two goals with different importances and the same velocity than there is with two goals with the same importance and two different velocities.

It is useless to define several service class periods with too similar a velocity goal and the same importance. Velocity values within the same importance should have a difference of at least 10% to 15%.

Test case

We ran a scenario to be able to show you how the choice of the goal can affect the WLM mechanism. See Example 6-4 on page 209.

For our test case, we use a job with an observed velocity capability of 25%. In other words, in an unconstrained environment, this job cannot get more than 25% of velocity.

We defined three service classes for this job with the same importance, and we start three clones of this job, one for each service class. The three jobs are similar, because they execute the same program with the same data:

- ITSOV01 imp 3 velocity 10
- ITSOV02 imp 3 velocity 25
- ITSOV03 imp 3 velocity 40

See Example 6-4.

Example 6-4 Sysplex Summary RMF Monitor III display

```
RMF V1R5   Sysplex Summary - WTSCPLX1        Line 1 of 31
Command ===>                                                 Scroll ===> CSR

WLM Samples: 399     Systems: 16 Date: 04/04/05 Time: 19.40.00 Range: 100   Sec

                      >>>>>>>>XXXXXXXXXXXXXXXXXX<<<<<<<<

Service Definition: spstpc              Installed at: 04/04/05, 17.54.17
     Active Policy: SPSTPC              Activated at: 04/04/05, 17.54.29

               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

BATCH    W           14                               0.000 0.000  0.000  0.000
ITSOV01  S  3    10  12                         0.87  0.000 0.000  0.000  0.000
ITSOV02  S  3    25  14                         1.81  0.000 0.000  0.000  0.000
ITSOV03  S  3    40  16                         2.58  0.000 0.000  0.000  0.000
CICS     W          N/A                               0.010 0.000  0.000  0.000
CICSDFLT S  2       N/A  0.015 99%       100%  0.50  0.010 0.000  0.000  0.000
DB2RES   W          0.5                               14.42 0.000  0.329  0.329
SCDB2STP S          0.5                               14.42 0.000  0.329  0.329
         1  2       0.5  30.00 AVG  0.329  AVG  0.01  14.42 0.000  0.329  0.329
OTHER    W           10                               0.010 0.000  3.414  3.414
```

Example 6-4 shows an RMF Monitor III Display report. Observe that the three service classes have almost the same execution velocity. But, because the goals were different, the performance index obtained by each service class is extremely different, too. In this case, WLM assigns a different dispatching priority to each job, as shown with the System Display and Search Facility (SDSF). See Example 6-5.

Example 6-5 SDSF display of the three jobs

```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF DA SC69  SC69     PAG    0 SIO  2442 CPU  99/ 97  LINE 1-8 (8)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP   JOBNAME  StepName ProcStep JobID    Owner    C Pos DP Real Paging    SIO
     ITSOV10  STEP1             JOB12116 CASSIER  A IN  F1  337   0.00 315.64
     ITSOV20  STEP1             JOB12124 CASSIER  A IN  F3  340   0.00 472.61
     ITSOV30  STEP1             JOB12135 CASSIER  A IN  F5  336   0.00 660.51
     LOAD10   STEP1             JOB12092 CASSIER  A IN  F7  342   0.00  88.93
     LOAD20   STEP1             JOB12093 CASSIER  A IN  F7  337   0.00  85.78
     LOAD30   STEP1             JOB12094 CASSIER  A IN  F7  336   0.00  50.91
     LOAD40   STEP1             JOB12095 CASSIER  A IN  F7  337   0.00  56.28
     LOAD50   STEP1             JOB12096 CASSIER  A IN  F7  337   0.00 675.31
```

In the Example 6-5, the DP column gives the dispatching priority of the three jobs ITSOV10, ITSOV20, and ITSOV30. They are, respectively, F1, F3, and F5. This follows the order of the performance indexes.

This is an important point to consider when you specify a goal for a service class.

6.4.4 Usage and number of periods

There is really no general rule for this element. It mostly depends on your installation, but you can consider the following items while planning your configuration.

In today's environment, the order of importance of the different periods must be equal or descending.

> **Note:** Depending on the functionality level used, you might get a warning or an error when you validate or try to activate a service definition that does not follow this rule.

Carefully consider the usage of a low-priority period as the last one based on:

- Consider if the job really become less important because it has a lot of work to do.
- Verify if the job acquires a lot of locks (performing DB2 or IMS updates): Decreasing its speed might cause global contention, especially if you use a discretionary goal. With this goal, a job could not get CPU resources for a while if the system is heavy constrained.

Generally, consider that the first period should be used to speed up a short job and that the second period should be used to decrease the performance goal for a long-running job.

Durations of periods should be carefully revisited. Often, durations are too short for batch service classes. Those durations should be reevaluated with the current workloads.

Check the WLM report to verify whether the number of jobs that ended during the first period is what you planned. See Example 6-6.

Example 6-6 Workload level report

```
  REPORT BY: POLICY=WLMPOL     WORKLOAD=BAT_WKL    SERVICE CLASS=BATMED     RESOURCE GROUP=*NONE
                                                   CRITICAL     =NONE
                                                   DESCRIPTION =medium priority batch

  TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE TIMES--   PAGE-IN RATES    ----STORAGE----
  AVG     24.90   ACTUAL            6.20.857   SSCHRT 217.7   IOC     13054    TCB          1421.1   SINGLE     0.0   AVG      223.63
  MPL     24.90   EXECUTION         6.19.483   RESP    47.0   CPU     29491K   SRB             2.3   BLOCK      0.0   TOTAL   5568.17
  ENDED      24   QUEUED               1.155   CONN     1.6   MSO     14472K   RCT             0.0   SHARED     0.0   CENTRAL 5568.17
  END/S    0.04   R/S AFFINITY             0   DISC     0.0   SRB     47626    IIT             1.3   HSP        0.0   EXPAND     0.00
  #SWAPS      0   INELIGIBLE             218   Q+PEND   4.0   TOT     44024K   HST             0.0   HSP MISS   0.0
  EXCTD       0   CONVERSION             262   IOSQ    41.3   /SEC    73374    IFA             N/A   EXP SNGL   0.0   SHARED     0.00
  AVG ENC  0.00   STD DEV           8.37.142                                   APPL% CP    237.5   EXP BLK    0.0
  REM ENC  0.00                                               ABSRPTN  2947    APPL% IFACP   0.0   EXP SHR    0.0
  MS ENC   0.00                                               TRX SERV 2947    APPL% IFA     N/A

  PER IMPORTANCE  PERF     --TRANSACTIONS--     -------------RESPONSE TIME--------------   -EX VEL%-   TOTAL   -EXE--
                  INDX     -NUMBER-     -%-     ------GOAL------  ---ACTUAL---     TOTAL   GOAL  ACT   USING%  DELAY%
  1   4           0.9           0       0                                                  25 27.2    21.4    57.4
  2   5           1.9          24     100                                                  20 10.5    10.0    85.5
TOTAL                          24     100
```

Example 6-6 shows the use of a WLM report type WGPER where you can check how many jobs ended during the first period. In this example, all 24 jobs or 100% ended in the second (and last) period. In this configuration, you need to consider increasing the duration of the first period.

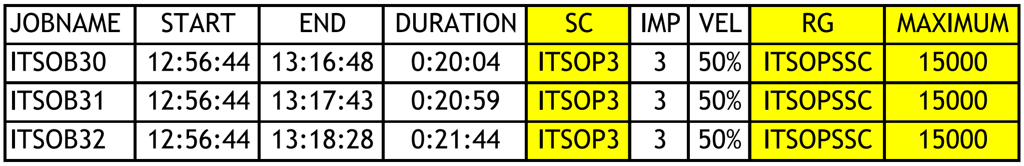
6.5 Resource Group usage

This section shows measurement related to Resource Group usage.

In the first test case, we submit three identical CPU bound jobs in a service class that has a Resource Group assigned. In the second test case, we submit the same jobs with three different service classes where the same Resource Group was assigned. For all cases, we measured the elapsed time of the jobs.

6.5.1 First test case

This test case shows the behavior of a job having a service class with a Resource Group assigned. We define a service class with a Resource Group with a maximum capacity of 15000 SU/s and submit three similar jobs in this service class. Our expectation is that the elapsed time for the three is extremely similar. See Figure 6-3.



JOBNAME	START	END	DURATION	SC	IMP	VEL	RG	MAXIMUM
ITSOB30	12:56:44	13:16:48	0:20:04	ITSOP3	3	50%	ITSOPSSC	15000
ITSOB31	12:56:44	13:17:43	0:20:59	ITSOP3	3	50%	ITSOPSSC	15000
ITSOB32	12:56:44	13:18:28	0:21:44	ITSOP3	3	50%	ITSOPSSC	15000

Figure 6-3 Jobs' elapsed time running in the same Resource Group (same service class)

Figure 6-3 shows the results. The small difference among the elapsed times is due to the time necessary for the Resource Group algorithm to properly cap the address spaces classified in the service class.

6.5.2 Second test case

Now, we want to assess how the importance of the goal of the service class is honored when a Resource Group is assigned to different service classes. The three service classes have the same goal, which is execution velocity of 50%, and different Importance levels from 3 to 5. We expect that the Importance 3 job will end first, then the Importance 4 job, and then the Importance 5 job. The three service classes have a Resource Group with a maximum capacity of 15000 SU/s. See Table 6-2.

Table 6-2 Jobs' elapsed time running in the same Resource Group (different service classes)

Jobname	Start	End	Duration	SC	IMP	VEL	RG	Maximum
ITSCB30	18:42:53	18:48:16	0:05:23	ITSOP3	3	50%	ITSOPSSC	15000
ITSCB40	18:42:53	18:50:23	0:07:30	ITSOP4	4	50%	ITSOPSSC	15000
ITSCB50	18:42:53	18:50:34	0:07:41	ITSOP5	5	50%	ITSOPSSC	15000

Table 6-2 shows the results. As expected, the Importance is honored; the elapsed time of the Importance 3 job is shorter than the elapsed time of the Importance 4 job, and so on. We performed the same test with two similar jobs running per service class. See Table 6-3 on page 211.

Table 6-3 Jobs' elapsed time (two jobs per service class)

Jobname	Start	End	Duration	SC	IMP	VEL	RG	Maximum
ITSCB31	18:51:11	19:02:19	0:11:08	ITSOP3	3	50%	ITSOPSSC	15000
ITSCB30	18:51:11	19:02:20	0:11:09	ITSOP3	3	50%	ITSOPSSC	15000
ITSCB40	18:51:11	19:05:41	0:14:30	ITSOP4	4	50%	ITSOPSSC	15000
ITSCB41	18:51:11	19:05:46	0:14:35	ITSOP4	4	50%	ITSOPSSC	15000
ITSCB51	18:51:11	19:08:06	0:16:55	ITSOP5	5	50%	ITSOPSSC	15000

Jobname	Start	End	Duration	SC	IMP	VEL	RG	Maximum
ITSCB50	18:51:11	19:08:16	0:17:05	ITSOP5	5	50%	ITSOPSSC	15000

Table 6-3 shows that two jobs running in the same service class have very close elapsed time and that the Importance is honored among service classes.

6.6 Impact of the number of engines on velocity

This section shows the impact of the number of engines on the execution velocity through two test cases. The execution velocity percentage is computed by dividing the number of using samples by the sum of using and delays samples. Every 250 milliseconds, WLM samples the state of each address space. Our samples are based on CPU bound jobs, so the using CPU states were the biggest contributors of execution velocity values.

In the first case, we started with an LPAR configured with ten engines on an IBM z990, and then we remove them until we have two engines left. The workload is made by 20 batch jobs running for all periods in 20 initiators.

In the second test case, we capped the LPAR and started with six engines. Then, we added one engine at a time until we reached a nine-engine configuration.

6.6.1 First case results

Figure 6-4 on page 213 shows the effect of decreasing the number of engines from ten to two on the execution velocity. The line in the graphic represents the execution velocity percentage scaled by the Y axis. The bars represent the performance index (PI) of the service class. The goal was an execution velocity of 50%. Even running with ten engines, the PI was greater than one because we had 20 batch jobs running on these ten engines. So, in this case, each job was competing for the engine.

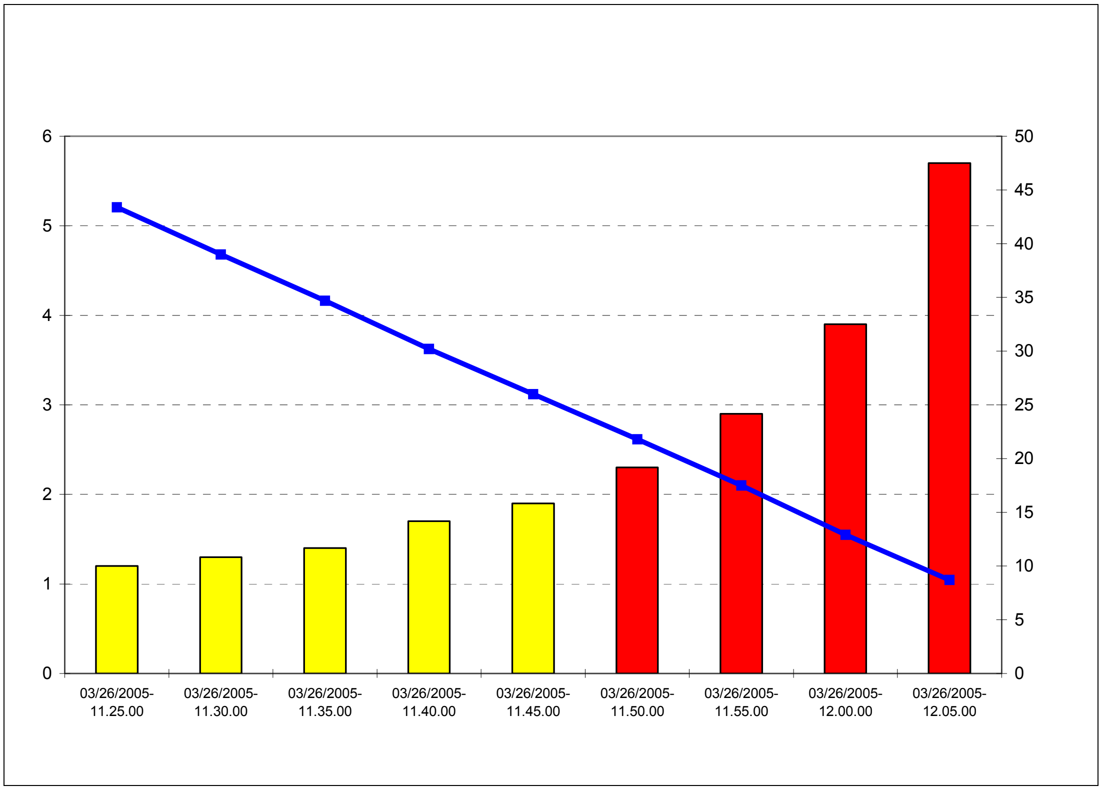


Figure 6-4 Decreasing the number of engines

Example 6-7 shows the WLM report with the ten engines' configurations at the beginning of the test case. The execution delays % samples are 56.5%.

Example 6-7 Extract of WLM RMF report with ten engines

```
GOAL: EXECUTION VELOCITY 50.0%     VELOCITY MIGRATION:    I/O MGMT  43.4%     INIT MGMT 43.4%

            RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
   SYSTEM                 VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU                               UNKN IDLE  USG  DLY  USG  DLY QUIE

   SC69        --N/A--    43.4  1.2  20.0  43.4  N/A  0.0 56.5 56.5                                0.1  0.0  0.0  0.0  0.0  0.0  0.0
```

Example 6-8 shows the report at the end of the test case where we are running with two engines. The execution delays % samples account now for 91.3%, while the PI jumps to 5.7 as the execution velocity percentage drops to 8.7%.

Example 6-8 Extract of WLM RMF report with only two engines

```
GOAL: EXECUTION VELOCITY 50.0%     VELOCITY MIGRATION:    I/O MGMT   8.7%     INIT MGMT  8.7%

            RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
   SYSTEM                 VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU                               UNKN IDLE  USG  DLY  USG  DLY QUIE

   SC69        --N/A--     8.7  5.7  20.0   8.7  N/A  0.0 91.3 91.3                                0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

6.6.2 Second case results

In the second case, we capped the LPAR configuration to have the same amount of CPU capacity during the test case. When adding an engine, take into account the multiprocessor (MP) factor effect. In the last interval of this experiment with nine engines, the global capacity of the machine decreases a little while the size of the logical uniprocessor is divided into 3.

Figure 6-5 shows the effect on execution velocity when increasing the number of engines from six to nine, keeping the same LPAR capacity. The line represents the execution velocity percentage scaled by the Y axis. The bars represent the performance index of the service class. The goal was an execution velocity of 50%. The goal is never reached, but the performance index is improving when adding logical engines.

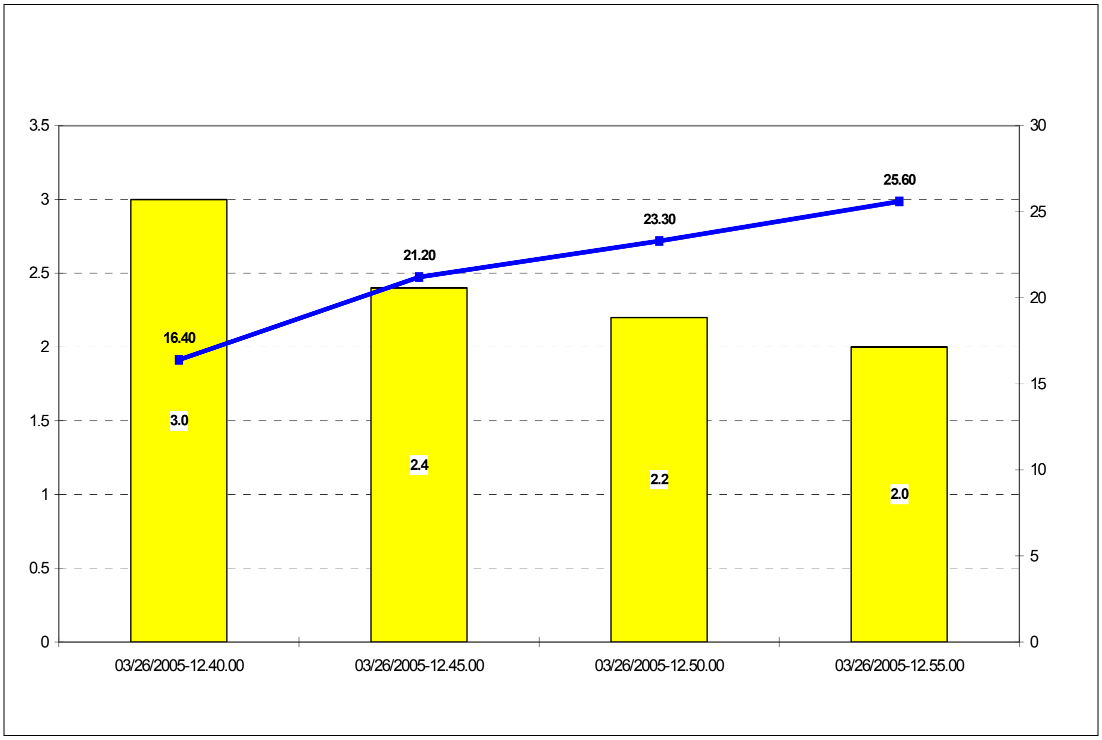


Figure 6-5 Effect of increasing the number of engines keeping the LPAR capacity

The curve analysis shows that the increase of the number of engines does not produce a linear effect on the execution velocity percentage.

Example 6-9 shows that the execution delays % samples account for 83.5% and the PI of 3.

Example 6-9 Extract of RMF WLM report with two engine configuration

```
REPORT BY: POLICY=SPSTPC     WORKLOAD=BATCH      SERVICE CLASS=ITSOB01    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=3
                                                    CRITICAL     =NONE

   TRANSACTIONS     TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--    ---SERVICE----    --SERVICE TIMES--   PAGE-IN RATES     ----STORAGE----
   AVG      19.99   ACTUAL              44.11.936   SSCHRT   1.2   IOC         200   TCB          902.7   SINGLE      0.0   AVG      8359.58
   MPL      19.99   EXECUTION           44.10.934   RESP     1.6   CPU      16814K   SRB            0.0   BLOCK       0.0   TOTAL    167094
   ENDED        8   QUEUED                  1.001   CONN     1.2   MSO      34343K   RCT            0.0   SHARED      0.0   CENTRAL  167094
   END/S     0.03   R/S AFFINITY                0   DISC     0.1   SRB         491   IIT            0.0   HSP         0.0   EXPAND     0.00
   #SWAPS       0   INELIGIBLE                  0   Q+PEND   0.2   TOT      51157K   HST            0.0   HSP MISS    0.0
   EXCTD        0   CONVERSION                  0   IOSQ     0.0   /SEC     170387   IFA            N/A   EXP SNGL    0.0   SHARED    19.99
   AVG ENC  0.00    STD DEV                     0                                   APPL% CP     300.7   EXP BLK     0.0
   REM ENC  0.00                                                  ABSRPTN  8524     APPL% IFACP    0.0   EXP SHR     0.0
   MS ENC   0.00                                                  TRX SERV 8524     APPL% IFA      N/A

   GOAL: EXECUTION VELOCITY 50.0%      VELOCITY MIGRATION:   I/O MGMT  16.4%     INIT MGMT 16.4%

            RESPONSE TIME EX    PERF  AVG    --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
   SYSTEM                 VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU                                UNKN IDLE  USG  DLY  USG  DLY QUIE

   SC69        --N/A--    16.4  3.0  20.0  16.4  N/A  0.0 83.5 83.5                                 0.1  0.0  0.0  0.0  0.0  0.0  0.0
```

Example 6-8 on page 213 shows the last interval with nine engines. The execution delays samples decrease and the PI is now 2, compared to the first interval where it was 3. See Example 6-10 on page 215.

Example 6-10 Extract of RMF WLM report with nine engine configuration

```
REPORT BY: POLICY=SPSTPC     WORKLOAD=BATCH      SERVICE CLASS=ITSOB01    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=3
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE TIMES--   PAGE-IN RATES    ----STORAGE----
   AVG     20.00   ACTUAL                   0   SSCHRT   0.1   IOC        13    TCB         911.0   SINGLE     0.0   AVG     8605.17
   MPL     20.00   EXECUTION                0   RESP     2.6   CPU     15809K   SRB           0.1   BLOCK      0.0   TOTAL    172097
   ENDED       0   QUEUED                   0   CONN     1.8   MSO     33210K   RCT           0.0   SHARED     0.0   CENTRAL  172097
   END/S    0.00   R/S AFFINITY             0   DISC     0.3   SRB      1075    IIT           0.0   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE               0   Q+PEND   0.4   TOT     49021K   HST           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION               0   IOSQ     0.0   /SEC   163512    IFA           N/A   EXP SNGL   0.0   SHARED    20.00
   AVG ENC  0.00   STD DEV                  0                                   APPL% CP    303.9   EXP BLK    0.0
   REM ENC  0.00                                               ABSRPTN  8176    APPL% IFACP   0.0   EXP SHR    0.0
   MS ENC   0.00                                               TRX SERV 8176    APPL% IFA     N/A

   GOAL: EXECUTION VELOCITY 50.0%     VELOCITY MIGRATION:   I/O MGMT  25.6%     INIT MGMT 25.6%

            RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
   SYSTEM                 VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU                               UNKN IDLE  USG  DLY  USG  DLY QUIE

   SC69        --N/A--    25.6  2.0  20.0  25.6  N/A  0.0 74.3 74.3                                0.1  0.0  0.0  0.0  0.0  0.0  0.0
```

To summarize the results of these tests, we can conclude that:

- Velocity is independent of speed.
- Due to the WLM sampling mechanism, the more engines you use (whatever their capacity), the higher probability you have to be sampled as using CPU. That is demonstrated in our second test case.

Note: IRD considerations:

When CPC is close to 100% utilization, to minimize LPAR overhead, IRD gives each LP as few logical CPs as possible, while still trying to make sure that each LP has access to a sufficient capacity configuration. By minimizing the number of logical CPs, it also increases the effective speed of each one.

Notice, in this example, that the CPU service unit consumption is larger with two engines than with nine.

7

TSO, STC, and APPC workloads

This chapter discusses how to classify Time Sharing Option (TSO), STC, and Advanced Program-to-Program Communication (APPC) workloads.

This chapter also includes information about the classification rules for each subsystem type and suggestions about their goal setting.

7.1 General considerations for TSO

TSO is an interactive workload. There are users at the terminals waiting for a quick response when they press Enter.

Each TSO user has individual address space in which only one transaction is executed at a time. Each command issued from the foreground is counted as a transaction. An exception for this rule is the use by the installation of the option CNTCLIST=YES, in IEAOPT*xx*. Using CNTCLIST=YES causes SRM to account for each command in the CLIST as a new transaction. Refer to *z/OS Initialization and Tuning Reference,* SA22-7592*,* for more information about this parameter.

Figure 7-1 illustrates a typical TSO environment: The users Joe and Bill log on to TSO; the logon command creates their user address spaces where their transactions will be executed.

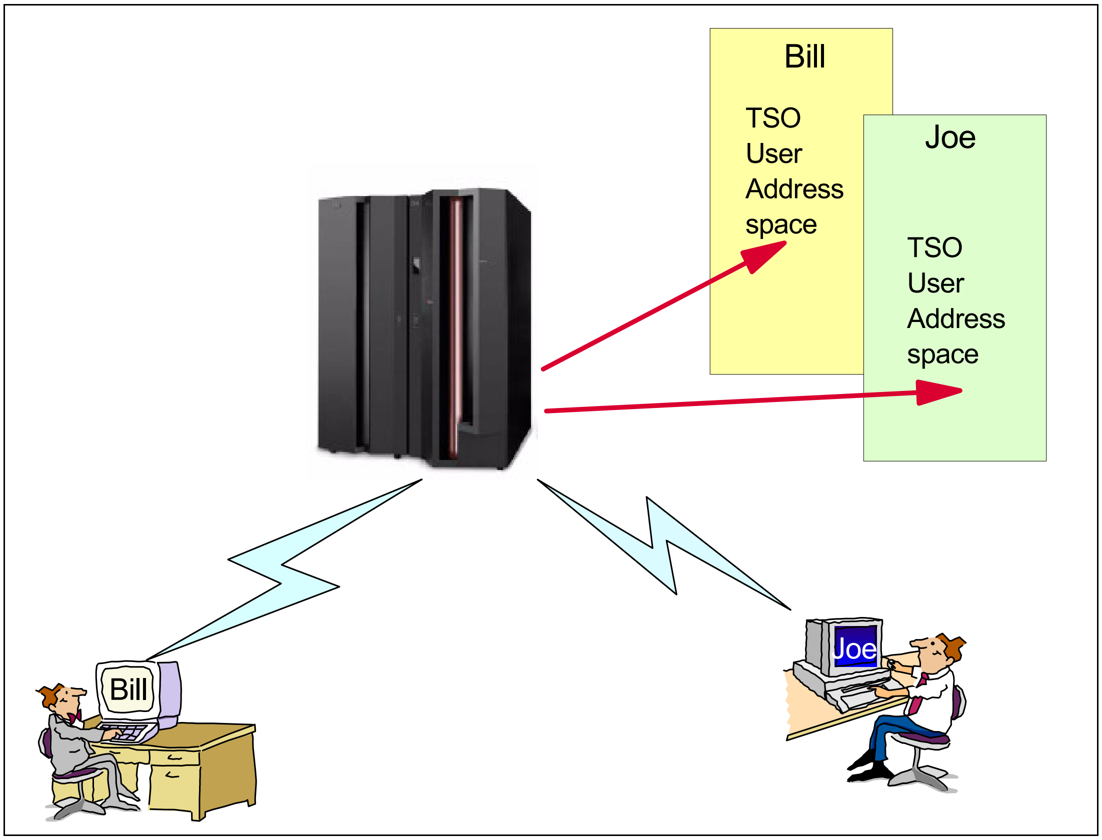


Figure 7-1 Typical TSO environment

7.2 Types of goals for TSO

For TSO transactions, we recommend that you use response time goals. The transactions are short and quick with enough completions to allow SRM to collect a reasonable statistical sample set on which to base decisions. Additionally, response time goals are less sensitive to upgrades of the processor power or capacity than velocity goals.

Workload Manager (WLM) supports two types of response time goals: An *average response time goal* and a *percentile response time goal*. At this point, we want to find out which might be better suited as a goal. Example 7-1 on page 219 shows a response time distribution captured by WLM for a service class. WLM always captures the distribution of ending transactions for its own decision process as well as for external examination. Externally, this information is available through monitoring products such as Resource Measurement Facility (RMF). The only difference is that internally WLM uses slightly higher granularity.

Example 7-1 Example of ended and running TSO transactions

```
SCPER report
          ---RESPONSE TIME---  EX   PERF   AVG   --USING%--  ------------ EXECUTION DELAYS % ------------  ---DLY%-- -CRYPTO%-    %
           HH.MM.SS.TTT         VEL  INDX  ADRSP   CPU   I/O  TOTAL                                         UNKN IDLE  USG  DLY QUIE
   GOAL    00.00.00.150  90.0%
   ACTUALS
   MVSA                  97.1% 57.1%  0.5   75.1   0.1   0.0    0.0                                          0.3  100  0.0  0.0  0.0

                                          ----------RESPONSE TIME DISTRIBUTION----------
      ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT------- 0    10   20   30   40   50   60   70   80   90   100
      HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET !....!....!....!....!....!....!....!....!....!....!
   <  00.00.00.075        10671            10671         94.5        94.5 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
   <= 00.00.00.090        10750               79         95.2         0.7 >
   <= 00.00.00.105        10812               62         95.7         0.5 >
   <= 00.00.00.120        10873               61         96.3         0.5 >
   <= 00.00.00.135        10923               50         96.7         0.4 >
   <= 00.00.00.150        10965               42         97.1         0.4 >
   <= 00.00.00.165        11005               40         97.5         0.4 >
   <= 00.00.00.180        11035               30         97.7         0.3 >
   <= 00.00.00.195        11056               21         97.9         0.2 >
   <= 00.00.00.210        11072               16         98.1         0.1 >
   <= 00.00.00.225        11074                2         98.1         0.0 >
   <= 00.00.00.300        11111               37         98.4         0.3 >
   <= 00.00.00.600        11170               59         98.9         0.5 >
   >  00.00.00.600        11292              122          100         1.1 >
```

In Example 7-1, 11292 transactions executed or ended through the observed time interval. By looking at the report, we can see that 94.5% of the transactions completed within 75 milliseconds and that only 1.1% of the transactions took a longer period of time or have not completed yet. In this example, we defined a percentile response time goal of 0.15 seconds for 90% of the transactions. Looking at the response time distribution, you can see that the majority of the transactions complete within the target goal. Only 2.9% of the transactions are not meeting the goals.

In this configuration, we can immediately understand the advantage of a percentile response time goal. It can be very sensitive to long-running outlying transactions. By using a percentile response time of 90% of all transactions to be completed within 0.15 seconds, the goal is achieved and this facilitates the policy adjustment algorithm. It is easier to adjust the resource access for the work so that 90% of the transactions run slightly faster and achieve the goal than to change the resource access to help a few extremely long-running transactions.

Using response time with percentile is generally better than average response time, because the worst response times tend to distort the average.

In addition, your installation probably uses service class periods for TSO. If your installation does not have service class periods, consider implementing them. As a best practice, three periods are usually enough for TSO users.

You can use multiple periods; we recommend two or three periods. You can punish those transactions that use more service than you think is a reasonable amount of service for a TSO user. You do this by setting the last period of a service class to a low-priority goal.

You can refine the values that you enter as you gain experience with your system behavior. You can adjust the values based on Workload Activity report data.

7.3 Best practices for TSO

You can choose to create a special TSO service class with more aggressive goals and with an importance level higher than the other TSO users. Support can use this special TSO service class in case of system problems, such as debugging or situations where the system hangs up.

During a situation where the system hangs up, you can change the service class of your TSO user ID using the RESET operator command:

```
E tsouid,SRVCLASS=scname
```

In this command, *tsouid* is your TSO user ID and *scname* is the service class name with high priority. In some cases, you need to run this command from the z/OS console. An alternative to this approach is to assign a TSO user ID to SYSSTC and have that user logged on all the time. This allows you to have a TSO session available even in situations where you cannot issue the RESET command.

7.4 Classification rules for TSO

For better handling of our classification rules, we suggest grouping together user IDs with similar characteristics and then assigning a service class to the group. For classification rules, you can use the following qualifiers:

- Accounting Information (AI)
- Perform (PF)
- Perform Group (PFG)
- Sysplex Name (PX)
- Sysname (SY)
- Sysname Group (SYG)
- Userid (UI)
- Userid Group (UIG)

Example 7-2 implements TSO classification with these assumptions:

- Any TSO user, who is not explicitly classified, is assigned to a low priority goal.
- TSO user IDs are classified in three user ID groups: TSOHI, TSOMED, and TSOLOW.
- Only certain special users can use a service class with a very aggressive goal.

Example 7-2 General TSO classification rules

```
Subsystem-Type  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type       Row 1 to 5 of 5
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : TSO         Fold qualifier names?   Y  (Y or N)
Description  . . . Use Modify to enter YOUR rules

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                 Service     Report
                                          DEFAULTS: SCTSOLOW    RTSO
 ____  1  UIG        TSOHI    ___                   SCTSOHI     ________
 ____  1  UIG        TSOMED   ___                   SCTSOMED    ________
 ____  1  UIG        TSOLOW   ___                   SCTSOLOW    ________
 ____  1  UI        GIAN*    ___                SCTSOHOT    ________
 ____  1  UI        ANNA*    ___                SCTSOHOT    ________
```

7.5 Reporting for TSO

Example 7-3 shows a RMF Workload Activity Report collecting TSO data.

In this case, we have defined three periods for the service class TSOHIGH used by TSO user IDs:

- The first period has percentile RT, where 85% of transactions should end within 0.7 seconds and Importance 1.
- The second period has percentile RT, where 75% of transactions should end within 3.0 seconds and Importance 2.
- The third period has percentile RT, where 50% of transactions should end within 10.0 seconds and Importance 3.

The most interesting fields for the TSO performance analysis are:

ENDED This value is the number of transactions ended within the interval of that period.

APPL % This field shows CPU utilization based on uniprocessor capacity. The value can exceed 100% in systems with more than one processor. To get the system utilization, this value has to be divided by the number of processors.

PERF INDX It is the calculated PI for the interval.

EXECUTION DELAYS % The delays under control of WLM/SRM.

DLY% The delays not controlled by WLM/SRM.

RESPONSE TIME DISTRIBUTION The breakdown response time related to the goal.

Example 7-3 RMF Workload Activity report for TSO

```
                                 W O R K L O A D   A C T I V I T Y
                                                                                                                     PAGE  29
      z/OS V1R3              SYSPLEX SYSPLEX1           START 12/16/2004-11.00.00 INTERVAL 001.00.02   MODE = GOAL
                        CONVERTED TO z/OS V1R2 RMF      END   12/16/2004-12.00.02

------------------------------------------------------------------------------------------------------------ SERVICE CLASS PERIODS

REPORT BY: POLICY=POLICY01   WORKLOAD=TSOWRKLD   SERVICE CLASS=TSOHIGH    RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=1
                                                 CRITICAL     =NONE

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
AVG      3.69   ACTUAL                 205   SSCHRT 151.6   IOC    274320   ABSRPTN       608   SINGLE     0.0   AVG     2847.18
MPL      3.60   EXECUTION              203   RESP  2718.9   CPU     7483K   TRX SERV      592   BLOCK      0.0   TOTAL   10236.8
ENDED   65081   QUEUED                   2   CONN   118.4   MSO         0   TCB         452.4   SHARED     0.0   CENTRAL 10236.8
END/S   18.08   R/S AFFINITY             0   DISC  2599.8   SRB    116616   SRB           7.0   HSP        0.0   EXPAND     0.00
#SWAPS  62458   INELIGIBLE               0   Q+PEND   0.6   TOT     7874K   RCT          18.7   HSP MISS   0.0
EXCTD       0   CONVERSION               0   IOSQ     0.0   /SEC     2187   IIT           4.1   EXP SNGL   0.0   SHARED    20.51
AVG ENC  0.00   STD DEV              9.250                                  HST           0.0   EXP BLK    0.0
REM ENC  0.00                                                               APPL %       13.4   EXP SHR    0.0
MS ENC   0.00

 VELOCITY MIGRATION:   I/O MGMT  34.2%    INIT MGMT 34.2%

         ---RESPONSE TIME---  EX   PERF   AVG  --USING%--  ------------ EXECUTION DELAYS % ------------   ---DLY%-- -CRYPTO%-    %
         HH.MM.SS.TTT        VEL  INDX  ADRSP   CPU   I/O  TOTAL   CPU                                    UNKN IDLE  USG  DLY QUIE
 GOAL    00.00.00.700  85.0%
 ACTUALS
 *ALL                  96.3% 34.2%  0.5  640.9   0.0   0.0    0.1   0.1                                     0.4 99.4  0.0  0.0  0.0
 SC01                  95.4% 30.6%  0.5    9.9   0.0   0.0    0.0   0.0                                     0.5 99.5  0.0  0.0  0.0
 SC02                  97.7% 50.6%  0.5   19.6   0.1   0.1    0.1   0.1                                     0.4 99.3  0.0  0.0  0.0
 SC03                 94.6% 56.9%  0.5   20.6   0.1   0.1    0.1   0.0                                    1.6 98.2  0.0  0.0  0.0
 SC04                 97.6% 70.4%  0.5    3.6   0.1   0.0    0.0   0.0                                    0.9 99.0  0.0  0.0  0.0
 SC05                 95.7% 53.1%  0.5   29.4   0.0   0.0    0.0   0.0                                    0.4  100  0.0  0.0  0.0
 SC06                 97.3% 35.6%  0.5  124.8   0.0   0.0    0.1   0.1                                    0.1  100  0.0  0.0  0.0
 SC07                 95.9% 30.2%  0.5  354.4   0.0   0.0    0.1   0.1                                    0.5 99.4  0.0  0.0  0.0
 SC08                 93.7% 23.0%  0.5    8.1   0.0   0.0    0.1   0.0                                    0.7 99.2  0.0  0.0  0.0
 SC09                 98.0% 55.2%  0.5    7.1   0.0   0.1    0.1   0.0                                    1.1 98.6  0.0  0.0  0.0
 SC10                 97.4% 18.8%  0.5   63.4   0.0   0.0    0.1   0.1                                    0.2  100  0.0  0.0  0.0

                                             W O R K L O A D   A C T I V I T Y
                                                                                                               PAGE  30
       z/OS V1R3              SYSPLEX SYSPLEX1          START 12/16/2004-11.00.00 INTERVAL 001.00.02   MODE = GOAL
                         CONVERTED TO z/OS V1R2 RMF      END   12/16/2004-12.00.02

                                       ----------RESPONSE TIME DISTRIBUTION----------
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
<  00.00.00.350        60706            60706         93.3        93.3  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.00.420        61345              639         94.3         1.0  >
<= 00.00.00.490        61807              462         95.0         0.7  >
<= 00.00.00.560        62165              358         95.5         0.6  >
<= 00.00.00.630        62454              289         96.0         0.4  >
<= 00.00.00.700        62651              197         96.3         0.3  >
<= 00.00.00.770        62836              185         96.6         0.3  >
<= 00.00.00.840        63012              176         96.8         0.3  >
<= 00.00.00.910        63138              126         97.0         0.2  >
<= 00.00.00.980        63262              124         97.2         0.2  >
<= 00.00.01.050        63362              100         97.4         0.2  >
<= 00.00.01.400        63800              438         98.0         0.7  >
<= 00.00.02.800        64622              822         99.3         1.3  >
>  00.00.02.800        65081              459          100         0.7  >

                                             W O R K L O A D   A C T I V I T Y
                                                                                                               PAGE  31
       z/OS V1R3              SYSPLEX SYSPLEX1          START 12/16/2004-11.00.00 INTERVAL 001.00.02   MODE = GOAL
                         CONVERTED TO z/OS V1R2 RMF      END   12/16/2004-12.00.02

REPORT BY: POLICY=POLICY01   WORKLOAD=TSOWRKLD   SERVICE CLASS=TSOHIGH    RESOURCE GROUP=*NONE      PERIOD=2 IMPORTANCE=2
                                                 CRITICAL     =NONE

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
AVG      0.44   ACTUAL                2.577   SSCHRT  58.4   IOC     117766   ABSRPTN      1013   SINGLE     0.0   AVG     1960.25
MPL      0.44   EXECUTION             2.356   RESP  3881.8   CPU      1479K   TRX SERV     1012   BLOCK      0.0   TOTAL    870.20
ENDED     529   QUEUED                  220   CONN   327.3   MSO          0   TCB          89.7   SHARED     0.0   CENTRAL  870.20
END/S    0.15   R/S AFFINITY              0   DISC  3553.9   SRB      23010   SRB           1.4   HSP        0.0   EXPAND     0.00
#SWAPS    771   INELIGIBLE                0   Q+PEND   0.5   TOT      1620K   RCT           0.2   HSP MISS   0.0
EXCTD       0   CONVERSION               25   IOSQ     0.1   /SEC       450   IIT           1.5   EXP SNGL   0.0   SHARED     3.01
AVG ENC  0.00   STD DEV              12.111                                   HST           0.0   EXP BLK    0.0
REM ENC  0.00                                                                 APPL %        2.6   EXP SHR    0.0
MS ENC   0.00

VELOCITY MIGRATION:   I/O MGMT  41.2%    INIT MGMT 41.2%

        ---RESPONSE TIME---  EX   PERF   AVG   --USING%--  ------------ EXECUTION DELAYS % -------------  ---DLY%-- -CRYPTO%-    %
        HH.MM.SS.TTT        VEL  INDX  ADRSP   CPU   I/O  TOTAL   CPU  I/O                               UNKN IDLE  USG  DLY QUIE
GOAL    00.00.03.000  75.0%
ACTUALS
*ALL                  83.9% 41.2%  0.8    2.0   1.4   3.0    6.2   4.6  1.7                               4.2 85.2  0.0  0.0  0.0
SC01                  25.0% 21.1%  1.3    0.0   0.0  18.2   68.2  68.2  0.0                               9.1  0.0  0.0  0.0  0.0
SC02                  86.5% 42.9%  0.9    0.0   5.6  29.9   47.2  34.5 12.7                              17.3  0.0  0.0  0.0  0.0
SC03                  95.0% 63.7%  0.5    0.0   5.5  30.8   20.7   7.1 13.6                              42.4  0.8  0.0  0.0  0.0
SC04                   100% 66.7%  0.9    0.0  18.8   6.3   12.5   0.0 12.5                              62.5  0.0  0.0  0.0  0.0
SC05                  88.0% 60.3%  0.6    0.0   8.4  39.0   31.2   9.2 22.0                              18.2  3.5  0.0  0.0  0.0
SC06                  93.1% 42.6%  0.6    0.7   0.8   0.7    2.0   1.5  0.5                               1.0 95.5  0.0  0.0  0.0
SC07                  79.2% 37.0%  0.9    0.1  11.2  14.8   44.2  36.9  7.4                              24.5  5.4  0.0  0.0  0.0
SC08                  75.0% 21.4%  0.8    1.1   0.1   0.1    0.5   0.0  0.5                               0.0 99.3  0.0  0.0  0.0
SC09                  86.7% 38.5%  0.7    0.0   3.4   3.4   10.8  10.1  0.7                               5.4 77.7  0.0  0.0  0.0
SC10                  76.9% 27.0%  1.0    0.0   2.6  14.7   46.8  42.4  4.4                              35.9  0.0  0.0  0.0  0.0

                                       ----------RESPONSE TIME DISTRIBUTION----------
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
<  00.00.01.500          317              317         59.9        59.9  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.01.800          350               33         66.2         6.2  >>>>
<= 00.00.02.100          386               36         73.0         6.8  >>>>
<= 00.00.02.400          408               22         77.1         4.2  >>>
<= 00.00.02.700          428               20         80.9         3.8  >>>
<= 00.00.03.000          444               16         83.9         3.0  >>
<= 00.00.03.300          460               16         87.0         3.0  >>
<= 00.00.03.600          471               11         89.0         2.1  >>
<= 00.00.03.900          484               13         91.5         2.5  >>
<= 00.00.04.200          487                3         92.1         0.6  >
<= 00.00.04.500          492                5         93.0         0.9  >
<= 00.00.06.000          504               12         95.3         2.3  >>
<= 00.00.12.000          519               15         98.1         2.8  >>
>  00.00.12.000          529               10          100         1.9  >>

                                              W O R K L O A D   A C T I V I T Y
                                                                                                                     PAGE  32
      z/OS V1R3              SYSPLEX SYSPLEX1           START 12/16/2004-11.00.00 INTERVAL 001.00.02   MODE = GOAL
                         CONVERTED TO z/OS V1R2 RMF      END   12/16/2004-12.00.02

REPORT BY: POLICY=POLICY01   WORKLOAD=TSOWRKLD   SERVICE CLASS=TSOHIGH    RESOURCE GROUP=*NONE      PERIOD=3 IMPORTANCE=3
                                                 CRITICAL     =NONE

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
AVG     10.84   ACTUAL            1.07.680   SSCHRT 317.3   IOC    593935    ABSRPTN       334   SINGLE     0.0   AVG     2416.68
MPL     10.80   EXECUTION         1.07.661   RESP  117819M  CPU     11997K   TRX SERV      333   BLOCK      0.0   TOTAL   26108.2
ENDED     152   QUEUED                  19   CONN   145.5   MSO         0    TCB         698.6   SHARED     0.0   CENTRAL 26108.2
END/S    0.04   R/S AFFINITY             0   DISC  117819M  SRB    412471    SRB          24.2   HSP        0.0   EXPAND     0.00
#SWAPS   5275   INELIGIBLE               0   Q+PEND   0.4   TOT     13003K   RCT           2.8   HSP MISS   0.0
EXCTD       0   CONVERSION               0   IOSQ     0.0   /SEC     3613    IIT           7.0   EXP SNGL   0.0   SHARED    85.01
AVG ENC  0.00   STD DEV           8.58.264                                   HST           0.0   EXP BLK    0.0
REM ENC  0.00                                                                APPL %       20.3   EXP SHR    0.0
MS ENC   0.00

VELOCITY MIGRATION:   I/O MGMT  32.4%    INIT MGMT 32.4%

        ---RESPONSE TIME---  EX   PERF   AVG   --USING%--  ------------ EXECUTION DELAYS % -------------  ---DLY%-- -CRYPTO%-    %
        HH.MM.SS.TTT         VEL  INDX  ADRSP   CPU   I/O  TOTAL   CPU  I/O                               UNKN IDLE  USG  DLY QUIE
GOAL    00.00.10.000  50.0%
ACTUALS
*ALL                  68.4% 32.4%  0.6   23.4   0.9   1.3    4.7   4.1  0.5                                6.1 87.0  0.0  0.0  0.0
SC01                   0.0%  0.2%  N/A    1.0   0.1   0.0   27.5  27.5  0.0                                1.2 71.2  0.0  0.0  0.0
SC02                   100% 80.0%  0.6    0.0  20.0  60.0   20.0  20.0  0.0                                0.0  0.0  0.0  0.0  0.0
SC03                  45.5% 62.9%  1.1   14.6   1.2   1.7    1.7   1.0  0.6                                9.0 86.5  0.0  0.0  0.0
SC04                   0.0%  5.9%  N/A    2.0   0.1   0.0    1.5   1.5  0.0                                0.0 98.4  0.0  0.0  0.0
SC05                  68.4% 18.6%  0.7    0.2   4.4   7.8   53.2  48.8  4.4                                8.4 26.1  0.0  0.0  0.0
SC06                  95.5% 19.0%  0.5    2.4   0.3   0.2    2.3   2.2  0.1                                0.8 96.4  0.0  0.0  0.0
SC07                  67.1% 13.8%  0.5    2.2   0.9   1.5   15.0  14.4  0.6                                1.8 80.7  0.0  0.0  0.0
SC08                   100%  0.0%  0.5    0.9   0.0   0.0    0.1   0.1  0.0                                0.0  100  0.0  0.0  0.0
SC09                  50.0% 30.3%  0.6    0.1   0.9  20.3   48.8  42.6  6.2                               30.0  0.0  0.0  0.0  0.0

                                       ----------RESPONSE TIME DISTRIBUTION----------
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
<  00.00.05.000           74               74         48.7        48.7  >>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.06.000           85               11         55.9         7.2  >>>>
<= 00.00.07.000           93                8         61.2         5.3  >>>
<= 00.00.08.000           97                4         63.8         2.6  >>
<= 00.00.09.000          101                4         66.4         2.6  >>
<= 00.00.10.000          104                3         68.4         2.0  >>
<= 00.00.11.000          107                3         70.4         2.0  >>
<= 00.00.12.000          110                3         72.4         2.0  >>
<= 00.00.13.000          112                2         73.7         1.3  >
<= 00.00.14.000          112                0         73.7         0.0  >
<= 00.00.15.000          114                2         75.0         1.3  >
<= 00.00.20.000          118                4         77.6         2.6  >>
<= 00.00.40.000          130               12         85.5         7.9  >>>>>
>  00.00.40.000          152               22          100        14.5  >>>>>>>>
```

Consider setting TSO goals so that most of the trivial transactions that use a low percentage of CPU end in the first period.

Analyzing the Response Time (RT) distribution, you can adjust your percentile goal based on the percentage *IN BUCKET* value. The percentages of execution delays show you which is the major bottleneck impacting your goal.

For a complete description of the fields in RMF reports, see *z/OS V1R6.0 Resource Measurement Facility (RMF) Report Analysis,* SC33-7991.

7.6 General considerations for STC

Many types of work within a z/OS system today run, or can be run, as a started task. For example, you can run CICS systems as started tasks (or as batch jobs) along with many system address spaces, including VTAM, JES, LLA, VLF, WLM, and many others.

WLM provides two internal service classes for this type of workload: SYSTEM and SYSSTC. These service classes are assigned by default to system functions.

STC address spaces are assigned to a default service class depending upon attributes specified on the ASCRE (create address space) macro and depending on whether the task is a system task or a privileged task. There is no external control of the HIPRI attribute of the ASCRE macro for started tasks. Table 7-1 shows the default assignments of service classes for started task address spaces.

Table 7-1 STC service class assignments

ASCRE attribute	Privileged or system task	Neither privileged nor system task

HIPRI	SC = ’SYSTEM’
	DP = x ’FF’
		SC = ’SYSTEM’
		DP = x ’FF’

NONURG	SC = ’SYSSTC’
	DP = x ’FE’
		SC = default for STC
		DP = managed to SC goal

In Table 7-1:

- DP = dispatching priority
- SC = service class name
- STC = started task subsystem type

SYSTEM service class

As you can see in Table 7-1, you can classify a non-system task or a non-privileged task in the SYSTEM service class, if the ASCRE macro that created the address space has the HIPRI attribute specified.

The system address spaces are automatically assigned to the SYSTEM internal service class.

Examples of these system tasks are:

- MASTER
- GRS
- DUMPSRV
- SMF
- CATALOG
- RASP
- XCFAS
- CONSOLE
- IOSAS
- SMSPDSE
- ANTMAIN
- JESXCF
- ALLOCAS
- IXGLOGR
- WLM

The system goal always has DP=255, IOP=255, and there is no specific storage protection because they are cross-memory servers. If the caller's address space suffers page faults, then storage protection is activated.

You do not need to create classification rules for them. However, if you want a report for some system address spaces, you must put them in the classification rules. You can see this in Example 7-4 on page 227.

If you want to control system address spaces, you can define a service class for them by specifying a classification rule in the STC subsystem entry. We do not recommend this way of managing system address space, and you must take this action very carefully. The system address spaces that you change might lose the high dispatching priority attribute and run at the dispatching priority of the assigned service class period. You can restore the high dispatching priority attribute by one of the following methods:

- Using the RESET operator command:

  ```
  E stcname,SRVCLASS=SYSTEM
  ```

- Changing the classification rules to explicitly classify the started task to SYSTEM and activate the policy.

> **Note:** *MASTER*, WLM, and some other *system-controlled* address spaces always run in the SYSTEM service class and cannot be reassigned via the service definition or via the RESET command.

SYSSTC service class

System tasks are those tasks with the privileged and system task attribute specified in the IBM-supplied program properties table (PPT). To add a new definition or override the default PPT entries, use the SCHED*xx* PARMLIB member. Refer to *z/OS Initialization and Tuning Reference,* SA22-7592, for a description of SCHED*xx* and PPT entries.

SYSSTC is the default service class for started tasks that are not classified in the SYSTEM service class, unless you provide another default in SUBSYS=STC.

The SYSSTC service class has DP=254, IOP=254, and no specific storage protection, because the address spaces are cross-memory servers. The multiprogramming level (MPL) is fixed and very high, resulting in no OUTR swaps for swappable STC address spaces. This is where important, light CPU address spaces, such as VTAM, JES, and IRLM should run. If you want to define a service class for some of these address spaces, we recommend that you use a high importance and high velocity goals.

We recommend that you classify all the STC with an adequate service class.

It is good practice to provide a default service class in the classification rules for subsystem type STC with a low importance so that any unclassified STC does not cause any performance issues. We recommend that you assign high importance tasks here such as VTAM, JES*x*, VLF, LLA, IRLM, TSO, monitors, auto operator packages, the OMVS kernel, APPC, and ASCH. You can create a transaction name group (TNG) for them and then define the group in classification rules.

A CPU-intensive started task is not appropriate for SYSSTC, because the task might use a large amount of processor cycles. However, if your processor is lightly loaded, SYSSTC might be appropriate, because that one task might not affect the ability of the remaining processors to manage the important work.

7.7 Classification rules for STC

A system address space always has SYSTEM or SYSSTC service class assigned. Non-system STCs, which are not classified, go to the default STC service class.

After choosing those started tasks to classify in the SYSSTC and SYSTEM service class, collect the others into a small number of similar groups, from higher to lower importance. We suggest three or four groups at a maximum. You can increase the velocity goal value, according to the importance of the group.

Usually started tasks are long-running address spaces. You can create a service class for short-running STCs, assigning them a percentile response time goal if you have at least ten ended transactions within 20 minutes.

Because the Transaction Managers (CICS, IMS, DB2, and so on) are long-running batch jobs or started tasks, you should assign them to a velocity goal. When setting a velocity goal for these STCs, set a goal that enables the regions to run sufficiently so the transactions they serve will meet the service level objectives.

When the Transaction Managers address spaces start up, they are classified as batch jobs or started tasks and are assigned a service class. When classified, they are managed by WLM to meet their specified velocity goals. You must specify a goal for the Transaction Manager's address spaces in order to ensure timely initialization during startup and proper treatment when they are not processing any transactions for a long period of time.

Normally, you do not define multiple service class periods for started tasks. They usually perform services for other address spaces and are long-running address spaces. Consequently, there is no reason to give them a lower priority as service consumption grows. Maybe, in cases where you have problems starting your online system in the middle of the day after an unscheduled outage, you can consider creating a service class with two periods: The first with a very high velocity goal and duration enough to complete initialization, and the second with normal velocity.

For classification rules, you can use the following qualifiers:

- Accounting Information (AI)
- Perform (PF)
- Perform Group (PFG)
- Sysplex Name (PX)
- Subsystem Parameter (SPM)
- Sysname (SY)
- Sysname Group (SYG)
- Transaction Name (TN)
- Transaction Name Group (TNG)
- Userid (UI)
- Userid Group (UIG)

Do not worry if you see some address spaces such as IRLM and VTAM with low execution velocity. This can happen when idle time is high, because WLM does not include idle time samples in the velocity calculation. WLM samples the state of each address space and enclave every quarter second. Consider 15 minutes, 3600 samples: If VTAM is 98% idle, only 72 samples are available for velocity calculation. A few delay samples can cause relatively large swings in velocity.

7.8 Types of goals for STC

All system address spaces should use SYSTEM or SYSSTC service class. You do not need to specify any goals for these service classes.

Because the non-system STCs are usually long-running address spaces, they should be assigned to a velocity goal. Keep in mind when setting a velocity goal for non-system STCs that it is better to set a goal that allows them to run sufficiently so the transactions they serve will meet their service level objectives.

7.9 Best practices for STC

All system-related address spaces should be classified to service class SYSTEM and all address spaces that either support the system or the operation of it should be classified to SYSSTC. We strongly recommend that you use the System Parameter (SPM) qualifier for the STC subsystem to classify all system tasks to SYSTEM or SYSSTC.

Example 7-4 shows a sample of a classification for a started task based on:

- Any started tasks not explicitly classified are given low priority.
- Started tasks are defined in three transaction name groups: STCHI, STCMED, and STCLOW.
- System defaults are used for MVS-owned address spaces.
- Separate reporting is used for *MASTER* and GRS.

Example 7-4 General STC classification

```
Subsystem-Type  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type     Row 1 to 11 of 11
Command ===> ____________________________________________ SCROLL ===> PAGE

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 ===>
          --------Qualifier--------               -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: SCSTCLOW    ________
 ____  1  TN         *MASTER* ___                  SYSTEM      MASTER
 ____  1  TN         GRS      ___                  SYSTEM      GRS
 ____  1  TNG        DB2      ___                  SYSSTC      ________
 ____  1  TN         %%%%IRLM ___                  SYSSTC      ________
 ____  1  SPM        SYSTEM   ___                  SYSTEM      ________
 ____  1  SPM        SYSSTC   ___                  SYSSTC      ________
 ____  1  TN         PCAUTH   ___                  SYSTEM      ________
 ____  1  TN         TRACE    ___                  SYSTEM      ________
 ____  1  TN         JES2AUX  ___                  SYSSTC      ________
 ____  1  TN         ANTAS000 ___                  SYSSTC      ________
 ____  1  TN         RRS      ___                  SYSSTC      ________
 ____  1  TNG        STCHI    ___                  SCSTCHI     ________
 ____  1  TNG        STCMED   ___                  SCSTCMED    ________
 ____  1  TNG        STCLOW   ___                  SCSTCLOW    ________
```

Example 7-4 on page 227 shows the use of the SPM rules. It assigns started tasks created with the high dispatching priority attribute to SYSTEM, and other privileged or system tasks to SYSSTC. In this way, WLM will manage started tasks that are recognized as special.

Note that explicitly defining the SPM rules, as in Example 7-4 on page 227, is optional. If they were removed from the example, then the high dispatching priority work is still assigned to SYSTEM, and the other privileged or system tasks are still assigned to SYSSTC, because those are the defaults. The reason for explicitly defining them here with the SPM rules is to protect yourself from inadvertently assigning them elsewhere in the rules that follow the SPM rules.

> **Important:** The placement of these SPM rules is crucial. If they had been declared first, then the two TN rules intended to assign *MASTER* and GRS to report classes would have never been reached.

We recommend that you classify at least JES2AUX, RRS, and ANTAS000 to SYSSTC. PCAUTH and TRACE should run at the highest priority (SYSTEM service class) to avoid contention on their own local lock when the system is at high CPU usage. Refer to APARs OA05709 and OA08960 for further details.

In addition, Example 7-4 on page 227 shows that the IRLM address spaces are classified to SYSSTC. In general, IRLM requires fast access to the CPU and does not consume too much of it. Therefore, it is predestined to get classified to SYSSTC.

You can use a similar rule for other work. If the work is very CPU sensitive, uses only very little of it (less than 5% of one CP), and does not show abrupt and high CPU demand, you can classify it to SYSSTC.

You can follow this rule as long as the total amount of work in SYSSTC does not increase too much. Your installation should try to keep the total CPU consumption of the SYSTEM and SYSSTC service classes below 10 to 15% of the total system capacity.

The first rule assigns *MASTER* to report class MASTER. (Note that “SYSTEM” is shown in the service class column. In this case, it is not a true assignment, because *MASTER* must always run in the SYSTEM service class anyway.) *MASTER* can also be classified using the SPM SYSTEM rule, but because separate reporting is wanted for *MASTER*, it is classified separately. GRS is handled similarly.

The DB2 subsystem is defined in IBM-supplied PPT with the system task attribute to assign DB2 address spaces to SYSSTC. If you do not want this assignment, you need to group DB2 address spaces (MSTR, DBM1, DIST, and SPAS), assign them to a TNG, and associate the group to a high importance service class. You need to define this rule before any SPM entries.

Started tasks in the STCHI transaction name group are assigned to service class SCSTCHI, STCMED in SCSTCMED, and STCLOW in SCSTCLOW. The goals on the service classes use importance to guarantee that started tasks of low value are sacrificed if necessary to keep medium-value started tasks meeting goals. This does not guarantee any relationship between the dispatching priorities that will be observed for service classes SCSTCMED and SCSTCLOW.

For a better analysis of the STCs, including the system STCs, consider assigning a report class in your classification rules in order to have the most granularity in your RMF reports.

7.10 Reporting for STC

Because the STCs have different specifications for workloads and importance (CICS, DB2, HSM, monitors, non-IBM products, and so on), approach the STC analysis from a subsystem point of view.

If you have created specific reporting classes for your analysis, look at the report class section of your RMF WLKD Activity report in order to find out how that STC (or group of STCs) is performing.

Example 7-5 shows a sample report where you can evaluate the performance of the SYSSTC service class.

Example 7-5 RMF Workload Activity report for STC

```
                                 W O R K L O A D   A C T I V I T Y
                                                                                                                     PAGE  13
      z/OS V1R6               SYSPLEX WTSCPLX1           START 03/28/2005-12.00.00 INTERVAL 000.59.59   MODE = GOAL
                              RPT VERSION V1R5 RMF       END   03/28/2005-12.59.59

                                        POLICY ACTIVATION DATE/TIME 03/28/2005 10.55.32

------------------------------------------------------------------------------------------------------------ SERVICE CLASS PERIOD

REPORT BY: POLICY=SPSTPC     WORKLOAD=SYSTEM     SERVICE CLASS=SYSSTC     RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=SYSTEM
                                                 CRITICAL     =NONE

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE TIMES--   PAGE-IN RATES    ----STORAGE----
AVG     45.56   ACTUAL            9.18.375   SSCHRT  85.6   IOC    861983    TCB          84.7   SINGLE     0.0   AVG     6531.42
MPL     45.55   EXECUTION         9.18.239   RESP   129.6   CPU      1758K   SRB          37.0   BLOCK      0.0   TOTAL    297512
ENDED      87   QUEUED                 135   CONN     2.5   MSO      2856K   RCT           0.5   SHARED     0.0   CENTRAL  297512
END/S    0.02   R/S AFFINITY             0   DISC   125.9   SRB    767819    IIT           3.6   HSP        0.0   EXPAND     0.00
#SWAPS   2965   INELIGIBLE               0   Q+PEND   0.8   TOT      6244K   HST           0.0   HSP MISS   0.0
EXCTD       0   CONVERSION              12   IOSQ     0.4   /SEC     1734    IFA           N/A   EXP SNGL   0.0   SHARED  2784.82
AVG ENC  0.00   STD DEV           18.12.960                                  APPL% CP      3.5   EXP BLK    0.0
REM ENC  0.00                                               ABSRPTN    38    APPL% IFACP   0.0   EXP SHR    0.0
MS ENC   0.00                                               TRX SERV   38    APPL% IFA     N/A
GOAL: SYSTEM

         RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
SYSTEM                 VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU  I/O                          UNKN IDLE  USG  DLY  USG  DLY QUIE

SC69        --N/A--    32.6      106.9   0.0  N/A  0.1  0.3  0.2  0.1                          12.3 87.3  0.0  0.0  0.0  0.0  0.0
```

The most interesting fields for the STC performance analysis are:

- APPL% CP: This field shows the used percentage of a CP in the interval.
- PERF INDX (not shown in the cases of SYSTEM and SYSSTC service classes): It is the calculated PI for the interval.
- EXECUTION DELAYS %: The delays under control of WLM and SRM.
- DLY%: The delays not controlled by WLM and SRM.
- IOSQ: It is the IOS UCB delay (software queue).

The PI, APPL%, and VEL fields are a good starting point to evaluate the performance and to understand if there is any potential problem in your system.

You can find more details about this report in *z/OS V1R6.0 Resource Measurement Facility (RMF) Report Analysis,* SC33-7991.

7.11 General considerations for APPC

WLM recognizes APPC work that has been scheduled by the APPC scheduler (ASCH). If your installation uses a different APPC scheduler, you need to consult the product's documentation for information about classifying its transactions. Figure 7-2 shows APPC communications using an ASCH scheduler.

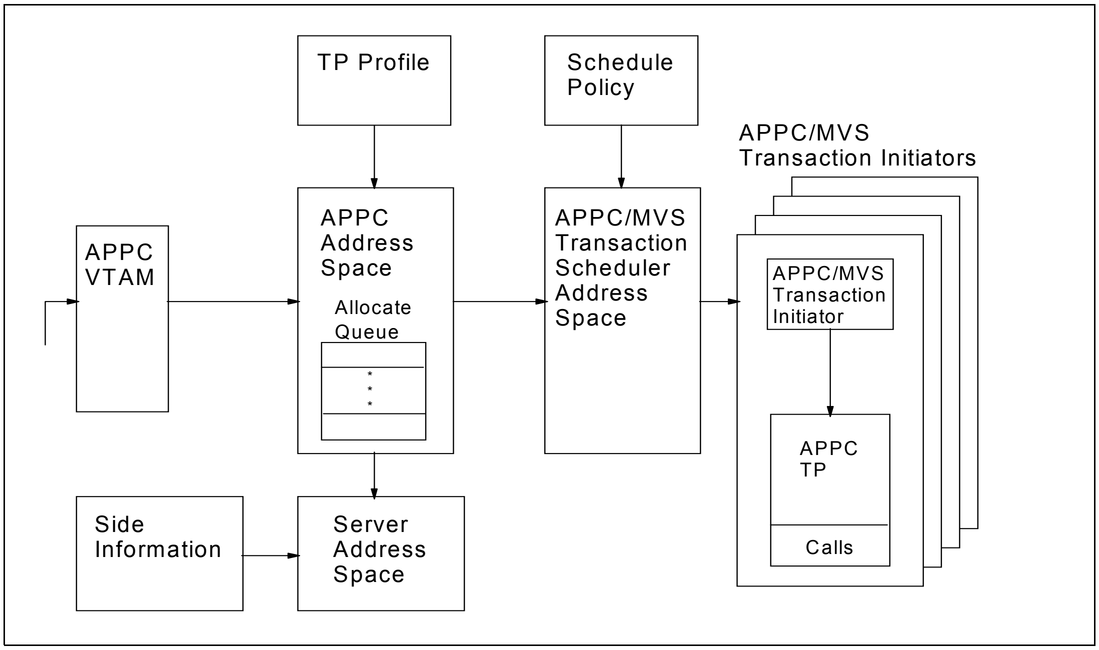


Figure 7-2 APPC environment using ASCH scheduler

An APPC transaction begins when the ASCH address space schedules a transaction program. That program is then started in an APPC initiator address space. This concept is very similar to the JES subsystem scheduling a batch job. The APPC transaction ends when the program returns to the initiator, similar to the completion of a batch job. Just as there are different profiles for batch jobs, there are different types of APPC transaction programs.

Your installation can use the VTAM generic resources function to improve availability and balance its APPC workload across the systems in a sysplex. VTAM passes to WLM the list of eligible APPL*n* instances. When all the systems in the sysplex have applications in this generic resources group, then WLM checks its service policy and checks the loading on each of the APPL*n* address spaces. It then chooses the instance that it calculates as the best instance able to handle a new session while meeting the service goals. WLM tells VTAM which APPL*n* it has chosen.

7.12 Classification rules for APPC

As with TSO and STCs, APPC work also needs to be classified using the WLM application. From the subsystem types, you have to select the ASCH entry in order to specify your APPC rules.

For classification rules, use these qualifiers:

- Accounting Information (AI)
- Sysplex Name (PX)
- Sysname (SY)
- Sysname Group (SYG)
- Transaction Class (TC)
- Transaction Class Group (TCG)
- Transaction Name (TN)
- Transaction Name Group (TNG)
- Userid (UI)
- Userid Group (UIG)

7.13 Types of goals for APPC

The type of goal you choose depends on the type of APPC Transaction Program (TP):

- If your installation is sure that all TPs will process a single request and have at least ten completions in 20 minutes, then the service class for these transactions can contain multiple periods with response time goals.
- Many APPC transaction programs keep a permanent, or lengthy, network connection and repeatedly process individual conversations across that network. These long-running transaction programs might stay connected to the network for an indeterminate time period. In these cases, you need to use a velocity goal.

If you are not sure that an APPC transaction program starts and ends for each network interaction, it is best to assign APPC transactions to single period service classes with velocity goals.

7.14 Best practices for APPC

By default or via SPM rules, ASCH and APPC address spaces are assigned to the SYSSTC service class. This ensures that the scheduler can quickly process requests for new APPC transaction programs.

7.15 Reporting for APPC

You can adjust your goals by using the RMF Workload Activity Report for the APPC service classes that you defined. You can obtain the number of ended transactions, the average response time, and velocity from periods representing peak periods for the ASCH workload. This information can help you adjust the service class goals.

8

DB2 workload considerations

This chapter provides a description of the characteristics of a DB2 workload, how to classify the different types, and how to report the related data.

8.1 General considerations

There are functions in DB2 that are important for business intelligence applications (sysplex query parallelism) or that are client/server-oriented (stored procedures). These new functions exploit workload manager (WLM) in order to improve performance management.

8.2 DB2 address spaces

Next, we describe how to set goals for DB2 environment management.

For DB2 address spaces, velocity goal is the appropriate goal. A small amount of the work done in DB2 is counted toward this velocity goal. Most of the DB2 threads work applies to the user goal.

8.2.1 DB2 address space velocity goals

This section tells you how to classify DB2 address spaces in order to satisfy the required velocity goals.

Use the following service classes for non-DBMS address spaces:

- SYSSTC service class for:
  - VTAM and TCP/IP address spaces
  - IRLM address space (IRLMPROC)

> **Important:** The VTAM, TCP/IP, and IRLM address spaces must always have a higher dispatching priority than all of the DBMS address spaces, their attached address spaces, and their subordinate address spaces. Do not allow WLM to reduce the priority of VTAM, TCP/IP, or IRLM to or below the priority of the other database management system (DBMS) address spaces.

- An installation-defined service class with a high velocity goal for:
  - DB2 (all address spaces, except for the DB2-established stored procedures address space):
    - %%%%MSTR
    - %%%%DBM1
    - %%%%DIST (DDF address space)

When you set response time goals for Distributed Data Facility (DDF) threads or for stored procedures in a WLM-established address space, the only work that is controlled by the DDF or stored procedures velocity goals is the DB2 service tasks (work performed for DB2 that cannot be attributed to a single user). The user work runs under separate goals for the enclave.

For the DB2-established stored procedures address space, use a velocity goal that reflects the requirements of the stored procedures in comparison to other application work. Usually, it is lower than the goal for DB2 address spaces, but it might be equal to the DB2 address space depending on what type of distributed work your installation does.

Consider the following other workload management considerations:

- IRLM must be eligible for the SYSSTC service class. To make IRLM eligible for SYSSTC, you do not need to classify IRLM to one of your own service classes.

- If you need to change a goal, change the velocity between 5 and 10%. Velocity goals do not translate directly to priority. Higher velocity tends to have higher priority, but this is not always the case.
- If you use I/O Priority management, WLM can assign I/O priority (based on I/O delays) separately from processor priority.
- z/OS workload management dynamically manages storage isolation to meet the goals you set.

8.2.2 WLM bufferpool management

DB2 bufferpools can occupy a large amount of storage. Starting with z/OS 1.8, bufferpool storage management is integrated with the WLM management of storage resources to assist in adjusting the size of a bufferpool. Bufferpool users can be CICS or IMS transactions represented by a Performance Block or DDF enclaves represented by a new bufferpool management only Performance Block. The bufferpool size and its hit ratio are recorded in this Performance Block.

In cooperation with WLM, DB2 is now able to efficiently manage the size of its buffer pools in a way that WLM can recommend DB2 to increase or decrease the bufferpool sizes. As an result, a new subsystem wait state BPMI is available indicating that the work manager is waiting for I/O because of a DB2 bufferpool miss. RMF indicates this new wait state in the state samples breakdown in the Subsystem section of the Postprocessor WLMGL report and in the Response Time Breakdown section of the Monitor III SYSWKM report.

Note: WLM algorithms verify the bufferpool size and adjust it if necessary. To prevent out of storage conditions, WLM tries to take storage from other bufferpools first.

8.2.3 How DB2 assigns I/O priorities

DB2 informs z/OS about which address space's priority is to be associated with a particular I/O request, and WLM handles the management of the request. Table 8-1 describes which enclave or address space is associated with I/O read requests.

Table 8-1 How read I/O priority is determined

Request type	Synchronous reads	Prefetch reads
Local	Application's address space	Application's address space
DDF or sysplex query parallelism (assistant only)	Enclave priority	Enclave priority

Table 8-2 describes to which enclave or address space DB2 is associated with the I/O write requests.

Table 8-2 How write I/O priority is determined

Request type	Synchronous writes	Deferred writes
Local	Application's address space	ssnmDBM1 address space
DDF or sysplex query parallelism (assistant only)	DDF address space	ssnmDBM1 address space

8.3 DB2 distributed environment

This section describes ways that you can tune your systems and applications that use Distributed Relational Database Architecture™ (DRDA) or DB2 private protocol for distributed data access.

DB2 supports two types of remote access between the requesting relational database management system (DBMS) and the serving relational database management system. The two types of access are DRDA access and DB2 private protocol access. When three-part named objects are referenced (or aliases for three-part name objects are referenced), DB2 chooses between the two connection types based on the bind option that you choose (or the default protocol set at your site).

Important: We recommend using DRDA if you are developing new applications and to migrate existing private protocol applications to DRDA, because no enhancements are planned for private protocol.

Refer to *DB2 UDB for z/OS V8 Administration Guide,* SC18-7413, for a complete description of DRDA and private protocol.

8.3.1 Thread and enclave relationship

Database access threads are created to access data at a DB2 server on behalf of a requester using either DRDA or DB2 private protocol. A database access thread is created when an SQL request is received from the requester. Allied threads perform work at a requesting DB2.

Database access threads have two modes of processing: ACTIVE MODE and INACTIVE MODE.

- ACTIVE MODE: A database access thread is always active from initial creation to termination.
- INACTIVE MODE: A database access thread can be active or pooled. (A database access thread that is not currently processing a unit of work is called a *pooled thread*, and it is disconnected.) When a database access thread in INACTIVE MODE is active, it processes requests from client connections within units of work. When a database access thread is pooled, it waits for the next request from a client to start a new unit of work.

Only when in INACTIVE MODE, a database access thread is terminated after it has processed 200 units of work or after the database access thread has been idle in the pool for a specified amount of time.

Note: For every database access thread, an independent enclave is created. The way DDF uses enclaves relates directly to whether the DDF thread can become inactive.

Refer to *DB2 UDB for z/OS V8 Administration Guide,* SC18-7413, for a complete description of DB2 threads in a distributed environment.

8.3.2 Duration of an enclave

The duration of an enclave differs between Active and Inactive mode; for this reason, the data reported in the SMF 72 record depends on the type of thread:

- INACTIVE MODE threads are treated differently. If the thread is always active, the duration of the thread is the duration of the enclave. When the thread is pooled, such as during think time, it is not using an enclave. In this case, the SMF 72 record does not report inactive periods.

- ACTIVE MODE threads are treated as a single enclave from the time they are created until the time they are terminated. Therefore, the entire life of the database access thread is reported in the SMF 72 record, regardless of whether SQL work is actually processed.

Figure 8-1 contrasts the two types of threads.

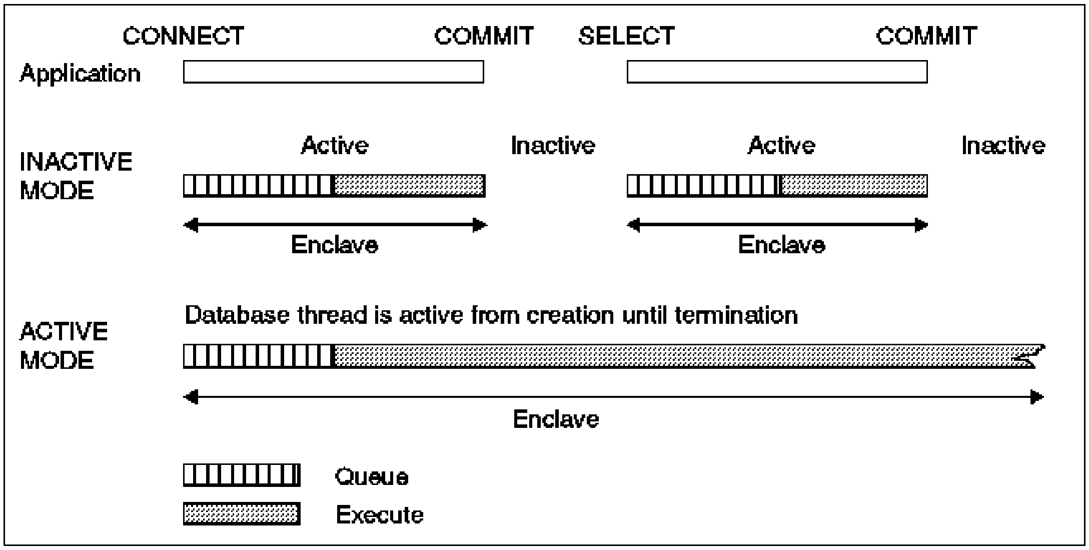


Figure 8-1 Contrasting ACTIVE MODE threads and POOLED MODE threads

8.3.3 DDF enclave creation

One DB2 request (transaction) is received by DDF from the network. DDF creates an enclave to process the SQL request using a DB2 thread. The thread executes DB2 code in cross-memory mode under the enclave priority. If data is missing, synchronous I/O requests are started during SQL processing. When data is retrieved, I/O interrupt processing is done by the DB2 DBM1 address space and control is returned to the DDF DB2 thread under the enclave priority. The SQL result is sent to the requester and the enclave is deleted as soon as the transaction ends.

The activity flow is shown in Figure 8-2 on page 238.

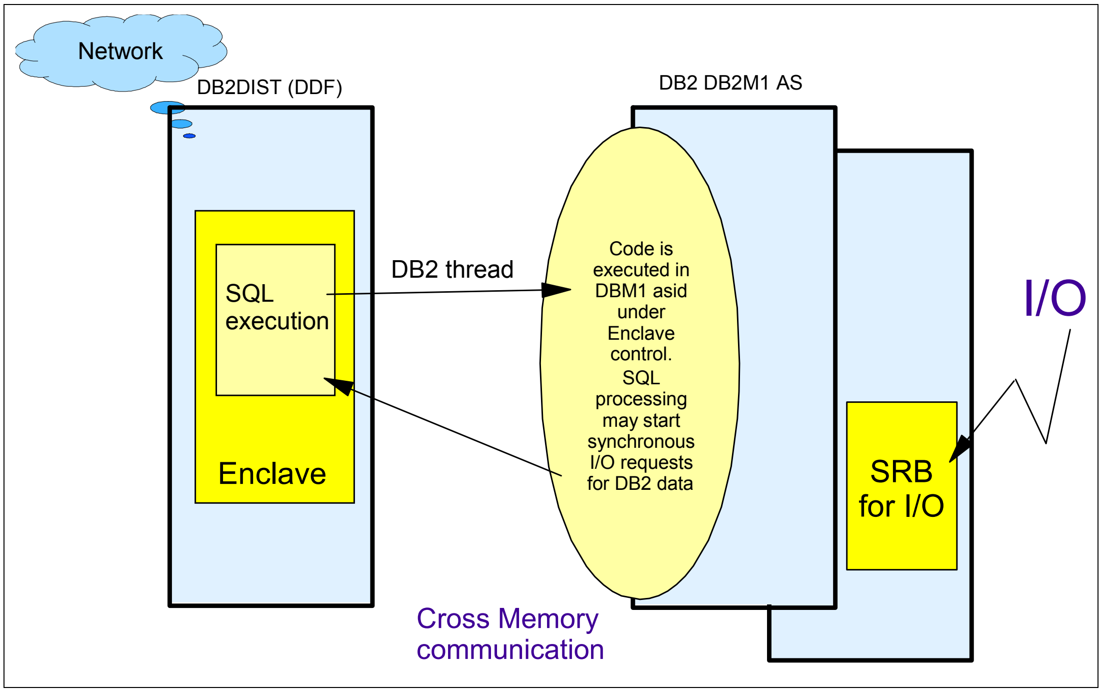

Network
DB2DIST (DDF)
DB2 DB2M1 AS
SQL execution
Enclave
DB2 thread
Code is executed in DBM1 asid under Enclave control. SQL processing may start synchronous I/O requests for DB2 data
I/O
SRB for I/O
Cross Memory communication

Figure 8-2 DDF independent enclave creation

8.3.4 Classification rules for a distributed environment

DDF receives requests from different clients: DB2 and other vendors' software. Each request becomes an enclave. You need to classify each request, because if you do not classify your DDF transactions into service classes, they are assigned to the default class with a discretionary goal.

You can have multiple filters to classify DDF threads, for example, their authorization ID, package name, user ID, or plan name. For a complete list, refer to *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602. Example 8-1 shows how to classify the remote requests.

Example 8-1 Classification rules for a distributed environment

```
                Modify Rules for the Subsystem Type     Row 1 to 14 of 14
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : DDF         Fold qualifier names?   Y  (Y or N)
Description  . . . DRDA Distributed Transactions

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                 Service     Report
                                         DEFAULTS: SCDB2LOW    ________
 ____  1  SI         DB8E     ___                   SCDB2MED    ________
 ____  2    UI         GIAN     ___                 SCDB2HI     RCGIAN_
 ____  2    PN         DIST*    ___                 SCDB2LOW    RCDIST__
 ____  1  SI         DB2B     ___                   SCDB2MED
 ____  2    UI         ANNA     ___                 SCDB2HI     RCANNA__
 ____  2    AI         PAYROLL  ___                 SCDB2HI     RCPAY___
```

We recommend that you use inactive threads with a two-period service class where the response time and the duration for the first period are defined in such a way that 80-90% of the transactions complete in the first period.

In Example 8-1 on page 238, the requests are handled by subsystem instance `DB8E` and are assigned by default to service class `SCDB2MED`. If the request comes from user ID `Gian`, it is associated to service class `SCDB2HI`, while the request with a plan name starting with `DIST*` is assigned to service class `SCDB2LOW`. Requests handled by subsystem instance `DB2B` are assigned by default to service class `SCDB2MED`. If the request comes from user ID `Anna`, it will be associated to service class `SCDB2HI`, while the request with accounting ID `PAYROLL` is assigned to service class `SCDB2HI`.

Note: For a detailed description and samples of how to classify DB2 Stored Procedures, refer to *DB2 for z/OS Stored Procedures: Through the Call and Beyond*, SG24-7083.

Considerations for JDBC requests

JDBC™ sends its requests to DB2 using Driver Type-2 or Driver Type-4.

The work of Type-2 comes from WebSphere Application Server to DB2 and it is classified under CB subsystem type. You are not required to classify it under DDF subsystem type.

For the Type-4/DDF workload, the DDF rules allow us to do classification. Not all of the DDF qualifiers are usable; for example, all the JDBC applications use the same packages so it is not a useful filter. If you want to base prioritization on application behavior, you can classify based upon the first stored procedure called in a transaction, but it is only usable if you call a stored procedure.

A simple approach is to use a different data source within WebSphere Application Server for each classification and programmatically use a specific data source. Each data source has either a different AUTHID associated with it or a JDBC driver collection. You can classify based upon either of those.

The only other approach is to use the DB2 client strings. The *client strings* are text attributes associated with the connection, which we can use for classification of the workload. The client strings can be set as a part of the data source definition or alternatively you can set them programmatically within the application so that every transaction can have a different value. We can use the client strings for workload classification, and we can also use them for end-to-end auditing (the fields are written in DB2 accounting reports). The recommended client strings are application name and user ID.

8.3.5 Types of goal

You can establish performance periods for DDF threads so that you can cause the thread’s performance objectives to change based upon the thread’s processor consumption. Thus, a long-running unit of work can move down the priority order and let short-running transactions get in and out at a higher priority.

To design performance strategies for these threads, take into account the events that cause a DDF thread to reset its performance period. The performance period is reset by terminating the enclave for the thread and creating a new enclave for the thread, as described in 8.3.2, “Duration of an enclave” on page 236.

Because threads that are always active do not terminate the enclave and thus do not reset the performance period to the first period, a long-running thread always ends up in the last performance period. Any new business units of work that use that thread will suffer the performance consequences. This makes performance periods unattractive for long-running threads. For always active threads, therefore, use velocity goals and use a single-period service class.

In Active mode, the enclave is classified when it starts. If the enclave is reused by different work with different goals, all work is assigned to the goal of the enclave; that is, it is classified using the characteristic of the first work joining the enclave.

You need to know how your distributed environment is configured before you define the goals.

8.3.6 Best practices

Evaluate filtering the remote requests using Subsystem Instance rule as the level 1 rule in order to separate different DB2 environments (for example, test and production). Then use the other available filters for the subsequent levels of classifications. Example 8-1 on page 238 implements this suggestion.

Because DDF receives requests of work from several clients that could also use dynamic queries, it is not easy to foresee peaks of workload introduced by unexpected long-running transactions. A way to circumvent this situation is to divide the known workloads in groups based on the types of query (static or dynamic). Then, for transactions using dynamic queries (more difficult to handle), assign a service class with a Resource Group specifying a MAX threshold in order to limit the CPU consumption.

For a better analysis of your workload, consider using report classes in your DDF classification rules in order to have the most granularity in your RMF reports.

8.3.7 Enclave reporting

The two most frequently used SMF records are types 30 and 72. The type 30 record contains resource consumption at the address space level. You can pull out total enclave usage from the record, but you must use DB2 accounting traces to see resource consumption for a particular enclave.

Each enclave reports its data to one type 72 record for the service class and to a type 72 record for the report class if specified in the service policy. You can use Workload Manager (WLM) classification rules to separate different enclaves into different service or report classes. Separating the enclaves in this way enables you to understand the DDF work better.

There is no separated SMF record for enclave except for multi-system enclave.

Table 8-3 shows the characteristics and accounting for each type of enclave.

Table 8-3 Enclave accounting

	Independent	**Dependent**	**Foreign**
Dispatchable unit	SRB/TCB	SRB/TCB	SRB/TCB
New transaction	Yes	No	No
Owner	Home address space at time of IWM4ECRE	Home address space at time of IWM4ECRE	Home address space at time of IWM4ECRE
Server	Address space where enclave work is dispatched	Address space where enclave work is dispatched	Address space where enclave work is dispatched

	Independent	Dependent	Foreign
Service class	Assigned based on WLM classification rules	Same as owner	Same as original enclave
CPU time	Owner’s SMF30CPT (total) owner’s SMF30ENC (for independent only)	Owner’s SMF30CPT (total) owner’s SMF30DET (for dependent only)	Owner’s SMF30MRI (for foreign indep.) owner’s SMF30MRD (for foreign dep.)
CPU service by address space	Owner’s SMF30CSU (total) owner’s SMF30ESU (independent only)	owner’s SMF30CSU (total)	CPU time / SMF30MRA / 256 * CPU coeff. (coeff. obtained by SMF72 record)
CPU service by period	Enclave’s R723CCPU or SMF72CTS	Enclave’s R723CCPU or SMF72CTS	Enclave’s R723CCPU or SMF72CTS
IOC service	Server’s SMF30 and SMF72 records	Server’s SMF30 and SMF72 records	Server’s SMF30 and SMF72 records
SRB service (nonpreemptible)	N/A	N/A	N/A
MSO service	N/A	Owner’s SMF30MSO, based on owner’s frame count	N/A

The RMF Workload Activity report summarizes resource consumption depending on the type of the enclave:

- Independent enclave:
  - CPU is accounted to the enclave service class.
  - I/O and MSO are accounted to the server address space service class.
- Dependent enclave:
  - CPU, MSO, and I/O are accounted to the server address space service class.

RMF Workload Activity report

Example 8-2 shows an RMF Report for a distributed data facility (DDF) service class:

- All storage-reported fields are zeros.
- In the Transactions section, you can see all of the information related to the enclaves: The average number of enclaves and how many ended in the interval.
- Only CPU Service Units are accounted and they should be accounted as TCB. The fields SRB, RCT, IIT, and HSt should be zeros.

Example 8-2 RMF post-processor for an independent enclave (DDF)

```
REPORT BY: POLICY=FIRSTPOL   WORKLOAD=DDF        SERVICE CLASS=DDFPRU     RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE RATES--   PAGE-IN RATES    ----STORAGE----
   AVG     56.19   ACTUAL               4.522   SSCHRT    9.0   IOC          0    ABSRPTN       126   SINGLE     0.0   AVG        0.00
   MPL     56.19   EXECUTION            4.522   RESP      3.6   CPU     12763K    TRX SERV      126   BLOCK      0.0   TOTAL      0.00
   ENDED    7229   QUEUED                   0   CONN      0.5   MSO          0    TCB         126.6   SHARED     0.0   CENTRAL    0.00
   END/S    4.02   R/S AFFINITY             0   DISC      2.8   SRB          0    SRB           0.0   HSP        0.0   EXPAND     0.00
   £SWAPS      0   INELIGIBLE               0   Q+PEND    0.3   TOT     12763K    RCT           0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION               0   IOSQ      0.0   /SEC      7091    IIT           0.0   EXP SNGL   0.0   SHARED     0.00
```

```
AVG ENC 56.19  STD DEV          1.50.515                                 HST          0.0   EXP BLK    0.0
REM ENC  0.00                                                            APPL %       7.0   EXP SHR    0.0
MS ENC   0.00
```

Enclave SDSF reporting

Example 8-3 is produced by the SDSF ENC command.

Example 8-3 SDSF enclave reporting (dependent enclave)

```
Display  Filter  View  Print  Options  Help
 -----------------------------------------------------------------------------
 SDSF ENCLAVE DISPLAY  SC69     ALL                     LINE 1-9 (9)
 COMMAND INPUT ===>                                            SCROLL ===> CSR
 NP   TOKEN            SSType Status   SrvClass OwnerAS Type Subsys   Workload
      600012A7CB       JES    ACTIVE   VEL95         38 DEP  JES2     OTHER
      540012A7CC       JES    ACTIVE   VEL95         40 DEP  JES2     OTHER
      480012A7CD       JES    ACTIVE   VEL95        107 DEP  JES2     OTHER
      200012A7CE       JES    ACTIVE   VEL95         98 DEP  JES2     OTHER
      680012A7C6       JES    ACTIVE   VEL95         55 DEP  JES2     OTHER
      380012A7C7       JES    ACTIVE   VEL95        100 DEP  JES2     OTHER
      340012A7C8       JES    ACTIVE   VEL95         28 DEP  JES2     OTHER
      280012A7C9       JES    ACTIVE   VEL95         39 DEP  JES2     OTHER
      4C0012A7CA       JES    ACTIVE   VEL95        109 DEP  JES2     OTHER
```

The TYPE Field shows on which kind of enclave we are working. In our example, the type is DEP.

The owning address space of the first enclave is 38.

8.4 DB2 Stored Procedures

In this section, we discuss the Stored Procedures DB2 facility and how it exploits WLM to manage the performance of individual client requests. Stored procedures is a special feature of DB2 and provides substantial improvements for distributed data access.

8.4.1 What are stored procedures

Note: The z/OS implementation of stored procedures changed from DB2 V4 to DB2 V7 and to DB2 V8.

The distributed environment brought a series of problems to modern data processing. One of these problems is the lack of central management for application installation and maintenance. Thousands of application copies are spread all over servers and client machines, making the task of keeping them consistent almost impossible.

Furthermore, these problems are not confined to a distributed database topology. For example, DB2 clients ask for data from DB2 servers in order to perform some processing against the data on the client workstation. To minimize such problems, the concept of DB2 Stored Procedures was introduced.

The DB2 Stored Procedures function consists of user-written structured query language (SQL) programs stored at a DB2 local or remote server. They can be invoked by a client application. This function enables a new kind of client/server application that has central management of the applications, with reduced traffic between the client and the server.

Local client applications, remote Distributed Relational Database Architecture (DRDA) applications, or remote data services (DB2 private protocol) can invoke stored procedures by issuing the SQL CALL statement. Figure 8-3 and Figure 8-4 on page 244 show examples of how to invoke DB2 Stored Procedures.

Stored procedures were introduced in DB2 for OS/390 Version 4.1. They are executed in a new address space, the stored procedure address space (ssnmSPAS).

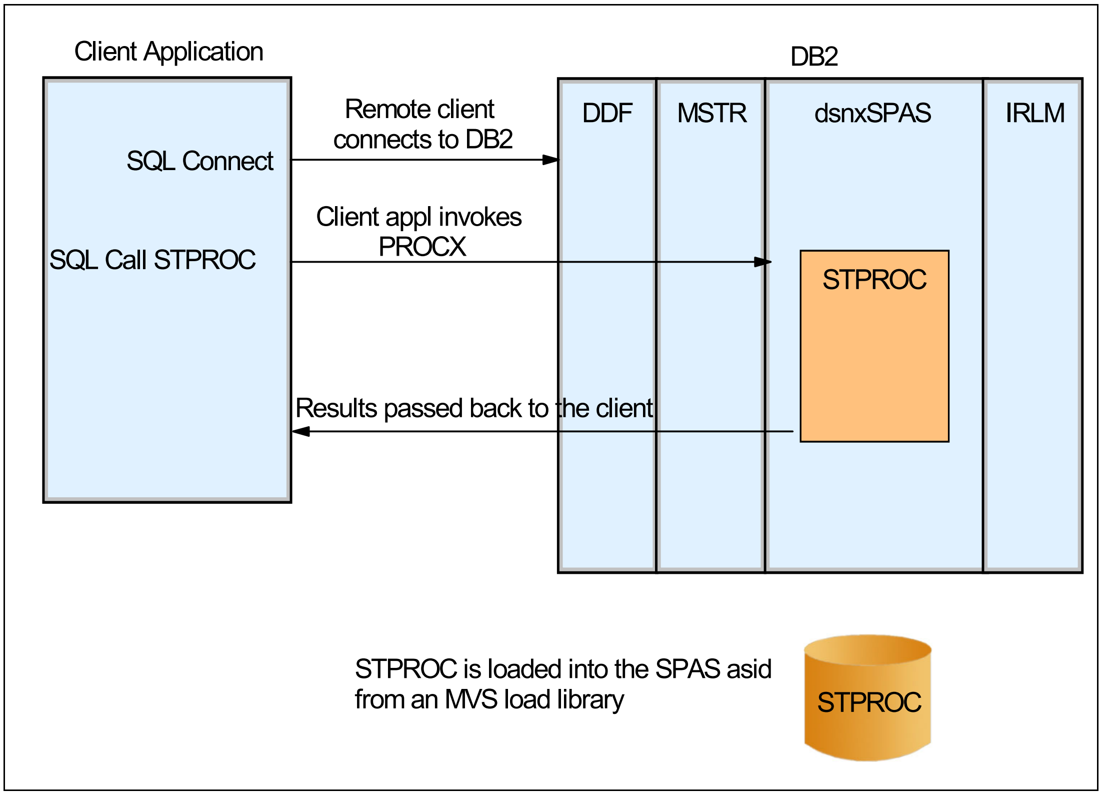

Client Application
DB2
SQL Connect
SQL Call STPROC
Remote client connects to DB2
Client appl invokes PROCX
Results passed back to the client
DDF
MSTR
dsnxSPAS
IRLM
STPROC
STPROC is loaded into the SPAS asid from an MVS load library
STPROC

Figure 8-3 Example of DB2-established stored procedures

In DB2 V4.1, you had a single stored procedures address space with multiple TCBs, each stored procedure running with its own TCB. You define the maximum concurrent TCBs in the DB2 Stored Procedure Parameter Panel (DSNTIPX) of the DB2 installation CLIST DSNTINST.

With DB2 V5 and the WLM enhancements in OS/390 Release 3, the stored procedures environment has been enhanced. With the extended enclave support in OS/390 Release 3, enclaves also support TCBs. Thus, WLM can assign dispatching priorities to stored procedure transactions based on the service class goals.

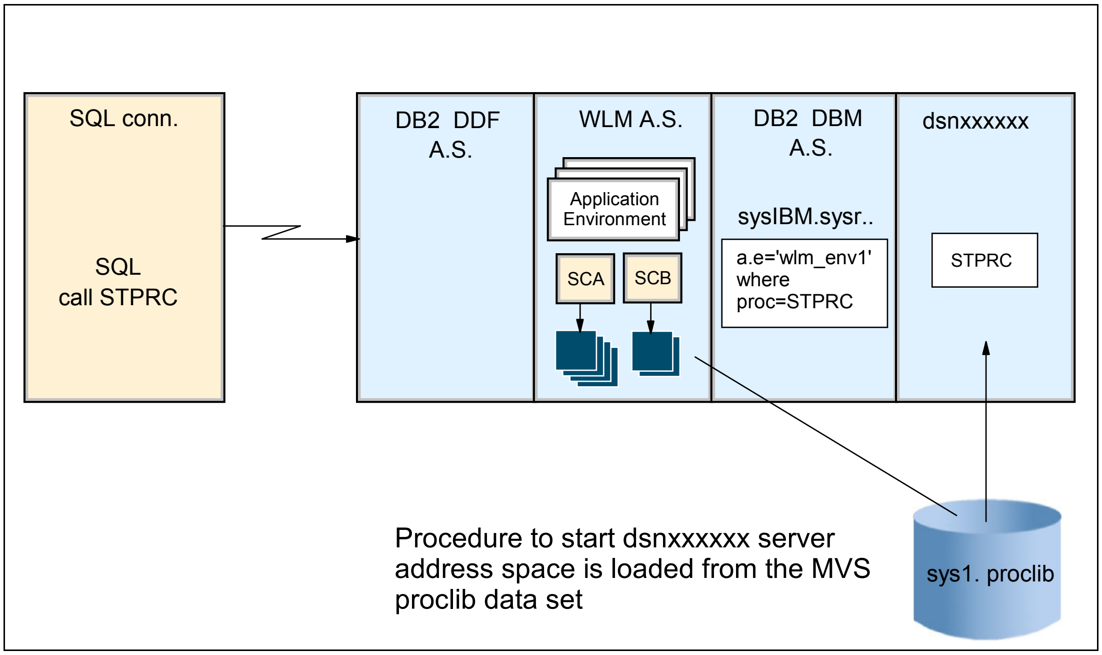


Figure 8-4 Example of WLM-established stored procedures

Now, you also have a choice between a single DB2-established address space or multiple WLM-established address spaces. Multiple DB2 address spaces for stored procedures with multiple tasks at each address space allow clients to run more concurrent transactions compared to a single address space. The multiple address spaces also provide improved program isolation for stored procedures.

Important: Only those stored procedures that were created in a release of DB2 prior to Version 8 can be run in a DB2-established address space. We do not recommend that you support this type of address space.

Table 8-4 on page 245 summarizes differences between stored procedures that run in WLM-established stored procedures address spaces and those that run in a DB2-established stored procedures address space.

Table 8-4 Difference between DB2 and WLM-established stored procedure address space

DB2-established	WLM-established
Use a single address space for stored procedures.	Use many address spaces for stored procedures and user-defined functions.
Incoming requests for stored procedures are handled in a first-in, first-out order.	Requests are handled in priority order.
Stored procedures run at the priority of the stored procedures address space.	Stored procedures inherit the z/OS dispatching priority of the DB2 thread that issues the CALL statement. User-defined functions inherit the priority of the DB2 thread that invoked the procedure.
No ability to customize the environment.	Each address space is associated with a WLM Application Environment that you specify. An Application Environment is an attribute that you associate on the CREATE statement for the function or procedure. The environment determines which JCL procedure is used to run a particular stored procedure.

For further discussion of DB2-established and WLM-established stored procedure address spaces, see *DB2 UDB for z/OS V8 Administration Guide,* SC18-7413.

8.4.2 WLM considerations for DB2 Stored Procedures

WLM is involved with DB2 Stored Procedures in three ways:

- It enforces goals to the DB2 Stored Procedures enclaves. These enclaves also support TCBs.
- It controls the number of DB2 Stored Procedure address spaces automatically, based on your service class goals through the following process:
  - First request comes in:
    - DDF creates an enclave to represent this request, assigning the service class according to the classification rules.
    - Now DBM1 recognizes that a STP request is included and inserts the request to the corresponding Application Environment.
    - WLM now creates a work queue for the Application Environment and the service class and queues the request here.
    - WLM starts the first server for this combination of Application Environment and work queue unconditionally. This server address space is always started and it is not based on any calculations for demand.
  - More requests for the Application Environment arriving:
    - If they all belong to the same service class, they are queued to the work queue that has already been created. If not, WLM starts additional server address spaces based on demand goal achievement and other considerations.
    - If a request for a new service class (same Application Environment) is detected, a new work queue is created and the first server address space will be started unconditionally.
    - The same is happening if a request for a different Application Environment arrives.

  - If the demand for a service class (Application Environment/work queue) drops:
    - Excessive servers become unbound after about five minutes of inactivity, although you might assist up to 15 minutes or more depending on the activity peaks and other internal considerations.
    - Unbound first means the server address space disappears from the work queue. It is still available for the Application Environment for another five minutes in case new activity comes in, but eventually WLM stops it.
    - This process is repeated for all server address spaces except the last one, which is bound to a service class work queue. This becomes unbound after one hour of inactivity. After that, WLM forgets about the work queue, and if new requests come in, the process starts all over again.

  If your installation requires you to keep a server address space always open, the only thing your installation can do is to send a dummy request once every hour that prevents the last server address space from going away.

  There is no mechanism provided for customer intervention in this WLM process. You can either set no limit on the number of startable address spaces in the Application Environment definition (which includes the JCL start procedure name), or set the limit to a single address space for an Application Environment.

  You can also manually control the number of stored procedure address spaces using the MVS start command. You choose this mode by omitting the JCL procedure name in the Application Environment for the stored procedures.

- It controls the distribution of the stored procedures across the server address spaces. You can use WLM to dedicate address spaces to stored procedures of the highest business priority to attempt to ensure that there is always an address space available. You can do this by assigning work to different Application Environments. You can group related stored procedures in the same address space to isolate them from others associated with different business functions. You can also isolate stored procedures completely from each other by configuring an address space to run only one procedure at time by using the DB2 parameter NUMTCB=1.

Considerations for enforcing WLM goals

When you access a stored procedure from a local client application, you do not need to define a specific performance requirement. All such calls inherit the performance requirements from the calling address space, because there is no independent enclave associated with the execution of the stored procedure. The stored procedure call is a continuation of the original transaction.

You only need to define specific performance requirements for the independent enclave under the DDF subsystem type when you access the stored procedure remotely via the DDF address space.

8.4.3 WLM definition for DB2 Stored Procedures

This section describes the WLM definition using DB2 Stored Procedures.

Using WLM-established address spaces

If you use WLM-established stored procedures address spaces, you must define a WLM Application Environment. WLM uses these definitions for its server address space management.

There are other tasks that must be completed before a stored procedure can run in a WLM-established stored procedures address space. One of these tasks is to assign the stored procedures to an Application Environment. You do this by updating the WLM_ENV column of the SYSIBM.SYSPROCEDURES table in your DB2 subsystem. You can use the following statement, for example:

```
UPDATE SYSIBM.SYSPROCEDURES
SET APPLENV='WLMENV2'
WHERE PROCEDURE='BIGPROC'
```

For a description of the other DB2-related tasks you must complete before using stored procedures in a WLM-established address space, see *DB2 UDB for z/OS V8 Administration Guide,* SC18-7413*.*

To define the WLM Application Environment, you have to use the WLM administrative application. The names of the defined environments have to match the names used in the DB2 definition for the stored procedures.

Example 8-4 shows an example of how to define your WLM Application Environment.

Example 8-4 Application Environment definition for DB2 Stored Procedures

```
Application-Environment  Notes  Options  Help
--------------------------------------------------------------------------
                    Modify an Application Environment
Command ===> ______________________________________________________________

Application Environment Name . : WLMENV2
Description  . . . . . . . . . . Large stored procedure Env.
Subsystem Type . . . . . . . . . DB2
Procedure Name . . . . . . . . . DBC1WLM2
Start Parameters . . . . . . . . DB2SSN=DBC1,NUMTCB=40,APPLENV=WLMENV2
                                 ________________________________________
                                 ___________________________________

Limit on starting server address spaces for a subsystem instance:
1   1.  No limit
    2.  Single address space per system
    3.  Single address space per sysplex
```

DB2 Stored Procedures have to run on the same image as the DB2 instance that started them, since all stored procedure address spaces depend on the same subsystem instance (DBC1, in this example). DB2 Stored Procedures do not support shipping transactions to server address spaces on another system.

You have to select option 1 No Limit (the default) to allow multiple servers to be started for stored procedures. If, for testing purposes, you want to limit the address spaces WLM can start for an Application Environment, select option 2. Option 3 is not applicable for DB2 Stored Procedures.

You can define parameters that should be used to start the address spaces in the Start Parameter line. For DB2, the NUMTCB parameter controls how many TCBs can start in this address space. Because every stored procedure runs in its own TCB, this parameter controls how many stored procedures can run concurrently.

You can separate your stored procedures into multiple Application Environments. But be aware that unless a particular environment or service class is not used for a long time, WLM creates on demand at least one address space for each unique combination of WLM Application Environment and service class that is encountered in the workload. For example, if there are five Application Environments that each have six possible service classes, and all those combinations are in demand, it is possible to have 30 stored procedure address spaces.

Important: To prevent creating unnecessary address spaces, create only a relatively small number of WLM Application Environments and service classes.

WLM routes work to stored procedure address spaces based on the Application Environment name and service class associated with the stored procedure. The service class is assigned using the WLM classification rules. Stored procedures inherit the service class of the caller. There is no separate set of classification rules for stored procedures. For more information about classification rules, see “Define your classification rules” on page 250. For information about the address space manager function of WLM, see 1.2.6, “Application Environment” on page 13.

Figure 8-5 shows how DB2 Stored Procedures use the server address space management function of WLM.

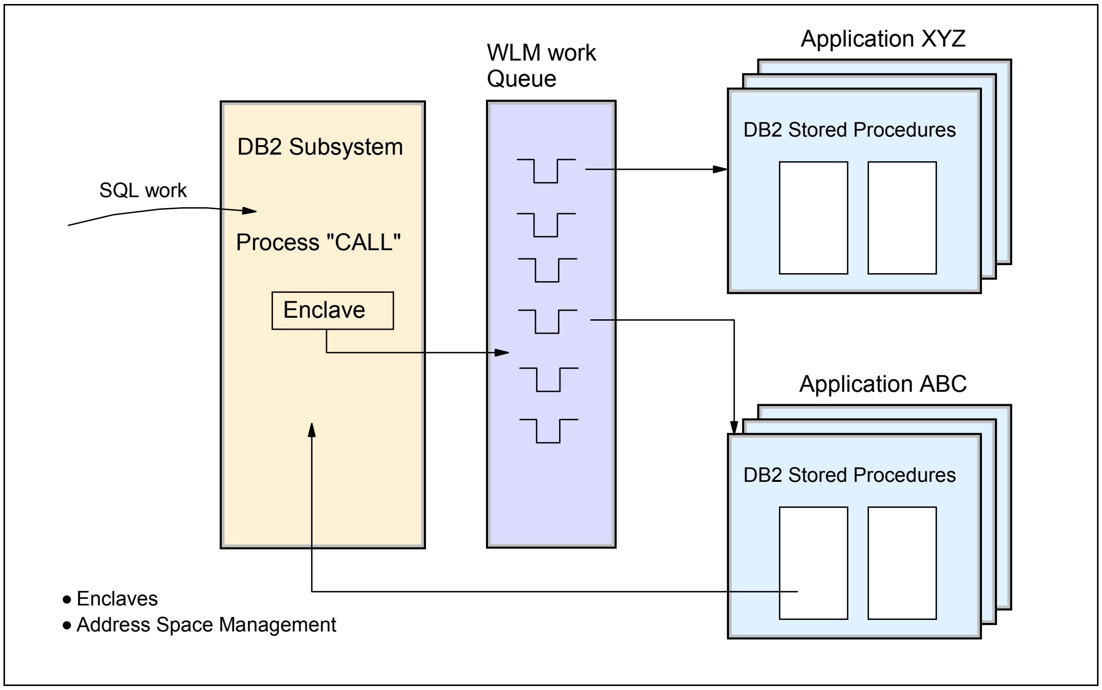


Figure 8-5 DB2 Stored Procedures and WLM

The steps are:

1. Before starting to process a transaction, DB2 needs to CONNECT to WLM. WLM then prepares the control blocks that anchor the Application Environments.
2. When a stored procedure call (SQL CALL) arrives, DB2 might create an enclave for the called stored procedure or use an existing one, depending upon the origin of the transaction:
   – If the origin is DDF, an independent enclave already exists, so DB2 uses this enclave for the stored procedure call.
   – If the origin is TSO, JES, or APPC, a dependent enclave is created that inherits the attributes of the originating address space and is considered a continuation of the originating transaction.

  - If the origin is CICS or IMS, a dependent enclave is created and the stored procedure call inherits the performance goal of the caller address space.

3. At this point, the enclave has an associated service class. There is also a check for the Application Environment associated with the transaction.
4. DB2 queues the request with an IWMQINS service call to a WLM-managed queue. WLM queues the request to the appropriate Application Environment queue. There is a queue for each unique combination of an Application Environment and service class.
5. If no address space serves this queue, a server address space has to be started. The server address space is started by the procedure defined in the Application Environment related to the called stored procedure. The main program is the shell program (DSNX9WLM). The shell connects to WLM using the IWMCONN service. Authorization is checked at this point.
6. A ready server address space serving the transaction environment issues an IWMSSEL service to select the request from the transaction environment queue.
7. Before passing control to the stored procedure, the shell invokes the begin execution processing by using the IWMSTBGN service of WLM. The TCB of the caller joins the enclave of the associated stored procedure. Resource consumption starts to be charged to the transaction or enclave at this point. If this is a dependent enclave, the resources are eventually charged to the originating address space.
8. The procedure ends and returns control to the shell, which invokes the end execution processing using the IWMSTEND service of WLM, and resource consumption ends. The result of the stored procedure is sent back to the originating client.

Figure 8-6 describes an example of a stored procedure called by a TSO user. In this case, a dependent enclave is created.

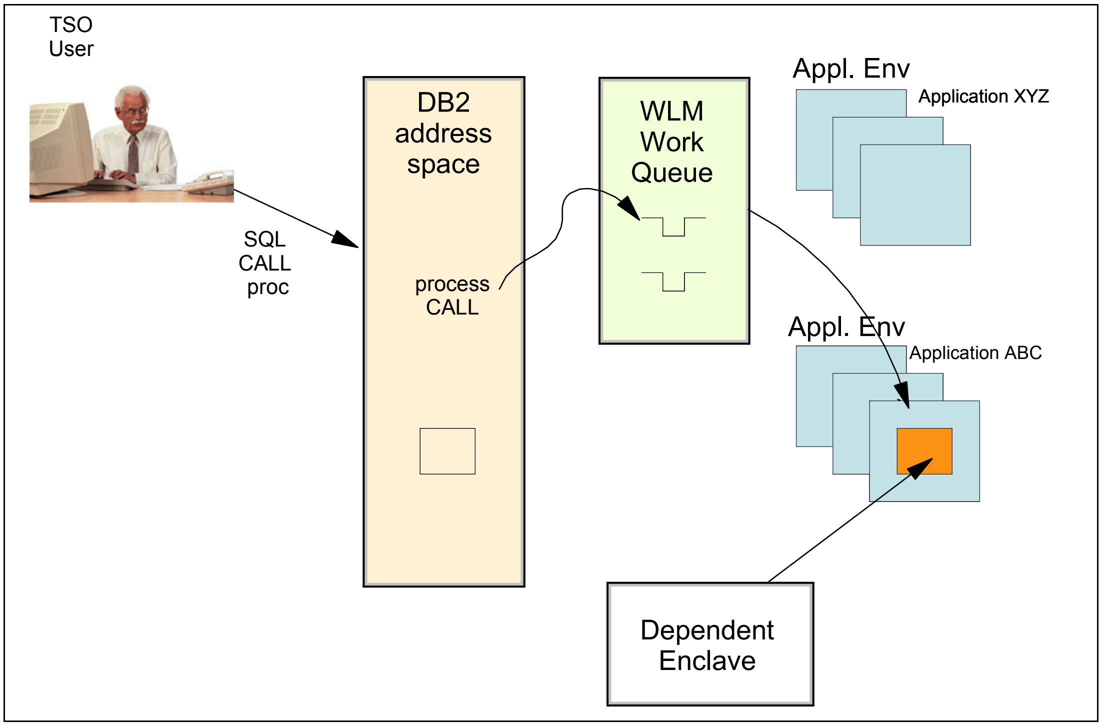


Figure 8-6 Stored procedure called by TSO user

Define your classification rules

You can define your own performance goals for a stored procedure *only* if you access the stored procedure remotely, because DDF creates an independent enclave for the incoming requests. All local calls inherit the performance attribute of the calling address space and are continuations of existing address space transactions (dependent enclave).

You have to define classification rules for the incoming DDF work.

> **Important:** If you do not define any classification rules for DDF requests, all enclaves get the default service class SYSOTHER. This is a default service class for low priority work, and it might not be what you want.

You have to define classification rules for SUBSYS=DDF using the possible work qualifiers (there are 11 applicable to DDF) to classify the DDF requests and assign them a service class.

> **Important:** You can classify DDF threads by, among other things, stored procedure name. But the stored procedure name is only used as a work qualifier if the first statement issued by the client after the CONNECT is an SQL CALL statement.

Other classification attributes are, for instance: account ID, user ID, and subsystem identifier of the DB2 subsystem instance, or LU name of the client application. For a description of all DB2 DDF work qualifiers and DDF threads classification, see *DB2 UDB for z/OS V8 Administration Guide,* SC18-7413.

Example 8-5 shows an example of the classification of DDF threads.

Example 8-5 Classification rules for stored procedures invoked from remote clients

```
. Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                Modify Rules for the Subsystem Type       Row 1 to 6 of 6
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : DDF        Fold qualifier names?   Y  (Y or N)
 Description  . . . Distributed DB2

 Action codes:  A=After    C=Copy        M=Move     I=Insert rule
                B=Before   D=Delete row  R=Repeat   IS=Insert Sub-rule
           -------Qualifier-------------            -------Class--------
 Action    Type       Name     Start                Service     Report
                                         DEFAULTS: DB2BATCH    RDB2
  ____  1  SI         DBC1     ___                  DDFPROD     ________
  ____  2    UI         SYSADM   ___                DDFHIGH     ________
  ____  2    PRC        PAYPROC  ___                PAYROLL     ________
  ____  2    LU         DDFL%%%  ___                DDFMED      ________
  ____  1  SI         DBCT     ___                  DDFTEST     ________
  ____  2    PRC_       PAYPROC ___                 PAYROLLT    ________
```

The first level subsystem instance (SI) qualifier specifies the DB2 subsystem the DDF work is for. The default service class in the DB2 subsystem DBC1 is DDFPROD. The SYSADM user gets a service class of DDFHIGH, and the client request coming from logical units (LU) starting with DDFL gets a service class of DDFMED. The stored procedure PAYPROC gets the service class PAYROLL in the production system DBC1, and it gets the service class PAYROLLT in the test system DBCT. All other users in the production system DBCT have the service class DDFTEST.

Defining your service classes

Important: Remember that WLM starts an address space for every unique combination of Application Environment and service class, so do not create too many service classes.

You have to define your service goals using the WLM administration application. You can define response time, velocity, and discretionary goals for stored procedures or other DDF threads in line with your business requirements.

Use execution velocity percent if you have fewer than ten ending transactions over a twenty-minute period. Use response time if there are sufficient completions, mainly for the first and second periods. The response time goal can be determined from an SLA or from RMF reports. Stored procedures enclaves do not present queue time, so the response time is equal to the service time. Use a discretionary goal for the third period.

Use period aging to protect the system against long-running (high-consuming) stored procedures and to let short-running transactions get in and out at a higher priority. Define a duration (DUR) that forces about 80% of the queries to finish in the first period (80/20 rule: 80% of the transactions use 20% of the resources). The first period is started when the enclave begins.

Performance goals for the server address spaces

The MVS performance objective of the DDF address space or the WLM-established stored procedures address spaces does not govern the performance objective of the user thread. Assign the DDF address space and WLM-established stored procedures address spaces to an MVS performance objective that is similar to the DB2 database services address space (ssnmDBM1).

The MVS performance objective of the DDF and stored procedures address spaces determine how quickly DB2 is able to perform operations associated with managing the distributed DB2 workload, such as adding new users or removing users who have terminated their connections.

8.5 DB2 sysplex query parallelism

In this section, we describe how sysplex query parallelism exploits the function of WLM.

8.5.1 Query parallelism

Query parallelism is a way of reducing the elapsed time for long-running queries that can either be I/O intensive or CP intensive. An I/O-intensive query scans large volumes of data and requires considerable I/O processing and minimal CPU processing. A CP-intensive query usually performs sorts and can have query functions such as joins on multiple tables, column and scalar functions, and grouping and ordering of data.

DB2 V5 has introduced sysplex query parallelism, which extends parallel processing to allow a single query to use all of the processing power of a data sharing group in a DB2 data sharing environment. Sysplex query parallelism now makes the Parallel Sysplex attractive as a complex query environment as well.

8.5.2 How sysplex query parallelism works

DB2 data sharing in a Parallel Sysplex configuration is a requirement for sysplex query parallelism.

If a query qualifies for parallel processing, DB2 determines the optimal degree of parallelism at bind or prepare time. DB2 can decide to distribute parts of the query for processing to other members in the data sharing group, or to process the query within the originating subsystem only. The distribution of query parts to another DB2 subsystem is done using XCF services.

Different DB2 members process different ranges of the data, and the results are returned to the application issuing the initial SQL request statement.

The original query must execute on a member of a data sharing group that has the COORDINATOR parameter set to YES in the DB2 installation dialog.

The COORDINATOR subsystem parameter controls whether this DB2 can send parallel tasks out to other DB2s in the data sharing group. If the COORDINATOR parameter is not YES at runtime, the query runs within a single DB2.

To be considered as candidates for assisting the parallelism coordinator, the other members must have YES specified in the ASSISTANT field of DB2 installation dialog. The ASSISTANT subsystem parameter controls whether this DB2 can receive parallel tasks from another DB2 in the data sharing group. If this DB2 is the coordinator for a particular query, then its ASSISTANT parameter is irrelevant.

A parallel query must start on a coordinator DB2 subsystem and can be spread over many assistant DB2 subsystems, or run only in the originated subsystem, depending on DB2 decisions and definitions. A DB2 subsystem can be the coordinator for one query and the assistant for an other query.

For optimal use of processor resources, run only one DB2 data sharing member at a time on a given CPC.

Figure 8-7 on page 253 shows all members of a data sharing group participating in processing a single query.

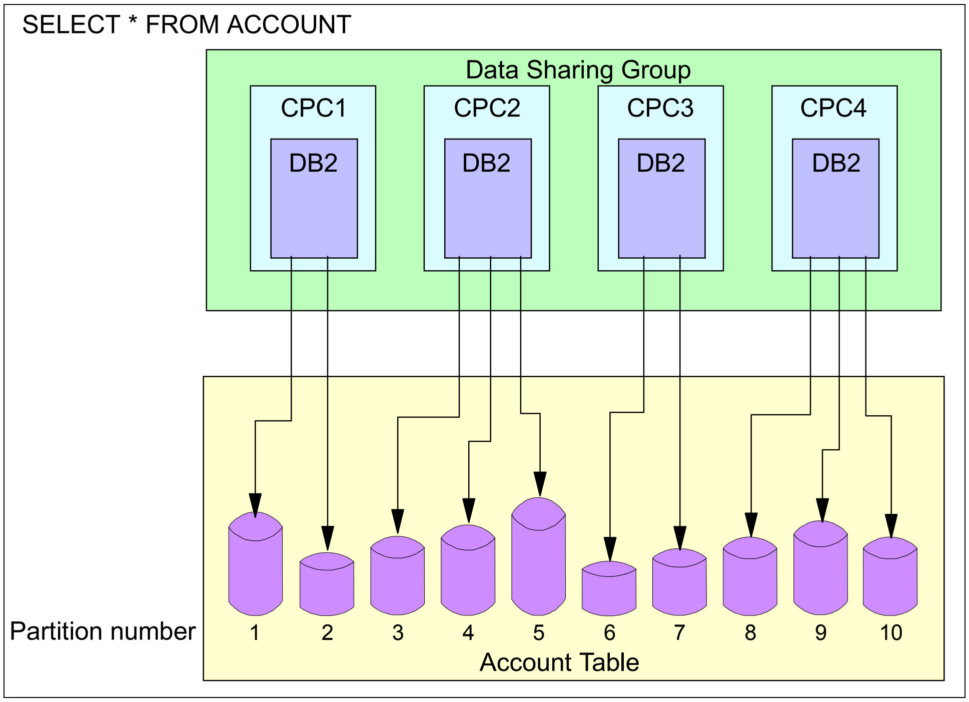


Figure 8-7 Parallel query processing

Different DB2 members are processing different partitions of the data.

This is a simplification of the concept; in reality, several DB2s can access the same physical partition. To take full advantage of parallelism, use partitioned table spaces.

For more information about sysplex query parallelism and how to implement it, see *DB2 UDB for z/OS V8 Data Sharing: Planning and Administration,* SC18-7417.

8.5.3 WLM considerations for sysplex query parallelism

It is important to define how you want MVS to handle the work for sysplex query parallelism.

You need to define group-wide goals for workload management for both work that originates on a particular DB2 and for work that is processed by that DB2 on behalf of another.

The task of classifying work that runs on the parallelism coordinator is the same as without parallelism. However, you must also classify work that runs on the assistant DB2. If you do not do this, the part of the query that runs on the assistant is, by default, discretionary work. Because the enclave created by the assistant DB2 has no associated service class, it gets the SYSOTHER service class. This has an assigned discretionary goal. Discretionary work runs at the priority usually reserved for very low-priority batch work.

DB2 in conjunction with WLM can assign query requests on a very granular level using the WLM classification rules. You can assign priority to your query at a user level, plan level, or package level.

You can define response time, velocity, or discretionary goals. You can define period aging to facilitate the flow of short-running and long-running work in the system. Consider also the use of Resource Groups in order to limit the amount of resources that a “runaway” or “hog” query might consume.

The performance philosophy behind the use of period aging is to give resources up-front to new work coming into the system in order to get the work completed. Longer-running queries fall through multiple service class periods and get fewer resources the longer they run. This automatically controls the priority of short-running and long-running work in the system, without the need for identifying the two categories of work. A good design goal for query workload is to keep short-running queries short and prevent longer-running queries from monopolizing system resources.

Your query workload (decision support, business intelligence, and so on) can run concurrently with your other online workload without affecting the performance of your other business-critical workloads.

8.5.4 WLM definition for sysplex query parallelism

As mentioned, the task of classifying the queries running on the parallelism *coordinator* DB2 subsystem is the same as without parallelism. The queries are associated with a service class according to the subsystem type of the originator of the query (TSO, CICS, JES, or DDF).

In addition to defining a service class for the coordinator, you must also define and assign a service class for the assistant enclave. If this enclave does not have a service class of its own, it gets the default service class SYSOTHER. This service class has a discretionary goal, that is, a goal reserved for very low-priority work.

General recommendations for the implementation are:

- Create classification rules and service class for the coordinator query.

  The pieces of the query that run in the coordinator DB2 are classified under the subsystem type of the originator of the query (for example, CICS, TSO, JES, or DDF). Refer to the specific subsystem type for guidance about how to classify such a workload.

- Create classification rules for the assistant enclave query.

  You have to define classification rules under the DB2 subsystem type, using any of the possible work qualifiers, to point to a service class. The qualifiers that are valid for a particular workload depend on where it originates. For example, if the query workload originates from TSO users, then the TSO qualifiers can be used for the assistant enclave. Ask for assistance from your DB2 database administrator to select the most appropriate work qualifiers.

- Create service classes for the assistant enclave query.

  Define one or more service classes representing the goal for your split-up enclave queries. We suggest that you use the same service classes that the coordinator uses, but if you want, you can define different service classes for the assistant.

Classification rules

You categorize your workload into service classes using a classification rule. The categorization is based on work qualifiers that identify a work request to the system. The first qualifier is the subsystem that receives the work request. This is very important to know when defining the classification rules for query parallelism.

The piece of a split-up query that runs locally, that is, the piece that runs on the coordinator, is classified according to the subsystem type of the originator of the query (for example, TSO, JES, or DDF). Only the queries that DB2 has created by splitting a single, larger query and distributing them to the assistant DB2s are classified under the DB2 subsystem type.

Because of this, you have to define classification rules for the coordinator and for the assistants. These rules can be different and can also use different goals.

Example 8-6 and Example 8-7 show examples of how to define your WLM classification definitions for your sysplex query parallelism workload. One definition is for the coordinator where the query originates locally from TSO, and the other definition is for the assistants where the distributed parts of the query run within the data sharing group.

Example 8-6 WLM classification rules for coordinator

```
Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type        Row 1 to 3 of 3
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : TSO        Fold qualifier names?   Y  (Y or N)
 Description  . . . Decision Support- Coordinator

 Action codes:  A=After    C=Copy         M=Move     I=Insert rule
                B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
                                                                More ===>
         -------Qualifier-------------            -------Class--------
 Action    Type       Name     Start                Service     Report
                                          DEFAULTS: DSLOW       RDSDEF
  ____  1  UI         CEO%%    ___                  DSHIGH      RDSHIGH
  ____  1  UI         USER%%   ___                  DSMED       RDSMED
  ____  2    AI__       D10*     ___                DSLOW       RDSLOW__
```

Example 8-6 shows the classification for the coordinator queries that originate from TSO. WLM associates work from user IDs that start with CEO to service class DSHIGH, and work from user IDs starting with USER to service class DSMED. If a user ID USER*xx* has an account ID starting with D10, then this work is associated with service class DSLOW. Requests that arrive from user IDs that do not match one of these rules are assigned to the default service class DSLOW.

Example 8-7 WLM classification rules for the assistant DB2 subsystems

```
Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type        Row 1 to 4 of 4
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : DB2        Fold qualifier names?   Y  (Y or N)
 Description  . . . Sysplex Query Assistants

 Action codes:  A=After    C=Copy         M=Move     I=Insert rule
                B=Before   D=Delete row   R=Repeat   IS=Insert Sub-rule
                                                                More ===>
         -------Qualifier-------------            -------Class--------
 Action    Type       Name     Start                Service     Report
                                          DEFAULTS: DSLOW       ________
  ____  1  SI         TSO      ___                  DSDISC      ________
  ____  2    UI         CEO%%    ___                DSHIGH      ________
  ____  2    UI         USER%%   ___                DSMED       ________
  ____  3      AI         D10*     ___              DSLOW       ________
```

The DB2 in the subsystem type is for all DB2s in the Parallel Sysplex where a split query can run under an assistant DB2. The subsystem instance (SI) level 1 classification rule classifies query work originating from other members of the data sharing group. In Example 8-7, the queries originated from TSO and were distributed to an assistant DB2. For the DB2 subsystem type, the qualifier SI corresponds to the transaction subsystem type on the coordinator DB2, in this case, TSO.

Because you define classification rules for different subsystems on the coordinator and the assistants, you also have a different set of work qualifiers. For a description of qualifiers supported by subsystems, see *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602.

Types of goals

Defining your workload goals depends on your business requirements and your workload characteristics. Therefore, while we cannot provide you with a definitive answer about how to define your goals, we give you useful advice regarding the goal for sysplex query parallelism.

You can define any of the three types of goals for your sysplex query parallelism work: Response time goals, execution velocity goals, or discretionary goals. You can also use performance aging in the form of service class periods.

The type and the numeric value for goals in the coordinator and assistants should be the same.

With regard to the type of goals, it is “business as usual”: Use execution velocity if you have fewer than ten ending transactions over a twenty-minute period. Use response time if there are sufficient completions to build up a good history for WLM to work with. The response time numeric value can be originated from an SLA or from RMF reports. For parallel query transactions, there is no queue time, so the response time is equal to the service time. Use a discretionary goal for the last period.

Important: Be aware that the response time for a query can vary widely depending on the degree of parallelism that determines the number of query parts. This might not always be predictable.

Use periods to protect the system against high-consuming, long-running queries. Use a duration (DUR) that forces about 80% of the queries to finish in the first period (80/20 rule: 80% of the transactions are using 20% of the resources).

With regard to period switching, be aware that with sysplex query parallelism, the query work for a particular query always starts out in the first period on each assistant with no accumulated service units. Use the RMF reports and DB2 performance monitor reports to examine the typical degree of parallelism and adjust the durations that you specify so that period switch occurs after the desired number of service units has been consumed.

Use a discretionary goal if all of your other workload is more important than your queries.

9

WebSphere Application Server workload considerations

This chapter provides a description of the WebSphere environment, a summary of the different elements that belong to this environment, and how you can classify them from a Workload Manager (WLM) perspective.

9.1 General considerations

Workload management on z/OS provides the following benefits to WebSphere Application Server applications. WLM:

- Balances server workloads, allowing processing tasks to be distributed according to the capacities of the different system images in the sysplex.
- Provides failover capability by redirecting client requests if one or more servers are unable to process them. This improves the availability of applications and administrative services.
- Enables systems to be scaled up to serve a higher client load than provided by a single system image configuration. With clustering, additional instances of servers can easily be added to the configuration.
- Enables servers to be transparently maintained and upgraded while applications remain available for users.
- Centralizes the administration of servers and other objects.

WebSphere for z/OS can use, along with WLM queuing manager services, WLM routing services and WLM sysplex routing services.

Workload management in z/OS is based on the concept of grouping work into service classes. The incoming work request is classified to a service class and the WLM schedules the resources to complete the work request according to this service class. Figure 9-1 shows how WebSphere work requests are classified into service classes.

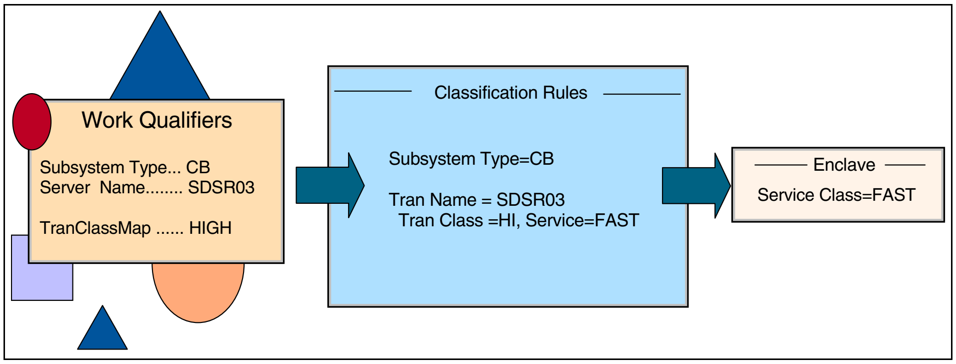


Figure 9-1 Classification of work requests

Figure 9-1 includes these components:

- Work qualifier: WebSphere for z/OS associates each work request with a work qualifier that identifies a work request to the system.

  The WebSphere application workload runs in subsystem CB.

  > **Note:** A CB subsystem is an IBM-supplied workload model for Component Broker-like workloads. It represents a set of CB objects that are grouped together and run in a logical server. It is used to manage transactions as WLM enclaves.

  The supported work qualifiers for subsystem CB are:

  CN Collection name, the server_generic_short_name for the WebSphere for z/OS Application server.

  SI Server instance name; this is the server_specific_short_name.

TC Transaction class assigned to the transaction using the transaction class mapping file for the server. This facility allows us to differentiate performance objectives between multiple transactions running on the same WebSphere server.

UI User name; the user name under which the work request is run.

- Classification rules: Classification rules associate a work request, as defined by its work identifier, to a WLM service class.
- Service class: The z/OS WLM organizes work into workloads and service classes. The service class for a group of work defines the performance goal and business importance:
  - Performance goals

    There are three kinds of goals: Response time, execution velocity, and discretionary. The response time goal indicates the response time for individual transactions, the execution velocity goals are suitable for started tasks or batch jobs, and the discretionary goals are for low priority work.

    A response time objective is usually consistent with the business requirement of a WebSphere application. Also, the response time option automatically generates response time distribution information to be reported through an RMF report. You might find this option useful when having to investigate response time issues.
  - Business importance of the work

    The business importance for a service class defines how important it is to achieve the performance goal for that service class. At runtime, the workload management component manages workload distribution and allocation of resources to competing workloads. High priority workloads get guaranteed consistent results, for example, response time, throughput, and so forth.

9.2 WebSphere on z/OS and WLM

WebSphere Application Server for z/OS V5 exploits WLM services to process incoming requests:

- Queuing services to manage the execution priority of address spaces and work requests.
- Enclave to manage performance of WebSphere transactions across multiple address spaces.
- Application Environments to dynamically manage the number of servant address spaces.

9.2.1 Queuing services

Queuing manager services manage execution of address spaces and the work requests they process to meet service class performance goals.

The classification of each transaction is managed by the controller region. The controller region acts as a queuing manager that queues work requests to workload management for execution in servant address spaces. Workload management maintains the queues for passing work requests from the controller region to each servant region.

The controller region listens for work requests and puts them on the Workload Management queue. The Workload Management component of z/OS dispatches the work to the servant region according to the WLM policy specified by the work identifier.

Figure 9-2 on page 260 illustrates the flow from the WebSphere controller region to the servant regions.

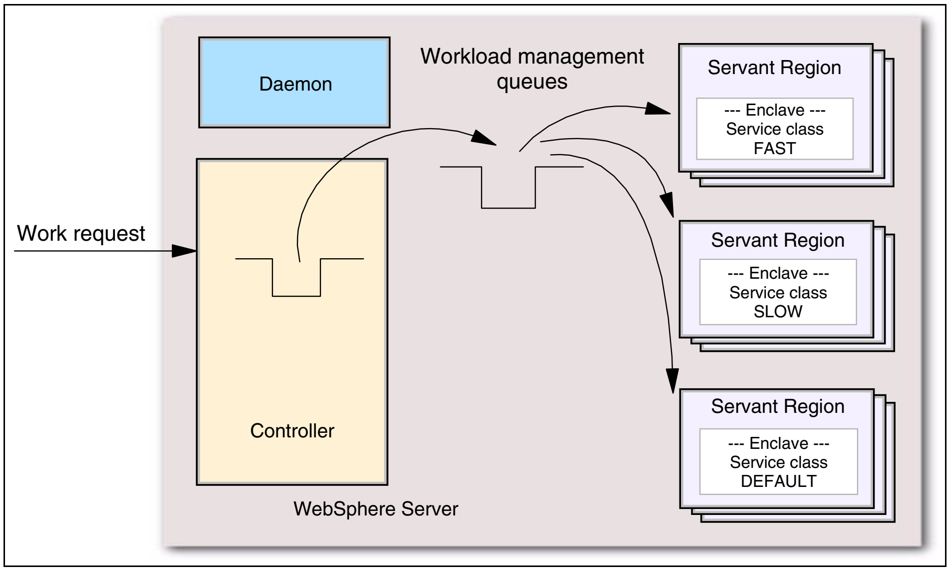


Figure 9-2 Interaction among WLM, controller, and servant regions

WLM temporal affinity

Temporal affinity is an extension to the WLM queue server manager interfaces that allow middleware to direct work requests to a specific server region (it is similar to CICS transaction affinity).

Temporal affinity enables WebSphere to create objects that live across multiple transactions that must be accessed by tasks that process transactions for the same client that are not persistent (therefore, temporal).

Existing WLM interfaces allow a WebSphere control region to queue work requests to a pool of server regions for a service class. The assumption is that a transaction starts with inserting the work requests and ends with leaving (potentially deleting) the enclave. There are cases where information must be preserved across multiple independent work requests. The information left behind lives only in the virtual storage of the address space. Follow-on work requests requiring this information must be directed to the server region that has this information.

The solution is that now the server region is able to obtain a region token at a select time (IWMSSEL). A new interface (IWMTAFF) allows the server or control region to mark the server region as needed by follow-on work requests. An extension of the insert mechanism allows the control region to direct a work request to a specific server region. WLM ensures that the server region stays alive until all temporal affinities have been removed.

SDSF displays information on the display active panel for queue server regions. See Example 9-1.

Example 9-1 SDSF display of temporal information

```
JOBNAME  U% Workload  SrvClass   SP    ResGroup       Server   Quiesce
ASAHIP   4   STCDEF   STC        1     TEMP-AFF       NO
```

If middleware, such as WebSphere, uses temporal affinity, then it is no longer possible to terminate a server region when such a temporal affinity exists. The server region uses a new interface IWMTAFF to tell WLM that it has an affinity to the client. It is also the responsibility of the server region to tell WLM that the affinity no longer exists.

9.2.2 Enclaves

Enclave services allow performance management of a transaction across multiple address spaces and systems inside a sysplex.

WebSphere Application Server for z/OS uses an independent enclave type that reports on resource consumption based on a performance goal associated to the transaction, unrelated to the performance goals of the address spaces in which the enclave dispatchable units execute.

The controller region creates the enclave and associates the transaction to this classified enclave. Then the transaction is queued, waiting to be served by an available thread in a servant region.

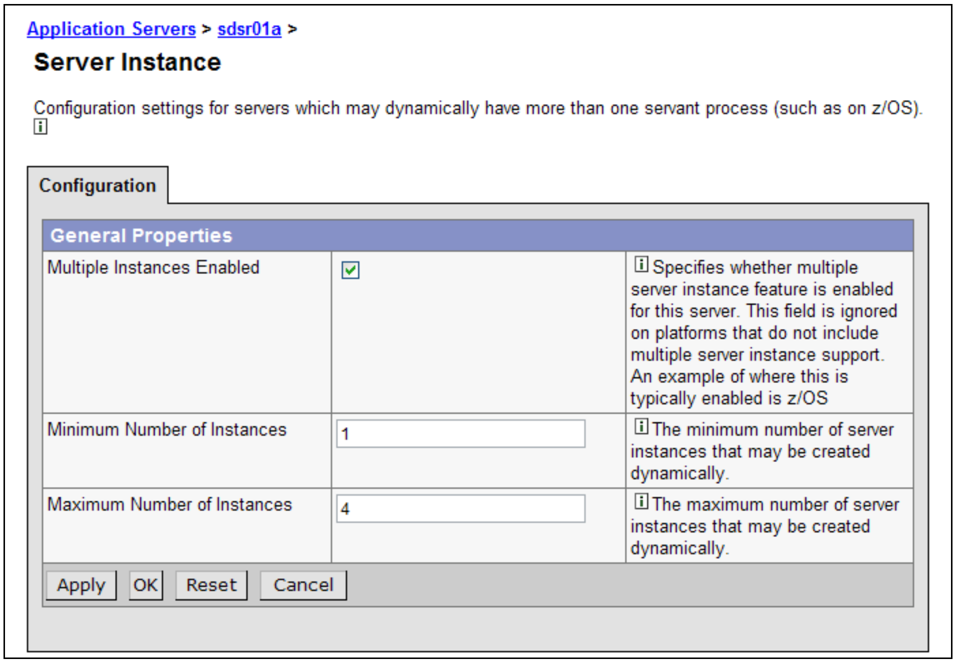
9.2.3 Application Environments

Application Environments allow WLM to start new servant address spaces in order to meet transaction performance goals. WLM starts a new servant region address space or stops a servant region address space as the workload varies.

WLM might start additional server regions to meet performance goals, based on the prioritization of its work compared to other work in the system, the availability of system resources needed to satisfy those objectives, and a determination by WLM whether starting more address spaces will help achieve the objectives.

By default, WebSphere servers are configured to only allow a single servant region. To enable multiple regions and specify the maximum and minimum number of servant regions, use the Administration Console, as shown on Figure 9-3.

From the initial panel, go to **Servers** → **Application Servers** → **[*server_name*]**, then from the Additional Properties section, select **Server Instance**.



Application Servers > sdsr01a >

Server Instance

Configuration settings for servers which may dynamically have more than one servant process (such as on z/OS).

Configuration

General Properties

Multiple Instances Enabled	☑	Specifies whether multiple server instance feature is enabled for this server. This field is ignored on platforms that do not include multiple server instance support. An example of where this is typically enabled is z/OS
Minimum Number of Instances	1	The minimum number of server instances that may be created dynamically.
Maximum Number of Instances	4	The maximum number of server instances that may be created dynamically.

Figure 9-3 Administration console: Minimum and maximum number of instances

Check the option **Multiple Instances Enabled** and specify the minimum number and maximum number of instances to set boundaries on how many servant regions WLM starts:

- Use *Minimum number of Instances* to start up a basic number of servant regions before the day's work arrives. This can save time waiting for WLM to determine that it needs more servant regions.
- *Maximum number of instances* is useful to cap the number of servant address spaces started by WLM if you determine that excessive servant regions can lead to service degradation.

The minimum and maximum number of instances specified also has an influence on the dispatching of transactions managed by WLM.

Transactions received by the WebSphere controller region are passed to servant regions through a set of WLM queues. The number of queues is determined by the number of service classes defined, and one servant region only serves one service class at a given time.

To ensure that you do not limit the parallelism of execution under full load, set the maximum number of instances at least as large as the number of defined service classes.

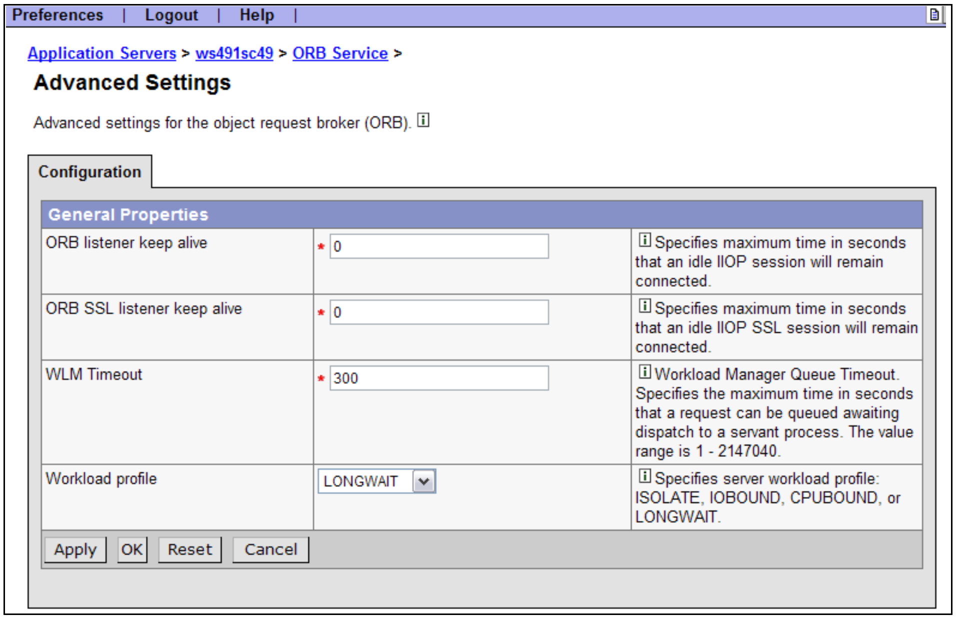
If you specify a maximum number of instances too low, there can be fewer servants available than WLM queues, and the result is a queue bottleneck under full load conditions. As a consequence, the system might experience queuing delays that result in transactions getting elongated response time.

Workload profile

Another performance factor to consider in order to determine the optimum number of servant regions is the number of threads available to each servant region, as defined in the workload profile.

The workload profile controls workload-pertinent decisions made by the WebSphere for z/OS runtime, such as the number of threads used in the server region. The default workload profile value is IOBOUND.

You can change the workload profile value through the administration console; go to **Application Servers** → **[*server_name*]** → **ORB Service** → **Advanced Settings**, as shown on Figure 9-4 on page 263.



Application Servers > ws491sc49 > ORB Service >

Advanced Settings

Advanced settings for the object request broker (ORB).

General Properties

ORB listener keep alive	* 0	Specifies maximum time in seconds that an idle IIOP session will remain connected.
ORB SSL listener keep alive	* 0	Specifies maximum time in seconds that an idle IIOP SSL session will remain connected.
WLM Timeout	* 300	Workload Manager Queue Timeout. Specifies the maximum time in seconds that a request can be queued awaiting dispatch to a servant process. The value range is 1 - 2147040.
Workload profile	LONGWAIT	Specifies server workload profile: ISOLATE, IOBOUND, CPUBOUND, or LONGWAIT.

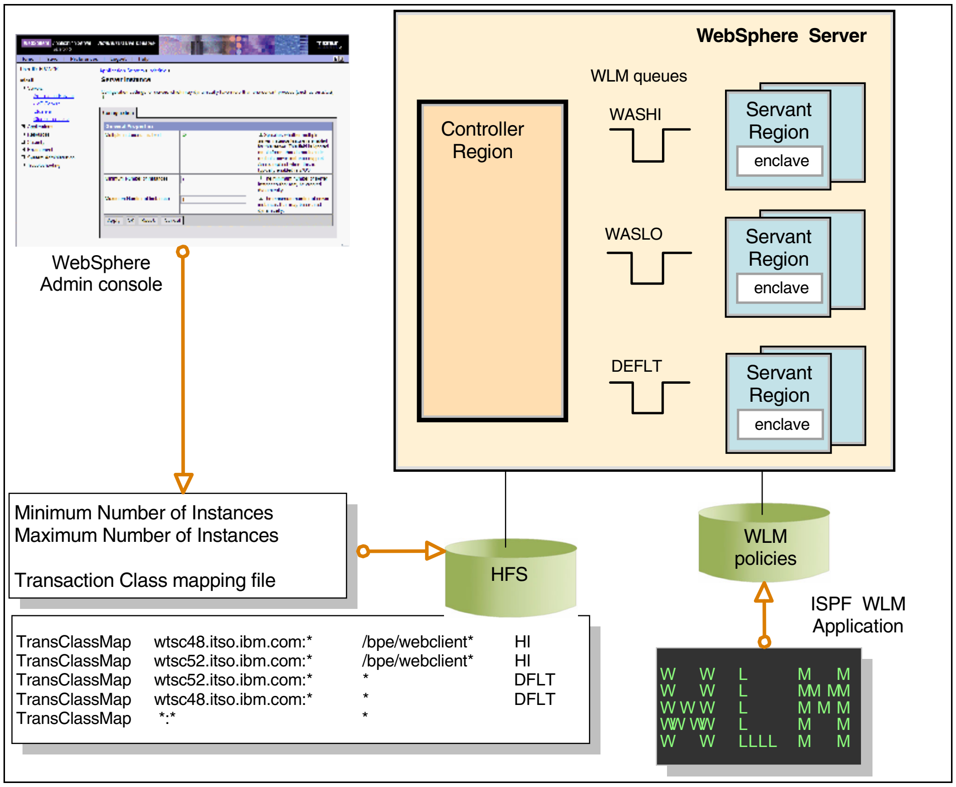
Figure 9-4 Administration console: Workload profile

You can specify the server workload profile as ISOLATE, IOBOUND, CPUBOUND, or LONGWAIT:

- ISOLATE: Specifies that the servant region is restricted to one single application thread.
- IOBOUND: Specifies more threads for applications that perform I/O-intensive processing. We recommend IOBOUND for most applications that have a balance of CPU activity and remote operation calls. The calculation of the thread number is based on the number of CPUs, using the formula MIN(30, MAX(5,(Number of CPUs*3))).
- CPUBOUND: Specifies that the application performs processor-intensive operations and, therefore, does not benefit from more threads than the number of CPUs. The calculation of the thread number is based on the number of CPUs, using the formula MAX((Number of CPUs-1),3).
- LONGWAIT: Specifies more threads than IOBOUND and is intended for applications that spend most of their time waiting for network or remote operations to complete. LONGWAIT allocates 40 threads.

9.3 Classifications rules

Figure 9-5 on page 264 shows the interaction between the WebSphere runtime and the WLM process. Later, you can learn how you can classify these WebSphere elements to WLM.

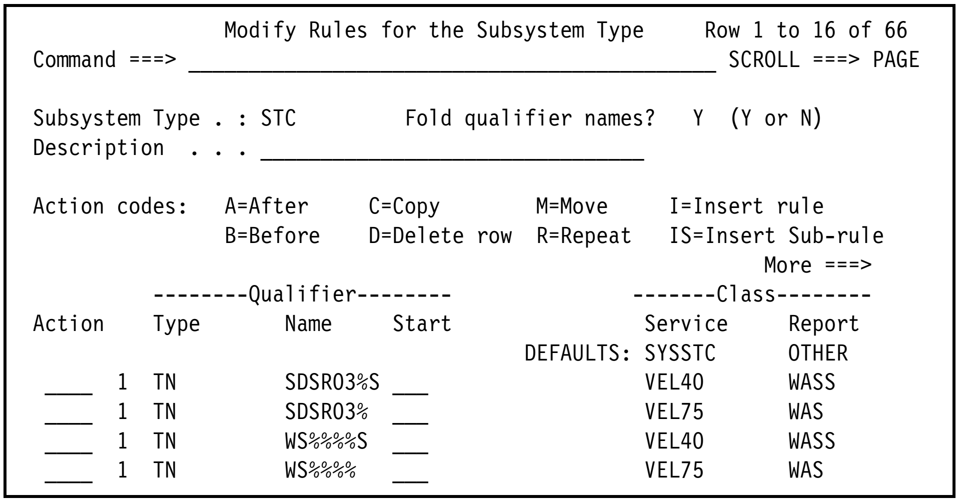


```
TransClassMap   wtsc48.itso.ibm.com:*   /bpe/webclient*   HI
TransClassMap   wtsc52.itso.ibm.com:*   /bpe/webclient*   HI
TransClassMap   wtsc52.itso.ibm.com:*   *                 DFLT
TransClassMap   wtsc48.itso.ibm.com:*   *                 DFLT
TransClassMap    *:*                    *
```

Figure 9-5 WebSphere runtime and Workload Manager

9.3.1 Classification rules for STC subsystem

You create the rules for STC subsystems using the WLM ISPF application, as shown in Figure 9-6.



```
                  Modify Rules for the Subsystem Type     Row 1 to 16 of 66
Command ===> ____________________________________________ SCROLL ===> PAGE

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 ===>
          --------Qualifier--------              -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: SYSSTC      OTHER
 ____  1  TN         SDSR03%S ___                  VEL40       WASS
 ____  1  TN         SDSR03%  ___                  VEL75       WAS
 ____  1  TN         WS%%%%S  ___                  VEL40       WASS
 ____  1  TN         WS%%%%   ___                  VEL75       WAS
```

Figure 9-6 WLM definition of WebSphere server regions, STC subsystem

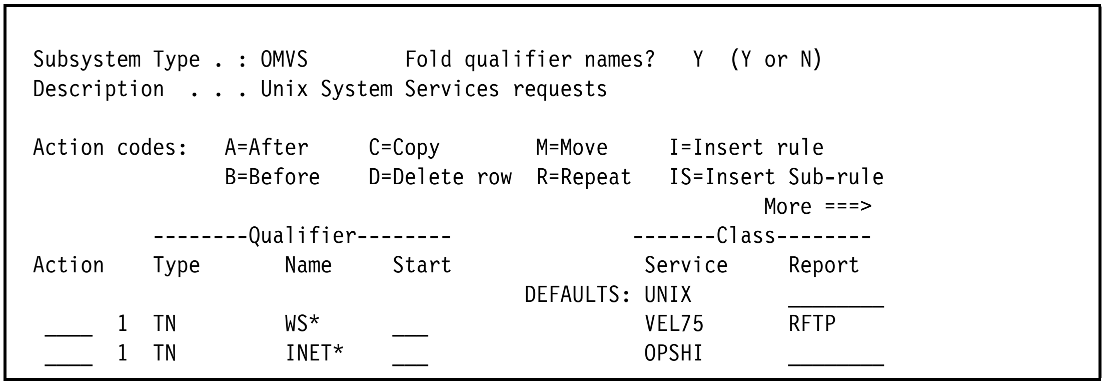
To create the rules:

- Assign the controller region (Daemon, Node Agent, Deployment Manager) to a STC service class with a high importance and a high velocity goal. There is a certain amount of processing in the WebSphere application's control regions to receive work into the system, manage the HTTP transport handler, and classify and route the input work. Because controller regions act as work routers, they must have high priority.

- Assign the servant and adjunct regions to an STC service class with a high enough velocity goal so that they can be initialized quickly when WLM determines that they are needed (although of a lower value than the controller region). You use this service class for tasks that run in the servant region under control of the step task and not as part of the enclave. The service class should be lower in the service class hierarchy than more important work, such as the controller region, production WebSphere enclaves, or CICS and IMS transaction servers.

  All threads are managed according to the CB-assigned service class, including those outside of the enclave. However, any CPU time measured in the servant region that is not part of the enclaves is charged to the STC-assigned service class (or reporting class) in the RMF Workload Activity report.

- Classify correctly the BPXBATCH shell scripts that represent the first two steps of the control region startup procedure to avoid elongating the startup time before BBOCTL gets control. To mitigate this fact, classify the BPXBATCH steps under the OMVS subsystem using the started task jobname for the TN field and assign a service class with high importance and high velocity. Figure 9-7 shows an example of the OMVS classification rules.



```
Subsystem Type . : OMVS        Fold qualifier names?   Y  (Y or N)
Description  . . . Unix System Services requests

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------              -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: UNIX        ________
 ____  1  TN         WS*      ___                  VEL75       RFTP
 ____  1  TN         INET*    ___                  OPSHI       ________
```

Figure 9-7 WLM OMVS classification rules for the WebSphere BPXBATCH steps

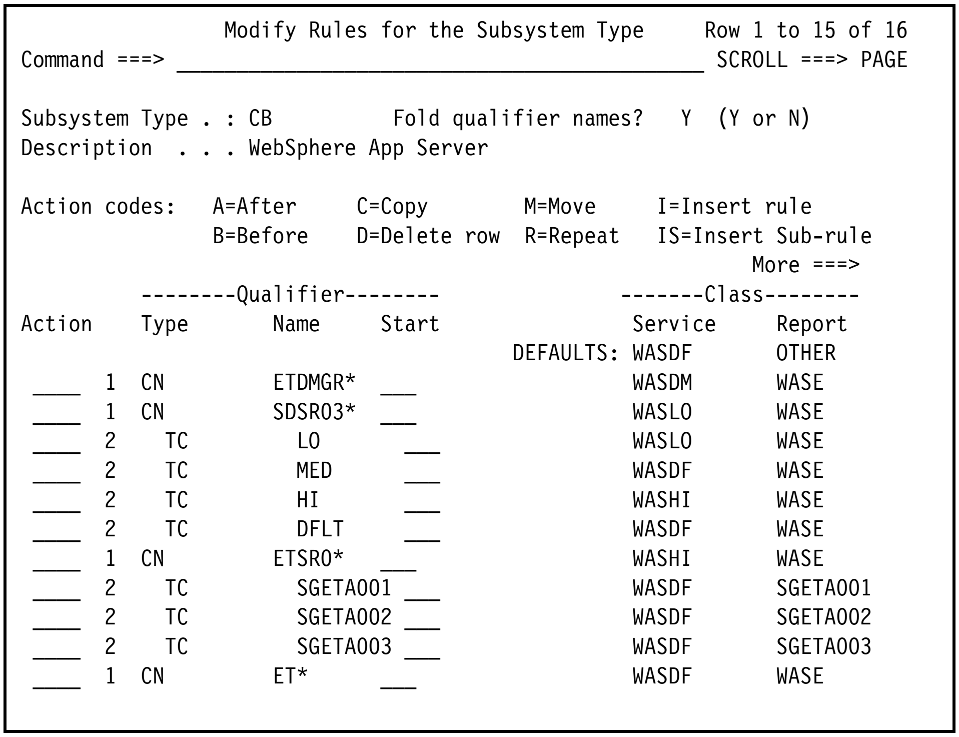
For further information, refer to:

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD102730`

You might also wish to define report classes for the address spaces' STC activity. This allows you to monitor the activity within the WebSphere regions for service tasks running outside of the enclave.

9.3.2 Classification rules for CB subsystem

WebSphere applications that run in the servant region as part of the dispatched enclave use these WLM classification rules. Each WebSphere transaction is dispatched as a WLM enclave and is managed within the servant region according to the service class assigned through the CB service classification rules. See Figure 9-8.



```
                 Modify Rules for the Subsystem Type     Row 1 to 15 of 16
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : CB          Fold qualifier names?   Y  (Y or N)
Description  . . . WebSphere App 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--------               -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: WASDF       OTHER
 ____  1  CN         ETDMGR*  ___                  WASDM       WASE
 ____  1  CN         SDSR03*  ___                  WASLO       WASE
 ____  2    TC         LO       ___                WASLO       WASE
 ____  2    TC         MED      ___                WASDF       WASE
 ____  2    TC         HI       ___                WASHI       WASE
 ____  2    TC         DFLT     ___                WASDF       WASE
 ____  1  CN         ETSR0*   ___                  WASHI       WASE
 ____  2    TC         SGETA001 ___                WASDF       SGETA001
 ____  2    TC         SGETA002 ___                WASDF       SGETA002
 ____  2    TC         SGETA003 ___                WASDF       SGETA003
 ____  1  CN         ET*      ___                  WASDF       WASE
```

Figure 9-8 WLM definitions of the servant regions, CB subsystem

The CB classification rules can be based on server name, server instance name (although not useful because instances share work), user ID (although not useful unless RUNAS=USER), or transaction class. You can assign a default transaction class to the servant, or you can use the virtual host name, port number, or URI template to map specific requests to a transaction class using a transaction mapping file.

We recommend that you use a percentage response time goal for these transactions; for example, 80% of transactions in less than 0.5 seconds, or you can have a high velocity default service class. Response time goals are better than velocity goals in a production environment. Velocity goals require recalibration every time that the environment changes (CPU or workload). The default service class is SYSOTHER with a discretionary goal.

Avoid using multi-period goals, because the second and subsequent periods are not aggressively managed.

Adjunct region

The adjunct region starts by the control region scheduling a dummy piece of work into the application environment for the CRA (separate from the WLM application environment for the regular servant regions). This will cause WLM to start up the CRA address space. As it starts up, it will be prioritized based on its classification as a started task.

After it gets initialized and picks up the dummy request from the WLM queue, it will be bound to the service class of the enclave associated with that request and the CRA will be managed according to the goals of that service class.

The dummy request is an internal piece of work that goes through the IIOP path for processing. In V6.0 it will be classified using the cluster name (as a collection name for WLM) and a transaction class (if it found one in the classification XML file using the IIOP rules). Starting with V6.1, you can use the InboundClassification type="internal" clause to specify a default classification for all internal work that originates from the CR headed for any SR (including the CRA). If the internal clause is missing, no transaction class is used for classification.

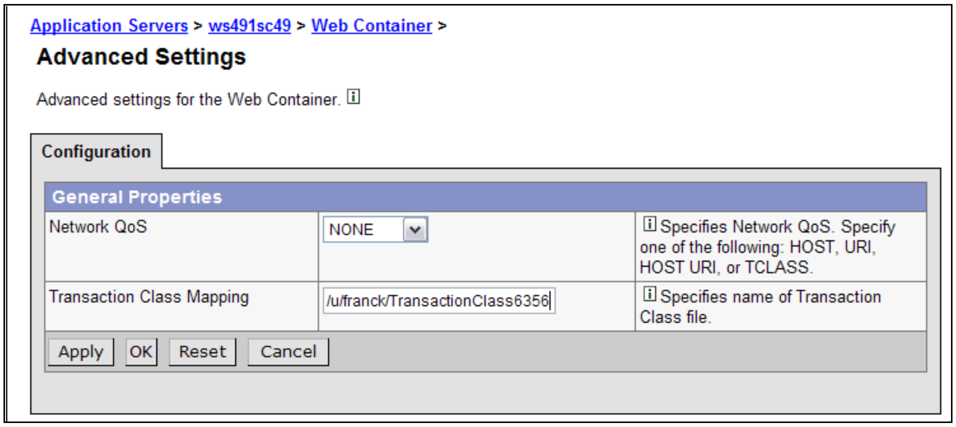
Assigning transaction classes

WebSphere Application Server V5.1 (at service level W510200) can assign a transaction class (TC) to a work item by using a *transaction class mapping* file for HTTP requests, or a *workload classification document* for HTTP, IIOP, or MDB inbound requests.

Using a transaction class mapping file

When you use a transaction mapping file, you must define it to the server through the administration console. See Figure 9-9.

From the initial panel, select **Servers** → **Application Servers** → **[*server_name*]** → **Web Container** → **Advanced Settings**.



Application Servers > ws491sc49 > Web Container >

Advanced Settings

Advanced settings for the Web Container.

Configuration

General Properties

Network QoS	NONE	Specifies Network QoS. Specify one of the following: HOST, URI, HOST URI, or TCLASS.
Transaction Class Mapping	/u/franck/TransactionClass6356	Specifies name of Transaction Class file.

Figure 9-9 Admin console: Transaction class mapping

A transaction class mapping file allows us to associate a set of URIs to a specific transaction class. At execution time, WLM uses the transaction class to associate the work request to a service class. Example 9-2 shows a transaction class mapping file used for our tests.

Example 9-2 Transaction class mapping file

```
TransClassMap wtsc49.itso.ibm.com:*   *               LO
TransClassMap wtsc49.itso.ibm.com:*   *               LO
TransClassMap wtsc66.itso.ibm.com:*   *               LO
TransClassMap wtsc66.itso.ibm.com:*   /bpe/webclient*  MED
TransClassMap wtsc48.itso.ibm.com:*   /bpe/webclient*  HI
```

Using workload classification documents

WebSphere for z/OS Version 5.1 introduces a common .xml file for the classification of inbound HTTP, IIOP, and MDB work, which we call the *WebSphere for z/OS Workload Classification Document*. The Workload Classification Document location is specified to the server through a WebSphere variable named wlm_classification_file. This variable can be set at the Cell, Node, or Server Instance level, but must be accessible and applicable to all the servers that inherit the specification from the Cell or Node level.

Important: We discourage you from using this new Workload Classification Document in conjunction with the old format HTTP classification and MDB classification files. If you need to use both methods, observe these rules for simultaneous usage:

- The use of the new Workload Classification Document supersedes the use of the old MDB classification file. WebSphere for z/OS does not use the endpoint_config_file variable.

  Migration consideration: If you are currently using a MDB classification document specified by the endpoint_config_file variable, then you must convert this .xml file to the new document format and refer to it with the wlm_classification_file variable.
- If the new Workload Classification Document is defined in a server that also has the old style HTTP classification document specified and there are NO HTTP classification rules in the Workload Classification Document, then the old format file contents are used to classify inbound HTTP requests. However, if there are HTTP classification rules present in the new Workload Classification Document, these classification rules are used instead of the classification rules in the old style HTTP classification document.

Consideration for using the Workload Classification Document:

- If the Workload Classification Document is not a well-formed, valid .xml document, then it is ignored by the server. In the case of an error, the following message appears in the job log with “{0}” replaced by the parser’s error message:

  ```
  BBOJ0085E PROBLEMS ENCOUNTERED PARSING WLM CLASSIFICATION XML FILE {0}
  ```
- WebSphere Application Server Controller Region only reads the Workload Classification Document during startup. You must restart the Application Server to implement any changes to the contents of the file.

Example 9-3 shows a sample Workload Classification Document (or file) with the descriptions of how to classify inbound IIOP, HTTP, and MDB requests.

Example 9-3 Sample WebSphere for z/OS Workload Classification Document

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE Classification SYSTEM "Classification.dtd" >
3 <Classification schema_version="1.0">

 <!--
         IIOP Classification Rules
 -->
4  <InboundClassification type="iiop"
                           schema_version="1.0"
                           default_transaction_class="A0">

5     <iiop_classification_info transaction_class="A1"
                             application_name="IIOPStatelessSampleApp"
                             module_name="StatelessSample.jar"
                             component_name="Sample20"
                             description="Sample20 EJB Classification">
6        <iiop_classification_info transaction_class=""
                                method_name="echo"
                                description="No TCLASS for echo()"/>
7        <iiop_classification_info transaction_class="A1B"
                                method_name="ping"
                                description="Ping method" />
5a    </iiop_classification_info>
```

```
8     <iiop_classification_info application_name="*"
                             module_name="*"
                             component_name="*"
                             transaction_class="A2"
                             description="TCLASS the rest to A2">
9        <iiop_classification_info transaction_class="A2A"
                                method_name="resetFilter"
                                description="Spl case resetFilter()"/>
8a    </iiop_classification_info>

4a </InboundClassification>
<!--
         HTTP Classification Rules
-->

10 <InboundClassification type="http"
                          schema_version="1.0"
                          default_transaction_class="M">
11    <http_classification_info transaction_class="N"
                             host="wsc1.washington.ibm.com"
                             description="Virtual Host wsc1">
12       <http_classification_info transaction_class="O"
                                port="9080"
                                description="Def wsc1 HTTP reqs">
13          <http_classification_info transaction_class="Q"
                                   uri="/gcs/admin"
                                   description="Gcs"/>
14          <http_classification_info transaction_class="S"
                                   uri="/gcs/admin/l*"
                                   description="Gcs login"/>
12a      </http_classification_info>
15       <http_classification_info transaction_class="P"
                                port="9081"
                                description=" Def wsc1 HTTPS reqs ">
16          <http_classification_info transaction_class=""
                                uri="/gcss/mgr/*"
                                description="GCSS Mgr"/>
15a         </http_classification_info>
11a      </http_classification_info>
10a</InboundClassification>

<!--
         MDB Classification Rules
-->

17 <InboundClassification type="mdb"
                         schema_version="1.0"
                         default_transaction_class="qrs">
18    <endpoint type="messagelistenerport"
                name="IVPListenerPort"
                defaultclassification="MDBX"
                description="ABC">
19       <classificationentry selector="Location=&apos;East&apos;"
                                classification="MDB1"
                                description="DEF"/>
20       <classificationentry selector="Location&lt;&gt;&apos;East&apos;"
                                classification="MDB2"
                                description="XYZ"/>
18a   </endpoint>
21    <endpoint type="messagelistenerport"
```

```
                name="SimpleMDBListenerPort"
                defaultclassification="MDBX"
                description="GHI"/>
17a</InboundClassification>

<!--
          Workload Classification Document for SDSR03x servers
          Change History
          --------------------------------------------------------_----------
          Activity                                       Date        Author
          Created                                        08-01-2004  MCC
-->

3a </Classification>
```

The numbers correspond to the following points:

- We denote start elements in Example 9-3 on page 268 with a number (3) and end elements corresponding to a start element have the same number suffixed with an “a” (3a).
- We do not explicitly denote child elements, but you can locate them in the example between the labeled start and end element. For example, element 16 is a child element of element 15, and you can see that element 16 is enclosed by start element 15 and end element 15a. Child elements are indented to assist the reader.

Considerations

Example 9-4 on page 272 contains the DTD that defines this xml document and that you need to use as the definitive reference for constructing a workload classification xml document. For our descriptive purpose, we can use the above sample to illustrate how to classify IIOP, HTTP, and MDB work in WebSphere for z/OS Version 5.1.

Elements 1, 2, 3, and 3a are required elements. Each Workload Classification Document must start with the first three statements and end with the </Classification> statement.

Element 1 is an indicator that you must save the Workload Classification Document in ascii in order for the Application Server to process it. If you create the document on z/OS in codepage IBM-1047, which is the normal codepage for files existing in the HFS, then you must convert the file to ascii prior to using it. There are two approaches for converting a working document into a document that the server can use:

- native2ascii

  Java SDK provides this utility to convert a file from the native codepage to the ascii codepage. For example, if the working xml document is called x5sr02.classification.ebcdic.xml and you want to create document x5sr02.classification.xml, you use the following command:

  ```
  /u/mccox -> native2ascii \
  > x5sr02.classification.ebcdic.xml >x5sr02.classification.xml
  ```

- iconv

  z/OS provides this utility to convert files from a designated codepage to a different designated codepage. For example, if the working xml document is called x5sr02.classification.ebcdic.xml and you want to create document x5sr02.classification.xml, you use the following command:

  ```
  /u/mccox -> iconv –f IBM-1047 –t UTF-8 \
  > x5sr02.classification.ebcdic.xml >x5sr02.classification.xml
  ```

In each case, the command line is too long to show on the page and is continued with the backslash (\) character to the next line. Alternatively, you can create the Workload Classification Document on your workstation and FTP the file to the correct location on z/OS in binary format. If you also create the Classification.dtd file in the same directory as the Workload Classification Document, then you can validity check the document prior to installing it in a server using any utility providing a validating parser. For example, you can use WebSphere Application Developer workbench to construct and validate the Workload Classification Document.

Element 2 provides the parser with the name of the DTD document provided by WebSphere for z/OS needed to verify the validity of the WLM classification document. This statement is required.

Element 3 is the root of the document. This element is required and you must specify the schema_version attribute. Currently, the only supported version number is 1.0. Element 3a is the *end element* corresponding to the root element.

This Workload Classification Document contains rules for classifying IIOP, HTTP, and MDB work. The *InboundClassification* element defines the type of work that is to be classified by type of work specific child elements. The statement is:

```
<InboundClassification type="iiop | http | mdb"
                         schema_version="1.0"
                         default_transaction_class="value">
```

The following rules apply to the *InboundClassification* element:

- The *type* attribute is a required attribute and its value must be either iiop, http, or mdb. Only one occurrence of an *InboundClassification* element can occur in the document for each type. Therefore, there can be at most three InboundClassification elements in a document. There is no required order for the InboundClassification elements, nor is an InboundClassification required for each type. If, for example, there are no MDBs deployed in a server, then there is no need for classification criteria.
- The *schema_version* attribute is a required attribute, and its value must be set to 1.0, because this is the only value currently recognized by WebSphere.
- The *default_transaction_class* attribute must be specified. The string value must be a valid WLM transaction class value, a null string (that is, “”), or a string containing eight or fewer blanks.
- An InboundClassification end element (that is, </InboundClassification>) must appear before the next occurrence of an InboundClassification element. InboundClassification elements cannot be nested.

In this Sample Workload Classification Document, you can see that all three types of work have classification rules defined. Elements 4 through 9 define IIOP classification rules, elements 10 through 16 define HTTP classification rules, and elements 17 through 21 define MDB classification rules. When an element has a separate *end element* statement, the corresponding end element statement is denoted with an “a”, for example, 3 and 3a.

The rules and xml statements for classifying IIOP work and HTTP work are similar; there is slightly different syntax. The rules and xml statements for classifying MDB work are essentially the same as the old rules of classifying MDBs. We examine each section in the Sample Workload Classification Document separately. Let us first look at the IIOP classification, which is new in WebSphere for z/OS V5.1; then the HTTP classification, which is different from before but basically the same as IIOP classification; and finally, the MDB classification, which is basically unchanged. See Example 9-4.

Example 9-4 DTD for the Workload Classification XML Document

```
<?xml version='1.0' encoding="UTF-8"?>
<!ELEMENT Classification (InboundClassification+)>
<!ATTLIST Classification schema_version CDATA #REQUIRED>
<!ELEMENT InboundClassification
((iiop_classification_info+|http_classification_info+|endpoint+))>
<!ATTLIST InboundClassification type (iiop|mdb|http) #REQUIRED>
<!ATTLIST InboundClassification default_transaction_class CDATA #REQUIRED>
<!ATTLIST InboundClassification schema_version CDATA #REQUIRED>
<!ELEMENT iiop_classification_info (iiop_classification_info*)>
<!ATTLIST iiop_classification_info application_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info component_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info description CDATA #IMPLIED>
<!ATTLIST iiop_classification_info method_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info module_name CDATA #IMPLIED>
<!ATTLIST iiop_classification_info transaction_class CDATA #REQUIRED>
<!ELEMENT endpoint (classificationentry*)>
<!ATTLIST endpoint defaultclassification CDATA #REQUIRED>
<!ATTLIST endpoint name CDATA #REQUIRED>
<!ATTLIST endpoint type (messagelistenerport) #REQUIRED>
<!ATTLIST endpoint description CDATA #IMPLIED>
<!ELEMENT classificationentry EMPTY>
<!ATTLIST classificationentry classification CDATA #REQUIRED>
<!ATTLIST classificationentry selector CDATA #REQUIRED>
<!ATTLIST classificationentry description CDATA #IMPLIED>
<!ELEMENT http_classification_info (http_classification_info*)>
<!ATTLIST http_classification_info host CDATA #IMPLIED>
<!ATTLIST http_classification_info port CDATA #IMPLIED>
<!ATTLIST http_classification_info uri CDATA #IMPLIED>
<!ATTLIST http_classification_info description CDATA #IMPLIED>
<!ATTLIST http_classification_info transaction_class CDATA #REQUIRED>
```

Classification rules for IIOP

The InboundClassification element with attribute type=“iiop” defines the section of the document that is applicable to IIOP classification. Specifically, an element, such as this example, starts the section:

```
<InboundClassification type="iiop"
                            schema_version="1.0"
                            default_transaction_class="value1">
```

IIOP work can be classified based on the following J2EE™ application artifacts:

- *Application name*: This is the name of the application containing the EJBs. It is the display name of the application, which is not necessarily the name of the .ear file containing all the artifacts.
- *Module name*: This is the name of the EJB™ .jar file containing one or more EJBs (there can be multiple EJB .jar files contained in an .ear file).
- *Component name*: This is the name of the EJB contained in a module (or EJB .jar) (there can be one or more EJBs contained in an EJB .jar file).
- *Method name*: This is the name of a remote method on an EJB.

WebSphere provides the flexibility of classifying IIOP work in various applications at any of these levels through the iiop_classification_info element. The format of the iiop_classification_info element is:

```
<iiop_classification_info transaction_class="value1"
                          [application_name="value2"]
                          [module_name="value3"]
                          [component_name="value4"]
                          [method_name="value5"]
                          [description="value6"] >
```

The iiop_classification_info element allows you to build filters based on the application, module, component, and method names, which assign TCLASS values to inbound requests.

The following rules apply to the iiop_classification_info element:

- You must specify the transaction_class attribute. The string value must be a valid WLM transaction class value, a null string (that is, “”), or a string containing eight or fewer blanks. Specifying a null or blank string allows you to override a default TCLASS setting (or one assigned by a higher level filter) and not have a TCLASS value for the request.
- You can specify the attributes application_name, module_name, component_name, and method_name with one of these:
  - The exact name of the application, module, component, or method (for example, “MAYPOLE”)
  - A wildcard value (for example, “MAY*”, which matches any string starting with “MAY”)
  - The asterisk character “*”, which matches any value.

  These attributes act as selectors or filters that either assign a transaction class or allow subsequent nested iiop_classification_info elements to assign the transaction class. You can specify one or more of the application_name, module_name, or component_name attributes on one iiop_classification_info element. You do not need to use all of the attributes to make a classification filter. Only use the granularity required. If for instance, there is only one application in the server, then the xml document describing the classification rules for that server does not need an iiop_classification_element with the application_name attribute specified.
- You can nest the iiop_classification_info elements in a hierarchical manner. This allows you to construct hierarchical classification filters based on attribute values. For example, consider this filter:

  ```
  <iiop_classification_info transaction_class="FAST"
                            application_name="MyAPP1"
                            component_name=”EJB1”/>
  <iiop_classification_info transaction_class="SLOW"
                            application_name="MyApp1"
                            component_name=”EJB2”/>
  ```

  And this filter:

  ```
  <iiop_classification_info transaction_class="MEDIUM"
                            application_name="MyAPP1>"
     <iiop_classification_info transaction_class="FAST"
                               component_name=”EJB1”/>
     <iiop_classification_info transaction_class="SLOW"
                               component_name=”EJB2”/>
  </iiop_classification_info>
  ```

  Both filters have the same effect when classifying requests on EJB1 and EJB2 in application MyAPP1. However, the second filter also classifies requests on any other EJB in the application “MyApp1”, unlike the first filter.
- Specifying an attribute value that conflicts with the parent element’s attribute value effectively negates the lower level filter. For example:

  ```
  <iiop_classification_info transaction_class="FAST"
                            application_name="MyAPP1">
     <iiop_classification_info transaction_class="SLOW"
  ```

```
                            application_name="MyApp2"/>
</iiop_classification_info>
```

This filter construction never results in EJB requests in MyApp2 assigned transaction class “SLOW”. The higher level filter only allows IIOP requests for application_name=”MyApp1” to be passed to a lower level filter.

- The order of iiop_classification_info statements is significant at each level. The first filter at a specific level that matches the attributes of the request is the one that is used, not the “best or most restrictive” filter. For example:

```
<iiop_classification_info transaction_class="FAST"
                            application_name="MyAPP"/>
   <iiop_classification_info transaction_class="SLOW"
                               component_name="*"/>
   <iiop_classification_info transaction_class="MEDIUM"
                               component_name="MySSB"/>
</iiop_classification_info>
```

In the above example, all the IIOP requests processed by EJBs in the application named “MyApp” will be assigned a TCLASS value of “SLOW”. This is true for requests to the EJB named “MySSB” as well. Even though “MySSB” is explicitly assigned a TCLASS value of “MEDIUM”, this filter is not applied. The filter preceding it matches and the TCLASS assignment will be made, and the remainder of the filters at that level will be ignored.

- The description field is optional. However, you need to use a description on all iiop_classification_info elements. This description string is printed as part of the operator command support and allows you to identify the classification rule used. Using a reasonably terse description is best, because it displays on the MVS console.

In the Sample Workload Classification Document, there are two *level one* IIOP filters defined. The first level one filter is defined by element 5 and the second by element 8.

The first filter is set up to explicitly assign transaction class values to the Sample20 EJB, residing in the StatelessSample.jar file, which is part of the application named IIOPStatelessSampleApp. Any method on this specific EJB will be assigned a TCLASS value of “A1” unless a lower level filter assigns a different TCLASS value. In this example, there are lower level filters defined with elements 6 and 7, which are specific only for the ping( ) and echo( ) methods:

- In the case of the ping( ) method, a null TCLASS value will be assigned (or no TCLASS value will be assigned). This will result in no TCLASS value used by WLM in the classification of the enclave.
- In the case of the echo( ) method, a TCLASS value of “A1B” will be assigned.

The second filter is set up to implicitly assign a transaction class to any other IIOP work not selected by the first filter, and assigns a transaction class of “A2” to work requests other than requests on the resetFilter( ) method, which is assigned a TCLASS value of “A2A”.

If you look closely at the IIOP classification section, you will notice two important things:

- The default_transaction_class attribute value of “A1” on the InboundClassifcation element will not be used. The second filter will effectively match all requests and assign TCASS value of “A2” or “A2A” as described above.
- The order of the two level 1 filters is important. If you reversed the order of these two filters in the document, the level 1 filter defined by element 5 is never used, because the very generic level 1 filter defined by element 8 matches every inbound request.

This example is quite simple. Because multiple attributes are specified on the first level filter, the nesting of iiop_classification_info elements is very shallow. There are, in fact, only two levels to either filter. You can certainly construct a filter with greater depth by specifying only one attribute per iiop_classification_level, which results in a filter of depth equal to 4.

And while it seems to make sense building filters that select at the application level and work down to the components residing in the application, there is no requirement that the high-level filter is based on application_name. What is important is that the filters that are created are constructed in a coherent manner so that the transaction classification value you want to assign is actually assigned.

We recommend that you keep the number of filters and the complexity of filters as simple as possible and yet retain the required level of classification needed by the installation. There is in fact a cost to associating a TCLASS value to inbound IIOP work. WebSphere for z/OS provides a new operator command that you can use to inquire about the classification of IIOP requests, the number performed, which classification was used, and a heuristic cost value of using a filter.

Classification rules for HTTP

The InboundClassification element with attribute type=”http” defines the section of the document that is applicable to IIOP classification. Specifically, an element like the following starts the section:

```
<InboundClassification type="http"
                            schema_version="1.0"
                            default_transaction_class="value1">
```

HTTP work can be classified based on the following J2EE application artifacts:

- Virtual Host Name: This is the host name in the HTTP header to which the inbound request is sent.
- Port Number: This is the port on which the HTTP catcher is listening.
- Uniform Resource Identifier (URI): This is the string that identifies the Web application. WebSphere provides the flexibility of classifying HTTP work in various applications at any of these levels through the http_classification_info element. The format of the http_classification_info element is:

```
<http_classification_info transaction_class="value1"
                          [host="value2"]
                          [port="value3"]
                          [uri="value4"]
                          [description="value5"] >
```

The http_classification_info element allows you to build filters based on the application, module, component, and method names that will assign TCLASS values to inbound requests.

The following rules apply to http_classification_info element:

- You must specify the transaction_class attribute. The string value should be a valid WLM transaction class value, a null string (that is, “”), or a string containing eight or fewer blanks. Specifying a null or blank string allows you to override a default TCLASS setting (or one assigned by a higher level filter) and not have a TCLASS value for the request.
- You can specify attributes host, port, and uri with one of these:
  - The exact name of the host, port, or uri (for example, “MAYPOLE”)
  - A wildcard value (for example, “MAY*”, which matches any string starting with “MAY”)
  - The asterisk, “*”, which matches any value

These attributes act as selectors or filters that will either assign a transaction class or allow subsequent nested http_classification_info elements to assign the transaction class. You can specify one or more of the host, port, or uri attributes on one http_classification_info element. You do not need to use all attributes to make a classification filter. Only use the granularity required. If, for instance, there is only one application in the server, then the xml document describing the classification rules for that server does not need an http_classification_info element with the uri attribute specified:

- You can nest http_classification_info elements in a hierarchical manner. This allows you to construct hierarchical classification filters based on attribute values. For example, consider the following filter:

```
<http_classification_info transaction_class="FAST"
                          host="MyVHost1.com"
                          uri=”/MyWebApp1/*”/>
<http_classification_info transaction_class="SLOW"
                          host="MyVHost2.com"
                          uri=”/MyWebbApp2/*”/>
```

  And this filter:

```
<http_classification_info transaction_class="MEDIUM"
                          host="MyVHost1.com>"
   <http_classification_info transaction_class="FAST"
                             uri=”/MyWebApp1/*”/>
   <http_classification_info transaction_class="SLOW"
                             uri=”/MyWebApp2/*”/>
</http_classification_info>
```

  Both filters will have the same effect when classifying requests to Web applications identified by context roots /MyWebApp1 and /MyWebApp2 in this Application Server hosting Web applications for virtual host “MyVhost1.com”. However, the second filter will also classify requests on any other context root in the Application Server, unlike the first filter.

- Specifying an attribute value as different from the parent element’s attribute value effectively negates the lower level filter. For example:

```
<http_classification_info transaction_class="FAST"
                          uri="/MyWebApp1/*">
   <http_classification_info transaction_class="SLOW"
                          uri="/MyWebApp2">
   </iiop_classification_info>
</iiop_classification_info>
```

  This specification never results in Web applications with context root /MyWebApp2 assigned transaction class “SLOW”. The higher level filter only allows HTTP requests with a context root of /MyWebApp1 to be passed to a lower level filter.

- The order of http_classification_info statements is significant at each level. The first filter at a specific level that matches the attributes of the request is the one that is used, not the “best or most restrictive” filter. For example:

```
<http_classification_info transaction_class="FAST"
                          host="MyVHost.com"/>
   <http_classification_info transaction_class="SLOW"
                             uri="*"/>
   <http_classification_info transaction_class="MEDIUM"
                             uri="/MyWebAppX/*"/>
</http_classification_info>
```

  In the above example, all the HTTP requests that processed the application server for virtual host “MyVhost.com” will be assigned a TCLASS value of “SLOW”. This is true for requests to Web applications with context root /MyWebAppX as well. Even though uri=“/WebAppX/*” is explicitly assigned a TCLASS value of “MEDIUM”, this filter is not applied. The filter preceding it matches and the TCLASS assignment is made and the remainder of the filters at that level are ignored.

- The Description field is optional, although you need to use it on all http_classification_info elements. It is printed as part of the operator command support and allows you to identify the classification rule used. Use a reasonably terse description because it displays on the MVS console.

In the Sample Workload Classification Document, there is one HTTP filter IIOP filter defined with two second level filters. Each of these second level filters has one or more third level filters. The level one filter (that is, **11**) is set up to explicitly assign transaction class values to all Web applications associated with virtual host “wsc1.washington.ibm.com”. Any Web applications in the server under any other virtual host name have the transaction class from the enclosing InboundClassification element (that is, TCLASS value of “M”).

The two second level filters, **11** and **15**, distinguish between HTTP and HTTPS requests. This is not obvious unless you know that port 9080 is for HTTP and 9081 is for HTTPS. Within the http requests there are two URIs, which are special-cased with third level filters (that is, **13** and **14**). Similarly, for HTTPS requests there is one URI, which is special-cased (that is, **16**).

This example is quite simple. Basically, the filters are established to assign transaction classes for special URIs within transports for one specific virtual host. And while it seems to make sense to build filters that select at the virtual host level and work down to the URIs defining the Web application, there is no requirement that the high-level filter is based on host=. If all of the Web applications in the server are under one virtual host, then perhaps context root (that is, uri=) is a better choice for setting up the level one filters. What is important is that you construct the filters in a coherent manner so that the classification you want to assign is the classification that is actually assigned.

We recommend that you keep the number of filters and the complexity of filters as simple as possible and yet retain the required level of classification needed by the installation. There is, in fact, a cost to associating a TCLASS value to inbound IIOP work. WebSphere for z/OS provides a new operator command that you can use to inquire about the classification of IIOP requests, the number performed, which classification was used, and a heuristic cost value of using a filter.

Classification rules for Message Driven Beans (MDBs)

Starting with WebSphere for z/OS Release 5.0.2, you can associate a Message Driven Bean to a transaction class within the CB subsystem. The filter elements that you can use for MDB classification are:

- Listener Port (Endpoint)
- Message Selector (XML tags)

As for the other transactions running in the CB subsystem, you can assign a default transaction class to the servant and a default transaction class to the MDBs that will not have the transaction class explicitly assigned. Example 9-5 shows a sample of the WLM definitions.

Example 9-5 WLM definitions for the MDB environment

```
Modify Rules for the Subsystem Type     Row 1 to 15 of 16
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : CB          Fold qualifier names?   Y  (Y or N)
Description  . . . WebSphere App Server

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

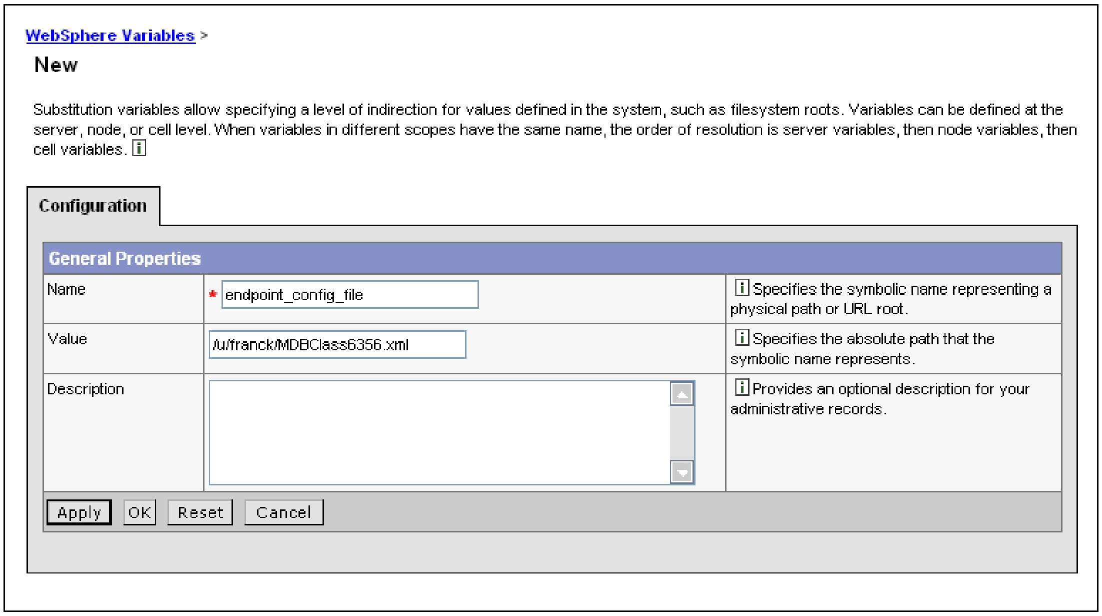
```
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                              More ===>
          --------Qualifier--------              -------Class--------
Action    Type       Name     Start                Service     Report
                                        DEFAULTS: WASDF       OTHER
 ____  1  CN         ETDMGR*  ___                  WASDM       WASE
 ____  1  CN         SDSR03*  ___                  WASLO       WASE
 ____  2    TC         MDB      ___                WASDF       WASE
 ____  2    TC         MDBD     ___                WASLO       WASE
 ____  2    TC         MDBQ     ___                WASLO       WASE
 ____  2    TC         MDBT     ___                WASHI       WASE
 ____  1  CN         ETSR0*   ___                  WASHI       WASE
 ____  2    TC         SGETA001 ___                WASDF       SGETA001
 ____  2    TC         SGETA002 ___                WASDF       SGETA002
 ____  2    TC         SGETA003 ___                WASDF       SGETA003
 ____  1  CN         ET*      ___                  WASDF       WASE
```

The WLM classifications for the enclaves representing the MDBs running in the SDSR03 server are:

- Enclaves running in the SDSR03 server with a Transaction Class other than MDB, MDBD, MDBQ, and MDBT are classified with the server default transaction class of WASLO.
- Message Driven Beans without a specific Transaction Class assigned are classified with the MDB default transaction class.
- Enclaves with a transaction class value of MDB, MDBQ, and MDBT are classified:
  - Transaction Class MDB → associated with service class WASDF
  - Transaction Class MDBD → associated with service class WASLO
  - Transaction Class MDBQ → associated with service class WASLO
  - Transaction Class MDBT → associated with service class WASHI

To be able to associate an MDB transaction to a transaction class, you need (similar to the transaction mapping file) an .xml file where the association is defined, and you must define it to the server through the administration console.

From the initial panel, select **Environment** → **Manage WebSphere Variables**, where you can specify the path and name of the mapping file for the variable *endpoint_config_file* (for example, u/franck/MDBClass6356.xml, as shown in Figure 9-10).



WebSphere Variables >

New

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Configuration

General Properties

Name	* endpoint_config_file	Specifies the symbolic name representing a physical path or URL root.
Value	/u/franck/MDBClass6356.xml	Specifies the absolute path that the symbolic name represents.
Description		Provides an optional description for your administrative records.

Figure 9-10 Environment variable definition

A transaction class mapping file allows us to associate an incoming MDB to a specific transaction class. At execution time, WLM will use the transaction class to associate the work request to a service class. Example 9-6 shows a transaction class mapping file.

Example 9-6 MDB transaction mapping file

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE InboundClassification SYSTEM "InboundClassification.dtd">
<InboundClassification type="mdb" version="1.0">
   <endpoint type="messagelistenerport"
      name="ListenerPort"
      defaultclassification="MDBD">
    <classificationentry selector="Operation='Query'"
                                  classification="MDBQ"/>
    <classificationentry selector="Operation='Trade'"
                                  classification="MDBT"/>
   </endpoint>
</InboundClassification>
```

In Example 9-6, the endpoint element describes each listener port that is defined in the server that you want to associate transaction classes with the MDB. The listener port to which a endpoint element applies is specified by the name attribute. The value of the name attribute must be the name of the listener port, as specified in the administration console for the server. The default transaction class associated with MDBs assigned to the listener class is specified by the defaultclassification element and will override the WebSphere default transaction classification assignment value (that is, MDBD).

In our example, we have a listener port called ListenerPort with the default transaction class assigned to ‘MDBD’.

You can have further classifications within the endpoint element, as shown in Example 9-6 on page 279. The *classificationentry* element applies only to the MDB whose selector clause appears in the selector attribute of the element. The MDBs, whose selector clauses matches this classification entry, will be assigned to the enclave.

After you have completed your definitions, the WebSphere environment variable named endpoint_config_file must point to the transaction class mapping file, which is normally set at the server level using the administration console, as shown in Figure 9-10 on page 279. You need to recycle the server in order to pick up the new definitions.

9.4 Types of goals

There are two predominant types of goal for this environment: Velocity goals to be assigned to control region servant address spaces and response time goals for the enclave because of the transactionality of this type of workload.

For the enclave, as discussed earlier, you can classify them and assign different goals depending on the type of workload and your service level agreement.

9.5 Reporting

From a reporting point of view, you need to evaluate performance data for all of the elements that are part of the WebSphere environment: servant regions, control regions, and the transactional part with enclaves. If you have created specific reporting classes associated with these elements, you can use the report class section in the RMF Workload Activity report and on the SYSINFO report to understand how the different elements behave.

Example 9-7 on page 281 shows an extract from the RMF Workload Activity report where the control region, servant regions, and the enclaves are associated to a different reporting class to better visualize their behavior.

Interesting information from the report is:

Transaction/second In the enclave reporting class session, the fields TRANSACTIONS AVG, TRANSACTIONS MPL, and AV ENC are equal and do not represent the number of enclaves or transactions processed in this interval of time. It represents the average number of enclaves or transactions active in this interval of time.

Response time In the enclave reporting class session, the field TRAN-TIME ACTUAL should be close to the value reported in EXECUTION time. The time includes the time waiting on the WLM queue.

CPU and service rates The CPU service units and the service unit per second are reported in the SERVICE CPU field and in the /SEC. The field APPL% represents the number of engines required to drive the work in the service class (the portion of this service class accounted against this reporting class).

Delays The field QMPL identifies how long this workload has been waiting for servant regions.

APPL% This field represents the number of engines (CP) required to drive the work in the service or report class.

CPU seconds per transactions This field represents the ServiceRate TCB per second / ENDED Transactions.

Example 9-7 Extract of RMF Workload Activity report

```
REPORT BY: ... REPORTCLASS=WAS       - Control Region
   TRANSACTIONS     TRANS.-TIME SS.TTT ---SERVICE--   --SERVICE RATES--
   AVG       1.00   ACTUAL           0 IOC        0   ABSRPTN     89615
   MPL       1.00   EXECUTION        0 CPU   522567   TRX SERV    89615
   ENDED        0   QUEUED           0 MSO    10159K  TCB          39.9
   END/S     0.00   R/S AFFINITY     0 SRB    61728   SRB           4.7
   #SWAPS       0   INELIGIBLE       0 TOT    10743K  RCT           0.0
   EXCTD        0   CONVERSION       0 /SEC   89630   IIT           0.0
   AVG ENC   0.00   STD DEV          0                HST           0.0
   REM ENC   0.00                                     APPL %       37.2

REPORT BY: ... REPORTCLASS=WASS      - Server Regions
   TRANSACTIONS     TRANS.-TIME SS.TTT ---SERVICE--   --SERVICE RATES--
   AVG       3.00   ACTUAL           0 IOC        0   ABSRPTN    183074
   MPL       3.00   EXECUTION        0 CPU   214567   TRX SERV   183074
   ENDED        0   QUEUED           0 MSO    30667K  TCB          39.9
   END/S     0.00   R/S AFFINITY     0 SRB    18553   SRB           4.7
   #SWAPS       0   INELIGIBLE       0 TOT    30900K  RCT           0.0
   EXCTD        0   CONVERSION       0 /SEC  259664   IIT           0.0
   AVG ENC   0.00   STD DEV          0                HST           0.0
   REM ENC   0.00                                     APPL %       15.0

REPORT BY: ... REPORTCLASS=WASE      - WebSphere Enclaves (Transactions)
   TRANSACTIONS     TRANS.-TIME SS.TTT ---SERVICE--   --SERVICE RATES--
   AVG     301.24   ACTUAL         332 IOC        0   ABSRPTN       118
   MPL     301.24   EXECUTION      323 CPU     3454K  TRX SERV      118
   ENDED   107561   QUEUED           4 MSO        0   TCB         261.4
   END/S   903.87   R/S AFFINITY     0 SRB        0   SRB           0.0
   #SWAPS       0   INELIGIBLE       0 TOT     3454K  RCT           0.0
   EXCTD        0   CONVERSION       0 /SEC      18   IIT           0.0
   AVG ENC 301.24   STD DEV         66                HST           0.0
   REM ENC   0.00                                     APPL %      215.8
             EX    PERF  AVG   --USING%--   ----- EXECUTION DELAYS % -----
            VEL    INDX ADRSP   CPU   I/O   TOTAL   CPU  QMPL
   GOAL     40.0%
   ACTUALS  45.3%   .89  13.4   0.1   0.0    36.1  23.5  12.6
```

Example 9-8 on page 281 shows an RMF Monitor III SYSINFO report where you can see that the enclaves have been processed, and you can use the service class or report class to verify the resource consumption.

Example 9-8 RMF Monitor III SYSINFO report

```
                       RMF V1R5   System Information             Line 1 of 21
Command ===>                                                  Scroll ===> CSR

Samples: 97      System: SC69  Date: 04/06/05  Time: 14.13.20  Range: 100   Sec

Partition:    A0A     2084 Model 318          Appl%:       2  Policy: SPSTPC
CPs Online:   2.0     Avg CPU Util%:    7     EAppl%:      5  Date:   04/04/05
IFAs Online:    -     Avg MVS Util%:    7     Appl% IFA:   -  Time:   17.54.29

Group     T WFL --Users--  RESP TRANS -AVG USG-   -Average Number Delayed For -
             %   TOT  ACT  Time  /SEC PROC  DEV   PROC  DEV STOR SUBS OPER  ENQ

*SYSTEM      93   98    0        0.12  0.2  0.1    0.0  0.0  0.0  0.0  0.0  0.0
*TSO        100    6    0        0.12  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*BATCH             0    0        0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
```

```
*STC         92   85    0        0.00  0.1  0.1    0.0  0.0  0.0  0.0  0.0  0.0
*ASCH              0    0        0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*OMVS              7    0        0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
*ENCLAVE    100    0  N/A         N/A  0.0  N/A    0.0  N/A  0.0  N/A  N/A  N/A
DB2RES    W 100    0    0  .021 16.53  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
WASHI     S 100    0    0  .021 16.53  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
OTHER     W        7    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
VEL70     S        7    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
SYSTEM    W  92   84    0  .000  0.00  0.1  0.1    0.0  0.0  0.0  0.0  0.0  0.0
SYSSTC    S 100   62    0  .000  0.00  0.1  0.1    0.0  0.0  0.0  0.0  0.0  0.0
SYSTEM    S  85   22    0  .000  0.00  0.1  0.0    0.0  0.0  0.0  0.0  0.0  0.0
TSO       W 100    6    0  .089  0.12  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
ITSOTSU   S        3    0  .002  0.02  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
ITSOTSY   S 100    3    0  .106  0.10  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
WBIFN     W        1    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
VELRRS    S        1    0  .000  0.00  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
WASE      R 100    0    0  .021 16.53  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
OTHER     R  92   84    0  .000  0.00  0.1  0.1    0.0  0.0  0.0  0.0  0.0  0.0
RTSO      R 100    6    0  .089  0.12  0.0  0.0    0.0  0.0  0.0  0.0  0.0  0.0
```

For more information about performance data about enclaves, refer to 8.3.7, “Enclave reporting” on page 240.

10

UNIX System Services considerations

In this chapter, we provide guidelines for defining and managing a UNIX System Services (USS) workload under Workload Manager (WLM).

We provide the basic USS concepts and how to integrate USS work in an WLM environment.

10.1 General considerations

Before going deeper into the USS workload management in WLM, we introduce basic concepts and terminology.

10.1.1 Unix System Services terminology

These are the most common concepts and terms used in UNIX environment:

- *Kernel*: The heart of the UNIX operating system is the kernel. The kernel is similar to the MVS BCP. It is responsible for interfacing between programs and hardware, scheduling work, I/O management, and storage management. The kernel also manages the work that is in the system.
- *Process*: In UNIX, a unit of work is a process. In MVS, when a non-UNIX user logs on, an address space is created with multiple tasks (TCBs) running in the address space to represent that user. In the UNIX environment, when a user logs in, a process is forked to allow the workspace to be dedicated to the new user. In z/OS, this new process is really an address space. This newly forked unit of work can fork additional units of work to create new processes, and parameters (arguments) are passed to the new process.
- *Fork*: One process can create another process by the use of the fork command. Fork is similar to an MVS attach in that it creates another unit of work that can execute independently. The difference in UNIX, however, is that the created process (called the *child* process, PID) has access to all of the open files of the creating process (called the *parent,* PPID). The child does not have to open the files, because they are already open. In z/OS UNIX, a new address space is created for each forked process. The address spaces are created by the WLM and managed in a WLM Application Environment.
- *Spawn*: In most UNIX programs, the fork of a process is followed by an exec to run the new program. When applications are ported to z/OS UNIX, one performance improvement that should be made to applications is to replace the fork and exec combination with an z/OS-specific service called spawn. Spawning a process on z/OS simply means a new process will be forked, and the new program will be executed all at once. It creates a child process that will execute a different program than the parent, either in a new address space scheduled by WLM, or in the same address space as the parent process (local spawn).
- *Threads*: A program executing in a process is actually executing on a thread within a process. Threads are separate dispatchable units of work within a process, and are a way for an application to execute independent units of work currently. The benefits are that all the threads in a process can share all the resources owned by the process. Some applications have just one or a few threads per process, and others have many threads per process.
- *Syscall*: Is a request by an active process for a system service by the kernel.
- *Daemon*: Long-running processes are called daemons and are similar to MVS-started tasks.

10.1.2 Unix System Services address spaces

In z/OS, USS code runs in different address spaces:

- OMVS: Address space containing support for Unix System Services
- BPXOINIT: Initialization process routine (process ID 1)
- BPXAS: Initiator started by WLM and used to create an address space for a forked process

Figure 10-1 shows the basic USS address spaces.

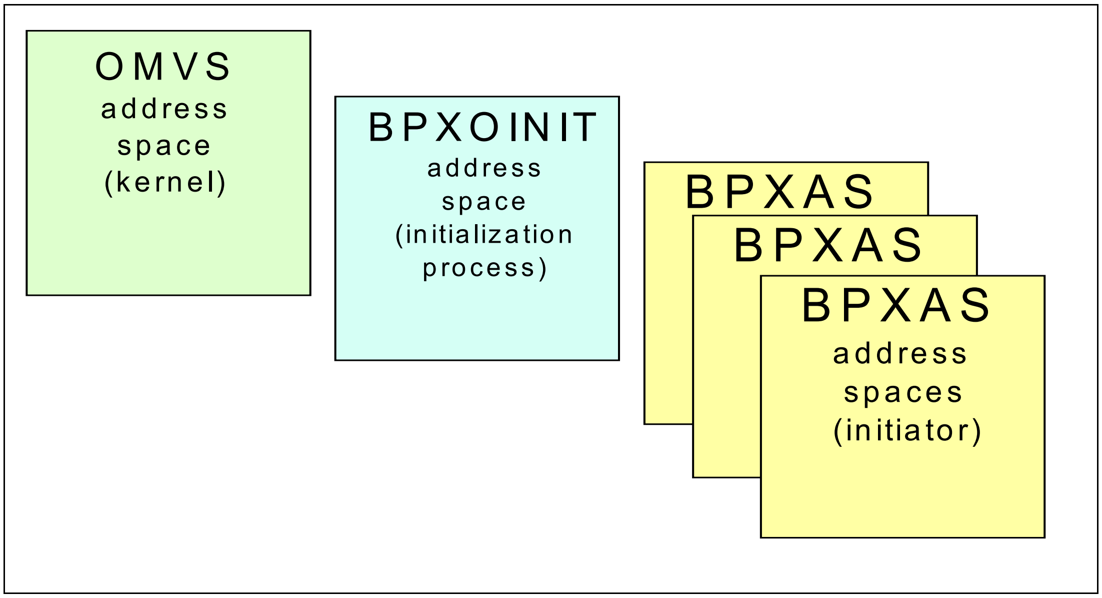


Figure 10-1 USS address spaces

You can execute UNIX work in several ways:

- Daemon: Process running in the background, not with any particular user.
- Dubbed address spaces: Traditional z/OS address space that calls a USS function. When a program uses a z/OS UNIX Assembler callable service, the z/OS address space will be dubbed a z/OS UNIX process. The address space will get a PID. The dub will not result in a new address space.
- Batch: The BPXBATCH z/OS utility allows UNIX processes to execute in an MVS batch environment.
- TSO: OMVS and ISHELL commands.
- Traditional USS programs: For example, rlogin and telnet requests coming from remote clients.

Figure 10-2 on page 286 summarizes these concepts.

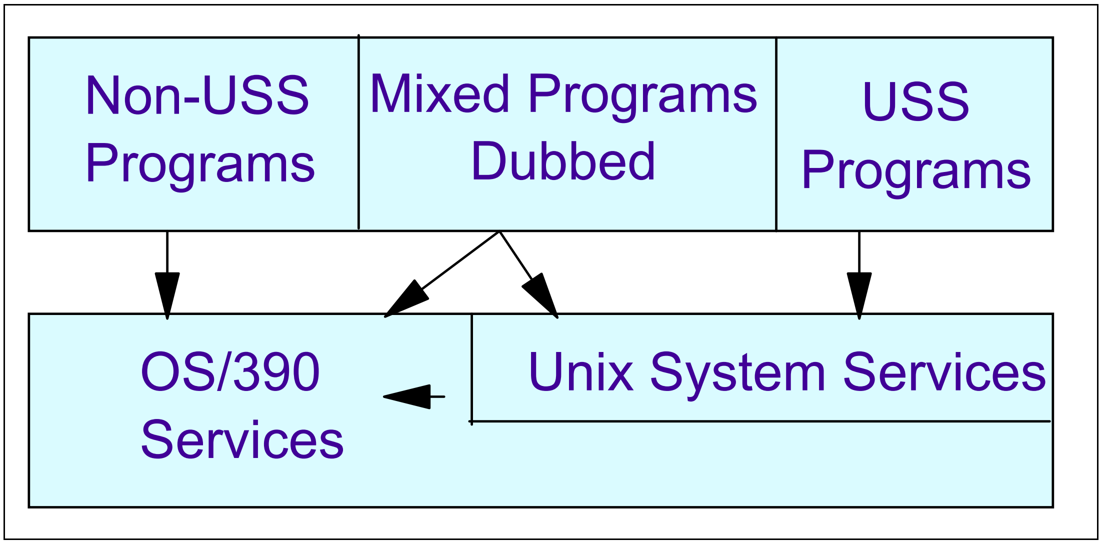


Figure 10-2 Work in z/OS USS environment

10.1.3 USS and WLM

USS uses WLM for three purposes:

- To create and maintain a pool of address spaces for fork and spawn function calls.
- To classify and manage all of the UNIX System Services address spaces.
- Enclave management across fork and spawn.

UNIX Services fork and spawn function calls

WLM is used to create and delete the UNIX Services fork and spawn address spaces.

WLM creates server address spaces (BPXAS) on demand and maintains a free pool of server address spaces to be reused by subsequent fork and spawn requests. These server address spaces are known as *WLM fork initiators*. An idle BPXAS Initiator will time out after 30 minutes of idle time.

A request for a WLM fork initiator is synchronous if the free pool of initiators is not empty, as shown in Figure 10-3 on page 287. When a request arrives, WLM resumes the SRB of the initiator task. When awakened, the initiator task correlates itself to the pending fork/spawn request. When the child task ends, its initiator SRB is suspended, and the initiator returns to the free pool.

> **Note:** The creation of these WLM fork initiators is based solely on demand, not on goal management.

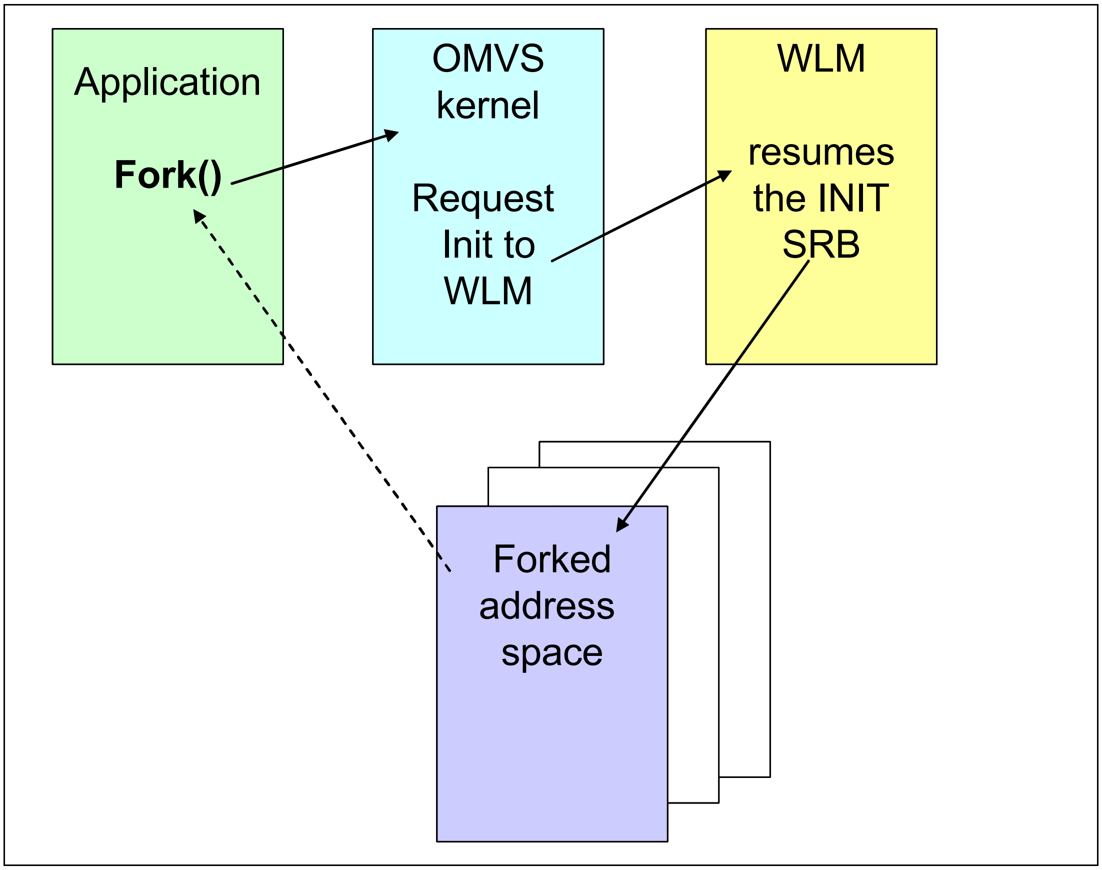


Figure 10-3 WLM fork and spawn synchronous processing

If the fork initiator pool is empty, WLM returns a non-zero return code to the USS, creates an ECB, and the application goes into a wait state, as shown in Figure 10-4. The request for an initiator is queued to WLM, which creates a new fork initiator using the SYS1.PROCLIB(BPXAS) procedure, adds it to the pool, and posts the ECB.

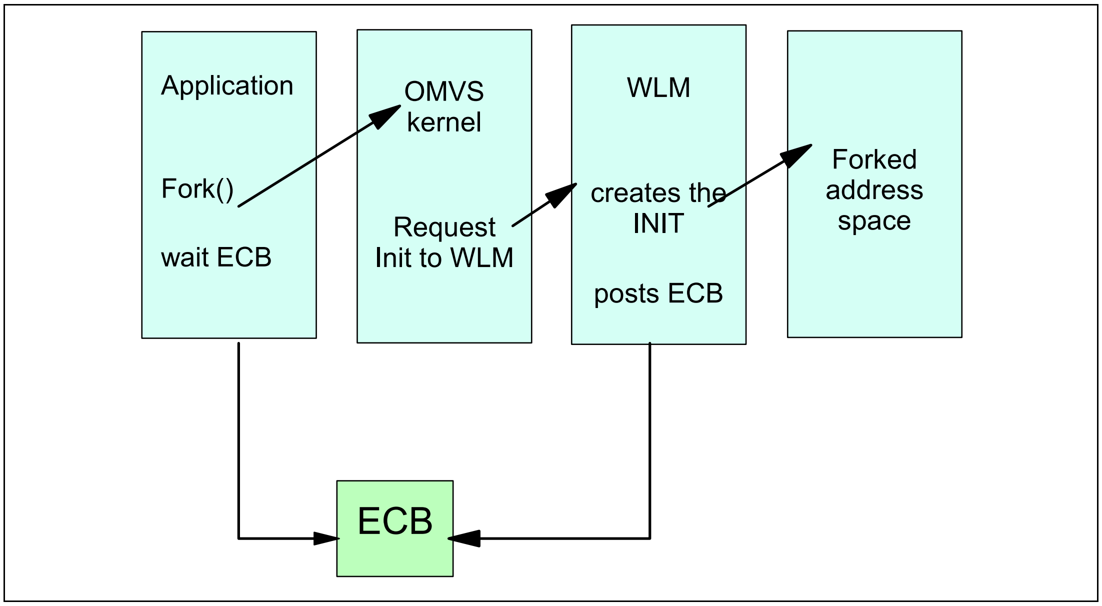


Figure 10-4 WLM fork and spawn asynchronous processing

Note: You do not need any tuning or definition in order to use WLM for the fork and spawn function calls.

WLM uses the BPXAS procedure in SYS1.PROCLIB to start a new initiator address space. The JOB and EXEC card information used to create these address spaces is overwritten when an actual fork or spawn is processed. During system initialization, running initialization routines can cause the creation of some BPXAS initiators (around 10), which become idle after initialization.

There are situations where you want to stop these initiators before they time out (for instance, if you want to shut down your system). However, you cannot stop your JES subsystem cleanly without stopping the idle BPXAS address spaces. To stop these address spaces, use the following command:

```
F BPXOINIT,SHUTDOWN=FORKINIT
```

You can display initiators by using an MVS DISPLAY command:

```
D A,BPXAS
```

Example 10-1 Displaying the WLM-created fork and spawn address spaces

```
BPXAS    BPXAS    IEFPROC  OWT  IO  A=006F   PER=NO   SMC=000
                                    PGN=N/A  DMN=N/A  AFF=NONE
                                    CT=000.064S  ET=01729.53
                                    WUID=STC07819 USERID=OMVSKERN
                                    WKL=OTHER    SCL=VEL70    P=1
                                    RGP=N/A      SRVR=NO  QSC=NO
                                    ADDR SPACE ASTE=03924BC0
BPXAS    BPXAS    IEFPROC  OWT  IO  A=001B   PER=NO   SMC=000
                                    PGN=N/A  DMN=N/A  AFF=NONE
                                    CT=000.002S  ET=NOTAVAIL
                                    WUID=STC18253
                                    WKL=SYSTEM   SCL=SYSSTC   P=1
                                    RGP=N/A      SRVR=NO  QSC=NO
                                    ADDR SPACE ASTE=039236C0
```

This command shows you all active or idle (but still in the WLM pool) address spaces that UNIX System Services is using or can use for fork and spawn function calls. Address spaces where no elapsed time is available (ET=NOTAVAIL) are idle and stay in the pool for new fork or spawn calls.

Fork, spawn, and enclave propagation

Enclaves can be created by USS on specific services or by any authorized user.

If the calling parent task, which issues the fork or spawn instruction is in a WLM enclave, the child is joined to the same WLM enclave. This allows WLM to manage the parent and child as one “business unit of work” entity for system accounting and management purposes.

USS uses the enclave propagation mechanism when an enclave is found at time of fork or spawn.

After APAR OA05701, USS provides a new environment variable, _BPXK_WLM_PROPAGATE, that can be set to 'NO', indicating to the kernel that WLM enclaves *not* created by USS will not be propagated on any of its services. If set to YES (default), USS still propagates any enclave that is found at the time of a fork, exec, pthread_create, FastCGI, or osenv request.

Figure 10-5 shows an example of enclave propagation through USS.

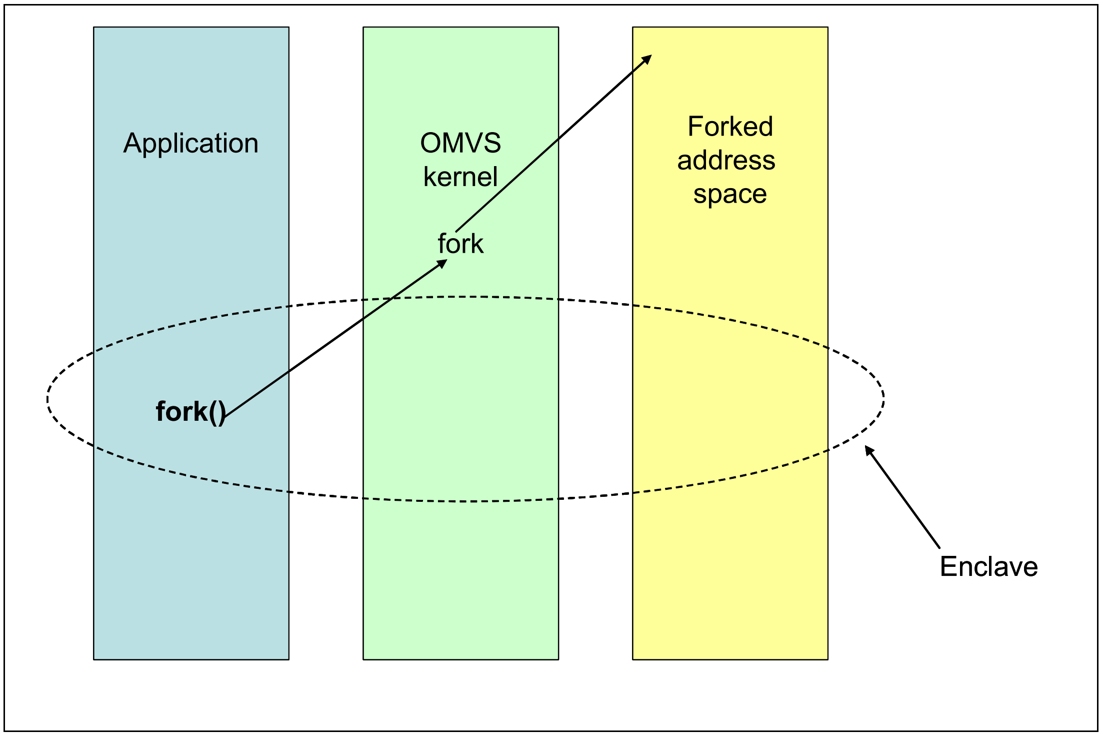


Figure 10-5 Forked child enclave propagation

10.2 Controlling the performance of your USS work

In an z/OS environment, you are able to control the performance of your UNIX work using functions provided by the UNIX services and WLM.

You must customize the WLM service definition for the UNIX System Services workload:

- Define services classes for USS work:
  - Define a service class for kernel, BPXOINIT, and BPXAS.
  - Define a service class for startup processes that are forked by the initialization process BPXOINIT.
  - Define a service class for other forked children.
  - Define a service class for daemons.
- Define classification rules for:
  - Kernel
  - Initialization process BPXOINIT
  - Startup processes forked by BPXOINIT
  - Forked child processes
  - Daemons

10.2.1 Types of goals and classification rules

In this section, we give general recommendations for defining your service classes and classification rules.

Definitions for USS started tasks

Put the kernel (classified with TN=OMVS) into a high-priority started task (subsystem type STC) service class. Another option is to keep the OMVS started procedure in the SYSSTC service class.

Put the initialization process BPXOINIT (classified with TN=BPXOINIT) into a high-priority started task (subsystem type STC) service class. Another option is to keep the BPXOINIT started procedure in the SYSSTC service class.

> **Note:** If not classified, OMVS and BPXOINIT will be assigned to the SYSTEM service class.

Put the fork initiators BPXAS (classified with TN=BPXAS) into a high-priority Started Task (subsystem type STC) service class. Another option is to keep the BPXAS started procedure in the SYSSTC service class.

Example 10-2 shows the classification rules for OMVS, BPXOINIT, and BPXAS address spaces.

Example 10-2 USS STC classification rules

```
                Modify Rules for the Subsystem Type     Row 1 to 10 of 85
Command ===> ____________________________________________ SCROLL ===> PAGE

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 ===>
          ---------Qualifier--------            -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: STCLOW
 ____  1  TN         OMVS     ___                  SYSSTC      ________
 ____  1  TN         BPXOINIT ___                  SYSSTC      ________
 ____  1  TN         BPXAS    ___                  SYSSTC      ________
```

Also, define a service class for daemons. Treat daemons the same as other started tasks of similar importance. Their service class normally has only one period with a velocity goal higher than the velocity goals of other forked children. If a daemon is critical for your business, consider putting it in the SYSSTC service class.

Definitions for OMVS subsystem type work

The OMVS subsystem type includes work requests processed in UNIX System Services forked address spaces. It does not apply to dubbed address spaces, such as batch programs that issue UNIX System Services callable services. If a TSO/E, batch, or started task uses UNIX System Services services, it does not change subsystem type and does not use subsystem type OMVS.

Subsystem OMVS classifies the following USS works:

- Startup processes forked by the initialization process BPXOINIT. They are identified by USERID=OMVSKERN. Give their service class a velocity goal that is higher than that of other forked children.
- Other forked child processes. You can assign them to different service classes based on USERID, ACCTINFO, TRXNAME, and so on. Specify a number of performance periods for the service class for forked children. Give performance periods for short-running work response-time goals or percentage response-time goals. Give velocity goals to performance periods for long-running work. You need to give this service class three performance periods, because it must support all types of kernel work, from short-interactive commands to long-running background work. You can set duration values using the SU/sec reported in the RMF monitor I Workload Activity report.

Note: The OMVS subsystem classification rule handles work requests processed in z/OS UNIX System Services forked children address spaces. The goals of the originating subsystem manage the work that comes from an enclave.

Example 10-3 shows the classification rules for the OMVS subsystem type. The first entry assigns the service class VEL70 to all of the processes forked by BPXOINIT that have the user ID equal to OMVSKERN. The other forked child processes fall into the default service class OMVSDEF, except for the listed TNs.

Example 10-3 Classification rules for OMVS subsystem work

```
                Modify Rules for the Subsystem Type       Row 1 to 6 of 6
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : OMVS        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 ===>
          --------Qualifier--------              -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: OMVSDEF     ________
 ____  1  UI         OMVSKERN ___                  VEL70       ________
 ____  1  TN         MIKE*    ___                  VEL40       ________
 ____  1  TN         MQVSBRK* ___                  VEL40       ITSOFN
 ____  1  TN         ERWCTG1  ___                  WAS48       ERWCTG1
 ____  1  TN         SDSR*    ___                  VEL70       WAS
 ____  1  TN         MQ8MBRK* ___                  STCHI       ________
```

The account information is normally inherited from the parent process of a UNIX System Services address space. In addition, when a daemon creates a process for another user, accounting data is taken from the WORKATTR of the RACF user profile. A user can also assign accounting data by setting the _BPX_ACCT_DATA environment variable, or by passing accounting information on the interface to the spawn or __spawn2 services.

The jobname for the UNIX System Services address space is used as the transaction name. By default, fork and spawn set the jobname value to the user ID with a number (1 to 9) appended. However, daemons or users with appropriate privileges can set the _BPX_JOBNAME environment variable to change the jobname for forked or spawned children. In this way, servers and daemons in UNIX System Services address spaces can easily be assigned performance attributes other than UNIX System Services address spaces.

The RACF user ID associated with the address space is used as the user ID work qualifier. This user ID is either inherited from the parent address process or assigned by a daemon process (for example, the rlogin or telnet daemon).

OMVS and TSO/E response time

When a user goes into the shell environment using the OMVS command from TSO/E, very long TSO/E response times (several seconds) might be recorded. This can affect those WLM goals for TSO users that are based on response time.

Normally, a TSO/E transaction starts when a user enters a command and ends when the command is completed. After the TSO/E command completes, a TGET WAIT is issued, indicating that the current transaction has completed and a new transaction will start when there is more work to be done.

In the OMVS shell environment, a transaction starts when a command is issued from the terminal. After the command is issued, polling is done to wait for output to return from the command. Every half second, there is a test for output and a test (TGET NOWAIT) for terminal input. This goes on for 20 seconds before the session goes into INPUT mode and does a TGET WAIT for terminal input only. TGET NOWAIT does not end the current transaction unless terminal input is found. If there is no more terminal input for over 20 seconds, the transaction does not end until the TGET WAIT is issued and the session goes into INPUT mode.

In effect, TSO/E users in the shell environment can experience response times of up to 20 seconds, often with little service consumption. Response times under 20 seconds occur only when users immediately enter the next command.

Example 10-4 shows the result of a test where we defined a service class with percentile response time of 20 seconds for 99% of transactions. After working under TSO, we activated the OMVS shell with the OMVS command and issued some short commands, waiting until the session went into Input mode. From the RMF Workload Activity report in Example 10-4, we can see that the OMVS transactions end after about 20 seconds and fall into buckets 5 and 6.

Example 10-4   RMF Postprocessor Workload Activity report bucket data

```
GOAL: RESPONSE TIME 000.00.20.000 FOR  99%

         RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
SYSTEM      ACTUAL%    VEL% INDX ADRSP   CPU  IFA  I/O  TOT  CPU                               UNKN IDLE  USG  DLY  USG  DLY QUIE

SC69           100     66.7  0.9   0.8   0.1  N/A  0.2  0.2  0.1                                0.3 99.2  0.0  0.0  0.0  0.0  0.0

                                       ----------RESPONSE TIME DISTRIBUTION----------
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
<  00.00.10.000          101              101         94.4        94.4  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.12.000          101                0         94.4         0.0  >
<= 00.00.14.000          101                0         94.4         0.0  >
<= 00.00.16.000          103                2         96.3         1.9  >>
<= 00.00.18.000          107                4          100         3.7  >>>
<= 00.00.20.000          107                0          100         0.0  >
<= 00.00.22.000          107                0          100         0.0  >
<= 00.00.24.000          107                0          100         0.0  >
<= 00.00.26.000          107                0          100         0.0  >
<= 00.00.28.000          107                0          100         0.0  >
<= 00.00.30.000          107                0          100         0.0  >
<= 00.01.20.000          107                0          100         0.0  >
>  00.01.20.000          107                0          100         0.0  >
```

Based on this behavior, because the wait time under OMVS is not idle time, we suggest that you use a percentile goal that takes into account the time spent by TSO users that work mostly under OMVS and not under TSO/ISPF, and that you set the goal accordingly.

This is applicable only to the OMVS command and not to the sessions started by the ISHELL command.

10.2.2 Control using UNIX functions

In a UNIX environment, there are some callable services that allow you to control the dispatching priority of a process, a process group, or a user. These services are setpriority(), chpriority(), and nice(). These services are also available in z/OS UNIX System Services and have an interface to WLM to provide the system control and monitoring support to which customers are accustomed.

To enable the nice(), setpriority(), and chpriority() functions, you have to use the PRIORITYGOAL statement in the BPXPRM*xx* parmlib member, and you must add additional service classes for kernel work. PRIORITYGOAL specifies a list of service class names that are used with the nice(), setpriority(), and chpriority() callable services when the system is running in goal mode.

Important: We do not recommend that you enable nice(), setpriority(), and chpriority() support. Instead, we recommend the exclusive use of normal WLM controls.

10.3 Reporting

You can collect OMVS data from multiple sources: SDSF, RMF III, RMF Postprocessor, and non-IBM monitors.

In this section, we discuss how to use several of these reports.

SDSF Processes panel

From the SDSF Main menu, select **PS** to display the panel. See Example 10-5.

Example 10-5 SDSF Process Display

```
SDSF PROCESS DISPLAY  SC69     ALL                     LINE 20-38 (39)
COMMAND INPUT ===>                                            SCROLL ===> CSR
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=SC69
NP   JOBNAME  Status                           State CPU-Time        PID       PPID ASIDX Command
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131110          1  0036 GFSCRPCD
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131111          1  0036 GFSCRPCD
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131112          1  0036 GFSCRPCD
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131113          1  0036 GFSCRPCD
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131114          1  0036 GFSCRPCD
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96     131115          1  0036 GFSCRPCD
     GIAN     SWAPPED,RUNNING                  1RI       5.18     131733          1  0075 EXEC
     NFSCLNT  RUNNING                          1R        3.96   16908292          1  0036 BPXVCLNY
     NFSCLNT  RUNNING                          1R        3.96   16908308          1  0036 GFSCMAIN
     EWLMMS2  RUNNING                          HR     2237.35   16908408   50462838  0028 /usr/lpp/
     RMFGAT   RUNNING                          1R    15612.82   33685507          1  004B ERB3GMFC
     CRON1    SWAPPED,OTHER KERNEL WAIT        1KI       0.90   33685512          1  0058 /usr/sbin
     PMAP     SWAPPED,FILE SYS KERNEL WAIT     1FI       0.73   33685533          1  004F PORTMAP
     NFSMVS   OTHER KERNEL WAIT                MK        6.38   50462727          1  004D GFSAMAIN
     NFSCLNT  MSGQ RECEIVE WAIT                1A        3.96   50462732          1  0036 BPXVCMT
     EWLMMS4  SWAPPED,WAITING FOR CHILD        1WI       0.49   50462838   67240053  001C /bin/sh
     DFSKERN  MVS PAUSE WAIT                   HG       61.23   67239941   84017166  005B DFSKERN
     DB8EDIST FILE SYS KERNEL WAIT             MF      138.15   67239968          1  0074 DSNVEUS3
     EWLMMS   SWAPPED,WAITING FOR CHILD        1WI       0.49   67240053          1  0037 -sh -c
```

Here you can see, for each process, information such as the process ID (PID), the parent process ID (PPID), the actual status of the process, the command that created the process, and so forth. The SDSF action character D displays additional information for the selected entry.

RMF III OPD

The RMF III OPD panel shows almost the same information as the SDSF PS panel. RMF III also gives you the ability to display historical data by pressing PF10 (backward reference) and PF11 (forward reference).

Example 10-6 RMF III: OMVS Process Data

```
RMF V1R5                     OMVS Process Data                  Line 1 of 39
Command ===>                                                 Scroll ===> CSR

Samples: 98     System: SC69  Date: 04/13/05  Time: 15.31.40  Range: 100   Sec

Kernel Procedure: OMVS     Kernel ASID: 0014       Option: PID     ALL
BPXPRM: OMVS=(6A)

-------------------------------------------------------------------------------
Jobname   User      ASID        PID       PPID  LW  State  Appl%  Total  Server

BPXOINIT  OMVSKERN  0086          1          0      MRI      0.0  19.43   FILE
DFS       DFS       0076     131074          1      HG       0.0  1.108   N/A
TCPIP     TCPIPOE   0069     131083          1      MR       0.5   3049   N/A
SYSLOGD5  OMVSKERN  0090     131085          1      1F       0.0  12.02   N/A
INETD1    OMVSKERN  0089     131087          1      1FI      0.0  0.666   N/A
TCPIP     TCPIPOE   0069     131088          1      1R       0.5   3049   N/A
```

Cursor sensitivity applies to the following fields:

- Jobname: The JOB report is invoked.
- Parent PID: Report is invoked again with the parent process shown on top of the window.
- Elsewhere on the process line: RMF OMVS Process Data - Details pop-up panel. Example 10-7 is an example of RMF OMVS Process Data - Details.

Example 10-7 RMF III OMVS Process Data - Details panel

```
                      RMF OMVS Process Data - Details

 Press Enter to return to the Report panel.

 Start Time/Date : 16.19.52 04/04/2005
 Command         : /usr/lpp/java/J1.4/bin/java
 Process-ID : 16908408    Parent Process-ID : 50462838
 Jobname    :  EWLMMS2    User              :   EWLMMS
 ASID       :     0040    Hexadecimal ASID  :     0028

 Appl%  :  0.4   Total CT     :  2242   LW-PID     :        0

 Server Information:
   Name : N/A
   Type : N/A    Active Files :   N/A   Max. Files :      N/A

 Process State : HR
 H: Multiple threads, pthread_create used
 R: Running
```

RMF Postprocessor

The RMF Kernel Activity report is generated using the SYSIN statement REPORT(OMVS).

The report is helpful for monitoring OMVS activities and configuration parameters.

In the SYSTEM CALL Activity section, you can obtain the number of system calls per second and the CPU time spent to process system calls. See Example 10-8.

Example 10-8 RMF Postprocessor OMVS System Call Activity data

```
                         OMVS SYSTEM CALL ACTIVITY
--------------------------------------------------------------------------------
                  MINIMUM  AVERAGE  MAXIMUM
--------------------------------------------------------------------------------
 SYSCALLS (N/S)   0.000    173.5     197K
 CPU TIME (H/S)   0.000    0.208    237.0
```

In the OMVS process Activity section (Example 10-9), the first row (MAXIMUM) shows the active values for:

- PROCESSES = MAXPROCSYS parameter
- USERS = MAXUIDS parameter
- PROCESSES PER USERS = MAXPROCUSER parameter

For each parameter, RMF shows you the CURRENT and OVERRUNS values:

- CURRENT: You need to monitor the AVERAGE values to determine if you set this parameter too high.
- OVERRUNS: Helps you to determine if you are experiencing slowdowns due to a value that is set too low.

Example 10-9 RMF postprocessor OMVS activity data

```
                                          OMVS PROCESS ACTIVITY
-----------------------------------------------------------------------------------------------------------
                          PROCESSES                        USERS                    PROCESSES PER USERS
 MAXIMUM  (TOT)            4096                              200                       32767
-----------------------------------------------------------------------------------------------------------
                  MINIMUM  AVERAGE  MAXIMUM      MINIMUM  AVERAGE  MAXIMUM      MINIMUM  AVERAGE  MAXIMUM
-----------------------------------------------------------------------------------------------------------
 CURRENT  (TOT)      38    43.79       53            1    1.734        3
 OVERRUNS (N/S)   0.000    0.000    0.000        0.000    0.000    0.000        0.000    0.000    0.000
```

11

Transactional workload considerations

This chapter provides guidelines to properly manage your overall CICS transactional workload environment.

This chapter offers you hints and tips to efficiently classify your CICS transactions and regions, and also explains how to set a report class to have detailed transaction information even when CICS is managed toward a region goal. This chapter explains and analyzes a CICS/batch CPU contention test.

11.1 General considerations for CICS

CICS has led the way in the implementation of new technology to support the transaction processing of your mission-critical business applications. This is just one reason why CICS is the leader in large-scale transaction processing today and is probably one of the reasons why you originally chose CICS as your transaction manager.

CICS evolved even further into an e-Business application server, providing improved application enabling capabilities as well as powerful Web and Java support.

So, properly defining CICS to WLM is key to guaranteeing the business quality of service of the IT infrastructure. This chapter provides information to set these definitions.

11.2 Types of goals

In this section, we describe the two types of management that you can use for CICS:

- Managing CICS toward a region goal

  The goal is set to a service class that manages the CICS address spaces: You can only use an execution velocity goal for this service class. This service class is assigned depending on the startup subsystem (JES or STC). In this case, we do *not* use the CICS subsystem classification rules.

- Managing CICS toward a transaction response time goal

  The goal can be average response time with or without percentile, and this is set to a service class assigned to a single transaction or a group of transactions. In this case, we use the JES or STC classification rules for the address spaces *and* the CICS subsystem classification rules for transaction.

You can dynamically switch from one mode to the other by changing the “Manage Region Using Goals of” field in the JES or STC subsystem classification rules panel. This can be done at the address space name level.

For example, you can use a transaction named TRX1 in a production or in a test environment. This transaction is classified in the CICS subsystem classification rules. If it is executed in the production environment, we want to use a response time goal (transaction management). However, if it is executed in the test environment, we want to use a velocity goal. This is because, in the test environment, we are managing the CICS region toward a region goal instead of a response time goal, which is considered inappropriate, because either the amount of transactions is not large enough or the response time is not suitable when using development tools.

If there are too many service classes for CICS or IMS with response time goals, the management of these transactions can become a little unpredictable, because WLM manages the regions based on the mix of transactions that runs in these regions. If this mix is too diverse or even if there are just a few service classes with response time goals but some of them with low activity, the mix can end in unpredictable results from a management point of view. This is one of the reasons why you might want to consider whether the response time goals for CICS and IMS are the best choice for your installation. Execution velocity goals might have their downsides, but they are much easier to correlate to what is running on the system.

WLM does not directly control each classified transaction, only the regions for the resources managed, such as CPU, processor storage I/O priority, and alias. Too much granularity for transaction service classes can be counterproductive. You usually obtain the best results when similar response time transactions are mapped to certain regions, but in the same installation, this might hard to achieve.

11.2.1 Region management goal

A single service class is needed to set up this management. This service class is assigned at the address space level in the JES or STC classification rules. See Example 11-1.

Example 11-1 Address space classification rule for region management

```
                Modify Rules for the Subsystem Type     Row 1 to 16 of 77
Command ===> ____________________________________________ SCROLL ===> PAGE

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 ===>
          --------Qualifier--------              -------Class-------- .......Manage Region
Action    Type       Name     Start              Service     Report  .......Using Goals Of
                                       DEFAULTS: SYSSTC      OTHER   ......
 ____  1  TN         SCSCP*   ___                ITSOCIRG    ________ .......REGION
```

Example 11-1 shows that all address spaces beginning with SCSCP have service class ITSOCIRG assigned. Notice that REGION is specified in the “Manage Region Using Goal of” field.

Service class ITSOCIRG is defined in Example 11-2.

Example 11-2 Service class definition for region management

```
                          Modify a Service Class              Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : ITSOCIRG
Description  . . . . . . . . . CICS regions
Workload Name  . . . . . . . . CICS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __
  __    1              2    Execution velocity of 70
```

Velocity is the goal for this service class.

CPU and storage-critical assignment in region management mode

If used, CPU critical has to be assigned at the service class level.

Long-term storage protection is assigned by specifying `YES` in the Storage Critical field in the rules for the address space.

Service class reporting in region management mode

In Example 11-3, we can see:

- The number of address spaces in this service class in the TRANSACTION column
- DASD I/O information
- CPU consumption in the field APPL% (expressed in percentage of a single engine utilization)
- Real storage used
- Service class samples, states, percentages, and performance index

Example 11-3 RMF SCPER report for ITSOCIRG service class

```
REPORT BY: POLICY=SPSTPC     WORKLOAD=CICS       SERVICE CLASS=ITSOCIRG   RESOURCE GROUP=*NONE      PERIOD=1 IMPORTANCE=2
                                                    CRITICAL     =NONE

   TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT   --DASD I/O--   ---SERVICE----   --SERVICE TIMES--   PAGE-IN RATES    ----STORAGE----
   AVG      6.00   ACTUAL                   0   SSCHRT 557.6   IOC    285185    TCB          191.6   SINGLE     0.0   AVG     10814.8
   MPL      6.00   EXECUTION                0   RESP     4.8   CPU      2204K   SRB           54.5   BLOCK      0.0   TOTAL   64888.2
   ENDED       0   QUEUED                   0   CONN     1.7   MSO         0    RCT            0.0   SHARED     0.0   CENTRAL 64888.2
   END/S    0.00   R/S AFFINITY             0   DISC     0.1   SRB    626491    IIT            2.1   HSP        0.0   EXPAND     0.00
   #SWAPS      0   INELIGIBLE               0   Q+PEND   0.3   TOT      3115K   HST            0.0   HSP MISS   0.0
   EXCTD       0   CONVERSION               0   IOSQ     2.7   /SEC    10384    IFA            N/A   EXP SNGL   0.0   SHARED    62.00
   AVG ENC  0.00   STD DEV                  0                                   APPL% CP      82.7   EXP BLK    0.0
   REM ENC  0.00                                               ABSRPTN  1731    APPL% IFACP    0.0   EXP SHR    0.0
   MS ENC   0.00                                               TRX SERV 1731    APPL% IFA      N/A

   GOAL: EXECUTION VELOCITY 70.0%     VELOCITY MIGRATION:   I/O MGMT  46.3%     INIT MGMT 46.3%

            RESPONSE TIME EX   PERF  AVG   --- USING% --- ---------- EXECUTION DELAYS % --------- ---DLY%-- -CRYPTO%- ---CNT%--    %
   SYSTEM                VEL% INDX ADRSP   CPU  IFA  I/O  TOT  I/O  CPU                          UNKN IDLE  USG  DLY  USG  DLY QUIE

   SC66        --N/A--    46.3  1.5   6.0  12.6  N/A 12.5 29.2 22.0  7.2                           0.0 45.7  0.0  0.0  0.0  0.0  0.0
```

In this report, there is no data showing the number of executed transactions and no data about the response time. The ENDED and ACTUAL fields are related to the CICS regions and report when an address space has stopped during the interval.

To obtain transaction-related information, even if you are in region management, refer to 11.5.1, “Reporting enhancement” on page 307.

11.2.2 Transaction response time management

There are two types of classification that are required for this management: address space classification in transaction management mode and transaction classification in transaction management model.

Address space classification in transaction management mode

You always need to classify the address spaces through the STC or JES subsystem classification rules. You only use the assigned service class goal during initialization and termination of the address space. The assigned service class goal is also used when there is no transaction during one minute. Be careful when assigning the velocity and importance goal value. Refer to Example 11-2 on page 299 to see how to assign a service class to a region.

Transaction classification in transaction management mode

Another set of service classes is necessary to assign response time goals to the transactions. Goals can only be average response times with or without percentile, and a service class must have only one period.

Other goals are not allowed:

- Velocity goals are not allowed because WLM does not know what happens at a transaction level inside the CICS region, but only knows resource usage for the CICS region itself. So, WLM does not know what each transaction is using or for what resource it is delayed.
- Multiple periods are unsupported for transactions in the CICS subsystem environments, because service units are accumulated for the address space, not for the individual transactions. Therefore, WLM cannot track a duration for those transactions.

See Example 11-4.

Example 11-4 Service class definition for response time

```
Modify a Service Class               Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : ITSOCI02
Description  . . . . . . . . . ________________________________
Workload Name  . . . . . . . . CICS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __
  __    1              2    80% complete within 00:00:00.025
```

Example 11-4 shows the definition of the ITSOCI02 service class. An average response time of 0.025 seconds is expected for 80% of the transactions and the goal has an Importance of 2.

CICS subsystem classification rules are used to assign service classes to a transaction or a group of transactions.

See Example 11-5.

Example 11-5 Transaction classification for response time management

```
                 Modify Rules for the Subsystem Type     Row 1 to 15 of 15
Command ===> ____________________________________________ SCROLL ===> PAGE

Subsystem Type . : CICS        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 ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                 Service     Report
                                         DEFAULTS: CICSDFLT    OTHER
 ____  1  SI         SCSCP*   ___                   ITSOCI01    ________
 ____  2    TN         DB2U     ___                 ITSOCI01    RCDB2U
 ____  2    TN         DB2N     ___                 ITSOCI01    RCDB2N
 ____  2    TN         DB2R     ___                 ITSOCI01    RCDB2R
 ____  2    TN         %X%      ___                 ITSOCI02    RCVSAM
 ____  2    TN         *        ___                 ITSOCI01    RCICOTH
```

Example 11-5 on page 301 shows how to assign ITSOCI01 to DB2U, DB2N, and DB2R transactions. The ITSOCI02 service class is assigned to all three character transactions having an X as the second character.

You can create service classes for groups of transactions with similar performance goals and business importance. Do not mix together the following types of transactions in the same service class or CICS regions, where possible:

- CICS-supplied transactions with user transactions.
- Routed with non-routed transactions.
- Conversational with pseudo-conversational transactions.
- Long-running and short-running transactions.
- Transactions with very different characteristics. For example, if you have two transactions that both have an average response time of one second, but one is CPU-intensive while the other is I/O-intensive, you should put them into different service classes.
- Transactions with very different response times. For example, do not put a transaction with an average response time of 20 seconds into the same service class as a transaction with an average response time of 0.5 seconds.
- Transactions with different levels of business importance.

The CICS address space receives resources and has a dispatching priority. A CICS transaction runs with the resources and dispatching priority of the CICS region. So, we recommend that you have more Application Owning Regions (AORs) than service classes on each z/OS image to make workload management via WLM easier. This decreases the chance of transactions with different goals executing in the same address space. If it happens, the transactions with less aggressive goals get a “free ride,” while WLM tries to honor the more aggressive and more important goals.

CPU and storage-critical assignment in transaction management mode

If you use CPU critical, you must assign it at the service class level. If any of the transactions executed in an address space have this attribute, then this address space will inherit this CPU protection.

Long-term storage protection is assigned by specifying `YES` in the Storage Critical field in the classification rules for specific transactions. After you specify `YES` for one transaction in a CICS service class, then all address spaces where this transaction executes are storage-protected. A preferred alternative is to assign storage protection to the region in which the transactions run. This ensures the region will be protected even when WLM loses sampling control (no activity).

To perform this setup, you need to modify the STC or JES classification rule.

11.3 Classification rules

The work qualifiers that CICS can use (and which identify CICS work requests to workload manager) are:

- LU: LU name
- LUG: LU name group
- SI: Subsystem instance (VTAM applid)
- SIG: Subsystem instance group
- TN: Transaction identifier
- TNG: Transaction identifier group
- UI: User ID
- UIG: User ID group

Note: Consider defining workloads for terminal-owning regions (TORs) only. Work requests do not normally originate in an application-owning region. The transactions are normally routed to an application-owning region from a terminal-owning region, and the work request is classified in the terminal-owning region. In this case, the work is not reclassified in the application-owning region. If work originates in the application-owning region, then it is classified in the application-owning region; normally, there would be no terminal.

You can use identifier group qualifiers to specify the name of a group of qualifiers; for example, GRPACICS could specify a group of CICS transaction identifiers (transids), which you can specify on classification rules by TNG GRPACICS. This is a useful alternative to specifying classification rules for each transaction separately.

11.4 CPU contention considerations

In this topic, we want to analyze the behavior of the CICS management modes in a CPU-constrained environment. The target system for our scenarios has two dedicated engines on a z900.

We are using TPNS to generate a constant transaction flow against our CICS subsystem while a variable amount of CPU bound batch jobs execute.

At the beginning of the test, CICS address spaces are the only workload running in the system. Then, one to four CPU bound jobs are started. Batch jobs are running in a service class with an execution velocity goal of 15% and an Importance of 5.

This scenario is executed in both CICS management modes (region and transaction). In the region management mode, CICS regions run in a service class with an execution velocity goal of 70% and an Importance of 2. In transaction management mode, CICS regions run in the same service class as the CICS regions, and the transactions are classified in Example 11-6 on page 304.

Example 11-6 Classification rules for transactions

```
                 Modify Rules for the Subsystem Type     Row 1 to 15 of 15
Command ===> ____________________________________________ SCROLL ===> PAGE

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--------                -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: CICSDFLT    OTHER
 ____  1  SI         SCSCP*   ___                  ITSOCI01    ________
 ____  2    TN         DB2U     ___                ITSOCI01    RCDB2U
 ____  2    TN         DB2N     ___                ITSOCI01    RCDB2N
 ____  2    TN         DB2R     ___                ITSOCI01    RCDB2R
 ____  2    TN         %X%      ___                ITSOCI02    RCVSAM
 ____  2    TN         *        ___                ITSOCI01    RCICOTH
```

Example 11-6 shows the classification used for the scenario. The first level relates to the generic applid. The same service class, ITSOCI01, is assigned to all transactions, except the VSAM transactions that run in ITSOCI02 service class. Different report classes are assigned to the transactions.

We defined the ITSOCI01 service class with an average response time with percentile goal of 95% with a response time of 0.015 seconds and an Importance of 2. ITSOCI02 service class has an average response time with percentile goal of 85% with a response time of 0.025 seconds and an Importance of 2.

For both tests, we collect the following data from RMF reports:

- Velocity, CPU delay, and APPL% of the CPU bound jobs
- Velocity, CPU delay, and APPL% of the CICS region
- Response time of transactions

11.4.1 CICS and batch jobs contention analysis

The CICS transaction flow is constant. The number (or amount) of batch jobs varies from zero to four.

The Importance of the CICS goal is 2, while the Importance of the batch goal is 5.

Figure 11-1 on page 305 shows the velocity, CPU delays, and APPL% variation depending on the amount of batch jobs running.

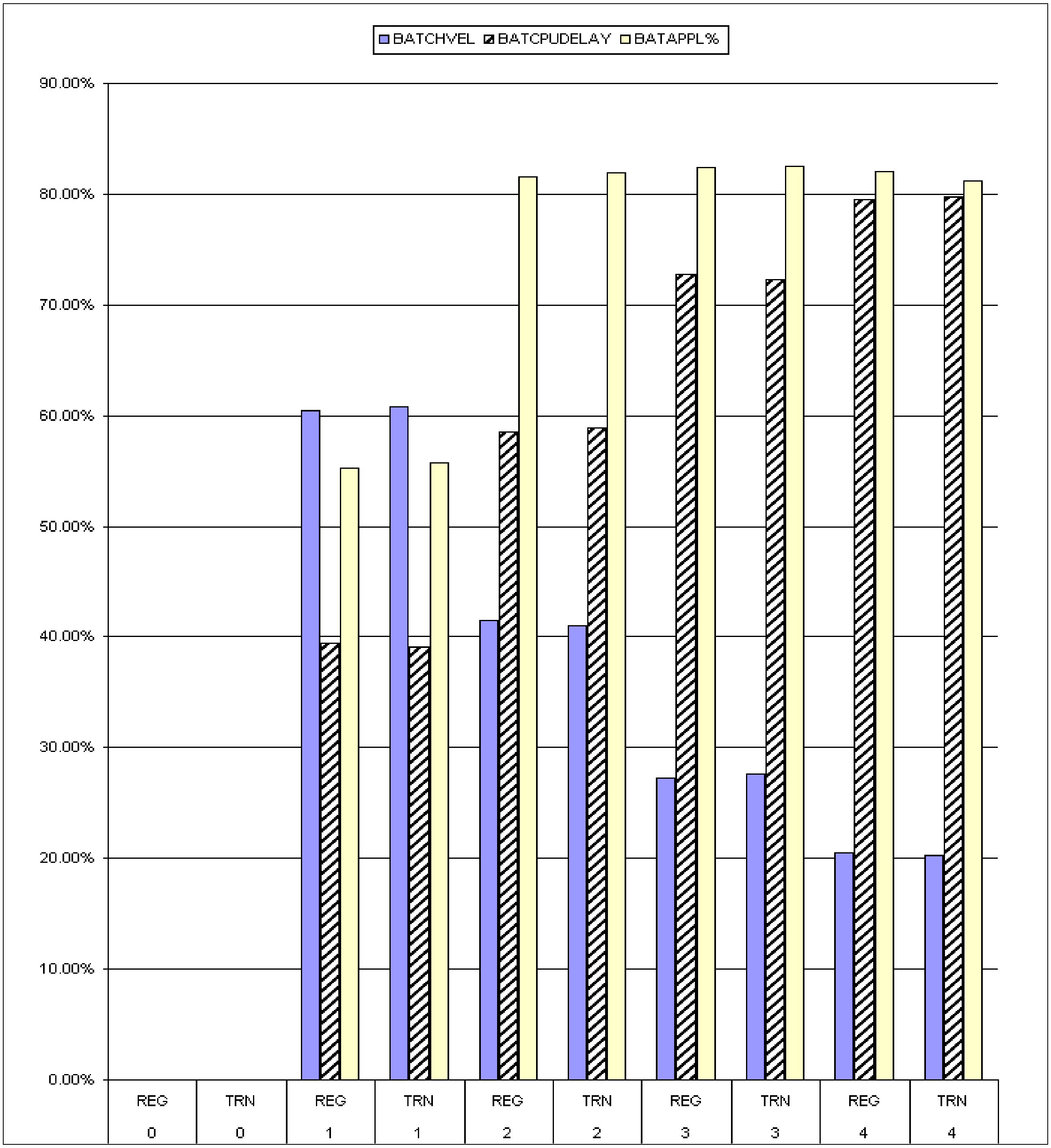


Figure 11-1 Jobs batch velocity, CPU delays, and APPL%

The X-axis in Figure 11-1 shows the amount of batch jobs varying from zero to four and, within this number, the management mode of CICS:

- REG for region management
- TRN for transaction management

Comparisons should be done within the same interval for batch jobs and CICS regions.

Execution velocity percentage obviously decreases as the number of batch increases because the number of engines is the same and there is more batch demand. The MVS busy percentage was 58% without the batch job, then 69% with one job, 86% with two jobs, and 100% with three and four jobs.

The increase of batch CPU delays is linear. The APPL% reached is 82% of one engine after the second job was started.

11.4.2 CICS regions contention analysis

Figure 11-2 shows the velocity, CPU delays, and APPL% variation of the CICS address spaces.

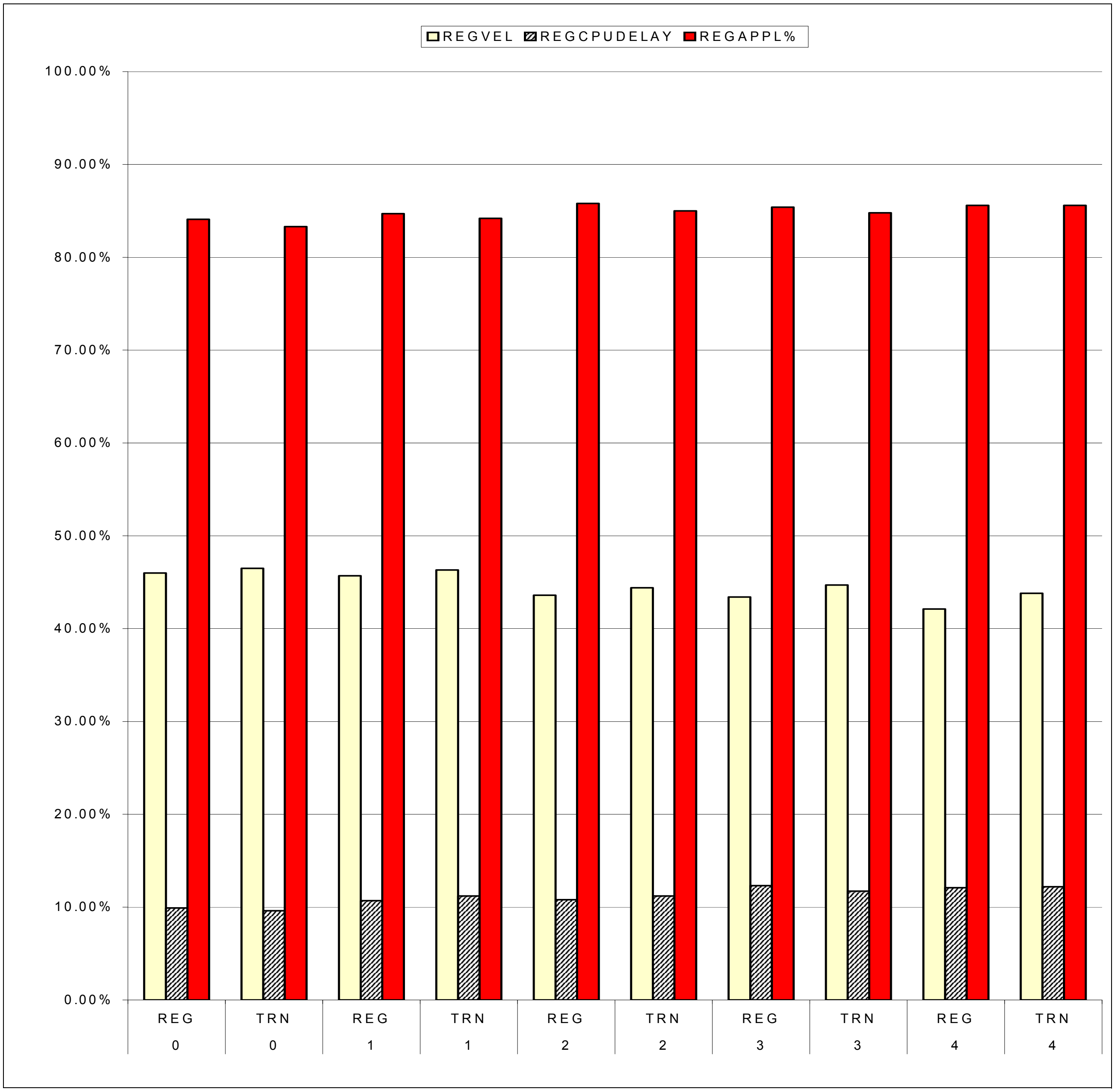
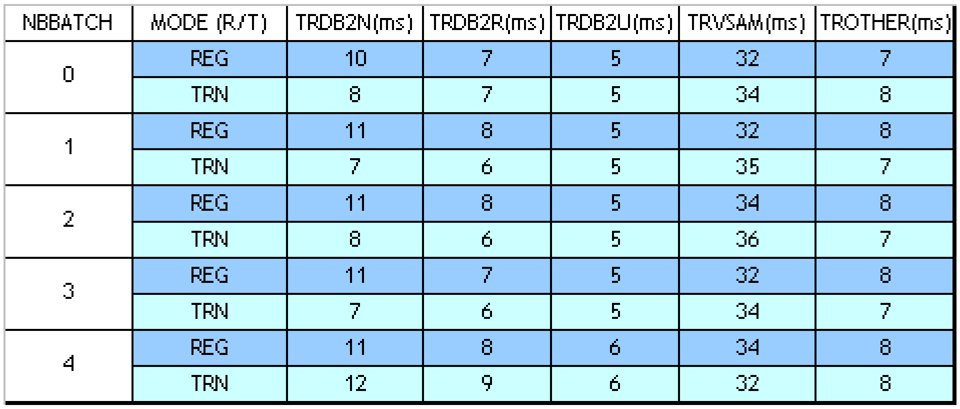


Figure 11-2 CICS regions' velocity, CPU delays, and APPL%

Figure 11-2 on page 306 shows an extremely slight increase of CPU delays due to the batch contention and a very small variation for the velocity. APPL% is steady. We can derive that, in both modes, batch jobs have no influence on the CICS throughput. You can verify this because the bars of the first interval (zero batch job) are nearly the same as the bars of the fifth interval with four jobs.

11.4.3 CICS response time contention analysis

We have shown that the batch jobs' CPU consumption has not affected the CICS region throughput. This section analyses whether there is an impact on CICS transactions' response time. See Figure 11-3.



NBBATCH	MODE (R/T)	TRDB2N(ms)	TRDB2R(ms)	TRDB2U(ms)	TRVSAM(ms)	TROTHER(ms)
0	REG	10	7	5	32	7
	TRN	8	7	5	34	8
1	REG	11	8	5	32	8
	TRN	7	6	5	35	7
2	REG	11	8	5	34	8
	TRN	8	6	5	36	7
3	REG	11	7	5	32	8
	TRN	7	6	5	34	7
4	REG	11	8	6	34	8
	TRN	12	9	6	32	8

Figure 11-3 Transactions' response time

Figure 11-3 shows the transactions' response time achieved during the test. From zero to four jobs running concurrently with CICS, we notice that response times have a very small increase for each management mode (region and transaction). This increase is slightly higher in region mode.

11.4.4 Contention analysis summary

The test shows that whatever the CICS goal management is, the WLM importance has properly protected the CICS environment.

If the batch workload has a different focus concerning data access (not the same DASD, not the same databases to prevent locks), CICS and batch can run at the same time when the WLM classifications are properly set.

11.5 Reporting

In a CICS/IMS environment, it is efficient and appropriate from a tuning and management point of view to have reporting information at the transaction level. Now, we explain how to get appropriate tuning and management even when CICS is managed toward a region goal.

11.5.1 Reporting enhancement

In this section, we describe the reporting enhancement that allows us to get a transaction detail report even if CICS is region-managed.

Our environment was made up of two TORs and four AORs that are executing a mix of VSAM-RLS and DB2 transactions generated by TPNS.

Service definition used for this test

For the six regions, we defined a service class ITSOCIRG with an execution velocity of 70% and an Importance of 2. This goal is assigned when we start the six STCs. See Example 11-7.

Example 11-7 Service class definition for region management

```
                          Modify a Service Class              Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : ITSOCIRG
Description  . . . . . . . . . CICS regions
Workload Name  . . . . . . . . CICS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __
  __    1              2    Execution velocity of 70
```

To assign this service class to the regions, we used the following parameters in the STC subsystem classification rules. See Example 11-8.

Example 11-8 STC classification rules

```
                Modify Rules for the Subsystem Type     Row 1 to 16 of 77
Command ===> ____________________________________________ SCROLL ===> PAGE

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 ===>
          --------Qualifier--------              -------Class-------- .......Manage Region
Action    Type       Name     Start               Service     Report  .......Using Goals Of
                                        DEFAULTS: SYSSTC      OTHER   ......
 ____  1  TN         SCSCP*   ___                 ITSOCIRG    ________ .......REGION
```

Example 11-8 shows the assignment of the service class ITSOCIRG: All our CICS regions are started tasks and their names begin with SCSCP. The “Manage Region Using Goals Of” field is obtained by pressing PF11 key two times.

Even if we specify region management, we define a response time goal service class for the transactions that are executed by these regions.

See Example 11-9 on page 309.

Example 11-9 Response time goal service class

```
                          Modify a Service Class               Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : ITSOCI01
Description  . . . . . . . . . ________________________________
Workload Name  . . . . . . . . CICS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description
  __
  __    1              2    95% complete within 00:00:00.015
```

Example 11-9 shows the ITSOCI01 service class that we assign to the transactions executed in the CICS regions through the CICS subsystem classification rules. In this case only, the defined goal has no effect on WLM management, because we have specified region management in the address space classification rule.

See Example 11-10.

Example 11-10 CICS subsystem classification rules

```
                 Modify Rules for the Subsystem Type     Row 1 to 14 of 14
Command ===> ____________________________________________ SCROLL ===> PAGE

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--------              -------Class--------
Action    Type       Name     Start               Service     Report
                                        DEFAULTS: CICSDFLT    OTHER
 ____   1  SI         SCSCP*   ___                 ITSOCI01    RCCICALL
```

Example 11-10 shows the classification assigned to our CICS system. The SI type qualifier is the CICS VTAM APPLID of our regions. We assigned a report class, RCCICALL, at the APPLID level, that receives the expected transaction level performance information. However, this simple setting produces the same detailed transaction activity report as if we were in response time management.

Example 11-11 on page 310 shows the RMF report of the RCCICALL report class. We have all of the expected information at the transactions level.

Example 11-11 Transaction activity report in region mode management

```
REPORT BY: POLICY=SPSTPC                         REPORT CLASS=RCCICALL                              PERIOD=1
                                                 HOMOGENEOUS: GOAL DERIVED FROM SERVICE CLASS ITSOCI01

TRANSACTIONS    TRANS.-TIME  HHH.MM.SS.TTT
AVG      0.00   ACTUAL                  10
MPL      0.00   EXECUTION                7
ENDED   43277   QUEUED                   0
END/S  144.26   R/S AFFINITY             0
#SWAPS      0   INELIGIBLE               0
EXCTD   40722   CONVERSION               0
AVG ENC  0.00   STD DEV                 25
REM ENC  0.00
MS ENC   0.00

           RESP  -------------------------------- STATE SAMPLES BREAKDOWN (%) -------------------------------  ------STATE------
SUB    P   TIME  --ACTIVE-- READY IDLE  -----------------------------WAITING FOR-----------------------------  SWITCHED SAMPL(%)
TYPE        (%)   SUB  APPL             CONV  I/O LOCK MISC                                                    LOCAL SYSPL REMOT
CICS  BTE  67.0  10.3   0.0   1.7  0.0  87.9  0.0  0.0  0.0                                                     88.8   0.0   0.0
CICS  EXE  21.4  73.0   0.0  13.5  5.4   0.0  5.4  0.0  2.7                                                      2.7   0.0   0.0
DB2   BTE   0.0   0.0   0.0   0.0  0.0   0.0  0.0  0.0  0.0                                                      0.0   0.0   0.0
DB2   EXE   0.6   100   0.0   0.0  0.0   0.0  0.0  0.0  0.0                                                      0.0   0.0   0.0
SMS   BTE   0.0   0.0   0.0   0.0  0.0   0.0  0.0  0.0  0.0                                                      0.0   0.0   0.0
SMS   EXE  24.8  25.6   0.0   0.0  0.0   0.0 65.1  9.3  0.0                                                      0.0   0.0   0.0

GOAL: RESPONSE TIME 000.00.00.015 FOR  95%

         RESPONSE TIME EX   PERF
SYSTEM      ACTUAL%    VEL% INDX

SC66          84.6      N/A  4.0

                                       ----------RESPONSE TIME DISTRIBUTION----------
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10   20   30   40   50   60   70   80   90   100
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|....|....|....|....|....|....|....|....|....|
<  00.00.00.007        29961            29961         69.2        69.2  >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<= 00.00.00.009        31523             1562         72.8         3.6  >>>
<= 00.00.00.010        32786             1263         75.8         2.9  >>
<= 00.00.00.012        34671             1885         80.1         4.4  >>>
<= 00.00.00.013        35386              715         81.8         1.7  >>
<= 00.00.00.015        36609             1223         84.6         2.8  >>
<= 00.00.00.016        37127              518         85.8         1.2  >
<= 00.00.00.018        37606              479         86.9         1.1  >
<= 00.00.00.019        38319              713         88.5         1.6  >>
<= 00.00.00.021        38611              292         89.2         0.7  >
<= 00.00.00.022        39072              461         90.3         1.1  >
<= 00.00.00.030        40128             1056         92.7         2.4  >>
<= 00.00.00.060        41634             1506         96.2         3.5  >>
>  00.00.00.060        43277             1643          100         3.8  >>>
```

This setting is useful when you do not want to or cannot manage your CICS at the transaction level, but still want detailed transaction level information.

This report can help you analyze your transaction response time and make a decision whether to migrate to transaction response time management. This can be useful when your CICS transactions are developed with certain fourth-generation languages where there is a single transaction name that manages the CICS context.

12

Uniprocessor considerations

As CPU speeds increase, many sites are moving to systems that are larger, but actually have fewer CPUs. This can cause difficulties, because as the number of CPUs decreases, the system becomes more sensitive to CPU intensive tasks. The extreme example is a single CPU system.

In this chapter, we investigate the way Workload Manager (WLM) manages access to the CPU, and we examine the implications for a single CPU system with our best practices recommendations.

12.1 CPU contention in a uniprocessor system

CPU contention is more of a problem on a single CPU system than on multiple CPU systems, because all lower priority tasks wait when any higher priority CPU bound task is running. You do not have any other CPUs that can handle other work.

Most work that runs on the system is relatively easily managed. It uses a small amount of CPU and then waits for something. This can be an I/O, a response from the user, or a number of other events. When it waits, the CPU is free for use by other tasks.

Some work, however, uses lots of CPU and rarely needs to wait for anything. This might be a job that performs complex calculations, with most or all of its data in memory. These tasks are referred to as *CPU bound*. Sometimes, tuning can actually produce these effects. A common tuning technique is to use buffers to eliminate I/O. By eliminating I/O, you eliminate one common reason for a task to wait and free the CPU.

On a two CPU system, it takes two simultaneous CPU bound tasks to force lower priority work to wait. With more CPUs, it takes even more simultaneous CPU bound tasks to force lower priority work to wait. With a single CPU, a single task can monopolize the whole system by itself, which means that CPU contention can become a problem even at low average utilizations on a single CPU system.

Upgrading to a faster CPU without increasing the number of CPUs is often not much help. A 50% faster processor might mean your online regions only hang for two minutes instead of three minutes, an improvement perhaps, but still an issue. We need to address the way to handle this configuration.

12.1.1 Dispatching priorities

The key to who gets to use the CPU is the dispatching priority. The dispatching priority is set by WLM to attempt to meet the goals of the workload.

Normally, you do not need to worry about dispatching priorities in a WLM system. Set the goals appropriately, and let WLM manage the priorities. However, when you have CPU contention problems in a single CPU system, it helps to understand the dispatching priorities to work out what is happening. The way that WLM manages dispatching priorities works best with a large pool of work that is served by many CPUs. When you only have a single CPU, it does not always work so well.

While there are many possible dispatching priorities, the actual numbers are irrelevant. When tasks are competing for the CPU, what matters is which task has the higher priority. The task with the higher priority will be able to use as much CPU as it likes before the lower priority task gets a share (ignoring capping, which only happens in specific circumstances). The lower priority task has to wait until the higher priority task voluntarily gives up the CPU.

For two tasks, A and B, there are only three possible dispatching priority arrangements:

- DP(A) > DP(B)

  B waits for A to release the CPU.
- DP(A) = DP(B)

  A and B share the CPU. If they both try to use it 100% of the time, they will receive approximately equal amounts of the CPU.
- DB(A) < DP(B)

  A waits for B to release the CPU.

We performed tests to measure how much service is received in different service classes running CPU bound work on a single CPU system. This allows us to see the effect of the WLM dispatching priority changes when different service classes are competing for the CPU, and the effect of different goals.

12.1.2 Test case

We created a simple REXX program that is designed to loop and consume CPU. The program reports every five seconds how many times it runs through the loop as a measure of how much service it received in that interval. The five second interval allows us to see what is happening within a single WLM adjustment interval.

Example 12-1 REXX program used for testing

```
/* rexx */

starttime = time(s)
lastinterval = starttime
loopcount = 0
intervaltime = 5
do forever
   if time(s) > lastinterval + intervaltime then
   do
      lastinterval = lastinterval + intervaltime
      outinterval = lastinterval - starttime
      say lastinterval outinterval loopcount
      loopcount = 0
   end
   else /* start another loop if we have caught up missed intervals */
   do
      loopcount = loopcount + 1
      x = 0
      do 33250 /* Adjust so that output is approx. the same as velocity */
         x = x + 1
      end
   end
end
```

We adjusted the loop so that the numbers reported by the job correspond approximately to the reported velocity; which meant it was simple to compare the service received with velocity goals.

We then submitted two jobs in each of three service classes, so that we could see how WLM handles the contention for the CPU. Because we had two jobs running in each service class, the highest expected velocity is close to 50% on a system with no other significant work. We ran two jobs in each service class so the velocities with which we were working would be lower. For some of the tests, we wanted goals of 30 or less in order to allow the service class to be capped.

The REXX exec recorded the time it started, so that if it received no service in an interval it would know that it was missed, and could write out 0 for that interval when it did receive service again. Initially, we found that some of the jobs did not receive enough service to even record the start time reliably, so we added a short, high importance first period to each of the service classes to get them started.

We only used batch jobs for these tests, but the principles apply equally well to other types of CPU bound work using similar goals.

We had two objectives with these tests:

- Run CPU bound work in three service classes and have all of them meet their goals.
- Ensure each service class receives at least some service. This avoids problems that can occur if a job is not dispatched at all. For example, a job needs to be dispatched to respond to the cancel command.

These tests might seem unlike real workloads, but they are designed to demonstrate the WLM response to CPU contention and the effects of changing dispatching priority. Most sites have some CPU bound programs. They might only run for a few minutes at a time, but while they run, you can see some of these effects on a single processor system.

Test one: Very high goals

The WLM goals for this test are shown in Table 12-1.

Table 12-1 Goals for test one

Service class	Goal	Importance
BATCHHI	Velocity 60	3
BATCHMED	Velocity 40	4
BATCHLOW	Velocity 20	5

Figure 12-1 shows the result of this test. Service class BATCHHI used all available CPU for most of the time. We can see that WLM briefly tried setting the dispatching priority for BATCHMED the same as BATCHHI, but the impact on BATCHHI was too severe, so the change was reversed. During this brief period, BATCHMED and BATCHHI received the same amount of service.

BATCHLOW received no service at all, and BATCHMED received no service for most of the test.

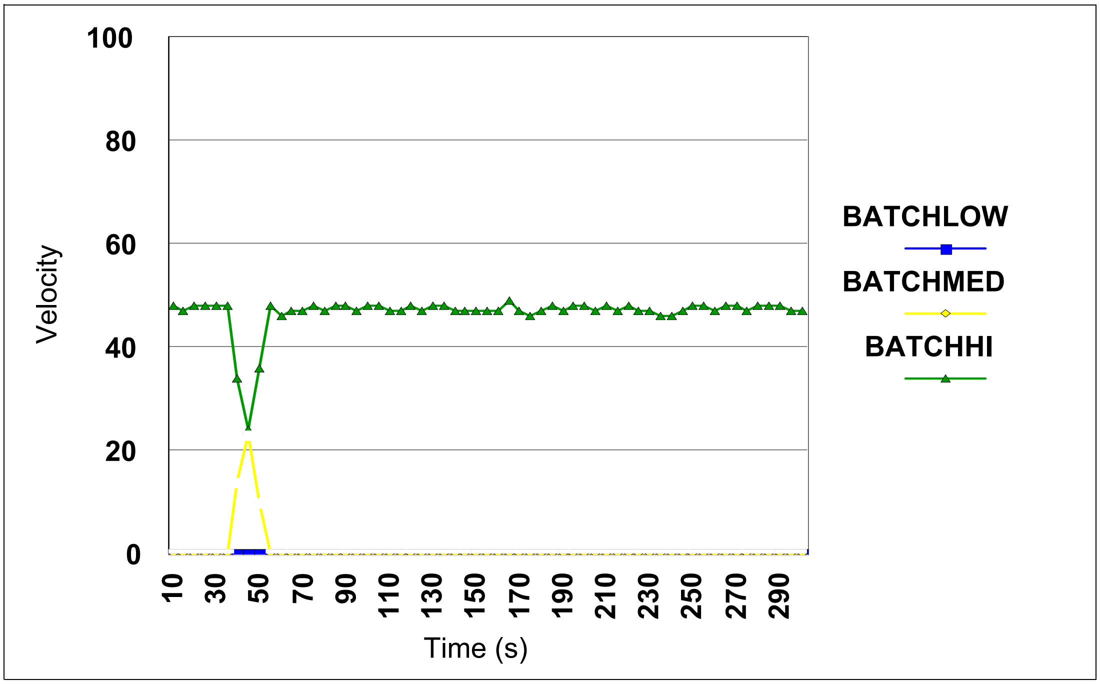


Figure 12-1 The result of very high goals

Test two: Reduced goals

For test two, we reduced the goals to try to make them achievable. We repeated the test several times, reducing goals in an attempt to find a level where each service class would run, while trying to maintain a distinction between the different goals.

In this test, we still found that the higher importance workload dominated the other work. The goals that we used are shown in Table 12-2.

Table 12-2 Goals for test two

Service class	Goals for three separate tests	Importance
BATCHHI	Velocity 30, 15, and 10	3
BATCHMED	Velocity 20, 10, and 5	4
BATCHLOW	Velocity 10, 5, and 2	5

The typical result is shown in Figure 12-2. WLM initially tries all three service classes at the same priority, and they all receive the same service. It then moves BATCHLOW below the others, and the service that BATCHLOW receives drops to 0. A short time later, BATCHMED is also moved below BATCHHI, and its service falls to 0.

The spike at the beginning is evidence of a common problem on uniprocessor systems. As the first CPU bound BATCHHI job started running, TSO was locked out for a short period. This stopped the second job in that service class from being submitted, so until WLM made an adjustment, only one job was running.

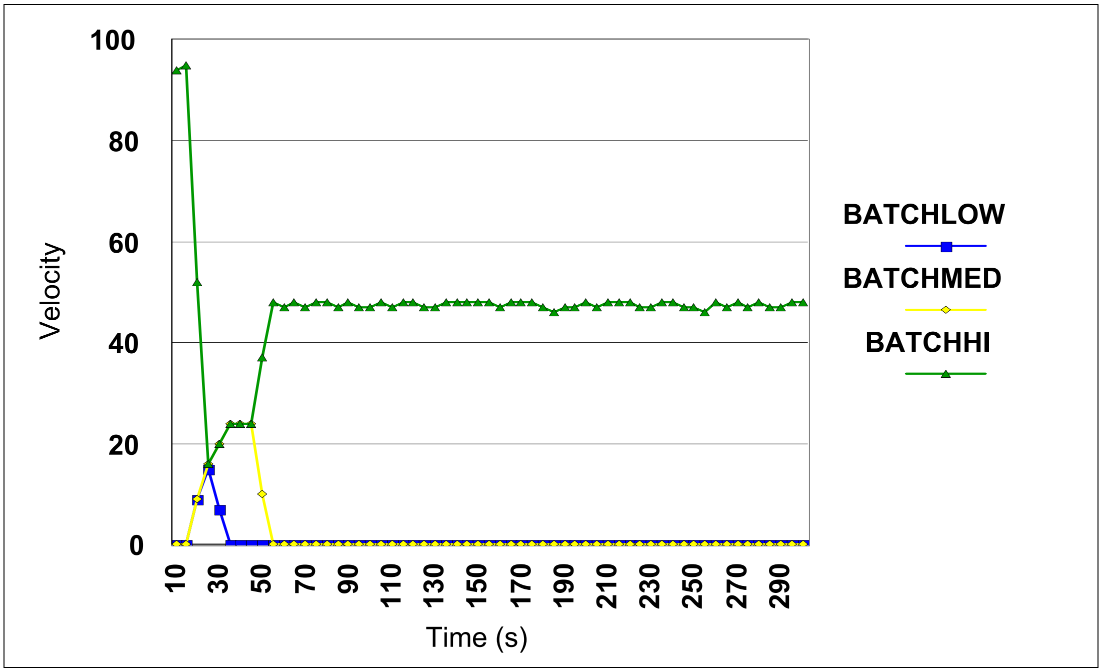


Figure 12-2 Typical result of lower goals

We were a little surprised that we were unable to get all of the work to run at the same time even with very low goals. Real world experience suggests that reducing the goals of CPU bound work will often help. This suggests that the work that we used was slightly too extreme, and it also indicates that WLM can be quite conservative in protecting high importance work.

Update: We ran this test again later. This time when the velocities were set to 10, 5, and 2, WLM set all three service classes to the same dispatching priority, and all three service classes received equal service. This was what we were expecting to see with low goals the first time we ran this test.

This demonstrates the problem with trying to control how WLM manages the work. WLM uses many different factors to decide what to do, and there are too many things that influence the outcome, even in a simple test.

It also shows the danger in repeating a test after you have written up the results.

Test three: Discretionary work

We performed some tests with the BATCHLOW service class set to discretionary. These tests demonstrate the effect of WLM capping overachieving service classes to give service to discretionary work.

Table 12-3 shows the goals that we used for test three.

Table 12-3 Goals for test three

Service class	Goals for three separate tests	Importance
BATCHHI	Velocity 15	3
BATCHMED	Velocity 10	4
BATCHLOW	Discretionary	

We had to run this test for a longer time before the situation seemed to stabilize. The results are shown in Figure 12-3 on page 317. This was the first test where we had each service class running consistently.

Initially, the results are similar to before. WLM set BATCHMED and BATCHHI to the same priority, and they received equal service. Then, BATCHMED is capped to try to give some service to the discretionary work. However, all the extra service is absorbed by BATCHHI. Both BATCHMED and BATCHHI are overachieving their goal, so they both get capped, and the discretionary work starts running.

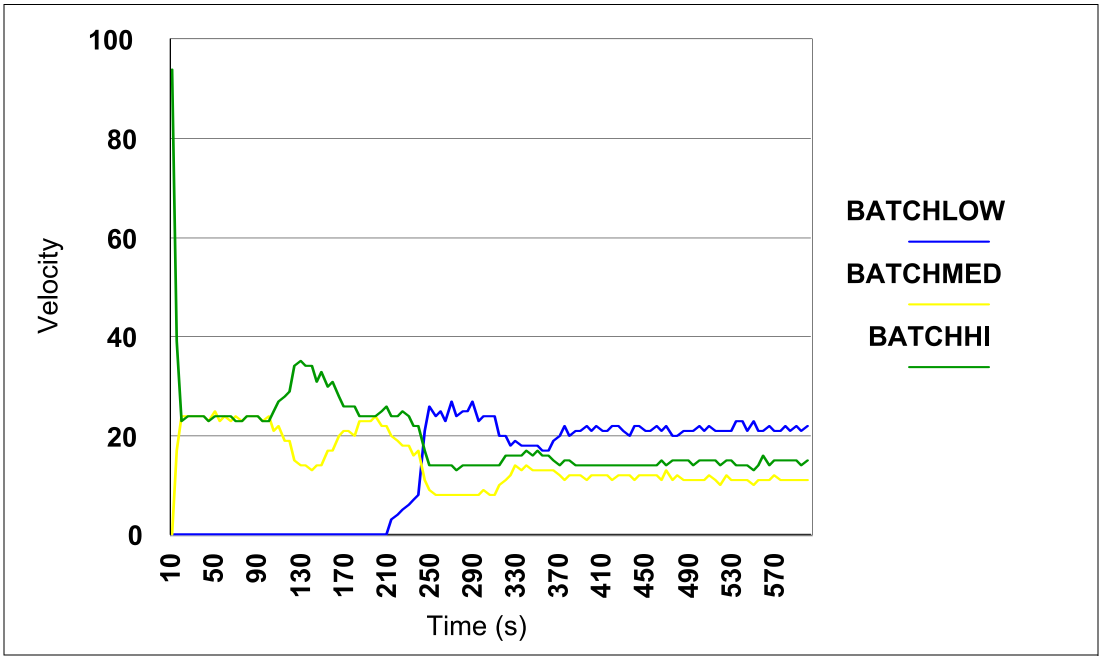


Figure 12-3 Results with discretionary work

There are several points to note about this result:

- In this situation, discretionary work received much better service than the same work with a low importance velocity goal in the previous tests.
- BATCHMED also received much better service than when there was no discretionary work. When BATCHLOW had a velocity goal, BATCHHI dominated the system. When BATCHLOW had a discretionary goal, BATCHHI was capped and BATCHMED benefited.
- BATCHHI and BATCHMED were both achieving the goals we set. Capping only occurs when a service class is significantly overachieving its goals. It is capped to a level where it is still overachieving, but by a limited amount.

In this test, BATCHLOW ended up with a velocity above the other two service classes. Do not read too much into that. The actual numbers are extremely dependent on the workload. If we had ten jobs in BATCHLOW, the reported velocity would have been more like 4%. The significant point is that all service classes receive service simultaneously, and the goals we set are achieved.

Test four: The effect of goals on capping

We repeated the previous test, but increased the goals of the BATCHMED and BATCHHI service classes. Table 12-4 shows the goals for this test.

Table 12-4 Goals for test four

Service class	Goals for three separate tests	Importance
BATCHHI	Velocity 30	3
BATCHMED	Velocity 20	4
BATCHLOW	Discretionary	

In this test, we can see that after some initial adjustments that were reversed, WLM again caps BATCHHI, which is overachieving its goal. The capping occurs because we have discretionary work, but the actual beneficiary is BATCHMED. BATCHMED absorbs all of the extra service, but does not achieve its goal, which means that it is not capped, and BATCHLOW still gets no service. However, this is still better than what we saw with a velocity goal for BATCHLOW. In that case, BATCHMED and BATCHLOW were both completely shut out by BATCHHI.

The capping still allows BATCHHI to exceed its goal. The BATCHHI goal is a velocity of 30, while the capped velocity is about 40.

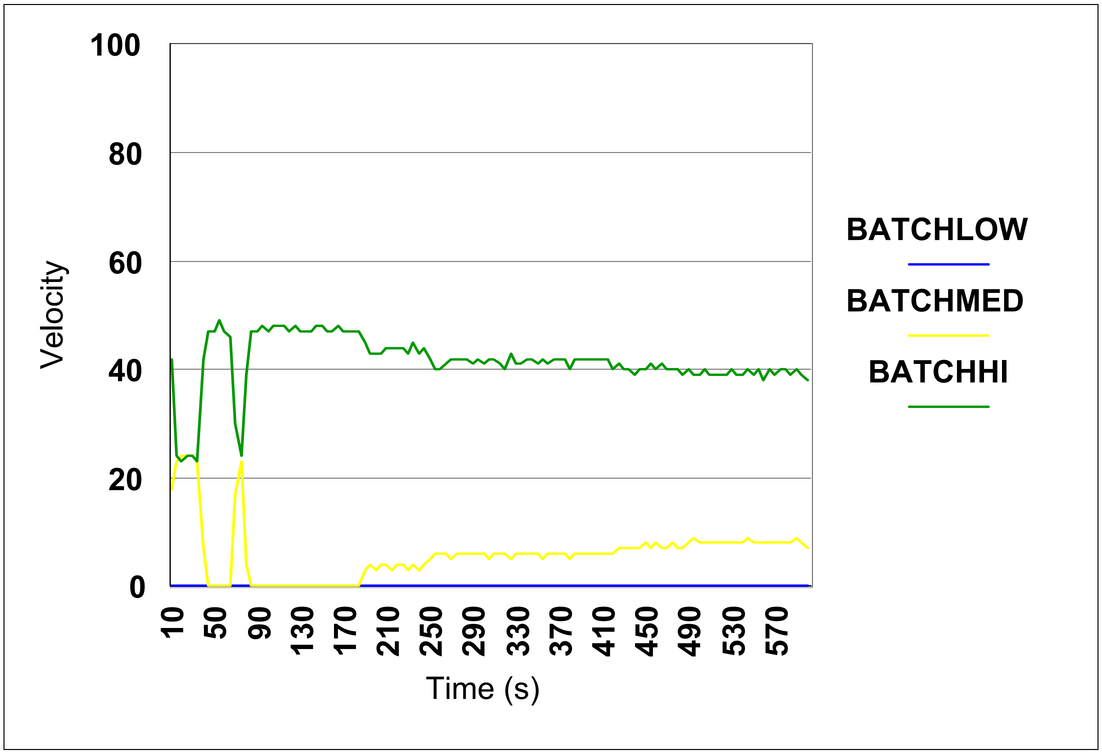


Figure 12-4 Results from test four: Discretionary work and higher goals

Test five: Resource Groups

Resource Groups are another method for controlling access to the CPU between competing tasks.

We set up Resource Groups to guarantee the BATCHLOW and BATCHMED service classes a minimum share of the CPU. We defined type 2 Resource Groups to guarantee service as a percentage of the LPAR. Table 12-5 shows the goals for test five.

Table 12-5 Goals for test five

Service class	Goals	Importance	Resource Group
BATCHHI	Velocity 30	3	
BATCHMED	Velocity 20	4	10% of LPAR
BATCHLOW	Velocity 10	5	5% of LPAR

Results of this test are shown in Figure 12-5 on page 319. The results with Resource Groups were quite different to previous results.

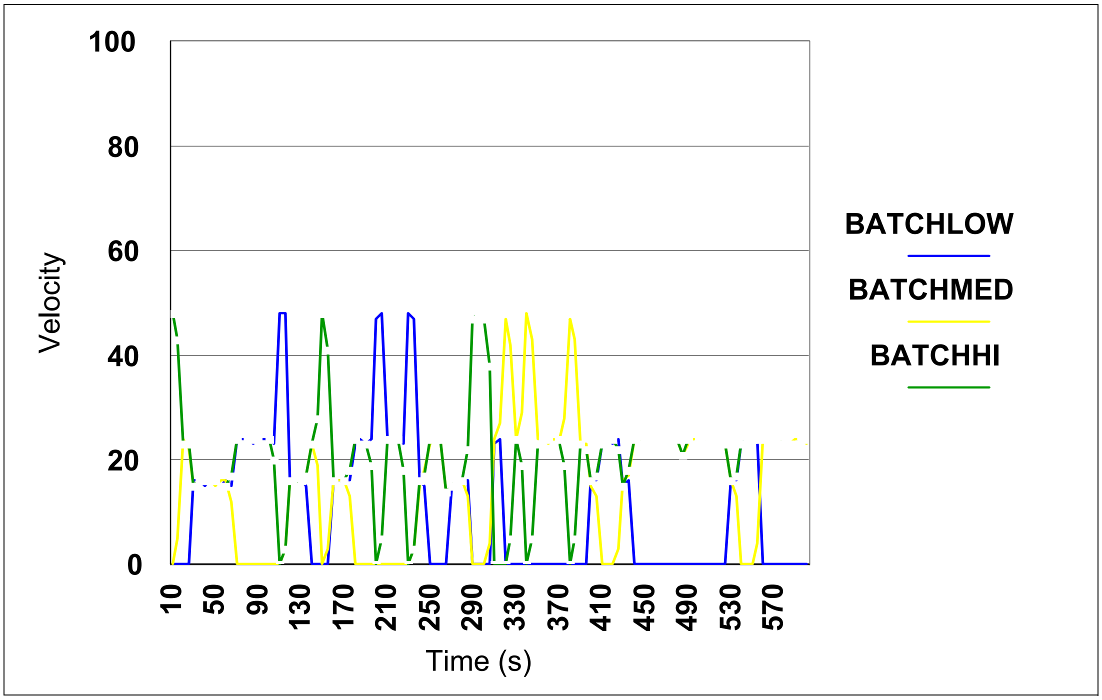


Figure 12-5 Running with Resource Groups

The Resource Groups give the minimum share to the service class when it is missing its goals, even if that causes more important work to miss its goals. WLM managed the Resource Group minimum shares by changing the dispatching priorities. We can see that there are short periods where BATCHLOW and BATCHMED are each elevated above the other service classes. BATCHHI occasionally does not get any service. The minimum share Resource Group does not protect BATCHMED or BATCHLOW from significant periods of no service; although on average, they receive the service specified.

During this period, we also noticed that TSO response time became erratic, with the looping batch jobs elevated above TSO from time to time. TSO period 1 was Importance 1, TSO period 2 Importance 2, so TSO is defined as more important than the batch. The problem was presumably because defining the Resource Groups allows the lower importance work to cause higher importance work to miss its goal.

These results suggest that Resource Groups are not very suitable for guaranteeing service for an online region; the response time is too erratic. Resource Groups can be useful for batch work, but it seems that there is also a risk to higher importance work.

Conclusions

Normally, WLM manages access to the CPU by changing dispatching priorities. When running CPU bound work in a single CPU system, that control is not very precise. To share the CPU between two CPU bound tasks, WLM has to make the priories equal and give them equal access to the CPU. A single task can lock out all lower priority tasks for as long as it uses the CPU.

To share the CPU unequally between CPU bound tasks or to restrict the amount of time that a high priority task can use the CPU requires more granular control. By having discretionary work in the system, you can effectively “turn on” a more granular control. Resource Groups give you some manual methods of controlling the share received by different service classes, but the distribution of service can be uneven.

12.2 Recommendations

This section contains suggestions for managing work in a single CPU system.

12.2.1 Number of service classes

Minimize the number of service classes that you use. Tasks in a single service class run at the same dispatching priority, and they will all get a fair share of the CPU. A task can only be starved of CPU by a task in a different service class. The more service classes you create, the more opportunities you have for one task to lock out others.

Try to ensure that all your service classes receive at least some service. Low importance service classes that receive no service can cause problems, such as jobs that you cannot cancel.

12.2.2 Goals

Velocity goals need to be meaningful figures. When you set dispatching priorities in the pre-WLM days, the relationship to other priorities was what was important. When setting velocities, the actual number is what matters. The relationship to other classes is not so important.

On a loaded single CPU system, velocities are likely to be quite low. In many circumstances, a velocity of 10 might be quite high. See 5.2.4, "Using execution velocity goals" on page 176 for more discussion.

For CPU bound work on a single CPU system, the velocity corresponds approximately to the percentage CPU that one task can use. So, if a single batch job should not be able to use 10% of the CPU on a loaded system, the batch service class probably should not have a velocity goal of 10%.

Do not use velocities to try to rank work according to importance. Use the Importance parameter; that is its purpose. Set the velocity to whatever the work really needs for acceptable performance. In some circumstances, you might have less important work with a higher velocity than more important work.

Velocity goals are probably the most difficult goals to use. Where possible, use response time goals. Response time goals are easier to relate to the performance that people see, and it is easier to understand whether the goals are reasonable.

12.2.3 SYSSTC

On a single CPU system, you need to be very careful about which work runs in SYSSTC. A CPU intensive task in SYSSTC will stop all your regular work until it gives up the CPU. WLM will not do anything to stop it.

Tasks that run in SYSSTC need to be well behaved. Ideally, they should only use the CPU for a fraction of a second at a time before releasing it. If you are aiming for subsecond response time from any regular work, then it cannot afford to wait a long time to be dispatched.

On the other hand, ensure that the tasks that need to be in SYSSTC run there. There are some tasks that must be dispatched quickly, or other problems can result. TCP/IP is an example; delays can cause more work because traffic is resent. SYSSTC provides protection for these tasks on a single CPU system.

Important: You must use SPM as described in 7.9, “Best practices for STC” on page 227 to prevent accidently declassifying system address spaces.

12.2.4 CPU Critical

CPU critical might be useful to ensure more important work is not delayed by less important work. However, if you set the CPU critical parameter for CPU intensive work, it restricts what WLM can do to share the CPU with less important work. Use it with caution.

12.2.5 Discretionary work

As we saw in our testing, discretionary work running can change how WLM manages the CPU, benefiting both discretionary and nondiscretionary work. In some circumstances, discretionary work can get better service than the same work running in a service class with a goal.

Discretionary is a nice, safe place to run work such as CPU-intensive batch jobs. It cannot starve other work of CPU. One way to implement this would be to use a two period service class. The first period can have a response time or velocity goal with a duration equivalent to a small number of CPU seconds, and the second period can be discretionary.

If you have plenty of discretionary work running in your system, and you are confident that it will receive service, discretionary can even be appropriate for very long-running CPU intensive TSO transactions, third period TSO perhaps.

13

Enterprise Workload Manager

This chapter provides an introduction to the IBM Enterprise Workload Manager (EWLM) and the correlations with WLM. It explains the fundamental concepts on which EWLM is based and describes the functionality of the elements that build this solution.

Note: For simplicity, this chapter refers to the z/OS Workload Manager as zWLM.

13.1 Introduction to EWLM

The Enterprise Workload Manager (EWLM) is a set of technologies and system services that enable system administrators to manage and monitor resources across multiple platforms.

The Enterprise Workload Manager (EWLM) enables you to define business-oriented performance goals. It provides an end-to-end view and management of the workload performance across multiple servers.

EWLM is an implementation of policy-based performance management and can be seen as an extension of the Workload Manager for z/OS philosophy. The scope of management is a set of servers that you group into what is called an EWLM *Management Domain*. The set of servers included in the Management Domain should have a logical relationship, for example, the set of servers supporting a particular line of business. The line of business might consist of multiple business processes spread across a few servers or a thousand servers.

There is a management focal point for each EWLM Management Domain, called the EWLM *domain manager*. The domain manager coordinates policy actions across the servers, tracks the states of those servers, and accumulates performance statistics on behalf of the domain.

On each server (operating system) instance in the Management Domain, there is a thin layer of installed EWLM logic called the EWLM *managed server.* The managed server is positioned between the operating system and the EWLM domain manager. The managed server layer understands each of the supported operating systems and gathers resource usage and delay statistics known to the operating system.

More importantly, the managed server layer gathers relevant transaction-related statistics from middleware applications. The application middleware implementations, such as WebSphere Application Server, understand when a unit of work starts and stops. The middleware also understands when a unit of work has been routed to another server for processing, for example, when a Web server routes a servlet request to a WebSphere Application Server.

The managed server layer dynamically constructs a server-level view describing relationships between transaction segments known by the applications with resource consumption data known by the operating system. A summary of this information is periodically sent to the domain manager, where the information is gathered together from all the servers in the Management Domain to form a global view.

An important aspect of the EWLM approach is that all data collection and aggregation activities are driven by a common service level policy, called the EWLM *Domain Policy*. This policy is built by an Administrator to describe the various business processes that the domain supports and the performance objectives for each process.

To permit the EWLM domain manager to construct an end-to-end view of each type of business transaction, the processing segments performed by each participating middleware instance must be correlated, or pieced together. For example, a transaction received by a Web server can flow to a WebSphere Application Server instance that might update a database. The transaction might even spawn by routing a sub-transaction to another server. It is the Domain Manager that correlates all of the information from the various managed servers, and it is capable of constructing the application and server topology used by a certain application.

The final aspect of EWLM is the EWLM *Control Center*, a browser-based application tailored to the needs of an EWLM administrator, analyst, and operator. This is where all the EWLM concepts come together, where you can create a service level *Domain Policy* and activate that policy on hundreds of servers with a single click. Reporting data is then available to let you view performance from a business perspective. And if you need the details, you can see the topology of servers and applications supporting each transaction type and understand where the time was spent. The information is organized in such a way that an administrator or analyst can easily drill down to what is relevant.

Figure 13-1 shows an example of EWLM Management Domain with the related components.

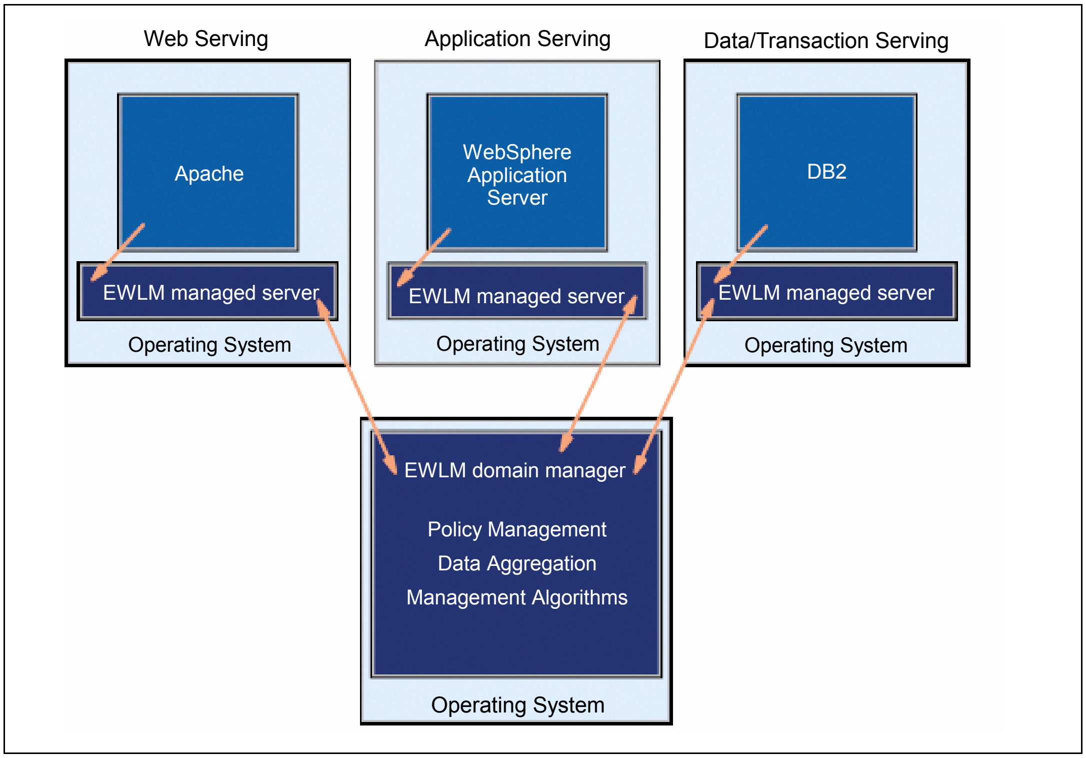


Figure 13-1 EWLM Management Domain

In addition to the monitoring capability, EWLM currently provides two management capabilities: *Partition management and load balancing.* In partition management, EWLM can dynamically manage CPU resources across multiple partitions within the same physical POWER5™ machine. Load balancing is where EWLM can monitor and provide recommendations to the Load Balancer about how to distribute work to achieve optimal performance. In the future, we can expect the addition of more and more management functionalities that will allow distributed transactions to be managed to end-to-end goals, especially on z/OS.

For further details about EWLM components, functions, installation, and usage, refer to *IBM Enterprise Workload Manager V2.1*, SG24-6785.

ARM-instrumented middleware

In order to manage transactions across different operating system platforms, the middleware has to be enabled (instrumented) to collect this data with the Application Response Measurement (ARM) V4.0 standard. *Instrumentation* provides the data to create the topology of a transaction, that is, through what server instances or application instances a transaction flows. In a typical Web transaction, the transaction starts at a Web server (edge server), flows to an application server, and then to a database server. Without instrumentation, the transaction looks like three separate processes with three separate response times rather than one transaction. The ARM-instrumented middleware creates a correlator to pass this information, and ARM data is captured only if a correlator is created or sent from an ARM-instrumented application. Figure 13-2 shows an example of ARM-instrumented middleware.

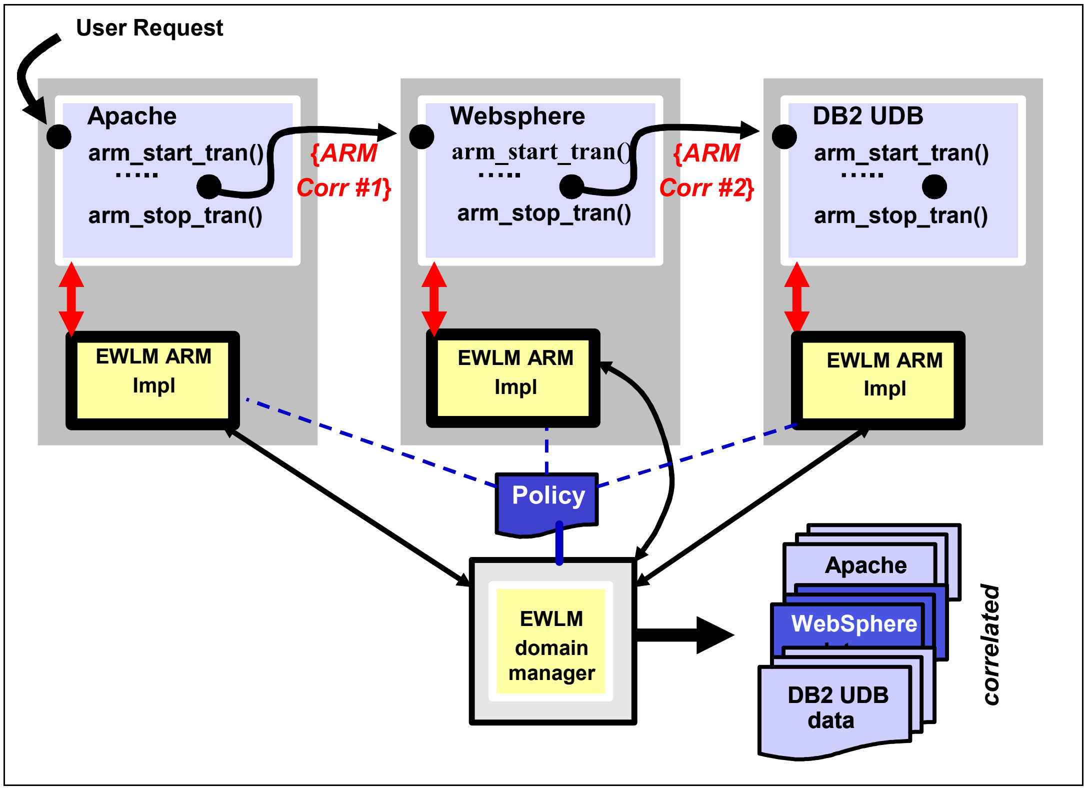
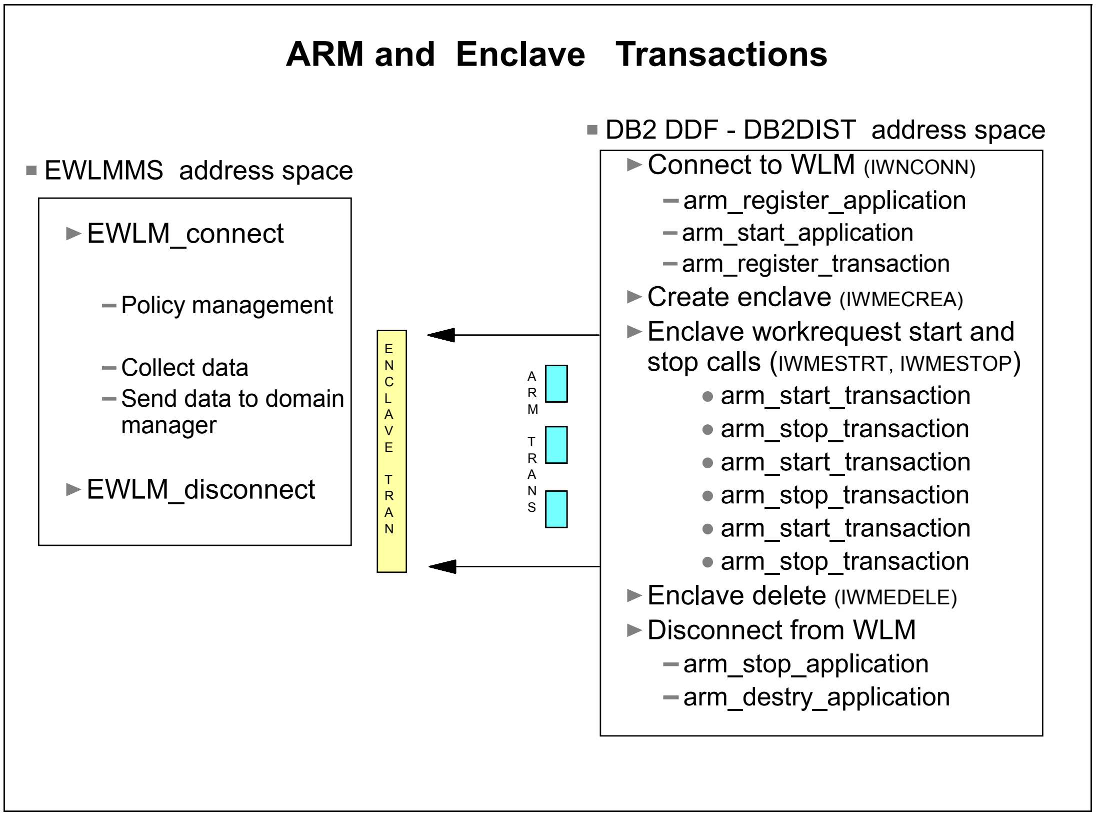


Figure 13-2 ARM implementation

If the application is not instrumented, EWLM can only manage the application by using the partition or process class. Partition and process classes cannot be used to classify work running on the z/OS platform.

ARM enablement on z/OS

Workload Manager on z/OS has provided workload management services to middleware, such as DB2, WebSphere, CICS, and IMS for a long time. For example, DB2 DIST uses WLM enclave services to schedule and process received work requests. In the current implementation, EWLM on z/OS leverages the existing WLM services and maps them to ARM calls, so that an EWLM-managed server can simply pick up the performance data and send it to the domain manager. Figure 13-3 on page 327 depicts a general flow using the distributed data facility (DDF) as a sample environment. In addition, z/OS also has added a native implementation of the ARM calls as described in the OpenGroup ARM standard. This allows applications written in C or C++ to use the ARM services for EWLM use. The EWLM product also adds a Java implementation of the ARM services.



ARM and Enclave Transactions

- EWLMMS address space
  - EWLM_connect
    - Policy management
    - Collect data
    - Send data to domain manager
  - EWLM_disconnect

- DB2 DDF - DB2DIST address space
  - Connect to WLM (IWNCONN)
    - arm_register_application
    - arm_start_application
    - arm_register_transaction
  - Create enclave (IWMECREA)
  - Enclave workrequest start and stop calls (IWMESTRT, IWMESTOP)
    - arm_start_transaction
    - arm_stop_transaction
    - arm_start_transaction
    - arm_stop_transaction
    - arm_start_transaction
    - arm_stop_transaction
  - Enclave delete (IWMEDELE)
  - Disconnect from WLM
    - arm_stop_application
    - arm_destry_application

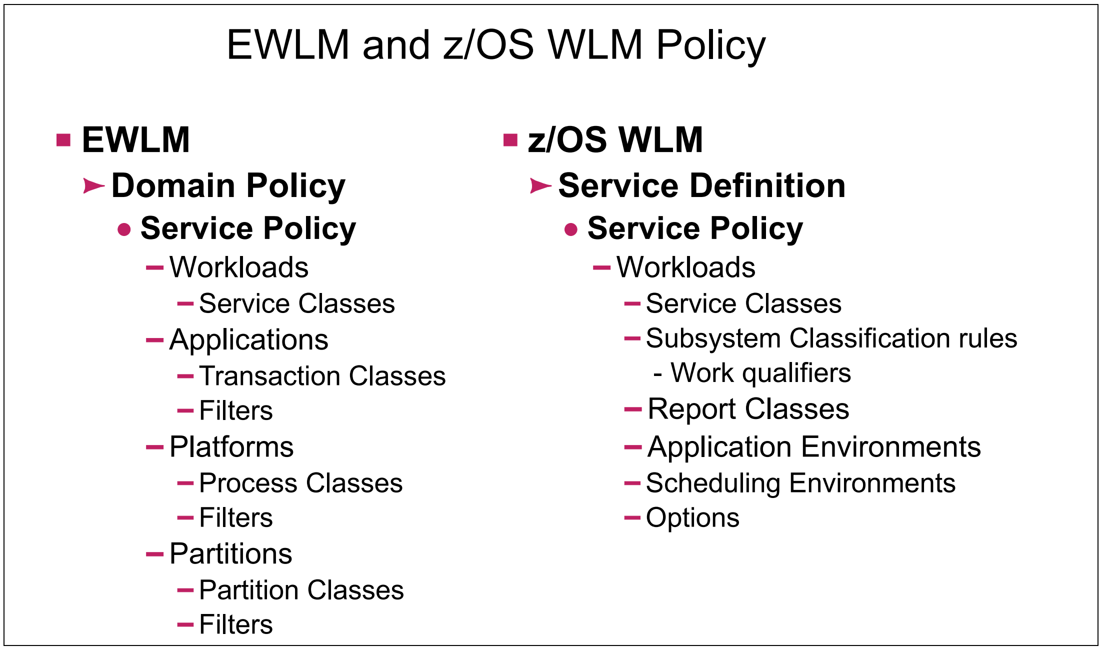
Figure 13-3 ARM and enclave

13.1.1 EWLM and zWLM policies

EWLM and zWLM policy structures are conceptually the same. The main difference is related to the work managers running on the distributed operating systems and to their logical components that receive and submit the work requests.

So, as far as the relationship between EWLM and z/OS WLM goes, at this time they are driven by different service policies, operate independently, and have different scopes. EWLM manages multi-tiered Application Environments that can run on different hardware platforms and operating systems including z/OS, and EWLM has an end-to-end scope of control. The z/OS WLM scope is limited to z/OS platforms. However, starting with z/OS 1.8, there is new functionality that starts converging these policies.

Figure 13-4 on page 328 shows the definitions within each type of policy.



EWLM and z/OS WLM Policy

- EWLM
  - Domain Policy
    - Service Policy
      - Workloads
        - Service Classes
      - Applications
        - Transaction Classes
        - Filters
      - Platforms
        - Process Classes
        - Filters
      - Partitions
        - Partition Classes
        - Filters
- z/OS WLM
  - Service Definition
    - Service Policy
      - Workloads
        - Service Classes
        - Subsystem Classification rules
          - Work qualifiers
        - Report Classes
        - Application Environments
        - Scheduling Environments
        - Options

Figure 13-4 EWLM and zWLM policies

EWLM and zWLM service classes are separate items. The EWLM service class is not automatically used by zWLM, even if there is a service class with the same name. zWLM always does its own classification.

The attribute values for work requests received by the entry application of an EWLM Management Domain might be different from those passed to z/OS WLM, especially if the classification on the distributed platform happened in an Application Environment different from the entry point on z/OS. For this reason, prior to z/OS 1.8, it might not always be possible to have a one-to-one mapping for EWLM service classes and z/OS WLM service classes. z/OS 1.8 introduce a new pseudo-subsystem, EWLM, that can be used to classify transactions based on the previous classification assigned by EWLM using either the service class or transaction class passed as part of the correlator. If the EWLM subsystem is not defined, the classification defaults to the work qualifier subsystem.

When the EWLM subsystem is defined, work can be classified to a zWLM service class based on the service class or transaction class assigned by EWLM. When EWLM classifies the work to a transaction class, it can use filters that are not available on z/OS. zWLM does not have the same information about the origin of the work if it has already passed through middleware on other platforms. However, by using the EWLM transaction or service class as a basis for assigning the zWLM service class, zWLM can indirectly use the same initial granularity used by EWLM in the workload classification.

Using the existing EWLM service class or transaction class classifications keeps the target performance goals defined in both policies synchronized. In this case, the response time defined within the zWLM service class should be a portion of the overall response time defined by the EWLM service class. The zWLM service class goal should be a subset of the EWLM goal to account for time spent on other platforms. For example, if the EWLM service class has a response time goal of 1.5 seconds, the zWLM goal might be 0.5 seconds. The numbers should obviously be adjusted to take into account the times needed for each hop in the transaction.

There does not need to be a one to one correspondence between EWLM service classes and zWLM service classes.

13.1.2 Platform support in EWLM 2.1

Domain managers

The domain managers are:

- Red Hat/SuSe Linux
- AIX® 5L™ and 5D
- i5/OS® 5.3
- Microsoft® Windows 2000/ Microsoft Windows 2003
- z/OS 1.6

Managed servers

The managed servers are:

- AIX 5L Version 5.2
- OS/400® 5.3
- Microsoft Windows 2000/Microsoft Windows 2003
- Sun™ Microsystems™ Solaris™ 8 (SPARC Platform Edition)
- Sun Microsystems Solaris 9 (SPARC Platform Edition)
- HP-UX 11i for PA_RISC
- Linux SuSe SLES 9 SP2
- z/OS 1.6

ARM-instrumented middleware

The ARM-instrumented middleware is:

- WebSphere V5.1.1 and above
- DB2 UDB 8.2
- Apache
- Microsoft IIS Web Server

Note: In the future, we can expect that more major subsystems, such as MQSeries, Lotus® Domino, and CICS will be enabled for EWLM. IMS support for EWLM will be available in R10.

13.2 Getting started

We already had installed and set up the EWLM-managed server on our z/OS system. We implemented a 3-tier Trade application: transactions were entering the system via an Web server running on Windows, connecting to a WebSphere Application Server on UNIX and accessing the data on a DB2 database located on z/OS using a type 4 JDBC driver. Figure 13-5 on page 330 shows the workflow of the trade application.

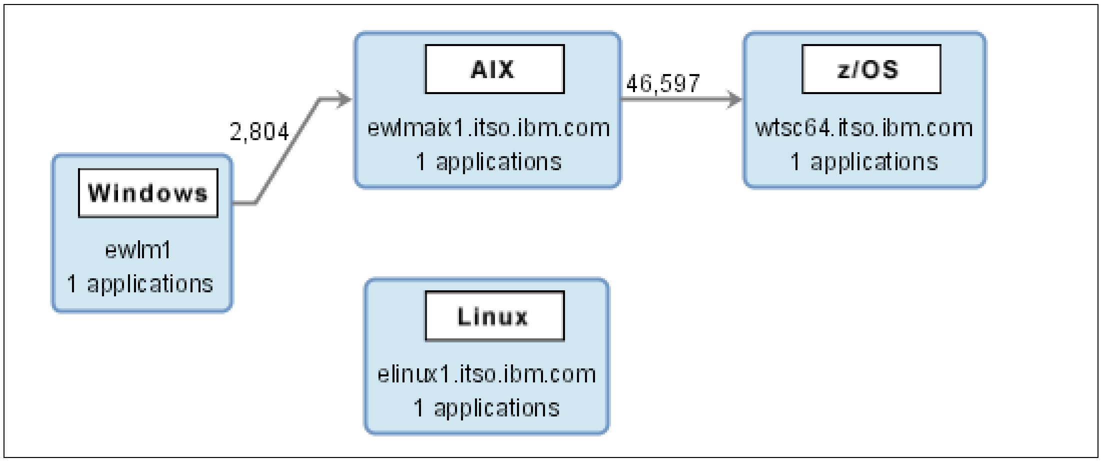


Figure 13-5 Workflow Trade application

This picture is actually taken from the EWLM control center. It shows that 2,804 Web requests issued 46,597 database messages to DB2 on z/OS.

If you need to install and set up the EWLM environment on z/OS, refer to *IBM Enterprise Workload Manager V2.1*, SG24-6785. We highly recommend that you read this IBM Redbook first to understand the complete EWLM environment and how it is implemented in z/OS.

To classify EWLM work on a z/OS operating system, a new subsystem type named EWLM is supplied by IBM. You have to define this subsystem to your zWLM service definition policy before you can start classifying EWLM work. If you do not set up this subsystem, EWLM work will be classified to the default work qualifiers service class defined by your installation.

13.2.1 How EWLM interfaces with z/OS

When the EWLM address space is started, it connects to WLM. Its main functions are:

- Manage the EWLM policy received from the domain manager.
- Collect the performance data and send it to the domain manager.

When middleware is started, it connects to WLM and specifies that it will participate in EWLM. The middleware, for example the DB2 DIST address space, is then registered to ARM on z/OS as an EWLM application participant.

In the case of DB2 DIST, ARM data is captured only if a correlator is sent from an ARM-enabled distributed application such as WebSphere; otherwise, DB2 continues processing distributed requests as it usually does.

When middleware receives a transaction, attribute values are passed with the correlator to zWLM to classify the work request with the zWLM classification rules and associate it to a service class.

From a monitoring perspective, performance data will be reported on both EWLM Control Center and on the RMF WLMGL Postprocessor report.

To verify if an application running on the z/OS system is ARM-enabled, enter the D WLM,AM,ALL command on your z/OS operator console. Figure 13-6 on page 331 shows an example of the output.

```
-D WLM,AM,ALL
   IWM075I  17.08.22  WLM DISPLAY 418
     EWLM ARM SERVICES ARE ENABLED
     EWLM MANAGED SERVER JOBNAME=EWLMMS7 ASID=0078
     EWLM POLICY NAME=ITSO TRADE6 SERVICE POLICY FOR WEEKDAYS
     NUMBER OF REGISTERED PROCESSES=2, APPLICATIONS=1
     ADDRESS SPACES CURRENTLY REGISTERED WITH EWLM ARM:
       JOBNAME=DB8EDIST ASID=007C
         APPLICATION=DDF
           IDENTITY PROPERTIES=0 CONTEXT NAMES=0
           STARTED APPLICATION INSTANCES:
             DB8E
               TRAN=0 GROUP=DB8E
           REGISTERED TRANSACTIONS:
             SYS_DefaultZWLMTransactionName
```

Figure 13-6 D WLM,AM,ALL

This shows us that EWLM ARM services are enabled for the DB2 DIST address space DB8EDIST. The EWLM policy name is ITSO TRADE6 SERVICE POLICY FOR WEEKDAYS.

These are the steps that you need to perform to use the EWLM subsystem classification:

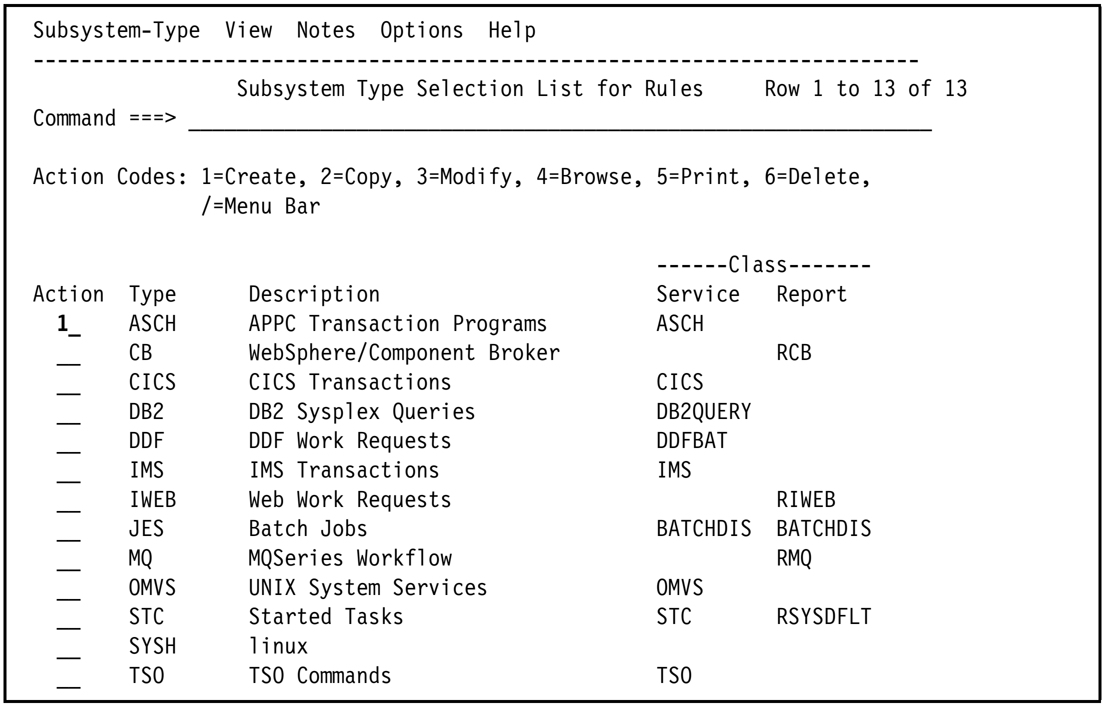
1. Define a service class for EWLM work.

   If you plan to use an existing service class, omit this step; otherwise, add a new service class to be assigned to the EWLM work. We assigned a new service class and named it SC_EWLM.

2. Define the EWLM Subsystem.

   To classify EWLM work on z/OS, add the new subsystem type EWLM to your zWLM service policy definition.

   Start your WLM ISPF application and create the new subsystem in your service definition policy. Select option 6 **Classification Rules** and then create the subsystem with option 1. Figure 13-7 on page 332 shows an example of the WLM ISPF application panel.



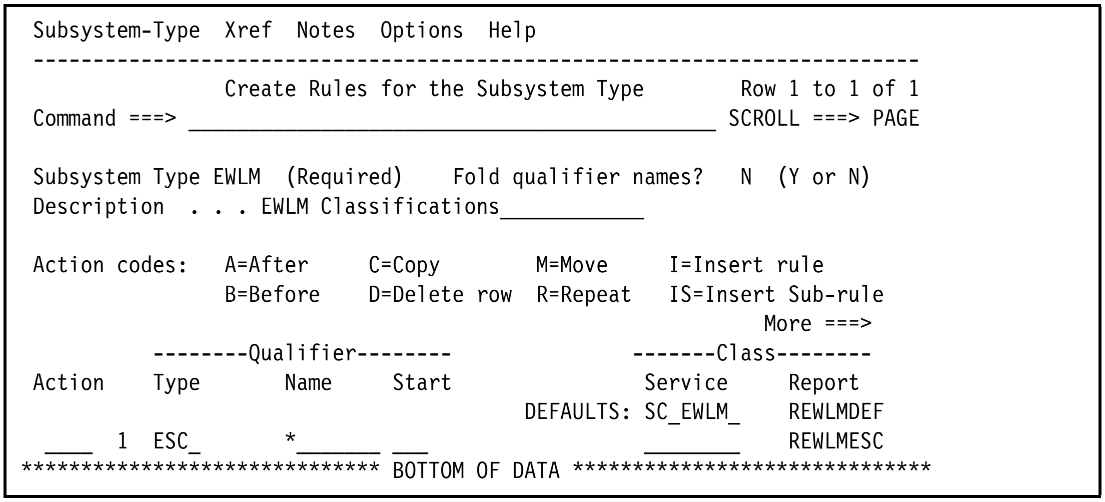
```
 Subsystem-Type  View  Notes  Options  Help
 --------------------------------------------------------------------------
                  Subsystem Type Selection List for Rules     Row 1 to 13 of 13
 Command ===> ______________________________________________________________

 Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
               /=Menu Bar

                                                        ------Class-------
 Action  Type      Description                         Service   Report
   1_    ASCH      APPC Transaction Programs           ASCH
   __    CB        WebSphere/Component Broker                    RCB
   __    CICS      CICS Transactions                   CICS
   __    DB2       DB2 Sysplex Queries                 DB2QUERY
   __    DDF       DDF Work Requests                   DDFBAT
   __    IMS       IMS Transactions                    IMS
   __    IWEB      Web Work Requests                             RIWEB
   __    JES       Batch Jobs                          BATCHDIS  BATCHDIS
   __    MQ        MQSeries Workflow                             RMQ
   __    OMVS      UNIX System Services                OMVS
   __    STC       Started Tasks                       STC       RSYSDFLT
   __    SYSH      linux
   __    TSO       TSO Commands                        TSO
```

Figure 13-7 WLM ISPF application window

We create the new subsystem and assigned the service class named SC_EWLM to the EWLM subsystem. Use **Fold qualifier names = N** to allow for lowercase characters on qualifier names. The subsystem contains two qualifier types to reclassify EWLM workloads:

– EWLM service class (ESC) qualifier to reclassify EWLM service classes
– EWLM transaction class (ETC) qualifier to reclassify EWLM transaction classes

We also made a wildcard (*) classification in the qualifier type EWLM service class (ESC) to be able to identify and report incoming work that has already been classified by previous EWLM instances to a separate report class. This report class then shows in the SDSF enclave panel or in the RMF reports. Figure 13-8 shows an example of the definition.

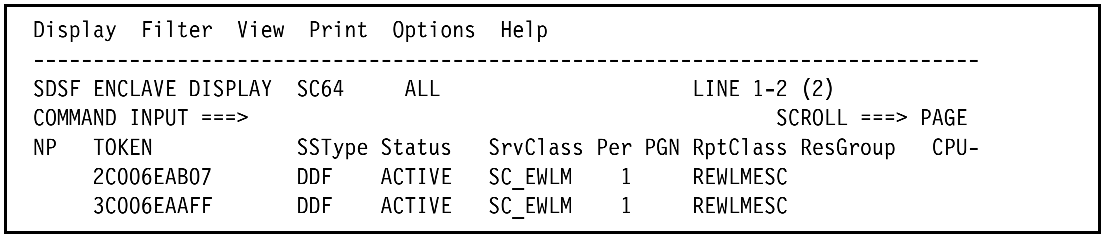


```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Create Rules for the Subsystem Type       Row 1 to 1 of 1
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type EWLM  (Required)    Fold qualifier names?   N  (Y or N)
 Description  . . . EWLM Classifications____________

 Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                 B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                                More ===>
           --------Qualifier--------               -------Class--------
 Action    Type       Name     Start                Service     Report
                                          DEFAULTS: SC_EWLM_    REWLMDEF
  ____  1  ESC_       *_______ ___                  ________    REWLMESC
****************************** BOTTOM OF DATA ******************************
```

Figure 13-8 EWLM classification rules

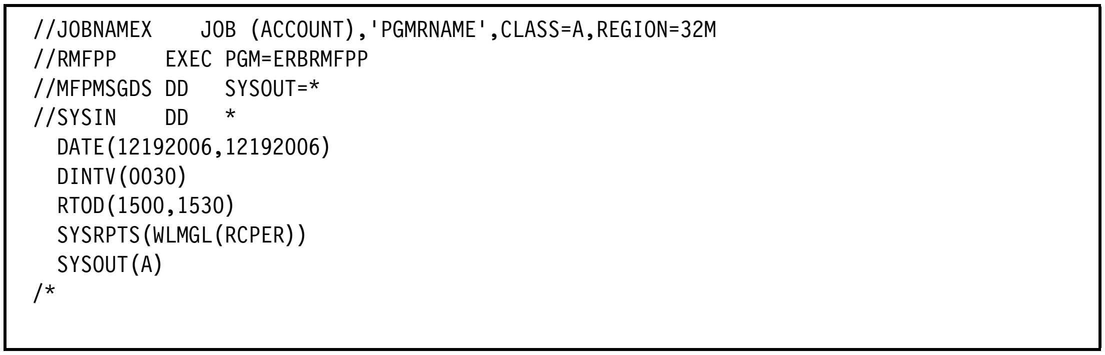
We now switch to SDSF to look for EWLM enclaves. Figure 13-9 shows an example of an SDSF window with an EWLM enclave.

If you do not see the enclaves, run the RMF Monitor I Postprocessor reports with the report class period option. An example of the RMF Postprocessor JCL is shown on Figure 13-10. Figure 13-11 shows the report created by the RMF Postprocessor.



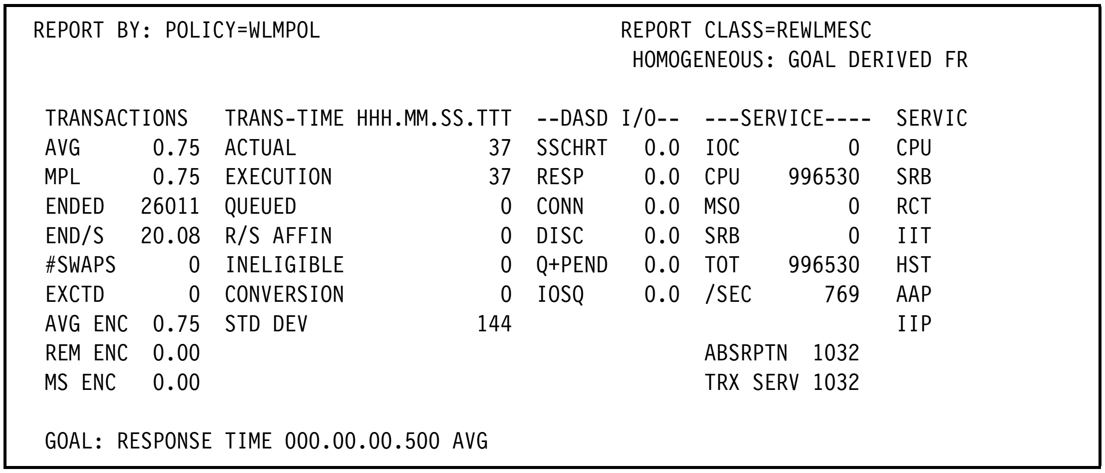
```
Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF ENCLAVE DISPLAY  SC64     ALL                     LINE 1-2 (2)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP   TOKEN           SSType Status   SrvClass Per PGN RptClass ResGroup   CPU-
     2C006EAB07      DDF    ACTIVE   SC_EWLM    1     REWLMESC
     3C006EAAFF      DDF    ACTIVE   SC_EWLM    1     REWLMESC
```

Figure 13-9   SDSF enclave panel



```
//JOBNAMEX    JOB (ACCOUNT),'PGMRNAME',CLASS=A,REGION=32M
//RMFPP    EXEC PGM=ERBRMFPP
//MFPMSGDS DD   SYSOUT=*
//SYSIN    DD   *
  DATE(12192006,12192006)
  DINTV(0030)
  RTOD(1500,1530)
  SYSRPTS(WLMGL(RCPER))
  SYSOUT(A)
/*
```

Figure 13-10   RMF I Postprocessor JCL



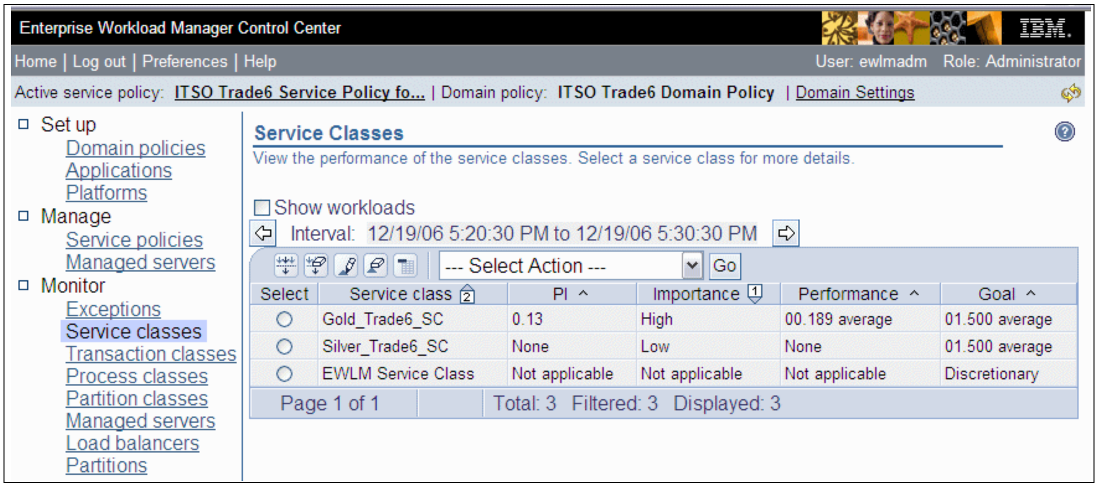
```
REPORT BY: POLICY=WLMPOL                         REPORT CLASS=REWLMESC
                                                  HOMOGENEOUS: GOAL DERIVED FR

 TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----  SERVIC
 AVG      0.75  ACTUAL                37  SSCHRT   0.0  IOC         0   CPU
 MPL      0.75  EXECUTION             37  RESP     0.0  CPU    996530   SRB
 ENDED   26011  QUEUED                 0  CONN     0.0  MSO         0   RCT
 END/S   20.08  R/S AFFIN              0  DISC     0.0  SRB         0   IIT
 #SWAPS      0  INELIGIBLE             0  Q+PEND   0.0  TOT    996530   HST
 EXCTD       0  CONVERSION             0  IOSQ     0.0  /SEC      769   AAP
 AVG ENC  0.75  STD DEV              144                                IIP
 REM ENC  0.00                                          ABSRPTN  1032
 MS ENC   0.00                                          TRX SERV 1032

 GOAL: RESPONSE TIME 000.00.00.500 AVG
```

Figure 13-11   RMF I Postprocessor report

13.3 Using the EWLM Control Center

The first step to view the EWLM data is to log on to the Control Center application using a browser interface. Once logged in, you can see the list of the Monitoring tasks on the left pane.

We clicked **Service Classes** to display the service classes. Figure 13-12 shows an example.



Enterprise Workload Manager Control Center

Home | Log out | Preferences | Help   User: ewlmadm   Role: Administrator

Active service policy: ITSO Trade6 Service Policy fo... | Domain policy: ITSO Trade6 Domain Policy | Domain Settings

□ Set up
- Domain policies
- Applications
- Platforms

□ Manage
- Service policies
- Managed servers

□ Monitor
- Exceptions
- Service classes
- Transaction classes
- Process classes
- Partition classes
- Managed servers
- Load balancers
- Partitions

Service Classes

View the performance of the service classes. Select a service class for more details.

□ Show workloads

Interval: 12/19/06 5:20:30 PM to 12/19/06 5:30:30 PM

--- Select Action --- Go

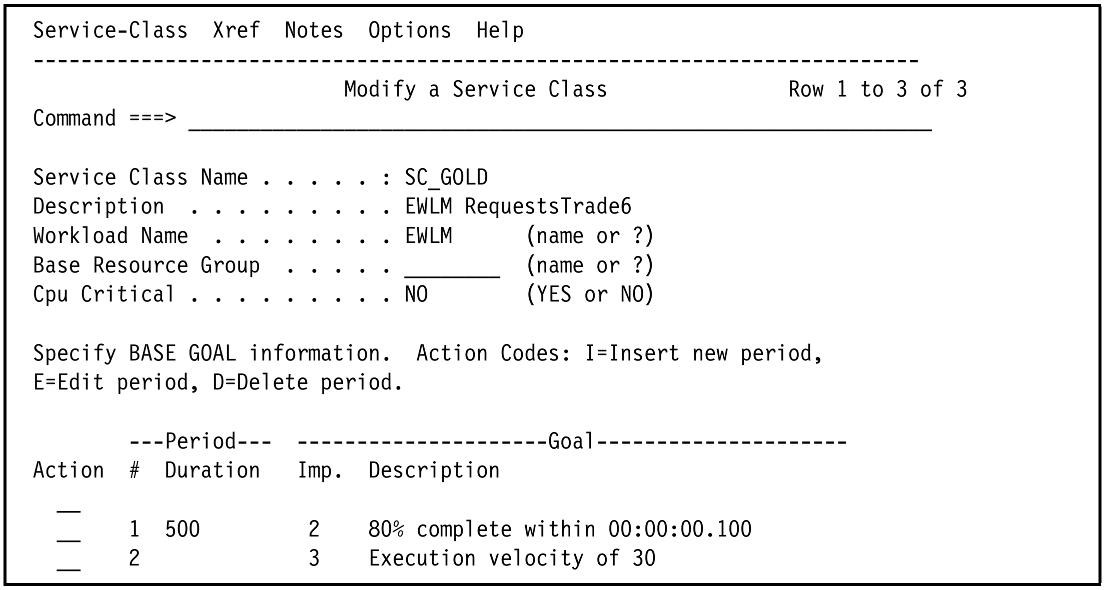
Select	Service class	PI	Importance	Performance	Goal
○	Gold_Trade6_SC	0.13	High	00.189 average	01.500 average
○	Silver_Trade6_SC	None	Low	None	01.500 average
○	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary

Page 1 of 1   Total: 3   Filtered: 3   Displayed: 3

Figure 13-12 WLM main panel

We have three service classes defined. The Gold_Trade6_SC is the only one active in this interval. The goal for Gold_Trade6_SC is set to 1.500 average response time, the current average response time is 00.189 seconds, the PI is 0.13, and the Importance is set to High.

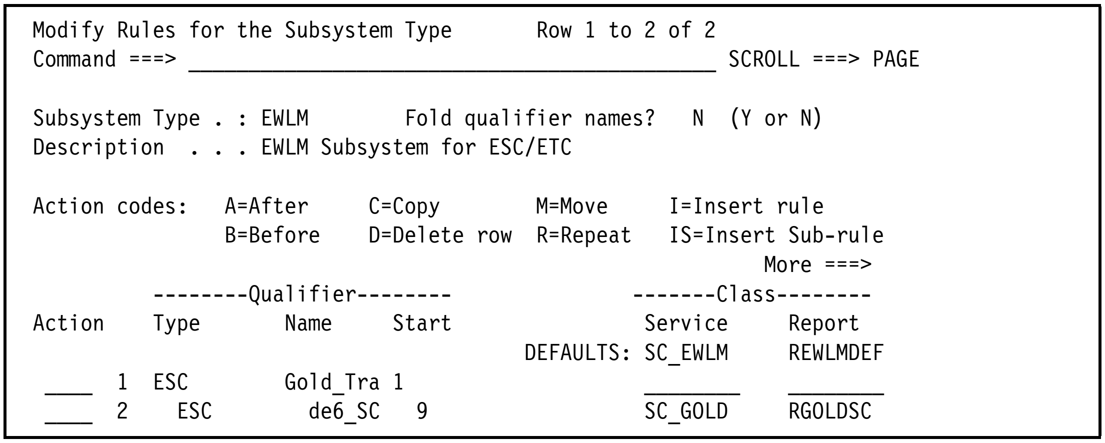
We decided to manage the Distributed Data Facility (DDF) transactions of this workload based on the EWLM service class Gold_Trade6_SC classification and assign an appropriate goal in the zWLM policy. In our zWLM policy, we added a new service class named SC_GOLD and set a two period goal. We also added a new workload definition to zWLM to identify this new type of work. We named the workload EWLM. Because we wanted the Trade6 transactions to process quickly by our z/OS hop, we set a response time goal of 0.1 seconds for the first period and started with a duration of 500 service units. Figure 13-13 shows an example of the service class definition.



```
Service-Class  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                        Modify a Service Class              Row 1 to 3 of 3
Command ===> ______________________________________________________________

Service Class Name . . . . . : SC_GOLD
Description  . . . . . . . . . EWLM RequestsTrade6
Workload Name  . . . . . . . . EWLM      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)
Cpu Critical . . . . . . . . . NO        (YES or NO)

Specify BASE GOAL information.  Action Codes: I=Insert new period,
E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
Action  #  Duration   Imp.  Description

  __
  __    1  500         2    80% complete within 00:00:00.100
  __    2              3    Execution velocity of 30
```

Figure 13-13 Trade 6 classification on the zWLM service policy

Now we define the EWLM Subsystem Type in the existing Classification Rules. We specified **N** on the “Fold qualifier names?” because the classification is case sensitive; otherwise, the classification will not match. Figure 13-14 shows an example.

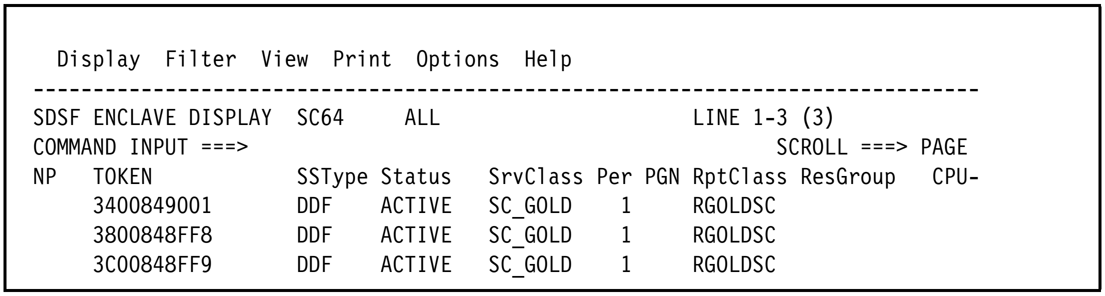


```
Modify Rules for the Subsystem Type       Row 1 to 2 of 2
Command ===> ___________________________________________ SCROLL ===> PAGE

Subsystem Type . : EWLM        Fold qualifier names?   N  (Y or N)
Description  . . . EWLM Subsystem for ESC/ETC

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                             More ===>
          --------Qualifier--------               -------Class--------
Action    Type       Name     Start                Service     Report
                                         DEFAULTS: SC_EWLM     REWLMDEF
 ____  1  ESC        Gold_Tra 1                    ________    ________
 ____  2    ESC        de6_SC   9                  SC_GOLD     RGOLDSC
```

Figure 13-14 Classification of Gold service class in the EWLM Subsystem

We installed and activated the new zWLM service definition.

If we now look at the enclave SDSF panel, we can see that the DDF transactions belonging to this workload are now managed under the SC_GOLD service class and reported under RGOLDSC report class. Figure 13-15 shows the enclaves running in report class RGOLDSC.



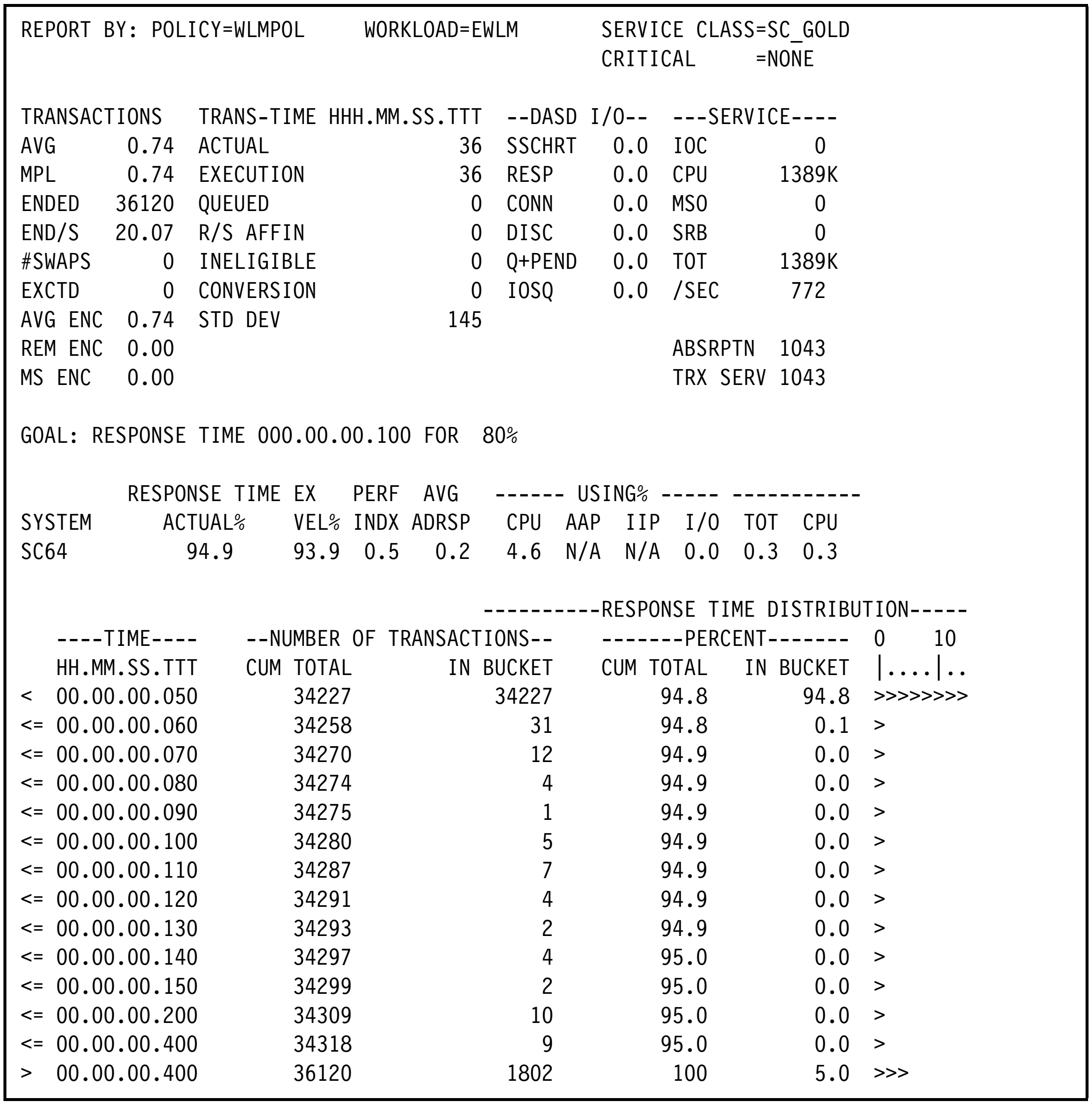
```
  Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF ENCLAVE DISPLAY  SC64     ALL                     LINE 1-3 (3)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP   TOKEN           SSType Status   SrvClass Per PGN RptClass ResGroup   CPU-
     3400849001      DDF    ACTIVE   SC_GOLD    1     RGOLDSC
     3800848FF8      DDF    ACTIVE   SC_GOLD    1     RGOLDSC
     3C00848FF9      DDF    ACTIVE   SC_GOLD    1     RGOLDSC
```

Figure 13-15 Enclaves in report class RGOLD

After completion of the next RMF interval, we run a service class period report to look at how our goal matched the transaction response time. Figure 13-16 shows the JCL that we used and Figure 13-17 on page 336 shows a screen capture of the report produced by RMF I Postprocessor.

```
//JOBNAMEX JOB (ACCOUNT),'PGMRNAME',CLASS=A
//RMFPP    EXEC PGM=ERBRMFPP
//MFPMSGDS DD   SYSOUT=*
//SYSIN    DD   *
  DATE(12202006,12202006)
  DINTV(0030)
  RTOD(1330,1400)
  SYSRPTS(WLMGL(SCPER))
  SYSOUT(A)
/*
```

Figure 13-16 RMF Postprocessor JCL



```
REPORT BY: POLICY=WLMPOL     WORKLOAD=EWLM       SERVICE CLASS=SC_GOLD
                                                 CRITICAL     =NONE

TRANSACTIONS   TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE----
AVG      0.74  ACTUAL                36  SSCHRT   0.0  IOC         0
MPL      0.74  EXECUTION             36  RESP     0.0  CPU      1389K
ENDED   36120  QUEUED                 0  CONN     0.0  MSO         0
END/S   20.07  R/S AFFIN              0  DISC     0.0  SRB         0
#SWAPS      0  INELIGIBLE             0  Q+PEND   0.0  TOT      1389K
EXCTD       0  CONVERSION             0  IOSQ     0.0  /SEC      772
AVG ENC  0.74  STD DEV              145
REM ENC  0.00                                          ABSRPTN  1043
MS ENC   0.00                                          TRX SERV 1043

GOAL: RESPONSE TIME 000.00.00.100 FOR  80%

         RESPONSE TIME EX   PERF  AVG   ------ USING% ----- -----------
SYSTEM      ACTUAL%    VEL% INDX ADRSP   CPU  AAP  IIP  I/O  TOT  CPU
SC64          94.9     93.9  0.5   0.2   4.6  N/A  N/A  0.0  0.3  0.3

                                       ----------RESPONSE TIME DISTRIBUTION-----
   ----TIME----    --NUMBER OF TRANSACTIONS--    -------PERCENT-------  0    10
   HH.MM.SS.TTT    CUM TOTAL        IN BUCKET    CUM TOTAL   IN BUCKET  |....|..
<  00.00.00.050        34227            34227         94.8        94.8  >>>>>>>>
<= 00.00.00.060        34258               31         94.8         0.1  >
<= 00.00.00.070        34270               12         94.9         0.0  >
<= 00.00.00.080        34274                4         94.9         0.0  >
<= 00.00.00.090        34275                1         94.9         0.0  >
<= 00.00.00.100        34280                5         94.9         0.0  >
<= 00.00.00.110        34287                7         94.9         0.0  >
<= 00.00.00.120        34291                4         94.9         0.0  >
<= 00.00.00.130        34293                2         94.9         0.0  >
<= 00.00.00.140        34297                4         95.0         0.0  >
<= 00.00.00.150        34299                2         95.0         0.0  >
<= 00.00.00.200        34309               10         95.0         0.0  >
<= 00.00.00.400        34318                9         95.0         0.0  >
>  00.00.00.400        36120             1802          100         5.0  >>>
```

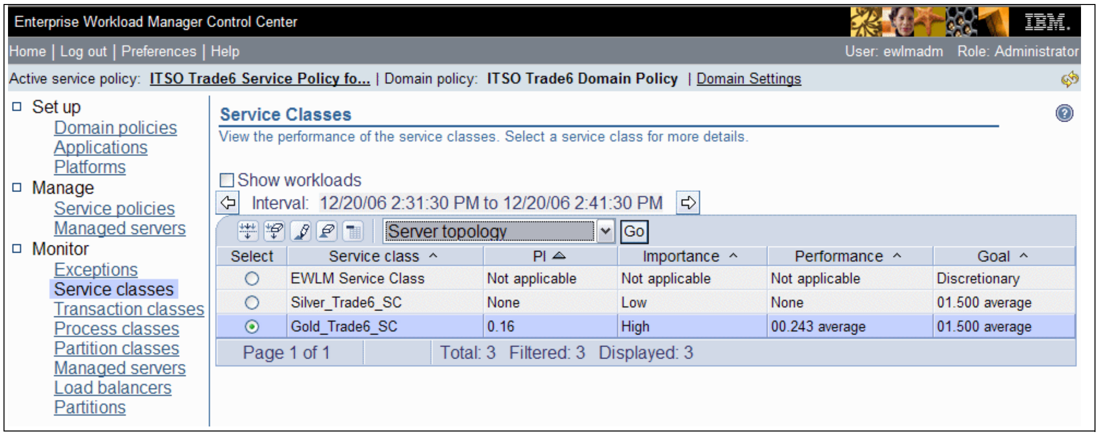
Figure 13-17 RMF Report on service class period

We see that almost all of the transactions exceeded their goal, and that the performance index is 0.5.

13.3.1 Monitoring the use of the EWLM Control Center

We now use the EWLM Control Center to monitor our service classes and transactions.

To view the server topology for our Gold_Trade6_SC service class, we selected **Service Classes**. We then selected the **Gold_Trade6_SC**, selected **Server topology** from the drop-down list, and clicked **Go**. Figure 13-18 on page 337 shows an example.



Enterprise Workload Manager Control Center

Home | Log out | Preferences | Help — User: ewlmadm Role: Administrator

Active service policy: ITSO Trade6 Service Policy fo... | Domain policy: ITSO Trade6 Domain Policy | Domain Settings

Service Classes

View the performance of the service classes. Select a service class for more details.

□ Show workloads

Interval: 12/20/06 2:31:30 PM to 12/20/06 2:41:30 PM

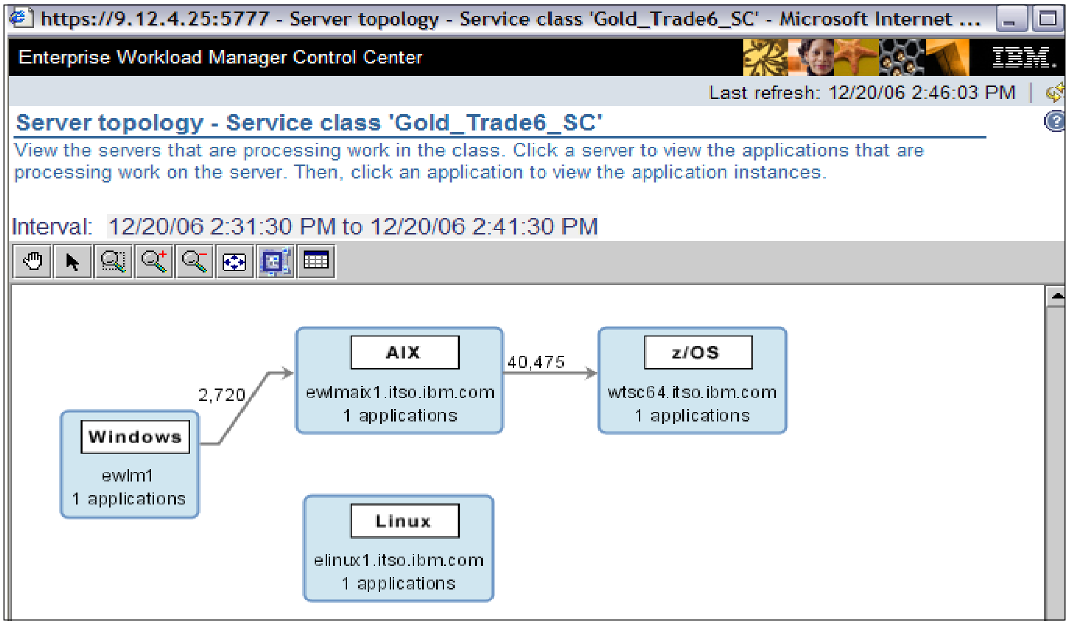
Server topology Go

Select	Service class ^	PI ^	Importance ^	Performance ^	Goal ^
○	EWLM Service Class	Not applicable	Not applicable	Not applicable	Discretionary
○	Silver_Trade6_SC	None	Low	None	01.500 average
●	Gold_Trade6_SC	0.16	High	00.243 average	01.500 average

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3

Figure 13-18 Monitor Service Classes

The server topology displayed shows transactions that are classified to the Gold_Trade6_SC presented using the related server topology. The detailed information shows 2720 requests issued on our Web server generating 40475 DB2 messages sent to DB2 on the z/OS system. You can see this in Figure 13-19.



https://9.12.4.25:5777 - Server topology - Service class 'Gold_Trade6_SC' - Microsoft Internet ...

Enterprise Workload Manager Control Center

Last refresh: 12/20/06 2:46:03 PM

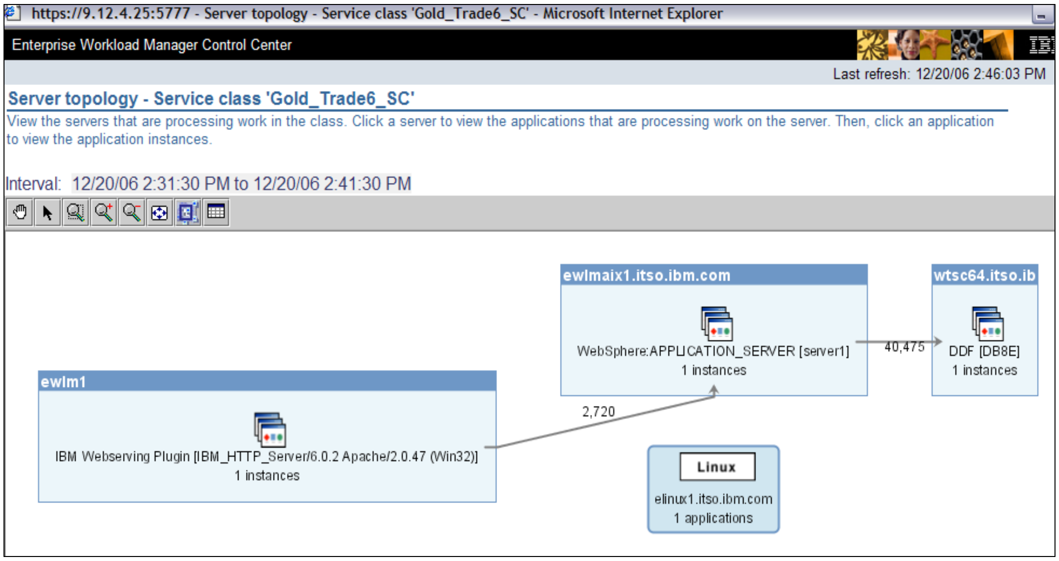
Server topology - Service class 'Gold_Trade6_SC'

View the servers that are processing work in the class. Click a server to view the applications that are processing work on the server. Then, click an application to view the application instances.

Interval: 12/20/06 2:31:30 PM to 12/20/06 2:41:30 PM

Figure 13-19 Display of server topology

From this panel, you can click each server icon to view the application instances running on each server (shown in Figure 13-20 on page 338).



https://9.12.4.25:5777 - Server topology - Service class 'Gold_Trade6_SC' - Microsoft Internet Explorer

Enterprise Workload Manager Control Center

Last refresh: 12/20/06 2:46:03 PM

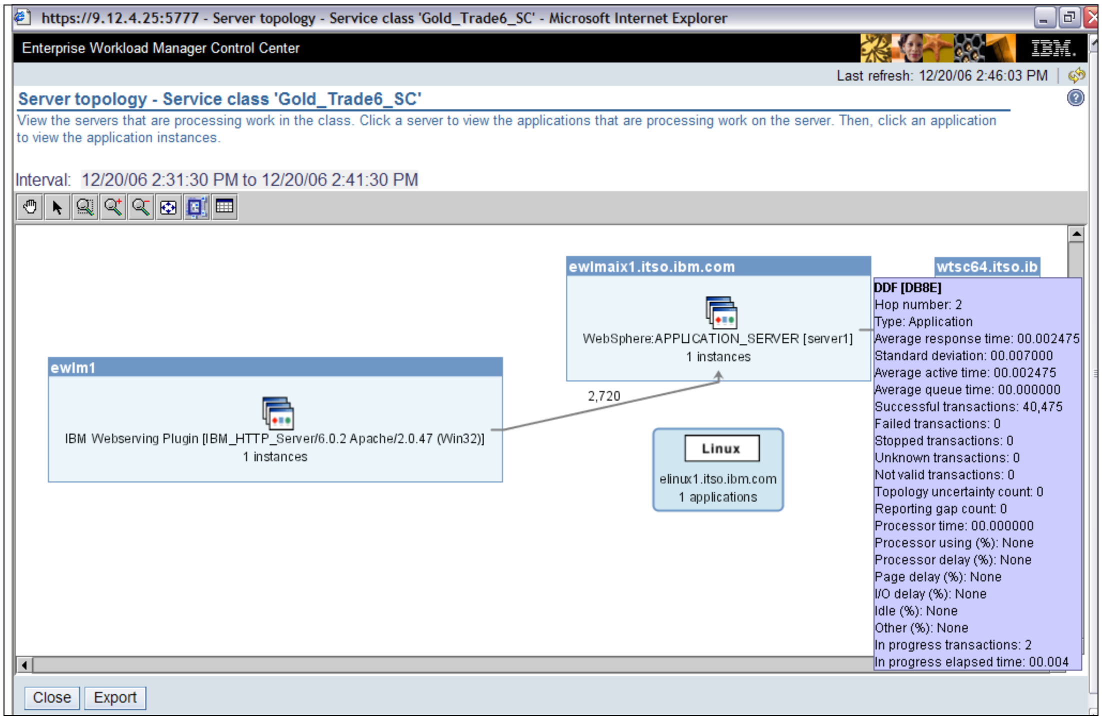
Server topology - Service class 'Gold_Trade6_SC'

View the servers that are processing work in the class. Click a server to view the applications that are processing work on the server. Then, click an application to view the application instances.

Interval: 12/20/06 2:31:30 PM to 12/20/06 2:41:30 PM

Figure 13-20 Application instances on servers

If you need additional details about the middleware instance, you can move your cursor over the application instance to view a detailed information window displayed on the panel (see Figure 13-21).



https://9.12.4.25:5777 - Server topology - Service class 'Gold_Trade6_SC' - Microsoft Internet Explorer

Enterprise Workload Manager Control Center

Last refresh: 12/20/06 2:46:03 PM

Server topology - Service class 'Gold_Trade6_SC'

View the servers that are processing work in the class. Click a server to view the applications that are processing work on the server. Then, click an application to view the application instances.

Interval: 12/20/06 2:31:30 PM to 12/20/06 2:41:30 PM

DDF [DB8E]
Hop number: 2
Type: Application
Average response time: 00.002475
Standard deviation: 00.007000
Average active time: 00.002475
Average queue time: 00.000000
Successful transactions: 40,475
Failed transactions: 0
Stopped transactions: 0
Unknown transactions: 0
Not valid transactions: 0
Topology uncertainty count: 0
Reporting gap count: 0
Processor time: 00.000000
Processor using (%): None
Processor delay (%): None
Page delay (%): None
I/O delay (%): None
Idle (%): None
Other (%): None
In progress transactions: 2
In progress elapsed time: 00.004

Figure 13-21 Server Topology: Detailed pane

The information shown includes response time measurements and also the name of the DB2 DDF instance, in our case, DB8E, and that DB2 is the hop number 2 for this workload. This shows how it is simple for you to keep track of a multi-tier application and have an idea of how the multiple application pieces behave in regard to their performance goal.

If application transactions are missing their goals, it is simple with EWLM to determine which hop or middleware might be in trouble and then focus on the specific section of the application with additional monitors, such as zWLM, DB2 PE, or RMF in case of z/OS.

We meant in this chapter to give you a brief summary of EWLM and how it works with zWLM. To learn more about EWLM and its features on z/OS and distributed servers, refer to the redbook *IBM Enterprise Workload Manager V2.1*, SG24-6785.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 341. Note that some of the documents referenced here may be available in softcopy only.

- *z/OS Intelligent Resource Director*, SG24-5952
- *OS/390 Workload Manager Implementation and Exploitation*, SG24-5326
- *VSAM Demystified,* SG24-6105
- *DB2 for z/OS Stored Procedures: Through the Call and Beyond*, SG24-7083
- *IBM Enterprise Workload Manager V2.1*, SG24-6785

Other publications

These publications are also relevant as further information sources:

- *z/OS V1R8.0 Resource Measurement Facility (RMF) Report Analysis,* SC33-7991
- *z/OS System Management Facilities (SMF)*, SA22-7630
- *z/OS MVS Initialization and Tuning Reference*, SA22-7592
- *z/OS V1R8.0 MVS Planning Workload Management,* SA22-7602

Online resources

These Web sites and URLs are also relevant as further information sources:

- LSPR for IBM zSeries and S/390

  http://www.ibm.com/servers/eserver/zseries/lspr/
- OpenGroup ARM standard

  http://www.opengroup.org/arm

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

$P SRVCLASS 194
$S SRVCLASS 194
$T SRVCLASS 194
_BPXK_WLM_PROPAGATE 288

A

administration application 30
administration policy management 30
alias devices 55
APPC 62, 230
Application Environment
  definition 13
application environment
  definition 140
Application Environments 261
  server address spaces 13
  work manager 13
application environments
  concepts 92
  stored procedures 246
ARM 325
assistant 252
Average response time 38, 128
average response time 172
average transaction response time 15

B

base devices 55
Batch 62
bottleneck 45
BPXAS 284
BPXBATCH 285
BPXOINIT 284, 289
buckets, response time 38
Business Unit of Work 31
business unit of work 6

C

capping 46, 317
  resource group 12
Capping overachieving work 183
CFCC 105
chpriority() 293
CICS 62, 298
classification rules 129
  DB2 250
  TSO 220
client SRB 33
CNTCLIST 116, 218
complex query 251
Component Broker 258
connect time 27
Control Center 324
controller region 265
coordinator 252
correlator 326
couple data set 121
CPAI 21
  formula 21
CPENABLE 116
CPU and storage critical 185
CPU bound tasks 312, 319
CPU contention 312
CPU CRITICAL 87
CPU Critical 321
CPU critical 302
CPU delays 35
CPU intensive TSO transactions 321
CPU service units 22
CPU time 20
  formula 20
CPU-bound 315
CPU-bound task 312
CPU-intensive batch jobs 321
Cycle time 21
  formula 21

D

Daemon 284
data collection 324
DB2
  client strings 239
DB2 Connect 100
DB2 stored procedures
  see stored procedures
DDF 234
delay 14
delay states 35
dependent enclave 248
device delay 54
device usage 54
disconnect time 27
Discretionary 11, 128, 316
discretionary 40, 50, 182, 316, 319
discretionary goal 50
discretionary goal management
  donors 50
discretionary goals 182
discretionary service class period 40
Discretionary work 321
dispatchable unit 30
  resources 31
Dispatching priorities 312
dispatching priorities 319
dispatching priority 31, 47, 312

domain manager 324
  definition 324
domain policy
  definition 324
donor 43
DRDA 236
Driver Type-2 239
Driver Type-4 239
DSNTIPX 243
dynamic internal service classes 69
dynamic service class 194

E
efficiency algorithm 56
enclave
  accounting 34
  characteristics 34
  dependant 34
  description 34
  dispatching priority 34
  independents 34
  management 64
  reports 240
  types 34
enclave exploiters 65
enclave propagation 286
enclave SRB 33
Enclaves 261
enqueue 46
enqueue promotion 116
enqueue promotion interval 91
ERV 115
ESS 55, 60
ETR 6, 17
  formula 18
EWLM 323
  policy-based 324
execution delay monitoring 65
Execution velocity 128
external service classes 69

F
Fork 284, 286
fork and spawn function calls 286

G
goal algorithm 56
Goals 320
goals
  response time 218
Group capacity 102

H
health indication 98
HTTP
  Classification rules 274

I
I/O delay 45
I/O delays 35
I/O performance metrics 25
I/O priority management
  decisions 54
  goals 53
I/O response time 25
  components 25
IBM Enterprise Storage Server 55, 60
ICF 105
Idle 14
IEAOPTxx 114
  zAAP 107
IFL 105
importance 3, 12, 319–320
Importance levels 180
IMS 62
INITIMP 117
INITIMP parameter 117
Instances 262
instrumented 325
Intelligent Resource Director 102
IOSQ time 26
IRD 102
  CSSP IOQP 54–55
IRLM locks 90
ISHELL 285
ISPF application
  security 119
ISTEXCGR 96
ITR 22
  formula 22
IWMMNTFY 68
IWMRPT 68
IWMSRSRG macro 98
IWMSRSRS macro 98
IWMSSEL 260
IWMTAFF 260

J
Java 31
Java code flow 106
JDBC 239
JES2 job selection 194
JES2 service class 194
JVM 31

K
Kernel 284

L
LINUX 103
load balancing 325
Load Balancing Advisor 99
Local PI 36
locks 46
LSPR 23

methodology 24
workload 24

M
maintaining service goals 169
managed server
definition 324
management domain 324
Managing Service Definitions 186
mapping file 266
MCCAFCTH 115
MDB
Classification rules 277
MFLOPS 21
MIPS 21
MPL 51
MPL (OUTR) delay 45
MSU 83
multisystem enclaves 34

N
nice() 293
Nonpreemtible Service Request 30

O
OMVS 284
orkload specification 126
OW28820 149

P
paging delay 45
parallel access volume 55
parallel access volumes 60
partition management 325
path length 21
PAV
static 55
PB 66
pending time 26
percentile response time 172
performance block 66
performance goal 10
performance group periods 219
performance index 36
calculation 37
local 36
sysplex 36
performance management
metrics 15
permanent service class 194
PI calculation example 38
Platform support in EWLM 2.1 329
policy adjustment
donor 43
receiver 44
Preemptible Client Service Request 30
Preemptible Enclave Service Request 30
Preemptible Task 30
Preemptible-Class SRB 33
Process 284
processor delay 45
processor performance
metrics 20
pthread_create 286

Q
Query Parallelism 254
Quiesced 14

R
RACF 119
receiver 43, 45
bottlenecks 45
selecting 45
receiver selection 45
Redbooks Web site 342
Contact us xv
region goal 298
report classes 183
Resource Grou 50
Resource Group 82
resource group 12, 76, 184
definition 127
Resource Groups 184, 318
Resource groups 319
resource groups 319
discretionary goal management 50
resource utilization 18
formula 18
response time 10, 12, 15–16
distribution 38
formula 16
Response time buckets 174
response time goals 172
response time goals for CICS and IMS 174
Response time with percentile 128
Revisiting Service Definitions 187

S
sampling 14
Scheduling Environment 14
definition 14
scheduling environment 141
SDP 19
formula 19
Security access 119
servant regions 265
server address space delay 45
server address space management 91
server address spaces 68
server topology 67, 69
server weight 98
servers
management 71
Service Class
SYSSTC 225

service class 3
  definition 128
  SYSTEM 224
Service Class periods 179
Service Classes 168
service classes 320
Service Definition 8
service definition 123
Service Level Agreement 6
Service Policies 7
Service policies 8
service units 138
set goals 170
setpriority() 293
single CPU system 312
SMF
  type 99 records 149
SMF 72 record 236
soft capping 101
Spawn 284, 286
spawn 286
specialty processor 104
Specialty Processors 104
SPECIFIC function 99
SPM rules 231
SQL 243
SRM 29
  plots 44
storage
  LRU algorithm 51
STORAGE CRITICAL 52
storage delays 35
Storage protection 52
storage protection 302
Stored Procedure 242
stored procedures 234
  application environment 246
  classification rules 250
  considerations 245
  defining 246
  desciption 242
  goals 251
  NUMTCB 247
  server address spaces 247
  service classes 251
  SYSIBM.SYSPROCEDURES 247
  transaction flow 248
  WLM_ENV 247
structured query language 243
subsystem work managers 30
swap page-in delay 45
swap protect time 51
Syscall 284
SYSIBM.SYSPROCEDURES 247
SYSOTHER 10, 50, 253
sysplex operation 30
Sysplex PI 36
sysplex query parallelism 236, 251
  assistant 252
  classification rules 254
  considerations 253
  coordinator 252
  description 236, 251
  group-wide goals 253
  workload classification 253
  workload goals 256
sysplex routing 98
sysplex routing services 97
sysplex scope 76
SYSSTC 10, 47, 320
SYSSTC service class 178
SYSTEM 10, 47
system resources
  contention 3
system scope 77
SYSZWLM_WORKUNIT 34

T
TCPIP 320
temporal affinity 260
Threads 284
transaction 6
  BTE phases 72
  EXEC phase 72
  phases 72
  response time 16
transaction class mapping file 266
transaction rate 18
transaction response time goal 298
TSO 62
TSO response time 319
TSO workload
  CNTCLIST=YES 218
  performance group periods 219
  response time goal 218
Type 1 Resource Group 76
Type 2 Resource Group 76
Type 3 Resource Group 76
Types of resource groups 76

U
Uniprocessor considerations 311
uniprocessor systems 315
UNIX 293
UNIX system services
  BPXAS 287
  chpriority() 293
  definition for fork and spawn 288
  dubbed address spaces 290
  enclave propagation 286
  fork and spawn 286
  goal mode 289
  nice() 293
  performance management 289
  pthread_create 286
  setpriority() 293
Unknown 14
user thinking time 15
using 14

V

Velocity 11, 38
Velocity goals 320
velocity goals 177
VTAM Generic resources 96

W

WebSphere Application Server 324
WebSphere environment
  reporting 279
When things aren't working 187
WLC 83
WLM
  concepts example 2
  constructs 3
  queue 92
  sampling 14
WLM capping 316
WLM CDS 122
WLM functions
  server address space management 91
WLM ISPF application 120
WLM Service Definition Editor 154
work classification 3
work requests
  identification 3
Workload Classification Document 267
Workload Licence Charge 83
Workload License Charges 101
workload management
  introduction 2
Workload Manager
  concepts 2
Workload profile 262
workload profile 262
workload routing 95
workloads
  sysplex query parallelism 236, 251
  UNIX System Services 286

Z

z/OS dispatcher 90
zAAP 104–106, 109
zIIP 109
zWLM 323

IBM

Redbooks

System Programmer's Guide to: Workload Manager

System Programmer’s Guide to: Workload Manager

Workload Manager overview and functionalities

How to classify your workloads

Best practices samples

This IBM Redbook gives a broad understanding of the Workload Manager component of the z/OS system. It covers basic aspects of Workload Manager (WLM) together with the new functions available in z/OS V1R8. The book provides a discussion about how to create WLM policies based on business goals and the types of transactions you run in your systems.

This book also provides information about how to implement the WLM functions, in addition to suggestions about how to classify different workloads to WLM so that it can best manage the different types of units of work. You will find information for effective performance analysis and a description of how to use the WLM functions to achieve a better throughput in your environment. This book also provides some best practices samples to better understand how to set up your classification and tune it.

SG24-6472-03 ISBN 073848993X