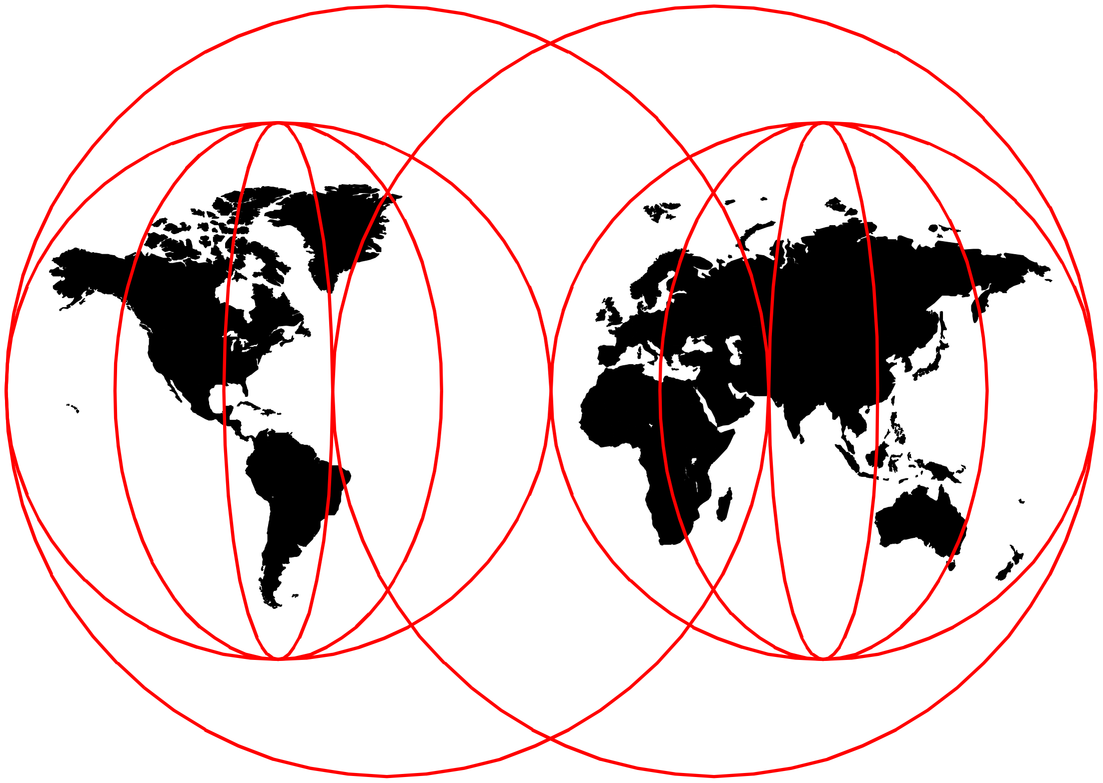
OS/390 Workload Manager Implementation and Exploitation

Dave Clitherow, Steffen Herzog, Alvaro Salla,
Valeria Sokal, Joan Trethewey



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5326-00

International Technical Support Organization

SG24-5326-00

OS/390 Workload Manager Implementation and Exploitation

May 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 231.

First Edition (May 1999)

This edition applies to Version 2 Release 6 of OS/390, Program Number 5647-A01

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Contents

Figures

Tables

Preface

This redbook provides detailed information on the Workload Manager (WLM) component of OS/390 Version 2 Release 6.

The material presented is aimed at systems programmers, performance analysts, and other people involved in defining how workloads are processed within an OS/390 system.

In-depth information is provided on how to classify different workloads to WLM, including new workload types such as business intelligence workloads exploiting DB2 and e-business workloads exploiting Web servers on OS/390.

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.

Dave Clitherow is an Advisory International Technical Support Specialist in Large Systems based in Poughkeepsie, New York. He writes extensively and teaches IBM classes worldwide on all areas of Parallel Sysplex implementation and exploitation. Before joining the ITSO in 1997, Dave worked in the UK as a Senior Services Specialist for IBM Global Services.

Steffen Herzog is an IBM I/T specialist in Germany. He has 8 years of experience in Large Systems. He holds a degree in Computer Technology from the Institute of Technology Ilmenau. His areas of expertise include OS/390, Parallel Sysplex and WLM.

Alvaro Salla is an IBM consultant SE in Brasil. He has 30 years of experience in Large Systems. He holds a degree in Chemical Engineering from Sao Paulo University. His areas of expertise include performance, computer architecture, and operating systems. He has extensive experience in writing redbooks.

Valeria Sokal is a systems programmer at Banco do Brasil. She holds a bachelors degree in Physics and Electrical Engineering from Rio Grande do Sul University, Brasil and an MBA in Software Engineering from Sao Paulo University. She has 10 years of experience in the mainframe area. Her areas of expertise include MVS, TSO/ISPF/PDF, SLR.

Joan Trethewey is an IBM Senior Services Specialist in Canada. She has 17 years of experience in Large Systems. She holds a degree in Computer Science and Combinatorics & Optimization from the University of Waterloo. Her areas of expertise include MVS, Parallel Sysplex, SMP/E, TSO/ISPF, CustomPac, and general systems management.

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

Gail Whistance
IBM Poughkeepsie

Merilee Osterhoudt
IBM Poughkeepsie

Don Ault
IBM Poughkeepsie

Mike Spiegel
IBM Poughkeepsie

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 247 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com`
- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Introduction

This chapter presents the concept of workload management and also a basic introduction to the Workload Manager (WLM) OS/390 component. Both WLM concepts and its terminology are covered.

You should not need to read this book cover-to-cover to find it useful. Feel free to turn directly to the topic of immediate interest to you.

1.1 Introduction to Workload Management

This section provides an introduction to the workload management disciplines, focusing on its goals and the problems it solves. In addition, there is a discussion of how these goals are achieved on the S/390 platform, both prior to WLM goal mode (compatibility mode) and with WLM goal mode.

1.1.1 Workload Management Definition

The purpose of *workload management* is to balance the available system resources to meet the demands of S/390 subsystems work managers such as CICS, Batch, TSO, UNIX System Services, and Webserver, in response to incoming work requests.

OS/390 tries to achieve the needs of the workloads (response time), for example as described in Service Level Agreements (SLAs), by attempting the appropriate distribution of resources without over committing them. Equally important, OS/390 maximizes system use (throughput) to deliver maximum benefit from the installed hardware and software platform.

1.1.2 Service Level Agreements

This section describes the concept of a service level agreement (SLA). It is strongly recommended (but not required) that you have an SLA in place when performing any workload management activity.

The human perception of the performance of a system is often subjective, emotional, and imprecise. The perception is related to throughput, response time and their frequency distributions. In order to make it more objective and more coupled with business needs, the concept of an SLA was introduced.

An SLA is a kind of contract (between IS and users), objectively describing the following:

- Average transaction *service time* or *response time* for:
  - Network
  - I/O
  - Processor
  - Sum of the previous
- Distribution of service time and response time (a measurement of how variable they are).
- *Throughput*, or external throughput rate (ETR). This measurement is not recommended to be in an SLA because ETR includes variables not under

control of the IS department, such as the number of users and user "think time".

- System availability figures (percentage of time the system is available to users).

A transaction or a business unit of work is the product of a human interaction (online or not) with a CICS subsystem, a Web server, a batch job, and so on. Figure 1 to shows an example of an SLA.

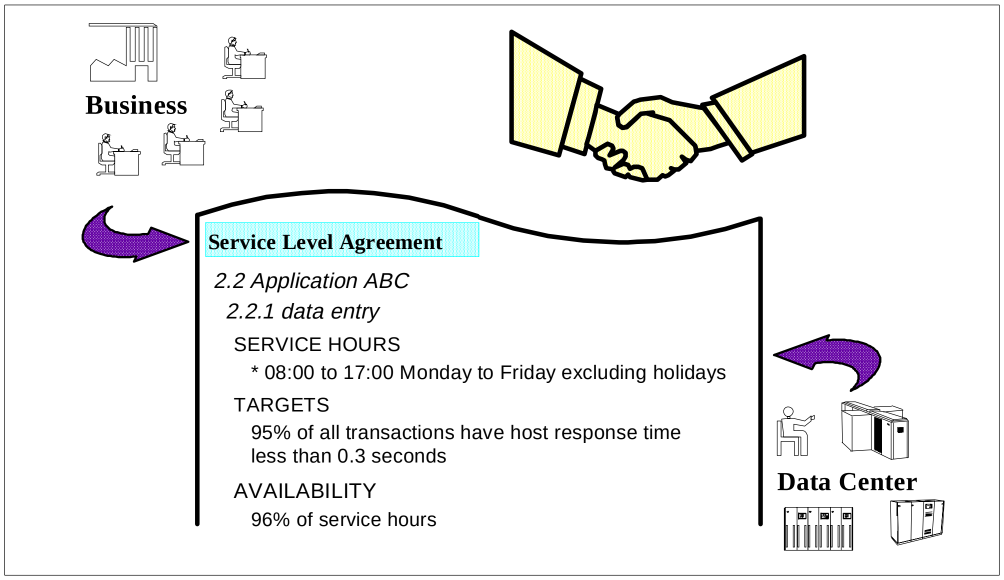


Figure 1. Service Level Agreements

1.1.3 OS/390 Workload Management in Compatibility Mode

Here we list some of the issues associated with workload management when in OS/390 compatibility mode:

- Translation complexity

  OS/390 running in compatibility mode requires you to translate your business goals, such as response time as described in SLAs, into the technical terms of System Resource Manager (SRM) languages (IEAIPSxx, IEAICSxx, and IEAOPTxx PARMLIB members). This translation process requires highly skilled staff, and can be protracted, error-prone, and eventually in conflict with the original SLA.

  Many of the SRM controls are geared towards implementation. So, you must tell SRM precisely *how* to process the work instead of just saying that a workload should be processed quickly. You need to understand the logic of swapping very well in order to tell SRM that one transaction is more important than another; then chances are that when resources are stressed, the second transaction is swapped out and the first is kept swapped in.

  Additionally, it is often difficult to predict the effects of changing a parameter, which may be required, for example, following a system capacity increase. This may result in unbalanced resource allocation, that is, feeding work one resource while starving it of another.

- Many systems (and subsystem) externals

One of the greatest strengths of OS/390 is the richness of functions it can deliver to its users. At the same time, it is equally true that one of the greatest weaknesses of OS/390 is the complexity of system tuning.

External tuning knobs, or controls, have proliferated among various subsystem work managers. Also, there are many monitoring and reporting products for keeping track of performance issues. Each subsystem has its own terminology for similar concepts. In addition, each has its own controls, and these are not coordinated with SRM. Furthermore, the subsystems have no knowledge of what happens in the OS/390 system. The subsystems send transactions to OS/390 with no regard for the available system capacity.

- Workload distribution among multiple OS/390 images

  It is very difficult to balance the load among different systems. There is no mechanism to automatically transfer workload among these systems and no overall sysplex view of where spare capacity exists.

1.1.4 OS/390 Workload Management in Goal Mode

Starting with MVS/ESA V5, a set of OS/390 components and other products implement new workload management techniques. Collectively, they address the problems described in 1.1.3, “OS/390 Workload Management in Compatibility Mode” on page 2. The following is a list of some of these components and products (presented in their base releases):

- New Workload Manager (WLM) MVS/ESA V5 component
- Modified SRM MVS/ESA V5 component
- DFSMS/MVS
- CICS/ESA
- IMS/ESA
- DB2
- IRLM
- CICSPlex/SM
- ACF/VTAM
- RMF
- SDSF

SRM implementation using the controls in IEAIPSxx and IEAICSxx was becoming unmanageable as new workloads were introduced, and as multiple systems were being managed together as a single image. With MVS/ESA V5, a new WLM-based performance management architecture was introduced. The next section introduces the solution, WLM *goal mode*.

1.2 Workload Manager (WLM) in Goal Mode

In this section we cover an overview of the WLM component, a major player in OS/390 workload management.

WLM was introduced as a new component of MVS/ESA V5 associated with SRM. From this release onward, MVS can run in either *compatibility mode* or *goal mode*, at the discretion of the installation.

When in goal mode, WLM allows:

- More effective use of a single system, due to the following productivity improvements:

    - Reduced complexity

      WLM provides fewer, simpler, and more consistent system externals that reflect goals for work expressed in terms commonly used in an SLA. You define performance goals for your workload, and WLM and SRM matches resources to meet those goals by constantly monitoring and adapting the system. It is now even more important that each installation develops an SLA based on its business needs and priorities. Even when there is no formal SLA, there are performance expectations that can be translated into an informal SLA.

      Installations no longer need to specify parameters to manage: CPU constraints, swapping decisions, storage isolation, expanded storage usage (no more criteria age), and so on. When in goal mode, an OS/390 system no longer uses the IPS and ICS members. The OPT is still alive - but with no expanded storage (ES) criteria age, no MPL happy values for contention indicators, no logical swap controls and so on.

    - Hands-off system management

      WLM enforces your performance goals through the manipulation of swapping, paging, I/O priority, dispatching priority, and number of server address spaces, without manual intervention.

    - Effective service management

      WLM provides consistent and consolidated performance data covering OS/390 and its subsystems. For example, CICS informs WLM of transaction externals, so WLM can map the work to the goals, and also informs WLM of internal performance data. RMF uses WLM services to get this data to build reports.

    - Effective performance management

      Every installation wants to make the best use of its resources, maintain the highest possible throughput, and achieve the best possible responsiveness. WLM helps by managing the trade-off between meeting your *response* goals (policy adjustment routines) for work and getting the best *throughput* (resource adjustment routines).

- Even more effective use of a Parallel Sysplex

  Parallel Sysplex has mechanisms for leveling the load across different OS/390 systems through dynamic workload distribution algorithms.

  WLM eliminates the need to manage each individual OS/390 image in a sysplex by providing a unique global performance policy. It also helps the capacity planning effort because it ensures work is sent to the systems with available capacity.

- Increased capacity and adaptability to new workload

  WLM provides a way for you to increase the number of systems and the type of workloads in your installation without greatly increasing the skill level or number of staff necessary to manage the environment.

WLM provides programming interfaces for communication with subsystems and performance reporters.

Summarizing the advantages of goal mode:

- Simplicity, using the same language as your business agreement with your customers (SLA)
- Automatic resource adjustment as the configuration changes
- Recognition by SRM of other subsystems' transactions besides TSO and batch
- Extremely well-suited to sysplex (base or parallel)
- More consistent allocation of resources than compatibility mode

1.2.1 New WLM Goal Mode Players

With WLM goal mode, the scenario of workload management activities has changed. We can now define two players, the Performance Administrator (a human) and the Performance Manager (SRM and WLM code) as follows:

- The Performance Administrator role

  The Performance Administrator is in charge of defining and adjusting performance goals. WLM introduces this role. The Performance Administrator is responsible for *defining* the installation's performance goals based on business needs (SLA) and current performance. This explicit definition of workloads and performance goals is called a *Service Definition*. WLM provides an online panel-based TSO/ISPF application (called the WLM application) for setting and adjusting the service definition.

  WLM collects performance and delay data in the context of the service definition. This data, as well as the application's goals versus actual data, are available to reporting and monitoring products, such as RMF, EPDM, PM for OS/390 or vendor products, for integration into more detailed reporting.

  The Performance Administrator can use these reports to analyze the goals and the current values. One way of analyzing the behavior of the system is to look at the service units consumed. These values are dependent on the service definition coefficients. Refer to 4.2, "General Recommendations for Migration" on page 132 to get more information about service definition coefficients.

  To summarize: the Performance Administrator performs the task of *defining* and *analyzing*.

- Performance Manager role

  The Performance Manager is the process WLM uses to decide how to match the available resources to the incoming work according to performance goals.

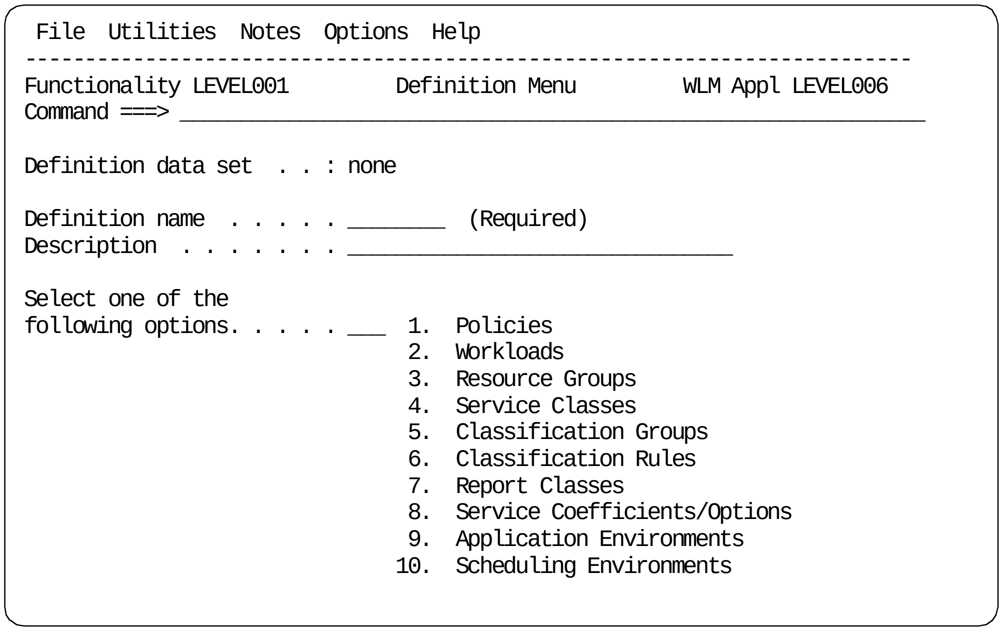
  WLM algorithms use service definition information to process work towards performance goals. The algorithms, with the cooperation with the subsystem work managers, use internal monitoring and feedback to check how well they are doing in meeting the goals.

  Therefore, the Performance Manager consists of the combined cooperation of various subsystems (CICS, IMS, JES, TSO/E, UNIX System Services) with the OS/390 Workload Manager (WLM) component.

  In summary, the Performance Manager executes the task of *managing* and *monitoring*.

1.2.2 WLM Application

The WLM ISPF-based application is a set of panels where the installation can enter and modify its service definition. Refer to Figure 2 to see the main panel.



```
 File  Utilities  Notes  Options  Help
--------------------------------------------------------------------------
Functionality LEVEL001        Definition Menu        WLM Appl LEVEL006
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 2. Main Panel of the WLM Application

For more information on the WLM application, refer to 3.3, “Service Definition Implementation” on page 104.

In your service definition, there are the various elements that WLM uses to manage the workloads. These include:

- Service policies: there is no predefined policy. You must define at least one policy, though it might be a null set of overrides.
- Workloads: these are arbitrary names, used to group various service classes together for reporting purposes.

  You must define at least one workload.
- Classification rules: these associate units of work (transactions, jobs and so on) with service classes.
- Service classes: within a service class, you define the goals for that work.
- Goals: the desired level of service that WLM uses to determine the amount of resource to give to a unit of work.

These elements are described more fully in the following sections. Figure 3 on page 7 shows the hierarchy of the definitions within a WLM service definition.

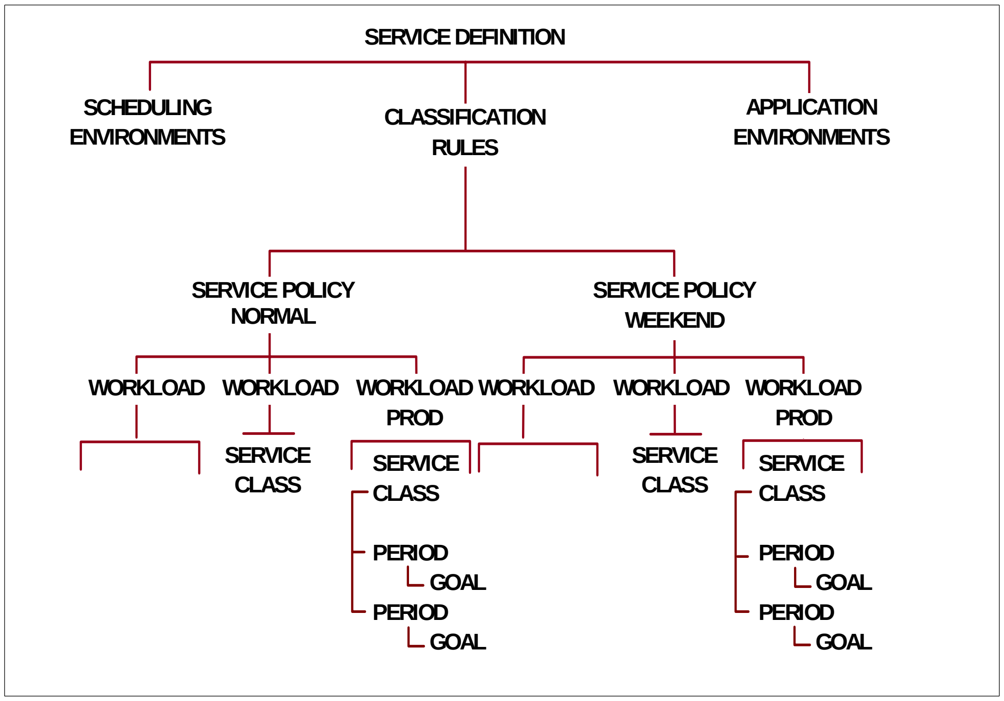


Figure 3. Hierarchy of Principal Elements in a WLM Service Definition

1.2.3 Introduction to Workload Classification

OS/390 is able to manage many different workloads, each with distinct business importance and processing characteristics. In order to accomplish this, the installation must categorize the arriving transactions to OS/390. In compatibility mode, the categorization is contained in the IEAICSxx SYS1.PARMLIB member. In goal mode, you use c*lassification rules*.

Classification rules are the rules WLM uses to associate a transaction’s external properties, also called work qualifiers (such as LU name or user ID), with a transaction goal. These goals are located in service class definitions. Classification rules and service classes are both defined in a service definition. Refer to Figure 4 for a pictorial view of the classification rules concept.

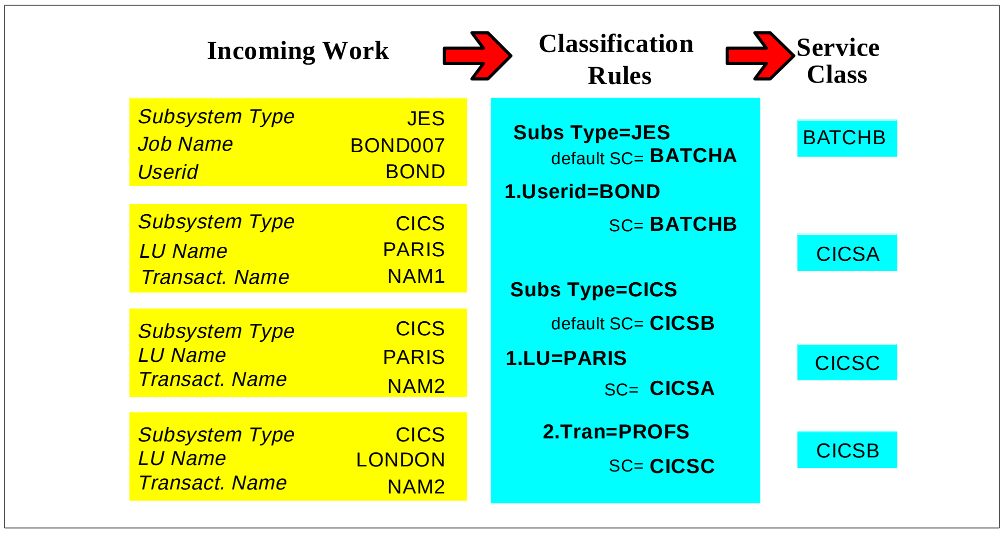


Figure 4. Workload Classification

Following are some examples of work qualifiers, which are used in Classification Rules:

- Subsystem type: IMS, JES2/JES3, TSO/E, STC, CICS, OMVS, DB2 (parallel queries), DDF, IWEB
- LU Name/Netid
- Transaction Name/Job Name. Transaction name for ASCH is the jobname on the JOB card in the APPC/MVS transaction program (TP) profile. For CICS it is transaction ID. For IMS, it is the CODE parameter in the IMS TRANSACT macro. For OMVS, it is the jobname in the JCL card defining the transaction program. For fork/spawn address spaces, this is the 7-character user ID plus a number from 0 to 9.
- Userid
- DB2 work qualifiers such as correlation ID, collection name, package name.

For a complete list of work qualifiers, refer to Chapter 4., “Migrating to WLM Goal Mode” 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 the example shown in Figure 5 on page 9:

```
* 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:
```

Figure 5. WLM Classification Rules Example

In the example in Figure 5, the PAYR transaction coming from Paris is going to be associated with service class CICSC and Report Class ABC.

A PAYR transaction coming from London is going to run in the CICSB service class by default.

For more information on classification rules, refer to 4.1, “Workload Classification Considerations” on page 129.

1.2.4 Service Definition and Service Policies

The service definition is defined to WLM through the WLM application. Refer to 3.3, “Service Definition Implementation” on page 104 more information on creating a service definition.

There is just one active service definition for the entire sysplex. The definition is given a name and stored in a WLM couple data set. The WLM couple data set is allocated via a utility function and accessed by all OS/390 images in the sysplex. Alternate or back-up service definitions can also be stored in an OS/390 partitioned data set.

A service definition contains just one set of workload classification rules, and consists of one or more *service policies*. There is just one active service policy at a time in the sysplex.

The active service policy can be switched by an operator command (like a SET IPS command in compatibility mode) to reflect changes in performance objectives.

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.

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 (CICS, IMS), by major application (production, batch, office), or by line of business (ATM, inventory, department).

The RMF Workload Activity report groups performance data by workload and also by service class periods within workloads.

1.2.5 Sampling Unit of Work States

Before we present the concept of a service class, you should first understand the meaning of unit of work (address space and enclave) states. This understanding will help you design the types of goals to use in a service class definition.

In order to enforce goals and to track the general performance of a sysplex, WLM samples the states of address spaces and enclaves every 250 milliseconds. Refer to 2.3.4, “Enclaves” on page 62, to get more information about enclaves. 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 (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 SRM resource (as ENQ or operator) or being idle for a reason other than those listed under 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 must add up to 100%; refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950.

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

- CPU

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

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

- Storage (paging and swapping)

  SRM 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 SRM decision) or is in the process of being paged in due to a swap in, the delay storage counter is incremented by *one* (single state). This is done at every sample.

  There is no *using* storage count.

- DASD I/O (optional)

  SRM 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 being delayed. This is done at every sample.

  SRM 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 or disconnect state, that is, executing the channel program. This is done at every sample.

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

1.2.6 Service Class

A service class (SC) is used to describe a group of work within a workload with equivalent performance characteristics, similar to performance group in compatibility mode. A service class can be associated with only *one* workload and can consist of one or more periods (through the DUR keyword). Three SCs (SYSSTC, SYSTEM and SYSOTHER) are automatically defined in a WLM policy. A service class has the following parameters:

- *Performance Goals*

  The goal types are:

  – Response time (average and percentile)

  – Velocity

  – Discretionary

- Business importance

  Each goal is qualified by its importance. Refer to 1.2.7, “Importance of a Goal” on page 13 to get more information about importance.

- *Resource group*

  A resource group optionally defines:

  – CPU service units required (protection)

  – CPU service unit limit (capping)

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

Since there are several types of goals, SRM needs some way to compare how well or how poorly work in one service class is doing compared to other work. That 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 with a workload having a percentile response time goal - and to determine which is farthest away from the goal (the largest PI).

The following list explains the meaning of the PI values:

- PI equal to one means that the SC period is exactly meeting its goal
- PI greater than one means that the SC period is missing its goal
- PI less than one means that the SC period is beating its goal

In a sysplex, a SC workload has a local PI for every OS/390 image, and a sysplex PI. RMF shows the sysplex and local PIs in the Workload Activity report.

WLM enforces performance goals by controlling access to resources.

1.2.6.1 Performance Goals

The concept of performance goals is vital to understand.

- *Response Time (RT) goals:* These are average transaction RTs or transaction RT percentiles (% of work that should complete within the response time). Unlike the RTO parameter in compatibility mode, RT goals do not cause artificial delays. Also, RT does not include network transmission. It does include queue time; for example, for batch workloads the response time includes the JES queue time. It is not applicable to STC address spaces because there is no concept of a transaction.

If you have a bad RT distribution, then using a percentile goal is better than using an average goal. For example, 9 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.

- *Execution Velocity goals*: These are a measure of the acceptable CPU, storage and I/O delays. This is similar to the RMF Monitor III workflow concept. In transaction-oriented service classes that control the IMS and CICS subsystems, velocity specification is not allowed because for these subsystems, using/delay sampling is done on an address space level only, not on a transaction level.

  The formula for the execution velocity is:

$$\text{velocity} = \frac{\text{using CPU + using I/O}}{\text{using CPU+using I/O + delayed by CPU + delayed by I/O + delayed by storage}} \times 100$$

  These using and delayed values are derived by sampling the address space and enclave states, as described in 1.2.5, “Sampling Unit of Work States” on page 10.

  The inclusion of I/O delay and I/O using in the formula is optional. You will activate the use of I/O delays when calculating the execution velocity by changing the *I/O priority management* setting to yes (using the WLM application, option 8); WLM then dynamically sets I/O priorities based on goals and I/O activity.

  Note: You might see some changes in your velocity values when you change the I/O priority option.

  There is a pragmatic reason for excluding unknown delay state samples from the execution velocity formula: some of the unknown delays include, in part, idle states that SRM cannot detect. Also, there is the possibility of including in the denominator the delays experienced by WLM initiators, that is, the delay suffered by a batch job before it starts.

  Velocity has the disadvantage (when compared with RT) of not being an intuitive concept and needs to be revisited after a hardware upgrade. Refer to 4.2.10, “Using Execution Velocity Goals” on page 140 for more information about execution velocity.

- *Discretionary goals:* This is a third type of goal, meaning “do the best you can”. It is usually used for batch jobs or very low importance test work.

  A discretionary goal has special properties such as:

  - The PI is always equal to 0.81.
  - It always runs in a low mean-time-to-wait (MTW) dispatching priority.
  - It is the goal of the SYSOTHER service class.
  - It is a universal donor (since OS/390 2.6, it can also be a CPU receiver).
  - It is only allowed in the last period of a service class definition.
  - It has a fixed IO priority of 248.
  - It has a goal importance equal to 6.

    - It is associated with an internal service class called $SRMDInn.
    - It is a candidate for individual storage control via Working Set Management.
    - It can get service when WLM caps overachieving SC periods (PIs less than one). This was introduced with OS/390 Version 2 Release 6.
- *SYSTEM*: A WLM-defined service class used as a default for certain system STC address spaces such as MASTER, GRS, DUMPSRV, SMF, CATALOG, RASP, XCFAS, CONSOLE, IOSAS, SMXC, ANTMAIN, JESXCF, ALLOCAS, IXGLOGR, WLM and others. It is assigned DP=255, IOP=255, and there is no specific storage protection because the address spaces are assumed to be cross-memory servers. If the caller's address space suffers page faults, then storage protection is activated.
- *SYSSTC*: A WLM-defined service class used as the default for system tasks and privileged address spaces. They are assigned a DP of 254 (before APAR OW19265 the DP was 253), IOP=254 and no specific storage protection because the address spaces are assumed to be cross-memory servers. The multiprogramming level (MPL) is fixed and very high, resulting in no unilateral swaps for swappable STC address spaces.

  This is where important low-CPU usage address spaces (VTAM, JES, DB2) should run. If you want to define a service class for some of these address spaces, a high *Importance* and a high *Velocity* are recommended.

1.2.7 Importance of a Goal

When the service class period is not meeting its performance goal, the importance specifies how important it is to you that the work meets its goal. It indicates which service class period should donate resources to a higher importance service class period that is not achieving its goal. The 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.

There are two more importance levels in addition to these five:

- Zero is implicitly assigned to the SYSTEM and SYSSTC service classes.
- Six is assigned to any service class period with a discretionary goal.

Importance values of zero and six cannot be specified by the Performance Administrator in the WLM service definition.

1.2.8 Resource Group (RG)

The Resource Group definition describes the amount of CPU capacity available to transactions associated with specific service class periods. You can use an RG to:

- Limit (cap) the amount of CPU capacity available to service class periods. Capping is used in situations where you want to deny the access of remaining

CPU cycles to a set of service class periods. This can, for example, be useful if you are running a service bureau.

- Guarantee some minimum CPU capacity to service class periods that are missing their goals. It can be used to mimic a throughput goal for batch work. You do this by defining this work as discretionary and protecting it with a minimum CPU service rate.

In an RG, you can define:

- ***Capacity Maximum***: Maximum CPU service rate. This is the CPU service rate figure for capping. Transactions running in the service class periods associated with this RG should not consume more than this capacity of service units per second. This value is enforced, taking precedence over the service goals declared in the associated service class periods.

  Note: Use maximums with care as they can allow the CPU to go idle.

- ***Capacity Minimum***: Minimum CPU service rate. This is the CPU service rate that should always be available for the transactions running in the service class periods associated with this RG. This minimum is guaranteed, if sufficient work exists and the goal is not being reached. Capacity minimum does not have precedence over the service goals declared in the associated service class periods when the goals are being met.

The CPU service units used for the calculation of the CPU service rate described in capacity minimum and capacity maximum are *raw* (or unweighted) CPU service units, which means TCB service units are not multiplied by the service definition coefficient (SDC) CPU coefficient.

Refer to Appendix B of *OS/390 MVS Planning: Workload Management*, GC28-1761 for a table of processor capacities in terms of unweighted service units per second.

As previously mentioned, capping is useful for a service bureau kind of installation. It is now possible to limit the CPU usage of a specific workload under MVS. This eliminates the need for creating a new logical partition for this workload just to enforce capping. Note, however, that capping is performed at a sysplex level, rather than at the system level.

For example, say an installation wants to know the maximum value to specify to cap a set of service class periods to 20% of the total sysplex CPU capacity. To determine this number, use the sum of the service rates for each system (that is, the number of raw service units per second of TCB/SRB) and multiply it by 0.20.

This service rate varies with the CPU speed, the number of CPUs in the CEC, and the number of logical CPUs in the LP. For example, the 9672 R65 has 2378.47 service units per second of CPU and RX5 has 1908.85 service units per second of CPU. The second is smaller due to the MP factor. However, in an RX5 logical partition with six logical CPUs, the number is 2378.47 service units per second of CPU.

If the associated SC has a discretionary service goal, WLM achieves the minimum as long as service goals running in any other defined SC are not impacted. If other service goals are impacted, then WLM does not maintain the minimum. Refer to 1.2.8, “Resource Group (RG)” on page 13 for more information on RGs.

1.2.9 Application Environment

One problem facing a Performance Administrator is determining the optimum number of address spaces in a multi-address space server subsystem. Usually this determination is done at the subsystem level in a static fashion.

Application environments are used to automatically control the number of server address spaces for Webserver, DSOM, SQL Stored Procedures, and MQ/Series work. These subsystem work managers use a set of address spaces to execute their transactions.

Application environments are defined in the WLM service definition. They allow WLM to do the following:

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

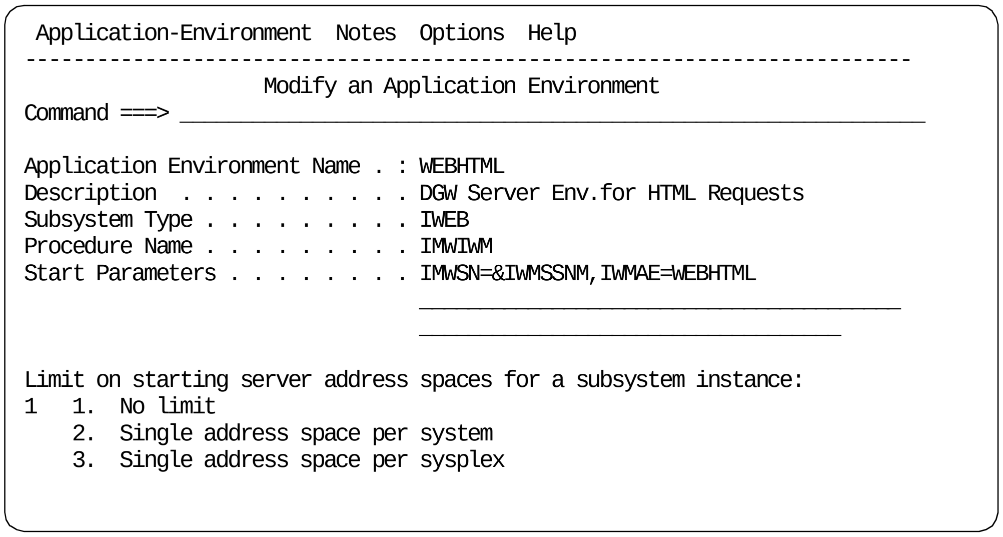
The Application environment (AE) can be thought of as a set of server address spaces belonging to the same subsystem instance, started by the same procedure, and having the same capabilities (in terms of program libraries, accessible data and security authorization). An AE can be used by more than one member of a sysplex.

Different transactions in an AE may 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 subsystem work manager. WLM samples the queues in order to decide if a new address space for the environment needs to be created. The subsystem work manager fetches work from these queues. WLM creates at least one subsystem work manager server address space per application environment/service class combination.

When defining an AE, you can specify the following:

- AE name: unique in the sysplex
- Work Manager: identifies subsystem 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 userid for security checking)
- WLM management options: special options governing WLM control of address spaces

Figure 6 shows an example of this panel in the WLM application.



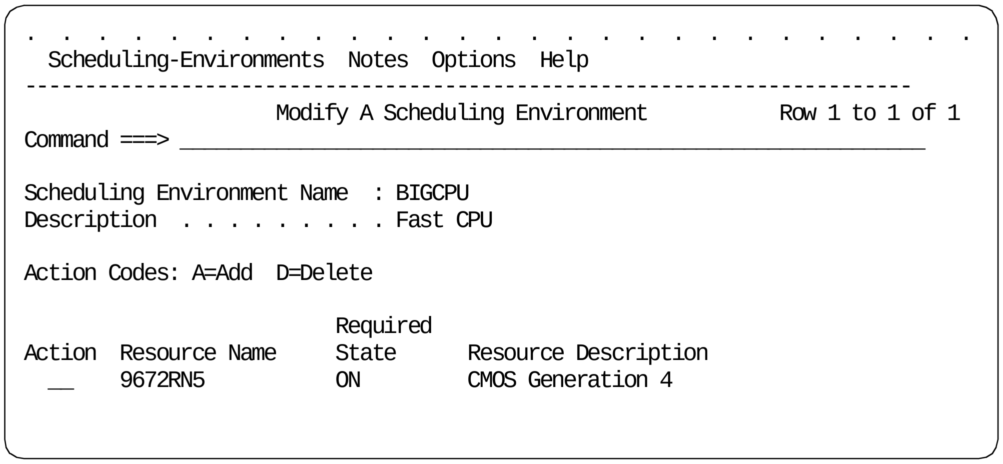
```
 Application-Environment  Notes  Options  Help
 ---------------------------------------------------------------------------
                     Modify an Application Environment
 Command ===> ______________________________________________________________

 Application Environment Name . : WEBHTML
 Description  . . . . . . . . . . DGW Server Env.for HTML Requests
 Subsystem Type . . . . . . . . . IWEB
 Procedure Name . . . . . . . . . IMWIWM
 Start Parameters . . . . . . . . IMWSN=&IWMSSNM,IWMAE=WEBHTML
                                  ________________________________________
                                  ___________________________________

 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 6. Application Environment Definition Screen

1.2.10 Scheduling Environment

A scheduling environment is another construct specified in the WLM service definition. It drives a function called resource affinity scheduling. Refer to 2.6.7, “Resource Affinity Scheduling” on page 85 to get more information about this function.

Figure 7 shows an example of the panel where a scheduling environment is defined.



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
   Scheduling-Environments  Notes  Options  Help
 ---------------------------------------------------------------------------
                      Modify A Scheduling Environment          Row 1 to 1 of 1
 Command ===> ______________________________________________________________

 Scheduling Environment Name  : BIGCPU
 Description  . . . . . . . . . Fast CPU

 Action Codes: A=Add  D=Delete

                          Required
 Action  Resource Name    State      Resource Description
   __    9672RN5          ON         CMOS Generation 4
```

Figure 7. Scheduling Environment Definition Screen

1.3 WLM Goal Mode Considerations

The following are some basic facts about WLM in goal mode that are helpful for understanding the other chapters in this book:

- If you want to define a policy in goal mode, it is mandatory to have the WLM couple data set allocated. The cross-system coupling facility (XCF), an OS/

390 component, is in charge of this allocation. XCF does not allocate such data sets if the system was IPLed in XCF-Local mode. You must be in *monoplex* or *multisystem* sysplex mode.

- You can run in goal mode without an allocated WLM couple data set. In this case you use a default policy with TSO running with a goal of 80% of transactions with a response time between zero and two seconds. The other workloads (not STC) run in service class SYSOTHER (goal = Discretionary). The STC workloads run in SYSSTC service class as usual.

  The default policy is adequate to allow the system to initialize so you can install and activate your own policy.

- Note that all the enclaves not associated with a specific service class default to the SYSOTHER service class. Refer to 2.3.4, “Enclaves” on page 62 to get more information about enclaves.
- When in a Parallel Sysplex, WLM does not use the coupling facility directly. It simply uses XCF for communicating with other WLM members. Each instance of the WLM address space joins an XCF group named SYSWLM.
- WLM uses the following methods to help transactions meet their goals:
  - Change access to resources for an executing transaction.
  - Increase or decrease the number of server address spaces serving a set of transactions.
  - Provide subsystems with sysplex capacity information to aid in distributing work across the sysplex.
- In order to achieve the installation goals, WLM tracks the following delays affecting the transactions:
  - CPU delay
  - Storage delay (page faults, swapping, and hiperspace activities)
  - I/O delay (optional)
  - Server queue delay (optional)

  If an important goal is not being met, WLM reacts depending on the type of the major delay:

  - CPU delay - dispatching priority (DP) is adjusted
  - Storage delay - storage isolation and/or swapping target MPL is adjusted
  - I/O delay - I/O priority is adjusted
  - Server queue delay - a server address space is added
- WLM understands the relation between server address spaces and transactions. If a CICS transaction is being delayed by a file-owning region (FOR), WLM can detect this and react to speed up the FOR address space.
- It is possible in a sysplex to have some OS/390 images in goal mode and others in compatibility mode.
- It is possible to switch a system into and out of goal mode while the system is running.
- WLM runs under its own task and address space because of the following reasons:

    - It communicates with other WLM instances in a sysplex through XCF protocol.
    - It owns data sets.
    - It uses the MVS Modify command.
- In compatibility mode there can be an active policy. There are WLM functions that are active in compatibility mode and require an active policy, such as the following:
    - Resource affinity scheduling (scheduling environments)
    - Application environments
    - Associating enclaves with a performance group number via the classification rules. The ICS can map a service class to a performance group number (PGN), where the DP of the enclave is set. In order to enable this mapping, the ICS has to be changed as follows:
        - A new service class keyword (SRVCLASS=) must be used.
        - The subsystem name keyword (SUBSYS=) is enriched, allowing all the subsystems supported by the classification rules in goal mode.
- The following lists the key functions of WLM:
    - Policy adjustment, to enforce the goals.
    - Resource adjustment, to improve the throughput. It includes working set management.
    - WLM services, to enable WLM to cooperate with subsystems to achieve WLM-defined goals and to provide meaningful feedback on how well WLM has achieved those goals.
    - Server address space management, using application environments, to allow WLM to control the number of server address spaces.
    - Dynamic workload balancing to select the best system or server to execute the transaction.
    - Resource affinity scheduling.
    - Business unit of work management (enclaves).
    - Batch Initiator management.

1.3.1 The New Goal Mode Cockpit

When in compatibility mode, you have many parameters and options to choose from. In goal mode, the majority of these controls vanish and you have just a few of them. The new ones with which you can command your workload management behavior could be called a “cockpit”. Refer to Figure 8 on page 19.

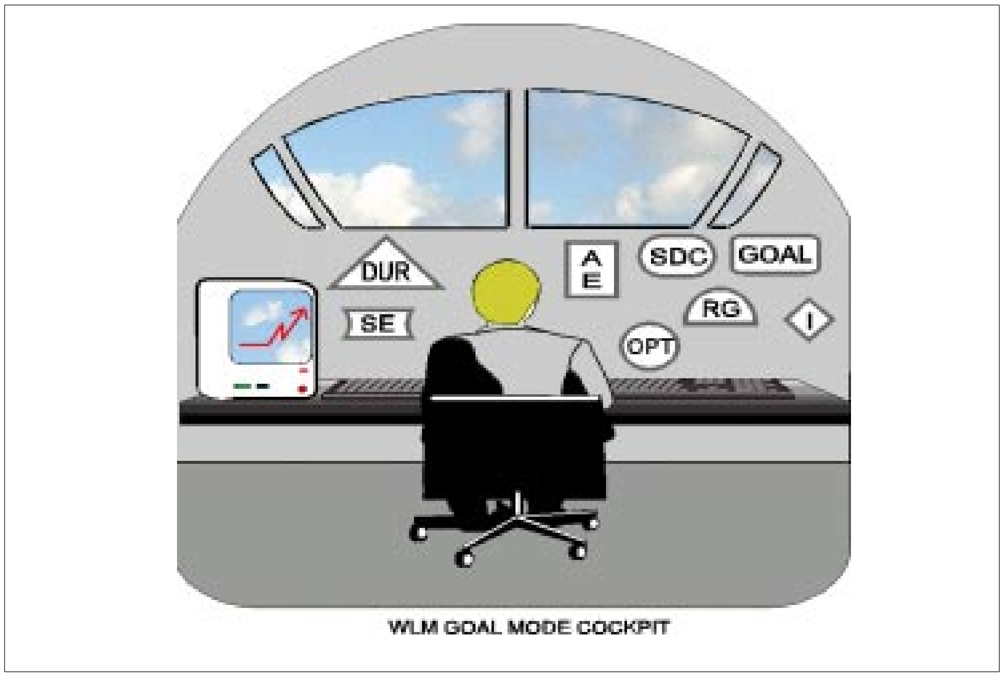


Figure 8. WLM Goal Mode Cockpit

Following is the list of variables you can use in goal mode in order to reach your objectives:

- The type of goal.
- The numeric value of the goal. (Be careful when setting unreachable important goals because WLM then could lose the capability of helping other service classes.)
- Relative Importance of the goal.
- The classification of work to assign the desired goals.
- Duration (when using multiple periods).
- CPU capping level (Resource Group).
- CPU protection level (Resource Group).
- Application environment variables. With these you can, for instance, specify the procedure name to be used to start the server address spaces.
- Scheduling environment content.
- Service definition coefficients, same as in compatibility mode.
- A few OPT keywords (ERV, available queue thresholds and fixed central storage thresholds).
- Faith and some I/O tuning.

Chapter 2. How WLM/SRM Works

This chapter provides a deeper insight into the workload management structure in an OS/390 environment. We describe the functions that comprise the workload performance management tasks.

It is a mistake to think that with the introduction of WLM, the SRM component has disappeared. SRM is still alive and is still responsible for managing the performance of the workloads. SRM manages work execution by dynamically managing system resources to achieve the goals associated with the work requests. For the purpose of this book, little distinction is made between WLM and SRM - only where it is required for deeper understanding.

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

This chapter also describes the decision-making process of workload management and how the CPU, storage, and I/O resources are adjusted based on customer goals. Furthermore, it describes the new services required in order to support new workload management concepts, like address space management and resource affinity scheduling.

We also cover some changes made to MVS/ESA and OS/390 that were necessary to solve many workload management issues and to support the new performance management concepts. These changes are the introduction of new types of dispatchable units.

2.1 OS/390 Workload Management Structure

Workload management, starting with MVS/ESA V5.1, can be divided into several functions:

- A new MVS component, that is, 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 modified SRM component formed by two major functions:
  - Policy adjustment
  - Resource adjustment
- New versions of *subsystem work managers* supporting the WLM services

Throughout this redbook the expression *subsystem work managers* means subsystems such as CICS, IMS, IWEB and others, exploiting WLM services. For a description about the subsystem work managers exploiting these new functions see Chapter 4, “Migrating to WLM Goal Mode” on page 129 and Chapter 5, “Considerations for New Workloads” on page 171.

2.2 Understanding Goal Mode

There are some structural differences in SRM before MVS/ESA V5.1 as compared to later releases. With the introduction of MVS/ESA Version 5.1, WLM can run in one of two modes: *compatibility* mode or *goal* mode.

In goal mode, the resource distribution is driven by WLM using the installation goals and not by the former IEAIPSxx and IEAICSxx PARMLIB members. In goal mode, there is no external means for a system programmer to control such things as dispatching priorities and working set size. This chapter describes the logic and algorithms used by in WLM goal mode.

There are two main functions influencing resource distribution:

- Policy adjustment
- Resource adjustment

Before we cover these two functions, we describe some of the algorithms used by policy adjustment and resource adjustment. They are here to familiarize you with the new concepts and new terminology of goal mode. When possible, correlations are established with the compatibility mode concepts.

2.2.1 SRM Algorithms

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. SRM manages these resources and the transactions using them by measuring the various delays that dispatchable units are experiencing.

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

CPU delays Being ready to run, but not being 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:

- Being delayed for a page-in operation from AUX
- Being swapped out, either logically or physically, with no MPL available for a swap-in
- Being swapped-out physically with being delayed because of AUX page-in operations.

I/O delays Having an I/O operation in the IOS queue or pending in the channel subsystem.

Besides the physical resource delays such as CPU, storage, and I/O, there are other tracked delays such as a job being delayed because of the lack of an appropriate initiator, or in general a transaction being delayed because the lack of a server address space. These delays are covered later in this chapter.

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

Access to DASD devices is managed by the sysplex I/O priority management function introduced in OS/390 Version 1.3.

2.2.1.1 Dispatching Priority Control

A *dispatching priority* is a number from 0 to 255 associated with a dispatchable unit of work (TCBs, SRBs). It is placed in an address space or in an enclave control block (for more information on enclaves, refer to 2.3.4, “Enclaves” on page 62). 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.

Some important and/or unique aspects of dispatching priorities in goal mode are as follows:

- 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. (This is also true in compatibility mode.)
- The time slicing function previously available through controls in the IEAIPSxx member if PARMLIB is no longer available in goal mode.
- Based on the performance index (PI) values and the reason for delay (CPU, for example), a service class period can be selected as a CPU receiver, and service class periods can be selected as CPU donors. As a consequence, the policy adjustment routine changes their relative dispatching priorities.
- There is only one mean-time-to-wait (MTTW) dispatching priority group in goal mode. This MTTW group is only used for service class periods with a discretionary goal. There are a set of dispatching priorities at the bottom of the range that only apply to discretionary units of work. These priorities are used to rank the I/O and CPU “boundedness” as is done in compatibility mode.
- If you *do not* activate the I/O priority function of WLM in goal mode, the I/O priority queueing is exactly as it would be accomplished with an IPS specification of IOQ=PRTY. Therefore, if a goal is not being achieved and the dispatching priority is raised to address that, then the I/O queueing priority is increased as well. If you have activated the I/O priority function (as we recommend), the I/O priority is completely independent of the dispatching priority. For more information see 2.2.1.9, “Sysplex I/O Priority Management” on page 38.

The compatibility mode external parameter DP (in IEAIPSxx) should not be confused with the service definition *importance* keyword. They have distinct functions. A high importance service class period does not necessarily imply a high dispatching priority. The *importance* value is used, for example, to determine which service class period should be a receiver of resource when there are one or more service class periods are not meeting their goals.

The RMF monitor II ASD report shows in the field DP PR the dispatching priority of the service class periods. Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information about this field. SDSF, in the Display Active (DA) screen, also shows the dispatching priority in the DP field.

2.2.1.2 Swap Control

The *domain* constructs used in compatibility mode for swapping control are no longer used in goal mode. Therefore, the IEAIPSxx domain controls once managed manually by the installation in compatibility mode are also gone. The following is a list of such controls:

- Constraints, where the target multiprogramming level values (ITMPL and OTMPL) are limited in a specific range.
- The RTO option.
- DSRV, ASRV, and FICIDX, where the installation assigns a relative importance to each domain (contention index).
- MPL adjustment constants as defined in OPT:
  - RCCCPUT: CPU utilization threshold
  - RCCPTRT: Page fault rate threshold
  - RCCUICT: UIC threshold
  - RCCFXTT: Fixed online storage threshold
  - RCCFXET: First 16 megabyte central storage threshold
  - MCCFXEPR: First 16 megabytes fixed storage %
  - MCCFXTPR: Online storage fixed %
- Swap rate scaling factor
  - SWAPRSF: Weight cost of doing exchange swap

Two significant modifications are implemented in goal mode with respect to swap control:

- The service class period absorbs the function of a domain.
- The variation of ITMPL and OTMPL is controlled - no external parameters are supported. Targets are set for service class periods based on goals and overall system throughput.

However, the following swap control options are still available in OPT:

  - MCCFXEPR: First 16 megabytes fixed storage %
  - MCCFXTPR: Online storage fixed %
  - SWAPRSF: Weight cost of doing exchange swap

MPL in_target (ITMPL) and MPL out_target (OTMPL) in goal mode

The meaning of ITMPL and OTMPL is adapted to the new goal mode concepts, as follows:

- MPL in_target (ITMPL)

  The number of service class period address spaces required to be swapped-in order 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.

SRM and Swap control in goal mode
The MPL targets are not set by external parameter. 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. SRM 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 2.2.2, “Policy Adjustment Function” on page 43 and 2.2.3, “Resource Adjustment Function” on page 52 for more information on this subject.

2.2.1.3 Swap Protect Time

In compatibility mode, an installation defines in IEAOPTxx (LSCTME keyword) in SYS1.PARMLIB a range for a system-wide *think* time. This value is used to specify a number of seconds that address spaces would be allowed to stay logically swapped in central storage. Until this time expired, the address spaces could not be moved to AUX or expanded storage, and so would not incur any swap in delay when they became ready. The system-wide think time is controlled in compatibility mode, based in the occupancy of central storage. This concept of logical swapping applies mainly to TSO address spaces in terminal wait state (waiting for transactions) and initiators in long waits.

In goal mode the logical swap algorithm is modified and the compatibility mode parameters in IEAOPTxx that control this feature are no longer available, such as:

- LSCTMTE=(xxxxxxx,yyyyyyy)]
- LSCTUCT=(xxxxx,yyyyy)]
- LSCTFTT=(xxx,yyy)]
- LSCTFET=(xxx,yyy)]

The *swap protect time* concept covered in this section is a replacement for the compatibility mode logical swap algorithm.

In goal mode, swap protect time is the time in milliseconds that a swapped-out address space remains in processor storage (central plus expanded) before becoming a candidate for AUX. It is a value set specifically for each service class period by SRM.

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 SRM 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.

The major differences between compatibility mode and goal mode in logical swap are:

- System think time in compatibility mode is a system-wide figure; swap protect time in goal mode is per service class period.
- Logical swap in compatibility mode protects the address space pages in central storage only; swap protect time in goal mode protects the address space pages in processor storage.
- In compatibility mode, the installation can have some control over the algorithm. In goal mode, there are no externals.

2.2.1.4 Storage Targets

SRM before MVS/ESA 5.1 had two algorithms to control the occupancy of storage, these were:

- Least recently used (LRU), to keep in storage the most recently used pages.
- Explicit storage isolation, keep in processor storage the most important pages for my business

An installation had control over these two algorithms through IEAOPTxx and IEAIPSxx parameters, and in many cases they run in parallel.

In goal mode, the 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).

Storage Isolation Algorithm in Compatibility Mode

Before MVS/ESA 4.2, the range for the target values associated with storage isolation had to be defined by the installation.

In general, some of the storage targets (the protective targets) keep SRM from taking pages from address spaces, while other storage targets (the restrictive targets) tell SRM which address spaces to take frames from first.

From MVS/ESA 4.2 upwards, but still in compatibility mode, the concept of implicit storage isolation was introduced. In this case certain *managed* address spaces have their storage targets totally controlled by a function called *working set management*. The reason for the implementation of this new concept was the difficulties posed to customers when defining the explicit range for target values.

Storage Isolation Algorithm in Goal Mode

In goal mode the storage isolation is totally implicit and there are no available externals.

The policy and resource adjustment routines control the use of storage through storage targets and the LRU algorithm. These targets look very much like the ones defined in compatibility mode by the working set manager (WSM).

In goal mode we have four types of storage targets:

Protective Processor: A storage target to protect a target number of pages in processor storage. Do not migrate pages from expanded storage 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 from expanded storage 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.

Refer to Figure 9 for a graphical representation of the relation between protective and restrictive targets.

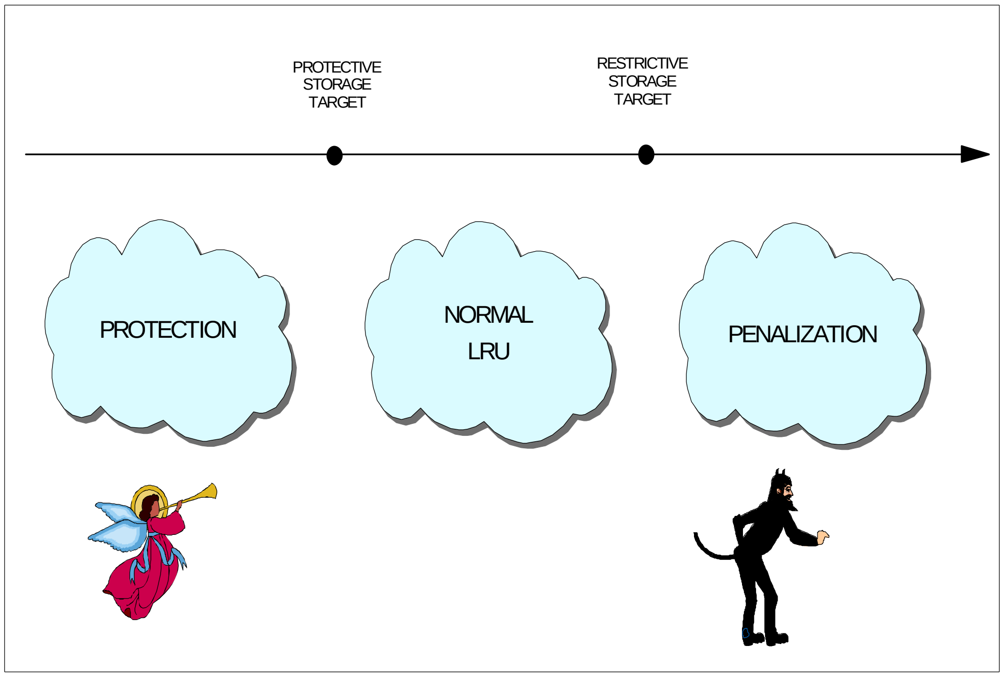


Figure 9. Storage Targets

Storage targets are applied to each individual address space. However, service class periods for trivial transactions, such as TSO first period, have a *single* protective processor target for all the address spaces because there is not sufficient time for a customized evaluation. Service class periods for non-trivial work may have different storage targets for each address space in the service class.

Policy adjustment raises the storage targets for a service class period receiver when its major delay is for storage delays and it is missing its goals. This delay indicates that the address spaces pages belonging to the service class period are suffering delays due to page faults from AUX. The service class period donors have a storage target decrease.

Resource adjustment also modifies storage targets of some service class periods to improve the use of storage.

For server address spaces, for example CICS and IMS regions managing transactions with response time goals, SRM proactively manages the working set, providing storage isolation for these address spaces. If a server address spaces' protective targets are significantly less that the resources it already owns, the protective targets are raised. This will provide the server a larger degree of storage protection should a sudden increase in demand for resources take place, giving time for the policy adjustment code to run and re-evaluate the situation.

2.2.1.5 Expanded Storage Processing

This processing decides where to send a page that is leaving central storage. There are two candidate storage devices: expanded storage (ES) and auxiliary storage (AUX).

In goal mode, this processing is controlled by the resource adjustment routine which decides where pages stolen from central storage should be sent: to ES, or to AUX. It is a replacement for the user-controlled criteria age values in compatibility mode.

Before we go deeper into goal mode processing, let us examine how it is implemented in compatibility mode.

Expanded storage processing in compatibility mode

The pages leaving central storage are classified in the following categories:

- Stolen single page (or stolen blocked pages)
- Changed trimmed pages of a swapped-out address space and its companion data spaces
- Working set pages of a swapped-out address space (and its companion data spaces)
- VIO window pages
- HSPSERV SWRITE pages moved from an address space to a standard hiperspace

The decision between AUX and ES is based on the type of the page (criteria age) and in the migration age concept.

Migration age measures, on average, how long (in seconds) a page has remained unreferenced in ES. Migration age is an inverse measurement of expanded storage contention. That is, when contention for expanded storage is high, the migration age is low; when contention is low, the migration age is high.

Migration age is a system indicator and is periodically derived by the OS/390 real storage manager (RSM).

Criteria age is associated with the page type. The page type depends on:

- Application type (batch, TSO, system, other)
- Page category

The lower the criteria age, the more important the page is to your business, so the greater the chance the page will go to ES, instead of AUX. Criteria ages are defined in the IEAOPTxx member in the keywords starting with ESCT. It is also possible to relate the pages of a domain with a specific criteria age, through the keyword ESCRATBX in IEAIPSxx.

The algorithm compares migration age with criteria age:

- If the criteria age supplied by the installation (or defaulted) is less than the migration age, then we meet the criteria and move the page to ES.
- If the criteria age supplied by the installation (or defaulted) is greater than the migration age, then we do not meet the criteria and move the page to AUX.

Expanded storage processing in goal mode

The first modification in goal mode is regarding the types of pages there are.

There are only four categories of pages defined in goal mode:

- Swap working set pages. These are the pages automatically brought to central storage at swap-in.
- VIO pages.
- Standard hiperspace pages, which are pages written to a standard hiperspace.
- Stolen pages and swap trim pages (the ones stolen at swap-out).

The concept of comparing criteria age with migration age was dropped. As a replacement, three policies were defined:

Protected:	Direct pages to ES, and protect a specific number of pages in ES.
LRU:	Direct pages to ES, and allow pages to be migrated in LRU order.
Space Available:	Direct pages to AUX unless there is significant space available on ES.

A page of a certain type may have one of the three policies associated with it, depending on WLM decisions.

The granularity of the controls is at the service class period level, except for those address spaces that are eligible to be managed with an individual ES policy where:

- The address spaces has a velocity goal.
- The address spaces has a discretionary goal.

- The address space is being managed as a server in a dynamic internal service class (DISC). Refer to 2.2.1.6, “Server Topology” on page 30 for more information on DISC.
- The address space has a long response time goal (> 20 seconds).

All address spaces for trivial transactions, such as TSO first period, have a constant service class period-wide policy and are managed together.

2.2.1.6 Server Topology

Server topology is an algorithm used to track, in a multi-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 is 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. Refer to 2.5.1, “The Work Manager Services” on page 69. The server topology is only implemented for CICS and IMS transaction response time goals. Other subsystems use enclave tranactions 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 being 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.

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

Customers 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 customer-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 (DISC) are used to manage to the external service classes goals. For example, assume that the WLM wants to help a given service class (ESC) representing some CICS transactions. WLM must know which address spaces must be given more resource because the CICS transactions in that service class run in multiple address spaces. A DISC is the set of address spaces that serve a given set of ESCs, and it is the set of address spaces in the DISC that will be given the resources.

DISCs have special names defined internally by WLM as *$SRMSnnn*. The number of $SRMSnnn service classes depends on how many external service classes are served and by how many combinations of server address spaces.

Server topology is the function used to create the DISC and map it to the external service classes (ESC) representing transactions with goals.

Each DISC contains a set of server address spaces that are serving the transactions belonging to the same set of external service classes. All of these address spaces will be managed together.

Each recognized server address space has counters indicating how many times it is detected serving each ESC. As stated previously, the incrementing of the counters is done by either PB sampling, or by report and notify processing.

Refer to Figure 10 on page 32 for an example of a server topology. The DISCs on the left are the set of address spaces that were found to be providing service to the ESCs on the right.

How the Server Topology is Determined

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

Imagine that address space server 1 was found servicing service classes A, B, and C. Address space server 2 was found servicing service class D. Address space servers 3 and 4 are serving service classes A and B. The following list shows who is serving whom:

```
server 1 = {A,B,C}
server 2 = {D}
server 3 = {A,B}
server 4 = {A,B}
```

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 which are equal (that is, those address spaces that serve the same set of external service classes, and which belong to the same resource group). For this example, this exercise would result in the following dynamic internal service classes being built:

```
Dynamic Internal Service Class X = {A,B,C} formed by address space server 1
Dynamic Internal Service Class Y = {D} formed by address space server 2
Dynamic Internal Service Class Z = {A,B} formed by address space servers 3, 4
```

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

Refer to Figure 10 on page 32 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 D in this figure, then the only address space that can be helped is the DISC representing the AOR2. But, if WLM wanted to help CICS transaction Type A, then the weighting factors would be used to determine whether to help the AOR1 address space (DISC X), or the set of address spaces for AOR3 and AOR4 (DISC Z).

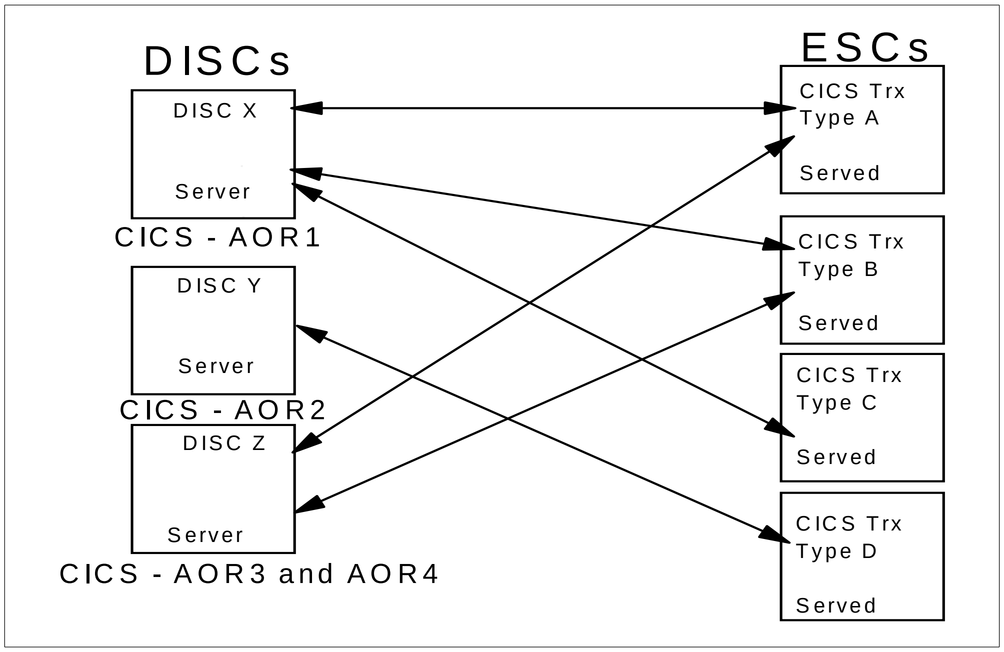


Figure 10. Server Topology Example

WLM uses the server topology in order to make its decision. WLM knows the delays of each address space and it knows how often each transaction was processed in each DISC. The ESCs have the goal and the PI, while the DISCs have the delays and the priority controls.

Assume that transaction Type A does not reach its goal and WLM chose this transaction to be a receiver. WLM has to decide between DISC X and DISC Z.

WLM recognizes, based on its address space samplings, that CPU is the major delay. The CPU delay for DISC X is 40 % and for DISC Z it is 20 %. (WLM can measure delays only for address spaces or enclaves, not for transactions.) WLM counts the number of transactions in each DISC. Transaction Type A was processed 100 times in DISC X and 150 times in DISC Z.

WLM computes the average weighed CPU delay for transaction type A:

```
(100 * 0.4) + (150 * 0.2) / 250 = 0.28
```

The same calculation is executed for the other resources as storage and I/O.

If ESC A needs to be helped, the delays calculated are used to evaluate which resource there is a need for.

If CPU is the resource required, then WLM computes the weight factor for every DISC:

```
DISC X:   #TRX * delay = 100 * 0.4 = 40
DISC Z:   #TRX * delay = 150 * 0.2 = 30
```

WLM recognizes that DISC X has the most impact on missing the goal of ESC A and it increases the dispatching priority of the address spaces belonging to this DISC. In our example it is only AOR1.

2.2.1.7 Performance Index Calculation

The Performance Index (PI) indicates how well a service class period is doing in meeting its goal, and allows it to be compared with other different types of goal (response time, velocity, percentile goal). Each service class period may have two types of PIs:

Sysplex PI: A PI that represents its global performance across all the systems in the sysplex.

Local PI: A PI that represents its performance in each local system of the sysplex. A service class may have more than one local PI if that service class has work executing on multiple systems in the sysplex.

The performance index is reported in the RMF Workload Activity report; both local and sysplex PIs are reported.

Through services provided by SRM, performance data is periodically exchanged between sysplex systems in goal mode. The information is sent between the systems and allows SRM to construct an approximate view of the status of the sysplex through the calculation of sysplex PIs. The aggregate view enables SRM 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 enforce globally the capping and the protection algorithms
- Information identifying the active service policy

WLM uses the PI value in the following ways:

- When the PI is equal to one, this means that the service class period is exactly meeting its goal.
- When the PI is greater than one, this means that the service class period is missing its goal.
- When the PI is less than one, this means that the service class period is exceeding its goal.

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.

Before OS/390 Version 2.4, or for systems without APAR OW25542 installed, WLM decisions were mainly influenced by the sysplex PI. The focus was to meet the service class period goal from a sysplex perspective. The local PIs on each system are only examined if all classes of work are meeting their goals based upon the sysplex PI, or if no action can be taken to help service class periods currently missing their goal. This could lead to a situation where a few users were having a very bad response time, but the overall average is meeting its goal. With the change introduced by OS/390 Version 2.4 (or OW25542, for earlier releases), the sysplex PI is still predominant but more attention is paid to individual systems.

The policy adjustment loop (see 2.2.2.1, “Policy Adjustment Loop” on page 43) uses the sysplex PI at the beginning of each interval (every 10 seconds) to decide whether there is an adjustment 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.

Sysplex PI evaluation remains the top priority, however, the local PI is no longer deferred until all sysplex PI processing is complete. With APAR OW25542 the sequence is:

1. Importance level 1 sysplex PI
2. Importance level 1 local PI
3. Importance level 2 sysplex PI
4. Importance level 2 local PI
5. Importance level 3 sysplex PI
6. and so on...

Performance Index for Goals

The following formulas for a PI calculation for a service class period depend on the type of the goal:

- Average Response Time Goal:

$$\text{Performance index} = \frac{\text{Actual average response time}}{\text{Goal average}}$$

- Execution Velocity Goal:

$$\text{Performance index} = \frac{\text{Goal execution velocity}}{\text{Actual execution velocity}}$$

- Response Time Percentile Goal:

$$\text{Performance index} = \frac{\text{Percentile actual}}{\text{Percentile goal}}$$

Note: Refer to “PI Calculation for Response Time Percentile Goal” on page 35 for an explanation of this calculation.

- Discretionary Goal:

  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*, as its PI indicates that it is always beating its goal.

A PI can be calculated over any interval. If you have RMF Monitor I with a 30-minute interval, RMF calculates a sysplex PI for that interval. Monitor III shows a PI calculated over any selected range.

PI Calculation for Response Time Percentile Goal

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. In Figure 11 on page 36, each *bucket* contains three pieces of information:

- The number of the bucket, from 1 to 14.
- The *percent of goal* refers to the response time intervals which start at 50% of the goal in bucket 1. Transactions that completed in an elapsed time less than half the response time goal are captured in the first bucket.

  Bucket 6 is 100% or equal to the goal and contains transactions that ended between 0.9 times and 1.00 times the response time goal.

  Bucket 12 is twice the goal.

  Bucket 13 contains counts from two to four times the goal, and bucket 14 is the number of transactions with a response time greater than 4 times the goal.

  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% the value of the goal.
- The third set of numbers for each bucket contains the number of transactions completed in each time interval and is referred to as the *transaction count*.

Note: Buckets in the 2-to-11 range are evenly distributed and serve to capture transactions that ended between the interval represented by 0.5 times the goal to 1.5 times the goal. In the example, bucket 2 contains the number of ended transactions with a response time between 1.0 and 1.2 seconds.

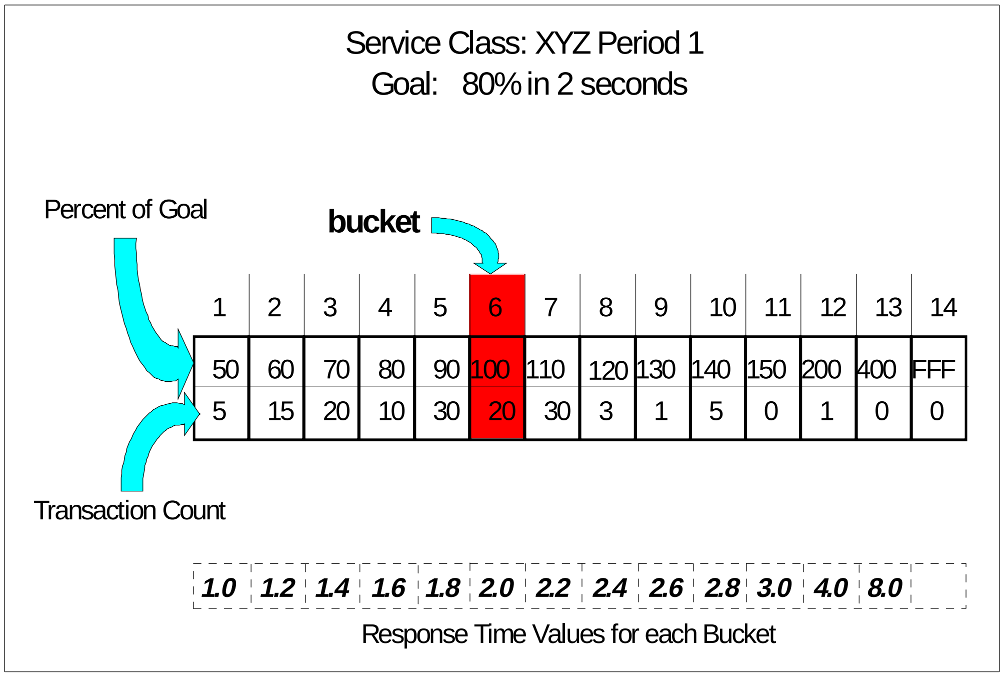


Figure 11. Response Time Distribution

When we add all the transactions that are captured in buckets 1 to 6, we know the number of transactions that achieved the goal. Using all this information, a reporter product can easily produce a response time distribution. This response time distribution is only provided in goal mode and is not reported for a report class.

In the example in Figure 11, the goal for the first period is: response time of 2.0 seconds for 80% of the transactions.

In Figure 11, the numbers at the bottom of the figure are the *percent of goal* numbers multiplied by the response time of 2 seconds. To get the actual time, the transaction distribution buckets are accumulated from the first bucket up to the bucket where the percentage of the defined goal is reached. The corresponding transaction time of this bucket defines the *actual time* as follows:

- Adding all the transactions completed, buckets 1 through 14 in the *transaction count* row: *140.*
- 80% of the transactions: *140 X 0.8 = 112.*
- Adding up buckets 1-n until we get a transaction count of at least 112, we get to bucket 7, representing an actual time of *2.2* seconds.
- The performance index = actual time divided by goal time. That means *2.2 divided by 2.0 = 1.1*.
- A PI of more than 1 means the goal is not being achieved.

RMF Response Time Report

RMF includes the response time distribution in several reports. In the Workload Activity report, the buckets are summarized to make up a four-line histogram. RMF Monitor III has a new report that includes these buckets. Other reports also use these statistics. For more information refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950.

2.2.1.8 Donor and Receiver Determination

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

1. Which service class periods require more resources to attain its goal.

   SRM selects one receiver from these candidates. The receiver is the most important work that needs help meeting its goal. For more information about selecting a receiver see 2.2.2.1, “Policy Adjustment Loop” on page 43.
2. Which resource bottlenecks (delays) are affecting the receiver. A service class that needs help must already have detected delays. A service class may be missing its goal, but if the delays are for resources that the SRM does not control, such locks or enqueues, then the service class cannot be helped, and will not be a receiver.
3. Which donors can donate resource to the receiver.

Refer to 2.2.1, “SRM Algorithms” on page 22, to get information about the SRM-tracked resources.

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.

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

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

This idea of plotting data was introduced in MVS/ESA Version 4 Release 2 with working set management, plotting the address space paging characteristic curve and the system paging curve.

For workloads containing subsystem transactions such as CICS and IMS, donors and receivers can be classified as:

- *Goal receivers/donors* when its service class period is served by another (a server) service class period. This service class period is called *served service class* or *external service class (ESC)*. These service classes are for CICS and IMS transactions where there may not be a one-to-one mapping of a transaction to an address space because:

    - One transaction can be run in multiple address spaces.
    - One address space may run transactions from multiple served classes.
- *Resource receivers/donors* - also called servers - when its service class period (DISC) is a *server* of another service class period (ESC).

The *resource* service class is the receiver or donor of resources which affect the performance of the *goal* service class. Resources are allocated to the server, or *dynamic internal service class* (DISC). The server's service class delay state samples are apportioned to the served service classes based on the time the server was serving transactions from the served service class. These delay samples are used to find the bottlenecks.

In other words, the goal receiver/donors are the CICS or IMS transactions, because they have a goal. But the resource receiver/donors are the address spaces (CICS or IMS regions) in which the transactions are processed.

For more information about external service class (ESC) and dynamically internal service class, see 2.2.1.6, "Server Topology" on page 30.

2.2.1.9 Sysplex I/O Priority Management

In releases prior OS/390 Version 1.3, CPU and storage were the only resources used for goal mode management.

OS/390 1.3 introduced changes that allow I/O resources to also be managed by WLM. Initially the IOS queue is the element of the I/O path to be managed. This queue is established at the unit control block (UCB), because a device can only serve one I/O request at time. Queueing at the UCB level occurs when a task attempts an I/O operation and the device is currently being used by another task of the same MVS image.

In compatibility mode, the installation has three methods for controlling the IOS queue on a PGN period basis:

- IOQ=FIFO, first in, first out.
- IOQ=PRTY, where the I/O priority in the UCB queue is identical to the CPU dispatching priority.
- IOQ=PRTY and IOP=xx, where the I/O priority in the UCB queue is declared in the IOP keyword.

WLM goal mode prior to OS/390 1.3 always queues I/O using the address space or enclave CPU dispatching priority.

One important feature introduced with OS/390 1.3 is the ability to dynamically set priorities in the UCB queues for DASD devices, because DASD is the predominant source of device contention for workloads. Paging I/O requests do not participate.

A complete I/O operation consists of three principal components:

- IOS queue time

  The sysplex I/O priority is used to queue requests at the UCB level.
- Pending time (delay time in the channel subsystem due to I/O elements busy, as controllers, channels and shared DASD devices)

There is no priority management inside the channel subsystem.

- Connect plus disconnect time

  Inside the DASD controller there are several queues (such as for accessing cache, internal data paths and actuators). For the current generation of DASD controllers (such as 9390, for example) the algorithm controlling such queues is FIFO.

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 have control over I/O priority.

Assigning a Priority to I/O requests

Each I/O request is performed by or on behalf of an OS/390 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.

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 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.

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 2.2.1.8, “Donor and Receiver Determination” on page 37. Therefore WLM must be able to determine the following:

1. If a service class period can significantly improve its goal achievement by raising its I/O priority - a potential receiver.
2. 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 may 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 may occur if I/O priorities are altered. This device delay time is the sum of the time each request was delayed by IOS (IOS queue time), the time delayed in the I/O subsystem (pending time), and internal control unit delays (some disconnect time).

  The delay state tracked by WLM only includes IOS queue time and pending time. There is currently no way of measuring the internal control unit delays.

Important

The device connect and disconnect time, as well as the pending time (including channel busy, controller busy, shared device busy) are obtained directly through channel measurement data. The IOS queue time is a sampled value.

WLM consolidates information from the whole sysplex. The consolidation takes place every 10 seconds. The selection of a receiver for I/O resources is no different than that for other resources. See 2.2.1.8, “Donor and Receiver Determination” on page 37 for additional information.

2.2.1.10 WLM/SRM Discretionary Goal Management

Since the first release of WLM (MVS/ESA V5), the discretionary type of goal has been intensively used by customers. It was designed to simplify the migration of a low priority type of work from WLM compatibility mode to WLM goal mode. A discretionary goal means “do the best that you can”, and usually applies to batch jobs.

Furthermore, such a type of 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 which 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.

The initial implementation of the discretionary type of goal poses a problem to the installation. The phrase “the best that you can” really means: *you only get CPU if the others do not want it.* As a consequence, we may have non-discretionary service class periods overachieving their goals (that is, having a PI consistently less than one) and being able to use CPU at expense of work with a discretionary goal.

Due to this implementation, some customers move low-priority workload from service classes having discretionary goals to other low importance service classes, with say a velocity goal, to ensure service, when other workloads are meeting their goals.

WLM/SRM Discretionary Goal Management Objectives

An improvement in WLM/SRM discretionary goal management is implemented in OS/390 2.6 with the following objectives:

- To assist discretionary work to get some CPU service when other types of work are overachieving their goals (PI consistently less than one). The new design is implemented by dynamically capping the overachieving work, therefore freeing CPU cycles to be used by the discretionary type of work.
- No required service definition modification or other external changes.
- To allow customers to move low-importance work back to a discretionary goal and take advantage of the MTTW function.

As a consequence, an installation may need to adjust current workload goals because they are now enforced more closely. Before OS/390 Version 2 Release 6, many service class periods far exceeded their goals even while shutting out discretionary work. This is not true, anymore.

The logic of this function, introduced in OS/390 Version 2.6, is based on the donor/receiver concept as explained in 2.2.1.8, “Donor and Receiver Determination” on page 37. The resource involved in the donation is the CPU.

There are two ways that a service class period donor can deliver CPU cycles to a receiver:

- By giving away dispatching priority to the receiver.

- By being CPU-capped and having the CPU excess used by the receiver.

In WLM/SRM discretionary goal management, the receiver has a discretionary goal and the donor does not. The donation is implemented though the use of capping. RMF shows CAPD delays for service class periods (SCPs) that are capped because of discretionary donations. 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 non-discretionary 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 1 minute.
- It has a PI less than 0.7.
- It has not been a significant receiver candidate for CPU lately.

The donor is capped through the implementation of an internally and dynamically defined resource group (RG). A maximum of ten internal RGs are allowed. This RG is single system in scope, meaning that the information on it is not sent around the sysplex as it is currently with “standard” RGs.

Associated with the RG there is a maximum CPU service rate (MAX). The lower this value, the more severe is the capping. Refer to 2.2.1.11, “Capping” on page 42 to get more information about resource groups and capping.

After finding the service class period donor (there can be more than one), WLM derives the value of MAX for the internal RG. This calculation has the following aims:

- To avoid performance deterioration of the donor's PI. This value cannot be higher than the discretionary PI, that is, 0.81.
- To satisfy the discretionary service rate demand.

WLM, after initially setting the MAX value, dynamically follows the effect of the MAX value looking for adjustments:

- If any capped service class period’s PI is greater than 0.81 (which is undesirable, since it is worse than discretionary) and its CPU service rate is above the MAX, then make the MAX value equal to the CPU service rate.
- If any capped service class period’s PI is greater than 0.81 and its CPU service rate is below the MAX, there is nothing WLM can do, so delete the dynamic RG.
- If the dynamic RG is not capping any donors for about five minutes, then delete the dynamic RG.

2.2.1.11 Capping

Capping is a technique used to enforce resource group CPU service rate maximums, and is also used by discretionary goal management in managing the dynamically created internal resource groups.

The OS/390 dispatcher divides elapsed time into 64 time slices. Dispatchable units from address spaces and enclaves belonging to a resource group are made non-dispatchable during some time slices in order to reduce access to the CPU to enforce the resource group maximum. The time slice where an address space is non-dispatchable is called a *cap slice*. The time slice where an address space is dispatchable is called an *awake slice*.

Capping delay is one of the delay states recorded by WLM during sampling, and they are reported with the other delay states in the RMF Workload Activity report. Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information on this report.

The granularity of the control is at the resource group level and not at the service class level. This means that 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 be based on the goals. Therefore, work in the resource group with a more stringent goal will be more likely to run when the group is not capped.

The CPU service rate (in order to see if the limit is being exceeded) is per resource group. In a resource group formed by several service classes, WLM enforces that the *total* CPU service rate is not going to be above the maximum.

Refer to 2.2.2.10, “Resource Groups Maximum (Capping)” on page 50 for more information on capping.

2.2.2 Policy Adjustment Function

The policy adjustment function is in charge of meeting the transaction’s service class goals and resource group service rate requirements as stated in the service policy. This is achieved by distributing resources among the service classes. Throughout this process some trade-offs need to be made and in general, the “golden rule” is to equalize PIs by importance within resource group constraints.

The policy adjustment function is invoked every 10 seconds. The heart of this function is the policy adjustment loop.

2.2.2.1 Policy Adjustment Loop

While the *policy adjustment loop* is a complex mechanism, we provide a simple description here and explain only its major steps and functions.

The loop can be seen as the following:

- Analyzing the system
- Recognizing the service class periods that need help
- Choosing one (and only one) service class period to be helped
- Finding out which resource that service class period needs most
- Looking for a service class period that can give up this resource
- Adjusting the appropriate priority or target to reflect that new resource distribution

The policy adjustment loop is illustrated in Figure 12 on page 44:

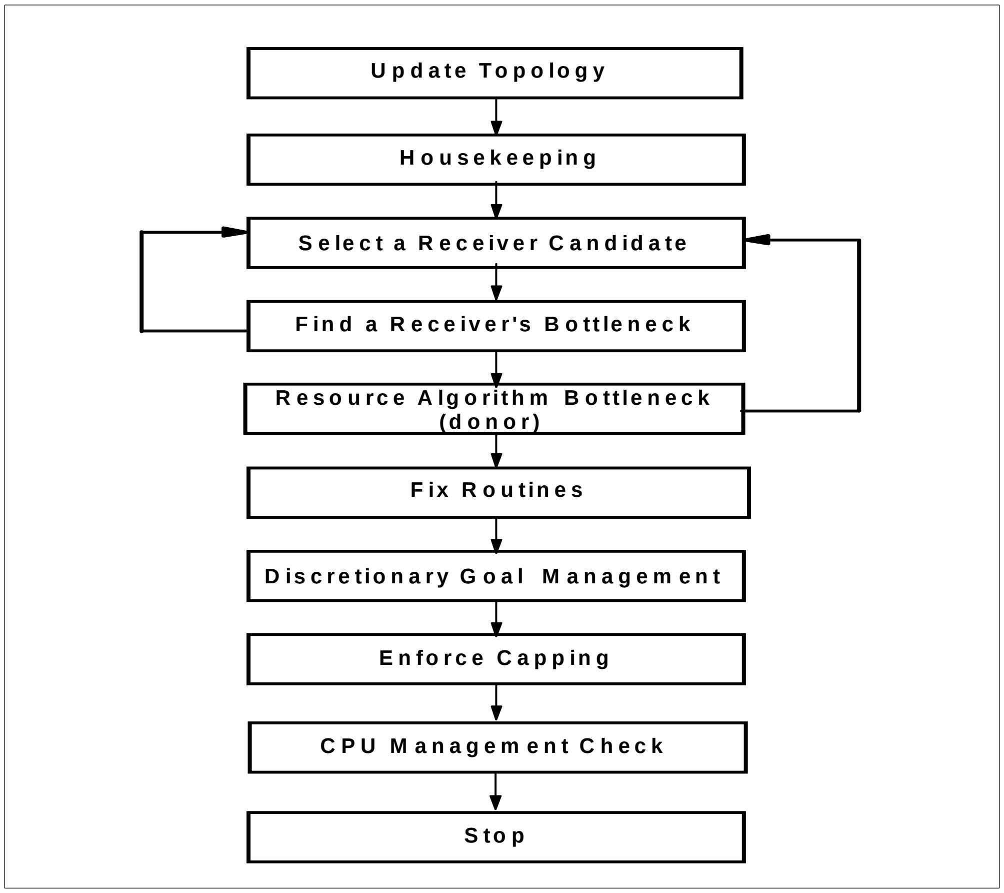

Update Topology
Housekeeping
Select a Receiver Candidate
Find a Receiver's Bottleneck
Resource Algorithm Bottleneck (donor)
Fix Routines
Discretionary Goal Management
Enforce Capping
CPU Management Check
Stop

Figure 12. SRM Policy Adjustment Loop Summary

All the actions executed in the policy adjustment loop are logged every 10 seconds in an SMF 99 record. These records can help in problem determination. However, SMF 99 recording can consume a lot of CPU cycles and DASD space and is not recommended for normal operation. For general reporting and tuning information for a system in goal mode, you can use SMF 72 records (RMF Workload Management report).

All the elements of the policy adjustment loop are more fully explained in the following sections.

2.2.2.2 Update Server Topology

Server topology is an algorithm used to determine in a multi-address space subsystem work manager, the distribution of the transactions and their service class across these multi-address spaces. Refer to 2.2.1.6, “Server Topology” on page 30, to get more information.

2.2.2.3 Housekeeping

This task includes sample counter consolidation, plot preparation and some data clean up.

2.2.2.4 Select a Receiver Candidate

The following describes the criteria used to select a candidate service class to be a receiver. These criteria are covered more generally in 2.2.1.8, “Donor and Receiver Determination” on page 37; nevertheless, here are more details.

If one of the criteria is met, then the search stops as 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, running in a resource group which is running below the resource group service rate minimum. The best receiver is chosen by largest amount below resource group service minimum, by most importance, by highest PI.
2. A service class period not meeting its goal, running in a resource group, between resource group service minimum and maximum (OK groups) 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.

   Note: This case is the only one in which a service class period with a discretionary goal (PI = 0.81) can be selected as receiver.
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 tough 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 MVS 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 (DISC) running important transactions with a PI greater than one, together with less important CICS transactions, then resources are given to the DISC to help the important ESC. Obviously, the less important ESCs will also get the benefit of these resources. This has been referred to as giving the low importance work a "free ride". For more information about ESC and DISC, see 2.2.1.6, “Server Topology” on page 30.

2.2.2.5 Find a Receiver's Bottleneck

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

- Processor delay
- Paging delay (local, common, xmemory, hiperspaces)

- Swap page-in delay
- MPL (OUTR) delay
- I/O delay (DASD device delay)
- Server address space delay

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 may cause the following effect:

> In a multi-task swappable type of workload, SRM 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 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, then the target address space is identified. This is so the cross-memory fix routine can help the target address space, 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:

  - Use the topology to understand the proportion of time (weights) each server DISC spent serving the chosen receiver ESC.
  - Calculate the delay samples for the receiver ESC.

    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 DISC address spaces are apportioned to the receiver ESC based on the previous calculated weights. In other words, the delay samples for the server DISC address spaces are scaled by the proportion of time the server DISC was serving this receiver ESC in relation to the amount of time the server period was serving any DISC.

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

2.2.2.6 Individual Resource Algorithm per Bottleneck

The individual resource algorithm selects donors for a receiver. The donation must be the same resource that is a bottleneck for the receiver. The individual resource algorithm is invoked for each bottleneck in the receiver, starting with the resource which has the most delay impact to the candidate receiver. The following is the flow for providing resources to receivers:

1. Select donors (by the reverse order of their PIs) and actions (such as donating storage, I/O or CPU).

   The following are the criteria to find a donor:

   - Service class periods meeting goals (including discretionary in resource groups with service above minimum): the best donor is chosen by lowest PI.
   - Discretionary periods below resource group service minimum: the best donor is chosen by the smallest amount below resource group service minimum.
   - Service class periods not meeting goals in resource groups running between minimum and maximum or service class periods not in a resource group not meeting their goals: the best donor is chosen by least importance, by lowest PI.
   - Service class periods not meeting goals in resource groups running below minimum: the best donor is chosen by smallest amount below resource group service minimum, by least importance, by smallest PI.

   As you see, importance is only a criteria when goals are *not* met.

2. Assess *receiver value*.

   A policy action is not taken to help a service class period unless there is sufficient receiver value. An action must have sufficient value to the receiver. If this were not done, the policy code could continually take action to help one service class period that has little effect, and ignore other service class periods that are less important, but not meeting their goals.

3. Assess impact to donors (net value). A receiver can receive resources from donors or work with discretionary goals. If, while analyzing a donor, the projected harm to the donor is more than projected improvements to the receiver, then another donor is selected. The “net value assessment” considers all external service policy specifications: resource group service minimums and maximums, importance and goals. Also, the projected PIs are going to be derived.

4. Loop through step 1 on page 47 to step 3 on page 47 for additional donors and actions as required.

5. Adjust policy for resource.

   The resources to help the receiver may also come out of what is referred as “discretionary resources”, which are those that can be reallocated with little or no effect on the system's ability to meet performance goals. An example of a discretionary resource is the amount of central storage occupied by logically swapped-out address spaces that have been swapped-out longer than their protect time.

If the net value for a receiver is not sufficient, this receiver cannot be helped and SRM goes back to the *Select an Receiver* step in order to choose another receiver.

2.2.2.7 Fix Routines

The fix routines are in charge of fixing the bottleneck by giving a specific resource from the donor(s) to the receiver. There is one such routine for every tracked delay resource:

- Auxiliary storage
- Multiprogramming level (MPL)
- Swap-out
- Processor
- I/O delay
- Queue delay

AUX_Storage_Delays_Fix Routine

This routine is invoked if a receiver service class period is experiencing its largest delay because the address spaces within the period are waiting for page-ins from auxiliary storage, such as:

- Private area paging
- Common area paging
- Cross-memory paging
- VIO paging
- Standard hiperspace paging

This routine addresses these paging delays through the use of the protective processor storage target (refer to 2.2.1.4, “Storage Targets” on page 26 for more details). This target protects an address space from losing processor storage due to UIC steal, physical swap-outs (working set and trimmed pages) and expanded storage migration; consequently, it is less delayed by page-ins from auxiliary storage.

The use of the protective storage depends on the particular case, for example:

- Private area paging for short response time goal period (trivial TSO)
- Private area paging for long response time goal period
- Common and cross-memory paging delays

For a detailed description of all the auxiliary storage delay fix routines, refer to *System/390 MVS/ESA Version 5 Workload Manager Performance Studies*, SG24-4352.

MPL_Delay_Fix Routine

This routine is invoked if a receiver service class period is experiencing its largest delay because the multiprogramming level (MPL) is less than the number of ready address spaces. In other words, the address spaces are swapped out due to a SRM decision and they are placed in the out-ready-queue (OUTR) queue.

The MPL of a service class period is the number of address spaces in that period that are swapped-in in central storage at any given time. An address space in a period experiences MPL delays when that address space is swapped-out and is ready to be swapped-in, but SRM has not (or cannot) increase the MPL in the period to allow the address space to be swapped-in. In order for MPL_delay_fix routine to alleviate a period’s MPL delay, it must allow the period's “swapped-out and ready” users into central storage. It is done by increasing the period's MPL in_target by one and maybe the period's MPL out_target by one as well. Refer to 2.2.1.2, “Swap Control” on page 24 to have a quick look at these two targets.

On the other hand, WLM must *find* and *ensure* that there is enough central storage available to contain another address space typical of that period. This is done through the invocation of the find_storage routine.

For a detailed description of the MPL delay fix routine, see *System/390 MVS/ESA Version 5 Workload Manager Performance Studies*, SG24-4352.

_Swap_Delay_Fix routine_

This routine is invoked if a receiver service class period is experiencing its largest delay because the address spaces within the period are waiting for page-ins from auxiliary storage. The solution is to increase the *swap protect time* of the service class period. Refer to 2.2.1.3, “Swap Protect Time” on page 25 to get more information about this time. For a detailed description of the swap delay fix routine, see *System/390 MVS/ESA Version 5 Workload Manager Performance Studies*, SG24-4352.

Processor Delay Fix

This routine is invoked if a receiver service class period is experiencing its largest delay because dispatchable units in this period are waiting for CPU resources.

Processor delays are addressed by increasing the dispatching priority of the receiver, or decreasing the dispatching priority of one or more donors, or both. Projections are made for the time the affected service class periods will consume and the new wait-to-using ratios the affected service class periods will experience after the dispatching priority changes. The time is projected based on the total processor time available, the maximum demand of the service class period, and the maximum demand of work at higher and equal dispatch priorities. The wait-to-using ratios are projected using the actual observed data collected for the wait-to-using ratio at each priority, adjusted for the maximum demand moved from one priority to another.

The projected times and wait-to-using ratios are used to calculate the change in processor_using and delay samples that work in the service class period will experience in the next interval. For non-served classes, these samples are used to calculate response time or velocity deltas and performance index deltas for the receiver and donors.

For served classes, the samples are calculated for all servers of receivers or donors and are then apportioned to the served classes based on the relative number of observations of each server serving each class. A proportional aggregate speed is then calculated for each served receiver or donor and the performance index delta is read off the proportional aggregate speed plot.

In both the served and non-served cases, the performance index deltas are then evaluated to determine if the trade-off is a good one. If there is net value to the dispatching priority trade-off, the change is made.

I/O Delay Fix Routine

This routine is invoked if a receiver service class period is experiencing its largest delay because of delayed I/O requests. The solution is to increase the I/O priority of the address space or enclave. This mapping is dynamically adjusted by SRM. For more information about I/O priority management, see 2.2.1.9, “Sysplex I/O Priority Management” on page 38.

2.2.2.8 Stop If One Receiver was Already Selected

If one receiver was already selected, then stop. Go ahead in the loop to process the discretionary goal management function and capping. If no receiver was helped, return to the *Select a Receiver* step to look for another receiver.

2.2.2.9 Discretionary Goal Management

This function is introduced in OS/390 Version 2.6. Its logic is based on the donor/ receiver concept as explained in 2.2.1.8, “Donor and Receiver Determination” on page 37. The resource involved in the donation is the CPU. For more information about this function, see 2.2.1.10, “WLM/SRM Discretionary Goal Management” on page 40.

2.2.2.10 Resource Groups Maximum (Capping)

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. A resource group is a WLM construct used by the installation to limit a maximum (capping) or to deliver a minimum of CPU capacity to the address spaces and enclaves belonging to the service class periods that constitute the resource group. These minimum and maximum capacities are measured in the unweighted CPU service rate consumed in the resource group across the sysplex. A resource group can be formed by distinct service classes with different importance values.

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.

Cap Time Slices

To implement capping, the elapsed time is divided into 64 time slices. Each time slice then represents 1/64th of the total elapse time, and each time slice represents 0.5 of an SRM second (with the TUNIT constant set to 2, which is the default).

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

Because two groups may accumulate service units at a different rate, 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* may 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 may 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 may 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 nondispatchable are reevaluated. The 64 time slices and the cap slices are then reassigned.

Attention

Keep in mind your service goals when you assign a service class to a resource group. Given the combination of the goals, the importance level, and the resource group capacity, some goals may not be achievable when capacity is restricted.

The RMF Workload Activity Report shows you the service class delay because of resource capping.

Data Used in the Capping Algorithm

Input data is needed in order to enforce capping in a resource group. The inputs to the capping algorithm are:

- The group maximum CPU service rate extracted from service policy
- The local CPU service rate at a particular importance value

2.2.2.11 CPU Management Check

When running in WLM goal mode, it is possible for high importance work to sometimes run at a lower dispatching priority than lower importance work as a result of WLM's goal-oriented CPU management. However, when the workload mix changes or CPU utilization increases, the higher importance work can start to miss its goals on the local system due to CPU delay. It may take WLM multiple intervals to address this problem if there are other problems to fix at a higher importance level. WLM makes only one policy adjustment per 10-second interval. This delay may be unacceptable for critical, response-oriented applications.

The WLM CPU algorithm has been enhanced by APAR OW37742 to detect situations where higher importance work is missing its goals on the local system due to CPU delay, and there is lower importance, heavy CPU work at a higher priority. The lower importance work will now have its priority lowered below the higher importance work in the first WLM interval where the situation is detected. These adjustments will be made in the same interval as the traditional policy adjustments.

Note: This means that WLM can now make multiple adjustments within one interval, rather than having to wait several intervals to make the necessary adjustments.

2.2.3 Resource Adjustment Function

The resource adjustment routines take care of the throughput, keeping the resources of the system effectively utilized.

The functions are:

- Detecting and addressing underutilized, overutilized and shortage conditions
- Managing individual address space working sets

Important

The resource adjustment routines respect the constraints set by the policy adjustment algorithms, avoiding taking actions that would cause work to miss goals. In other words, goals have precedence over throughput.

The resource adjustment actions are usually not applied to a service class period, but to address spaces.

Working set management introduced in MVS/ESA V4.2 is still a major component in goal mode. The major differences between compatibility and goal mode are:

- Goal mode manages central storage and expanded storage; compatibility mode only manages central storage.
- In goal mode, address spaces may also be selected for monitoring by the policy adjustment routine.

2.2.3.1 Concepts in Resource Adjustment Routine

The following are some of the terms and concepts used internally by the resource adjustment routines. Some of them are common to the policy adjustment routines and are repeated here.

Productive CPU Rate	The ratio of the time an address space is doing useful work to the time it is in storage. It is defined as TCB + SRB time (measured in milliseconds) divided by residency time (measured in seconds).
AUX Paging Cost Rate	A measure of the CPU cost of paging by the address space. It is defined as the

	calculated (not measured) CPU cost of paging to/from AUX measured in milliseconds of CPU time over residency time.
Total Paging Cost Rate	A measure of the CPU cost of paging by the address space. It is defined as the calculated (not measured) CPU cost of paging to/from expand/AUX measured in milliseconds of CPU time over residency time.
Net Productive CPU Rate	A measure of how much an address space adds to running the CPUs in a system productively. It is defined as the productive CPU rate for an address space minus the paging cost for an address space.
Unmanaged CPU Paging Cost Rate	The CPU cost of paging done by address spaces that do not have a restrictive storage target.
Process Storage OK1 Point	The amount of processor storage an address space needs to have so that it is spending less than 5% of its time paging from AUX.
Central Storage OK1 Point	The amount of central storage an address space needs to have so that it is spending less than 5% of its time paging from expanded or AUX.
Monitored Address Space	An address space that is having data collected to help manage its working set. Monitored address spaces have points plotted on their address space paging plot and address space central paging plot.
Managed Address Space	Address space with either a restrictive central and/or expanded storage target set by the working set manager.
Discretionary Storage	Frames that can be reallocated without significantly affecting the work in the system. Included are available frames, frames containing pages with a high Unreferenced Interval Count, and any amount of frames containing pages above a restrictive target. There is a discretionary count for both central and expanded store frames.

2.2.3.2 Address Space Working Set Management

The management of the address space working set size provides system-managed use of processor storage. This is done by dynamically adjusting the storage targets of selected address spaces. For more information about storage targets, see 2.2.1.4, “Storage Targets” on page 26.

Working set management divides all address spaces into the following classes:

- Unmanaged address spaces
- Monitored address spaces
- Managed address spaces

The storage targets of unmanaged address spaces are not managed. Address spaces in service class periods that have short response time goals or that are in the SYSTEM or SYSSTC service classes are never managed individually.

At each resource management interval, the swapped-in address spaces are searched looking for non-monitored address spaces that should be monitored. The following are basically the set of conditions that an address space must meet before it is considered for monitoring:

- It has collected enough CPU time so we have some reliable data about it.
- One of the following must be true:
  - The address space is paging significantly.
  - The address space working set is at least as large as the average monitored address space.
  - The policy adjustment code wants this address space monitored.

WSM only manages the working set of an address space if it is monitored. A monitored address space has data collected about its storage usage in the form of two plots:

- The address space “paging plot” tracks how an address space performs in relationship to the amount of processor (central + expanded) storage the address space has.
- The address space “central storage paging plot” tracks how an address space performs in relation to the amount of central storage the address space has.

2.2.3.3 Resource Adjustment Actions

Sampling of storage related conditions is done every 250 milliseconds. After 2 seconds, resource management collects data about what happened over the last interval and then plots points on address space plots. Next, a sequential list of system conditions is verified. If one of the following conditions is present, actions may be taken:

- The system is overutilized.
- The system-unmanaged CPU paging is high.
- The system AUX paging is high.
- An OK1 action is needed.
- The system is underutilized.
- There is a phase change.
- It is time to invoke working set management.

2.2.3.4 System Over-Utilized

The system is considered overutilized if it meets one of the following criteria:

- On average, a significant number of address spaces were waiting for the CPU.

- The percentage of central storage fixed is above the RCT fixed high thresholds RCCFXTTH and RCCXETH (as in compatibility mode).

If the system meets one of these criteria, the MPL out target of a service class period is decreased so that an address space is swapped-out and demand for system resources is reduced. First, attempts are made to find a discretionary period to lower the MPL out target. If no discretionary period is eligible, a period with goals is chosen in standard donor order. The chosen period's MPL out target is lowered enough to cause one address space in the period to be swapped-out. If the system is still overutilized, no other action is taken during the interval, whether or not a period was found to reduce MPL. For more information about MPL targets, see 2.2.1.2, "Swap Control" on page 24.

2.2.3.5 System-Unmanaged CPU Paging Cost High

System-unmanaged CPU paging cost is the cost of CPU due to paging done by all address spaces that do not have a restrictive storage target and that are address spaces which are not managed by the resource adjustment routine.

It is considered high when over 10%, that is, when more than 10% of the CPU time is spent performing paging. One reason the system might be paging heavily is that one or more large jobs (managed or not) do not fit in available storage. In such a situation, it is probably better to let these jobs page and keep the rest of the system with little paging.

Therefore the resource adjustment function attempts to convert some of the unmanaged paging to managed paging:

- First it tries to reduce unmanaged paging by reducing the restrictive central target of an already managed address space by a value (squeeze). The idea here is to decrease the working set of a managed address space in order to decrease the unmanaged paging. If this reduction takes the address space plot curve to a high paging rate, then the address space is swapped-out since it is not likely to get any useful work done with such a low target. In general the same address space is *squeezed* until the unmanaged paging problem is solved or the address space is swapped-out.
- If no address space can be squeezed, the next step is to try to manage a new address space. The following criteria are used in choosing an additional address space to manage.
  - An address space in a period meeting goals is chosen before one missing goals.
  - The least important address space is chosen.
  - The address space furthest from OK1 is chosen.
  - The largest address space is chosen.

The address space chosen to be managed is given a restrictive central target below its current size. If no address space is chosen to be managed, a period is chosen to have its MPL reduced, discretionary first and then in donor order.

2.2.3.6 System Auxiliary Paging High

This situation happens if the system AUX paging rate is higher than the DASD paging subsystem can handle. The sign that the paging subsystem is overloaded is that paging-in from AUX starts taking longer to process due to the queue time building up. SRM uses the “system paging responsiveness plot” to recognize this situation; see Figure 13 on page 56 for an example of this plot. There is one of these plots per system. Points are plotted every minute.

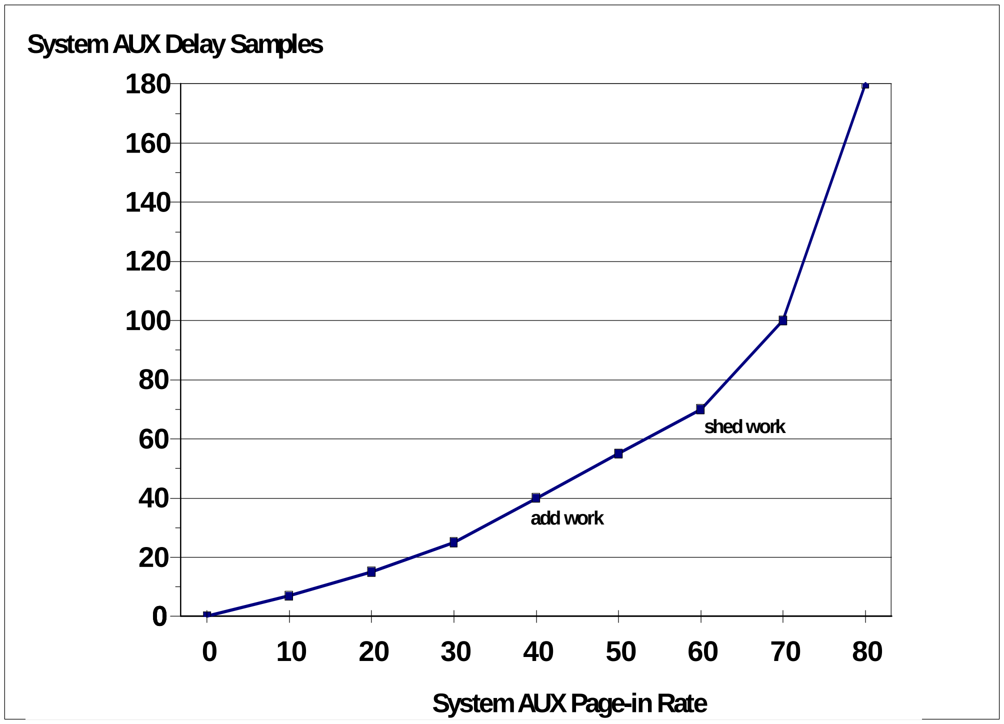


Figure 13. Paging Responsiveness Plot

The plot has the following attributes:

- X value: the system AUX non-blocked page-in rate. This rate does not include the pages that are automatically brought in a block together with the faulting page.
- Y value: the total AUX delay samples (private, xmemory, common).

If the DASD paging subsystem is doing well, then the plot will show that AUX delay samples are growing proportionately with the systems’s AUX page-in rate. If the paging subsystem becomes overloaded, the plot will show AUX delay samples growing significantly faster than the systems's AUX page-in rate.

The first point at which the rate of growth of AUX delay samples becomes more than twice the AUX page-in growth rate is called the “shed work point”. The last point at which the rate growth in AUX delay samples is close to proportional with the AUX page-in growth rate is called the “add work point”. When the current point on the plot is at or above the shed work point, a period is chosen to reduce MPL in the same fashion as the overutilized case.

2.2.3.7 OK1 Action

The next condition verified is the need for executing an OK1 action. An OK1 action steadily increases the protective target of one address space until the OK1 point is reached. The advantages of an OK1 action are:

- Lowering the address space page-in rate.

- Finding the central storage OK1 point of a monitored address space. Finding an address space's OK1 point fills out the address space's central paging plot.

An address space is only given one chance to go through an OK1 action because these actions have the potential to be disruptive to the system.

When an OK1 action is started for an address space, it is given a protective central target above its current size. During each interval after that in which the OK1 action is continued, the protective target increases by a value if both the following conditions are true:

- The address space absorbed the last target change; that is, the current working set is greater or equal to the protective target.
- There is some available discretionary storage.

If an address space is already going through an OK1 action, the action continues for that address space. Otherwise the following criteria are used to pick a new address space to go through an OK1 action:

- There must be enough discretionary central storage for the address space to increase the protective target.
- Address spaces that already have a protective target are chosen over those that do not.
- Address spaces that do not have a restrictive target are chosen over those that do.

If no address space meets any of these criteria, no OK1 action is done.

The OK1 action is ended when *one* of the following is true:

- An OK1 point is plotted on the address spaces’ central paging plot.
- The address space is chosen to be *squeezed* because of too much system unmanaged paging.
- UIC steal could not free enough frames.

2.2.3.8 System is Underutilized

The next condition verified is if the system is being underutilized. In the situation where it is underutilized, more work should be brought into the system by raising a period’s MPL. The system is considered underutilized if *all* of the following conditions are true:

- On average, during the last interval, no more than a few address spaces were waiting for the CPU.
- The system unmanaged paging cost was under 10%.
- The amount of central storage fixed is below the RCT low thresholds (RCVFXIP, RCVMFXA).
- The system is not, and has not recently been, in a storage shortage.
- The current point on the system paging responsiveness plot is below the add work point; see Figure 13 on page 56.

If *all* these conditions are met, select_receiver is called to find, in receiver order, a period with at least one out-and-ready user. If such a period is found, its MPL out target is raised by one.

2.2.3.9 Phase Change

The next action is to attempt a “phase change”. The idea behind phase change is for an address space that has had a restrictive target for a long time, to increase its central and/or processor storage restrictive target by a delta value. This is done to detect if the address space's paging characteristics have changed such that it would run faster with just a few more frames. Without increasing the restrictive target, we might never find this out since the restrictive target would stop it from getting these frames. If an address space has never gone through a phase change, it is eligible 3 minutes after getting a restrictive target. It is not eligible again for 6 minutes, then 9 minutes and so on.

Jobs are selected for a phase change based on the following criteria:

- They have had a restrictive target long enough.
- There is enough discretionary storage to raise the restrictive target in question (central or processor storage).

2.2.3.10 Working Set Management Algorithm

Finally, if no other actions have been taken and there are monitored address spaces, WLM tries to reduce the amount of CPU time the system spends on paging, by efficiently allocating storage among monitored address spaces. The primary difference between working set management in goal mode and compatibility mode is that in goal mode, it deals with efficiently allocating expanded storage in addition to AUX storage.

Consider the following example showing how WLM can greatly increase the amount of productive work done by a system. On a system with 120 MB of central storage, 3 jobs are run that getmain a 50 MB area and reference each page in this area sequentially in a tight loop.

Without working set management, all three jobs might be run simultaneously, with none of the jobs able to keep their storage. Each job would spend most of its time paging and much of the system’s CPU time will be spent paging.

Working set management, on the other hand, will understand that two of these jobs will fit without paging and keep one job swapped-out. In this case, most of the system’s CPU time will go towards running the application's code of the jobs getting productive work done. Exchange swapping will ensure that each job gets a chance to run.

If working set management wants an address space to have more central storage, it raises the address space’s protective central storage target. Similarly, working set management raises an address space’s protective processor storage target to give an address space more processor storage. When working set management wants to take central or processor storage from an address space, it gives the address space the appropriate restrictive target. If working set management decides an address space can do very little productive work with the storage it has been allocated, it swaps the address space out.

For a batch job, working set management can keep the address space swapped-out up to ten minutes past the time it would normally be swapped-in. In the case of a TSO user, the address space can only be swapped-out an extra 30 seconds.

2.3 Understanding the New Dispatchable Units in OS/390

Before MVS/ESA 5.2.0, the only recipient of resource consumption and priorities was at the address space level. The address space can be swapped-in or swapped-out, it can be given more or less processor storage, and its priority can be adjusted, to control the CPU resources given to the associated dispatchable units, that is, SRBs and TCBs. This approach is effective for traditional workloads such as TSO and batch.

However, to face the challenges of a client/server environment, a new set of dispatchable units needed to be designed.

2.3.1 Concept of Dispatchable Unit

In data processing literature you may find the following definition of a *process:*

“A serial execution of a computer logic directly connect to a business need.”

Examples of processes are a payroll application, an end-user query, or a report preparation. A process should not be confused with a program; to clarify that, consider the two following sentences:

- I am going to the movies, driving my car serially through the streets.
- To perform the payroll application, the CPU executes serially a set of programs.

In these examples, going to the movies and the payroll application are both processes, while the car and the CPU are the executors and the streets and the programs are the same class of objects.

Academically, a process is a unit of:

- Priority, for the use of resources as CPU, I/O, and storage inherited from the user transaction goal.
- Account, for keeping information about resource consumption.
- Addressing, for referencing private and common virtual storage.
- Security, for guaranteeing the privacy of the information being processed.

In OS/390, processes are called *dispatchable units*. Originally two types of dispatchable units were defined:

- *Task,* which is preemptible, long-lived, problem or supervisor code. Its creation demands a long path length of instructions (ATTACH macro) and it is represented by a TCB control block getmained below the 16 MB virtual line.
- *Service request,* which is non-preemptible, short-lived, local or global priority supervisor code (usually). Its creation has a short path length of instructions (SCHEDULE macro), and it is represented by a SRB control block usually getmained above the 16 MB virtual line.

  Non-preemptible dispatchable units such as local and global SRBs, once dispatched, continue to run until they incur a voluntary suspension (such as lock held or page fault) or they complete. This happens even if higher priority work is ready to use the CPU. External and I/O interrupts are serviced (usually SRBs are interrupt-enabled), but the SRB is re-dispatched after each interrupt is serviced.

It is important to note that because TCBs and SRBs in OS/390 always run in address spaces, then the properties of priority, accountability, addressing and security are allocated in a common place, that is, the address space construct.

As a final point, we introduce the concept of a *transaction*. It is a computer interaction with a user (batch or online). Its execution may require more than one dispatchable unit, usually executed in sequential fashion (parallel queries are an exception).

2.3.2 The Need for New Dispatchable Units

The performance management difficulties that modern workloads posed to OS/390 were that OS/390 MVS did not provide an anchor for a transaction with the exception of address spaces. All client requests came through one address space and were managed under the priority for that address space.

In a modern multiaddress space application scenario such as client/server, we may have the following:

- 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 non-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
- A SRB scheduled from one address space and running in other address spaces

As you can see, it is very difficult to assign priorities and keep track of the units of account. The business unit of work is typically the user transaction, while the MVS unit of work is a TCB or SRB in an address space. This results in:

- 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 end-user transaction. So, the server can execute tasks on behalf of clients which have different goals, but all tasks execute with the goal of the server address space.

On top of these difficulties, other aspects should be taken in consideration:

- Creating many tasks for parallel-intensive applications causes the overhead of the ATTACH macro and the lack of space in LSQA below the 16 MB virtual storage line.
- SRBs are non-preemptible and consequently run without suspension till they end, causing problems to the priority concept.

Then, when attempting to solve the problems of distributing work requests to server address spaces and managing them under the caller performance goal, or using multiple CPUs for one request, OS/390-related limitations dominate.

In order to address such problems, new dispatchable units need to be 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 (in compatibility or goal mode)
- Low overhead (less costly than ATTACH)
- Retaining the possibility of defining service class or performance group periods
- Retaining the concept of Resource Group, if in goal mode

2.3.3 New Dispatchable Units

To respond to these requirements, a new kind of dispatchable unit was introduced in MVS/ESA 5.2.0: the preemptible-class SRB. Together with the new SRB, a new OS/390 construct called an *enclave* was introduced.

2.3.3.1 Preemptible-Class SRBs

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 may be suspended by the dispatcher at any time to run other work at the same or higher priority.

Any dispatchable unit that is expected to consume a large amount of CPU we would naturally want to be preemptible in order to let the dispatcher run the work that the installation has directed it should be running.

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

Client SRBs

A *client SRB* is a preemptible dispatchable unit that runs in one address space (server) but executes work on behalf of some other address space (client). In other words, the unit of priority and account is 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 CPU service in the client address space’s current SRM transaction. When the client address space switches to a new performance period, so do any client SRBs running on behalf of the client address space.

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 may trigger a period switch at swap-in, but will not do so *before* swap-in.

Client SRBs are created by using an option on the macro IEAMSCHD.

IBM introduced client SRBs in order to allow DB2 to increase the CPU parallelism of queries. (Originally, the significant work of a query was done in the caller's task.) This design had an advantage for performance management because each parallelized query can be managed according to the DB2 user's dispatching priority rather than according to DB2's dispatching priority. Another point of concern was that there was no easy way to introduce CPU parallelism with existing TCB or non-preemptible SRB constructs.

Together, these attributes allow DB2 to create dispatchable units that look, act, and are managed as if they were part of the address space on whose behalf they are executing (for example a batch job or TSO user), rather than part of the DB2 address space.

DB2 Version 4 CPU parallelism exploits client SRBs. It permits multiple client SRBs to run parts of a complex query at the same time in a single system, in order to reduce the elapsed time of the query.

Enclave SRBs
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.

Refer to 5.1.1, “DB2 Sysplex Query Parallelism” on page 171 for more information on DB2 parallel queries.

Enclave SRBs are also created by the macro IEAMSCHD. This macro creates the SRB and joins it into the enclave. Refer to 2.3.4, “Enclaves” on page 62 for more information on enclaves.

2.3.4 Enclaves

The concept of an *enclave* was introduced in MVS/ESA 5.2. In OS/390 1.3, enclaves were enhanced and also referred to as *business units of work*. In this section we cover the enhanced version of enclaves.

An enclave is an OS/390 construct serving as a unit of *priority* and *account* for an arriving transaction. An enclave is not a dispatchable unit but may encompass several dispatchable units such as tasks and preemptible SRBs. These SRBs are called *enclave SRBs*. Do not confuse enclaves with enclave SRBs. With some approximation, we may say that in a sense an enclave looks like an address space. Their similarities are:

- Both enclaves and address spaces can be generically named as a business unit of work.
- Both are units of priority (inherited from the transaction) and unit of account (CPU and I/O resources) for their dispatchable units.
- For both, you can define performance periods (DUR keyword)) for changing goals or priorities as their dispatchable units consume resources.

- For both, you can define WLM goal mode resource groups for capping and protection.
- Response time, velocity or discretionary goals are allowed.
- Both may contain several dispatchable units, such as tasks. You first create the construct and later you associate the task to the constructs:
  - Via ATTACH macro in the address space.
  - Via ATTACH plus IWMEJOIN macros in the enclave.
- In both, WLM measures (by sampling) CPU delay and I/O delay.
- Both have a single-system scope, that is, their dispatchable units can run in one OS/390 image.

The dissimilarities are:

- Only address spaces may contain non-preemptible SRBs and client SRBs.
- Only enclaves may contain enclave SRBs.
- Only address spaces are units of addressing and security for their dispatchable units. As a result, an enclave dispatchable unit must also run in an address space.
- Only address spaces can be the unit of accounting for I/O, SRB and storage resources (MSO service units) for its dispatchable units. For enclaves, the MSO, IOC and SRB service units' figure is always zero.
- Only address spaces can be swapped-out.
- Only address spaces can own storage.

2.3.4.1 Enclave Properties

The following are some properties and general comments about enclaves:

- There are two types of enclave
  - Independent enclaves

    An independent enclave is used to represent a new transaction that has no association with a particular address space. The home address space when IWMECREA is issued is the owner of an independent enclave. CPU service consumed by the enclave is accumulated in the SMF 30 record of the owning address space and the SMF 72 record of the enclave's service class or performance group period.
  - Dependent enclaves

    A dependent enclave is used to continue an existing transaction that will be running under dispatchable units not associated with the current home address space. The home address space at the time IWMECREA is invoked becomes the owner of the enclave. A dependent enclave derives its performance goal from the owning address space, and all CPU service consumed by the enclave is accumulated in the SMF 30 record of the owning address space and the SMF 72 record of the owning address space's service class or performance group period.

    A dependent enclave is used when there is an existing address space defined with its own performance goal that needs to be extended to programs running under dispatchable units in other address spaces.

- The accounting for resources consumed by an enclave depends on whether it is a dependent or an independent enclave.

  A dependent enclave is a logical continuation of the transaction already active in client's address space. Therefore, CPU and MSO service for a dependent enclave is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the address space's transaction. MSO service for the enclave is calculated based on the frame count of the owning address space, not on frame usage in the address space.

  For an independent enclave, CPU service is included in the SMF 30 record of the owning address space, and in the SMF 72 record for the enclave's service class or performance group period. MSO service is not calculated for an independent enclave.

  For both dependent and independent enclaves, IOC service is included in the SMF 30 and 72 records associated with the address space where the enclave work is executing. SRB service for enclaves is *always* zero.

- Enclaves allow the management of individual transactions flowing through address spaces, something that simply has never been possible before.
- Each enclave is associated with a single transaction, which starts when the enclave is created and ends when the enclave is deleted. For example, DB2 distributed data facility (DDF) creates an enclave at first SQL statement and deletes the enclave at SQL commit.
- You can have enclaves in WLM goal mode or WLM compatibility mode.
- The dispatchable units of an enclave (an enclave SRB or a task) are scheduled into a target address space, but execute work on behalf of an enclave. Dispatching controls (I/O and CPU) are derived from the enclave:
  - In goal mode, the dispatching controls are managed using the service class period goal (response, velocity, discretionary) of the enclave as assigned by the classification rules.
  - In compatibility mode, the dispatching controls are managed using the performance group period of the enclave as assigned by the ICS classification rules. The following is an example of ICS/IPS parameters for DDF subsystem transactions:

    ```
    ICS:
    SUBSYS=DDF,
       SRVCLASS=SC1, PGN=10
    IPS:
    PGN=10, (DP=F51, DUR=30K)
    ```

    The service class is obtained from the work qualifiers in the active service definition classification rules. Then, to classify an enclave, the WLM classification rules are always used. For more information see 5.1.3, "WLM Definition for Sysplex Query Parallelism" on page 174 and *OS/390 MVS Initialization and Tuning Reference,* SC28-1752.
- If you do not have an active WLM service policy (you can have one also in compatibility mode), an enclave gets the performance objectives of the creating address space.
- A subsystem work manager creates an enclave though the WLM services macro IWMECREA. The enclave is deleted by the macro IWMEDELE.

- Dispatchable units can join an enclave through the following macros:
  - WLM services macro IWMEJOIN for tasks.Tasks can dynamically leave (macro IWMELEAV) and join an enclave as they finish one piece of work and begin another. When a task leaves an enclave, resource consumption automatically reverts to the address space where TCB executes.
  - Supervisor services macro IEAMSCHD for enclaves SRBs. This macro also creates the enclave SRB.
- CPU time consumed by each enclave SRB or task is accumulated back to the enclave and is reported as enclave-related CPU service in SMF type 30 records for the address space which created the enclave.
- TCBs and enclave SRBs running in one address space may belong to different enclaves; see Figure 14 on page 66.
- Enclaves have performance period capability. This means that the system can react to long-running or resource-consuming units and vary their goal on the fly.
- Enclaves have WLM goal mode resource group capability. This means that capping and protection are available to the enclave transactions.
- Tasks in enclaves automatically associate the enclave with the address space where they are dispatched, so WLM can manage the storage of those address spaces to meet the goal of the enclave.
- WLM services are provided to dynamically connect/disconnect a TCB or an SRB to or from an enclave. This allows the existence of a server address space with a permanently created TCB, with the TCB dynamically switching from one enclave to another, without the need of attaching and detaching the task.
- Association of a TCB with an enclave triggers association of all daughter tasks.
- WLM queueing manager services require the use of enclaves. Refer to 2.5.4, “The Queuing Manager Services” on page 73 to get more information about queueing manager services.
- With enclaves there are no performance blocks (PB) as there are with CICS and IMS. Therefore it is not possible for RMF to report the internal states of a transaction inside a subsystem work manager, as is done for CICS/IMS/DB2 using the exception delay monitoring services.

Important

The major conceptual difference between dependent enclaves (or client SRBs) and independent enclaves is that dependent enclaves or client SRBs are used for a request from a client address space (inside the same OS/390 system) to a server address space. Therefore, the priority of the enclave or client SRB is the one of the client address space.

Indepentent enclaves are used for requests from clients *outside* OS/390 images. Therefore a specific priority must be assigned from the classification rules to such an enclave because there is no client address space.

In Figure 14, enclave 1 includes SRB A1 of address space A, and TCB B2 and B3 of address space B. Enclave 2 includes SRB A2 from address space A, SRB B1 and TCB B1 from address space B, and SRB C2 from address space C.

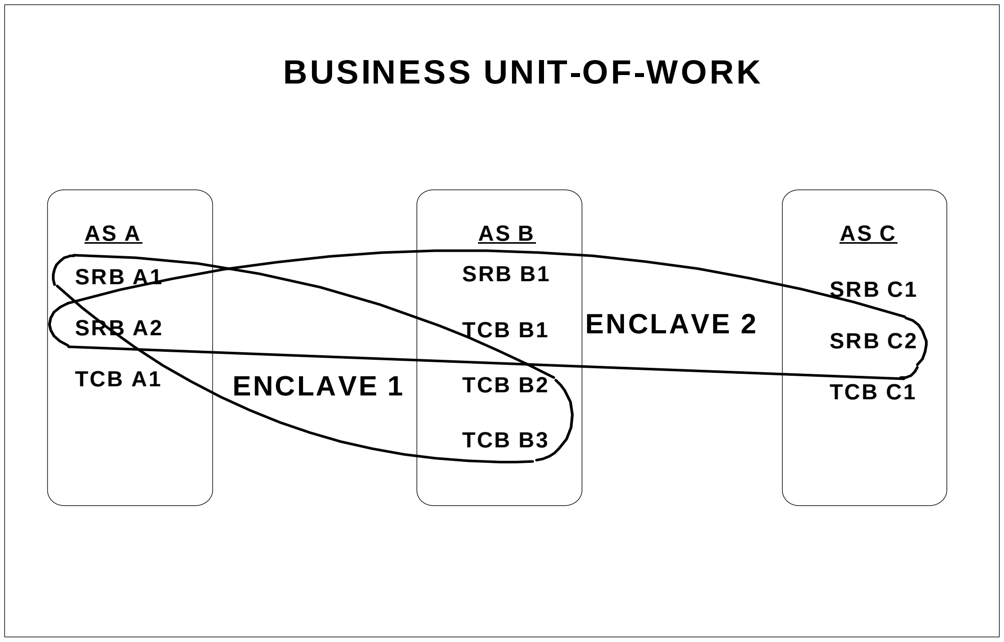


Figure 14. Enclaves Example

The first use of independent enclaves is DB2 Distributed Data Facility (DDF).

When running on MVS 5.2.0 or above and DB2 V4 or above, DDF always creates an enclave per DDF request to anchor a corresponding transaction.

This was a big issue for DDF. Prior to DB2 Version 4 and MVS/ESA 5.2.0, DDF provides no ability to prioritize among requests according to their business value, since the requests are not reported as transactions to MVS. All requests are managed like the DDF address space. That meant that the controls for this address space must be set so that the processing requirements of the requests most valuable to the business are satisfied. Any other requests that flow through the DDF address space also get this favorable treatment, even if they are not as important to the business.

Summarizing the benefits of the enclave construct, this support improves the ability of the system to manage client/server workloads by enabling WLM to manage the individual transactions within a server address space. WLM differentiates among dispatchable units associated with different business units of work, providing an additional level of manageability and control solely through the existing WLM view of goals for work. This allows control of resource consumption for transactions that span multiple address spaces, and address spaces that serve multiple transactions simultaneously.

The exploiters of the OS/390 enclave construct include:

- DDF for distributed DB2 requests
- DB2 sysplex query parallelism

- IWEB for Web server requests
- DB2 SQL stored procedures
- Component Broker
- MQ/Series Workflow

2.3.5 Accounting Service for Enclaves and Client SRBs

Originally, MVS had one pair of counters for CPU time reported in the RMF Workload Activity report:

- A counter named CPU for task time.
- A counter named SRB for service request time.

Both counters are on an address space basis.

With the introduction of preemptible SRBs, the account of CPU time has been changed to include the work done by preemptible-class SRBs. The first counter is renamed to preemptible CPU time. The second does not change:

- A counter named preemptible CPU time, including CPU time consumed by tasks and preemptible service requests (SRBs) running in an address space, or in an enclave owned by the address space.
- A counter named SRB, including CPU time consumed by non-preemptible service request (SRB) running in an address space.

For each CPU time counter there is an equivalent CPU service unit counter. The definition of I/O service is unchanged.

Refer to Figure 15 on page 68 to see a pictorial view of these new counters.

The MSO service is unchanged as well, but because independent enclaves do not occupy central storage pages, the product of CPU time consumed and central pages is always zero. So the MSO service of an enclave is always zero, which implies that the enclave CPU time is not included in any MSO calculation. Dependent enclaves also do not occupy central storage pages, but in this case the enclave is associated with an address space, so the MSO service calculation for the address space includes the CPU consumed by the enclave.

RMF in the workload activity report shows the CPU time consumed by service classes using the new counters. Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information.

For a detailed look at how time is accounted for when enclaves are being used, refer to *OS/390 MVS Programming: Workload Management Services*, GC28-1773. In the section about enclave accounting, there is a table titled *Enclave*

Characteristics and Resource Accounting describing the accounting characteristics for both goal and compatibility mode.

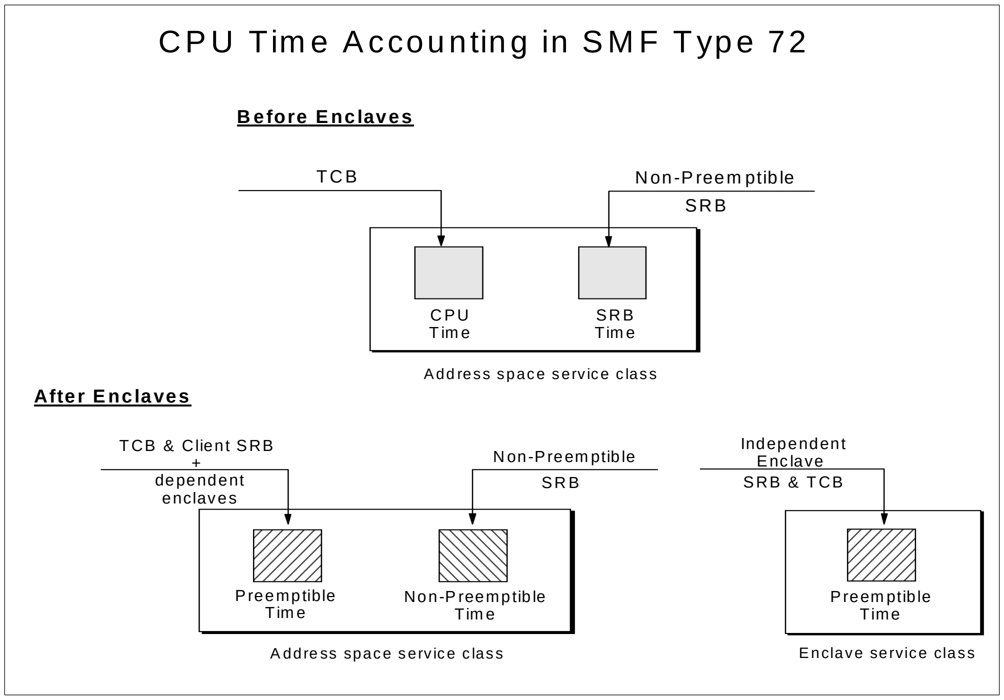


Figure 15. Accounting Service for Enclaves and Client SRBs

2.4 Introduction to the Workload Management Services

The workload management services enable MVS to cooperate with subsystem work managers to achieve installation-defined goals for work, to distribute work across a sysplex, to manage servers and to provide meaningful feedback on how well workload management has achieved those goals.

To change from resource-based performance management to goal-oriented workload management, many transaction managers, data managers, and performance monitors and reporters need to take advantage of the services MVS workload management provides.

This section describes the services available for subsystem work managers, like transaction and data managers. The services available for performance monitors and administrative application programs are not covered in this section.

For more information about the workload management services see *OS/390 MVS Programming: Workload Management Services*, GC28-1773.

2.5 The Subsystem Work Manager Service

The workload management services for subsystem work managers allow an installation to process work towards performance goals defined in a service policy. Workload management uses the information provided by the subsystem work managers through the services to match system resources to work in order to meet these goals.

Workload management matches system resources to meet the performance goal assigned to a service class. This management involves handling address space-related resources, such as processor storage, multi-programming level (MPL), dispatching, and I/O queueing.

A work manager can use many different combinations of the workload management services. Which services and which combinations are used by the subsystem depend on the benefit it gets from using them, whether the programming environment allows it to use them, and the structure of the subsystem work manager using the services.

Subsystem work managers, like CICS or IMS, use the *execution delay monitoring services* to get information about how well the transactions are processed and which resources are needed to meet the goal.

However, this service supports only response time goals. If a subsystem needs to assign velocity or discretionary goals, or the subsystem has a multiaddress space structure, another subsystem work manager service must be used.

In the next sections the subsystem work manager services are described. There is also information about which service is suitable for what kind of work manager. For more information about these services and about considerations when using the services see *OS/390 MVS Programming: Workload Management Services*, GC28-1773.

2.5.1 The Work Manager Services

The work manager services allow workload management to associate incoming work with a service class. When the work is associated with a service class, MVS knows the performance goal and importance level associated with the work, as well as understanding which address spaces are involved in processing the work request.

Work manager services allow WLM to:

- Recognize a subsystem work manager and the transactions it processes
- Classify the incoming requests and recognize the service class goals associated with the transactions
- Recognize the address spaces that are processing the transactions

Based on this information, WLM can determine whether goals are being met, and which work needs resources to meet the goals.

The workload manager services are used by the traditional workloads like CICS and IMS. Table 1 shows a summary of the work manager services.

Table 1. Work Manager Services

Service	Purpose
IWMCONN	Connect a work manager subsystem to WLM, request that WLM Work Management services be made available to the connected address space, and optionally pass topology information to WLM.

Service	Purpose
IWMCLSFY	Classify the incoming request and associate it with a service class defined in the active service policy.
IWMQRY	Obtain a service class goal and importance for a service class period.
IWMDISC	Disconnect a work manger from WLM.

But what if your work manager has a client-server structure or provides services through other address spaces and has additional objectives such as:

- Dynamic management of server address spaces
- Management of server work requests as part of the originating unit of work
- Resource management and/or reporting of individual requests
- Balancing workload among servers across a sysplex

In these cases, the queueing manager, or the routing manager, or enclave services are used instead of the work manager services. Examples of such work managers are DCE, DSOM, DB2 stored procedures and sysplex query parallelism and the Domino Go Webserver.

2.5.2 The Execution Delay Monitoring Services

These are used by CICS and IMS.

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 which keeps track of the execution delays encountered by a work request.

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

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 their view of the current state of a work request, such as ready state, idle state, or waiting state. The actual state may be different. For example, a work request may 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. Table 2 gives a summary of some execution delay services.

Table 2. Execution Delay Monitoring Services Examples

Service	Purpose
IWMMCREA	Create a monitoring environment, also called performance block
IWMMCHST	Record the state, such as ready, waiting, idle of a work request
IWMMABNL	Record an abnormal event for work
IWMMINIT	Initialize monitoring environment with information about a work request
IWMRPT	Report on the completion of the work associated with the service class
IWMWQWRK	Identify where transactions are executing

2.5.3 The Enclave Services

Up to and including MVS/ESA V5.1.0, the only recipient of resource consumption was the home address space. This meant that, in a client/server configuration, the server address space, even working on behalf of a client, would be charged for its resource consumption and be managed according to its own goals and not to a client's goals.

A partial solution to this problem was given in MVS/ESA V5.2.0 with the introduction of enclaves. But at this time, only SRBs could use an enclave.

In OS/390 Release 3, the enclave concept was extended to TCBs, so that a enclave can include now both SRBs and TCBs. Refer to 2.3, “Understanding the New Dispatchable Units in OS/390” on page 59 to get more information on enclaves.

An *enclave* is an anchor for a transaction that can be spread across multiple dispatchable units and multiple address spaces. The transaction can be one dispatchable unit in a single address space, or multiple dispatchable units running in one or more address spaces. Multiple enclaves can exist simultaneously in a single address space.

The value of using an enclave to represent a transaction is that the resources used to process the transaction can be accounted to the transaction itself, rather than to the address space or spaces that the transaction runs in. In addition, you can assign a performance goal to the enclave, which means that as a transaction consumes system resources, it can switch periods to run with a new goal (in goal mode) or a different set of performance group characteristics defined in the IEAIPSxx SYS1.PARMLIB member (in compatibility mode).

Enclaves allow you to manage a transaction that spans multiple address spaces and dispatchable units separately from the server address spaces it runs in. For example, the dispatching priority for a unit of work in goal mode could be set based on the business objectives of the enclave.

Additionally, the use of enclaves allows the management of multiple transactions with different goals in a single address space, without the more important work being impacted by the less important.

Any number of tasks and SRBs can be grouped together in an enclave:

- Enclave SRBs offer the advantage that they are preemptible and will not tie up the system. SRBs in enclaves work well for higher volume, small requests, as SRBs have very little overhead compared to tasks. The subsystem can create an enclave using the IWMECREA macro, and then schedule SRBs to run in the enclave using the IEAMSCHD macro.
- Tasks in enclaves automatically associate the enclave with the address spaces where they are dispatched, so workload management can manage the storage of those address spaces to meet the goal of the enclave. The enclave can perform functions that require a task environment, such as supervisor calls. Tasks can dynamically leave and join an enclave as they finish one piece of work and begin another. This allows the existence of a server address space with a permanent TCB, with the TCB dynamically switching from one enclave to another.

TCBs and SRBs of an address space can belong to different enclaves, but an SRB or TCB can belong only to one enclave.

Queueing manager services require the use of enclaves.

Table 3 shows a summary of some enclave services.

Table 3. Enclave Services Example

Service	Purpose
IWMECREA	Create an enclave.
IEAMSCHD	Schedule an SRB into the enclave.
IWMEJOIN	Join an task (TCB) to an enclave. Once a task has joined an enclave, all future processing is on behalf the transaction represented by the enclave.
IWMELEAV	Leave an enclave (task only).

2.5.3.1 Enclaves Compared to Execution Delay Monitoring Services

You cannot use tasks in enclaves and execution delay services in the same address space.

Using enclaves has the following advantages over execution delay services:

- Isolation of transactions

  Enclaves allow separate dispatching priorities to be assigned to work running in the same address space. Therefore, workload management can manage this work to different performance goals. Without enclaves, all work in an address space runs at the same major dispatching priority.
- Period control

Enclaves can run in a service class with multiple periods. Because resource consumption is tracked for individual enclaves, the enclave can move from one period to the next as it consumes CPU resource. The goals for the periods can be chosen to favor short transactions over long ones within a single address space.

- Full goal support

  Enclaves support response time, velocity, and discretionary goals, whereas transactions reported using execution delay services can be managed only to response time goals.

- Server address space management

  Enclaves are independent from an address space, so a transaction that moves from the originating address space to one or more server address spaces can be managed as a single transaction.

2.5.4 The Queuing Manager Services

A *queueing manager* is a subsystem or application that queues work requests to WLM for execution by one or more server address spaces.

The queueing manager services are intended for queueing managers to use to manage server (execution) address spaces and the work requests they process to meet service class performance goals. Through queueing manager services, WLM maintains the queues for passing work requests from the queueing manager to its servers.

WLM dynamically starts and maintains server address spaces as required to meet the queueing manager's workload. Therefore, installations do not have to manage the address spaces manually, nor do they have to monitor workload fluctuations that change the number of address spaces needed for the work to meet its goals. Workload management automatically adjusts to changes in the workload.

The design of server address space management addresses the following questions:

- If the objectives 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 the number of server address spaces be decreased?
- Will the creation of a new server address space adversely impact the performance of other more important goals?
- How do you report resource consumption and response time of units of work that span more than one business unit of work (enclave)?

For queue managers using the services, WLM spreads the work across multiple address spaces, providing workload isolation and greater scalability based on workload demands. For a queueing manager that queues and executes work all in the same address space, sometimes encountering storage overlay problems, the services provide an incentive to change to a multiple address space configuration.

Attention

The queue manager services have a single-system scope. That means all server address spaces started and managed by this service run in the same OS/390 image as the queue manager address space.

DB2 stored procedures and scalable Domino Go Webserver for example, use this WLM service to permit workflow from their network-attached address spaces into server address spaces for execution.

The queueing manager services use application environments. For more information about the WLM application environment see 2.6.1, “Server Address Space Management” on page 77, and *OS/390 Release 3 Implementation*, SG24-2067. At server address space initialization, it connects to an application environment.

In order to use these services, a work manager must also use enclaves.

Table 4 shows a summary of some queue manager services:

Table 4. Queueing Manager Service Examples

Service	Purpose
IWMCONN	With the QUEUE_MANAGER=YES parameter, establish a address space as a queue manager to WLM so the address space can begin queueing work requests to its server address spaces.
IWMCONN	With the SERVER_MANAGER=YES parameter, establish the caller as a server address space to WLM so it can begin to receive work requests from the WLM-managed queue.
IWMQINS	Insert a work request onto WLM queues so that its execution in a server address space can be managed by WLM.
IWMSSEL	Select a work request from WLM queues for execution in a server address space.

2.5.5 The Routing Manager Services

These services are used by the SOM/DSOM and Smartbatch subsystems.

A *routing manager* is a subsystem that establishes and manages connections between a client and a server address space. The routing manager handles these connections rather than individual work requests; the requests are processed only after they arrive in the server address space.

The routing manager is responsible for balancing the client connections across a set of eligible servers, with the assistance of the routing manager services.

Routing manager services perform two main functions:

- Automatically starting and maintaining server address spaces as needed by the workload across the sysplex. Installations then do not have to manage the address spaces manually, nor do they have to monitor workload fluctuations that change the number of address spaces needed. WLM automatically adds servers to adjust to changes in the workload.

- Balancing the workload among the servers in the sysplex by deciding on the best server and providing the server routing information when a server is requested by the routing manager.

The design of routing manager services addresses the following objectives:

- Isolate different types of requests into separate server address spaces for integrity, security and operational reasons
- Have MVS add server address spaces to meet the goals
- Let MVS balance the workload across a sysplex by selecting the best system on which to start a server

Routing manager services combines the use of the “find server” function with application environments and enclaves.

When defining an application environment, you must specify whether workload management can start multiple or single address spaces for the subsystem. In the case of a routing manager, both Option 1, No limit, and Option 3, Single address space per sysplex are valid.

The routing manager will not queue a request to a WLM queue, like a queue manager. Instead, it returns routing information to the client. The client uses this information to contact the selected server.

The routing manager can start server address spaces across a sysplex, but it starts only one address space for an application environment per system image in the sysplex.

If necessary, the client request will be suspended until a new server address space is created. The first server is started on the system with the most available capacity. Subsequent servers are started on other systems in the sysplex when the work running in the existing servers is not meeting it goal. However, if the workload diminishes, WLM does not decrease the number of servers.

Table 5 shows a summary of some routing manager services.

Table 5. Routing Manager Services

Service	Purpose
IWMCONN	With the ROUTER=YES parameter, establish the address space as a routing manager so it can begin to request server routing information through IWMSRFSV.
IWMCONN	With the SERVER_MANAGER=YES and SERVER_TYPE=ROUTING parameters, establish the address space as an eligible server for requests coming from a routing manager. WLM will balance the workload among the eligible servers.
IWMSRFSV	Find the best server for this work request. If no server exists for a request, start one.

2.5.6 The Sysplex Routing Services

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 having each installation have to make the 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 following information:

- Location name
- Network ID
- LU name

WLM then makes this information available to all OS/390 images in the sysplex.

When a client wants to route work to a server, it issues the IWMSRSRS macro for a list of registered servers. To help the client decide where to route the request, this macro returns, for every eligible server, an associated weight. The weights represent the relative number of requests each server should receive. Various capacity considerations are used to calculate the weights. The weights allow changes in system load and server availability to be factored into work distribution. A client should issue the macro on a regular basis to stay current with the changing conditions.

Note: More than half of the systems running in the sysplex should be running in goal mode. Workload balancing can be done more intelligently if all servers are located on systems running in goal mode.

Routing Manager compared to Sysplex Routing Services

Routing manager services differ from sysplex routing services in the following ways:

- Routing manager services provide automatic management of address spaces.
- Routing manager services include the server's performance index when selecting the best available servers.
- With routing manager services, workload management decides on a server and passes its identity to the routing manager, instead of offering the routing manager a choice of several servers.

2.5.7 The Scheduling Environment Services

These services are utilized by JES2 and JES3 for resource affinity scheduling.

A *scheduling environment* is a list of resource requirements, allowing you to ensure that transactions are sent to systems that have the appropriate resources to handle them. Resources can represent actual physical entities, such as a data base or a peripheral device, or they can represent intangible qualities such as a certain period of time (like second shift or weekend).

These resources are listed in the scheduling environment according to whether they must be set to ON or set to OFF. A unit of work can be assigned to a specific system only when *all* of the required resource states are satisfied.

There is a WLM service to change resource state settings. The resource state can be changed only on the system in which the program is executing. The following OS/390 command uses this service:

```
F WLM,RESOURCE=resourcename,(ON|OFF|RESET)
```

2.6 WLM Functions

This section describes WLM functions which are used by subsystem work managers.

2.6.1 Server Address Space Management

Some work managers, such as DB2, provide services through other address spaces. The services may be provided by one or more address spaces.

In OS/390 Version 1.3, WLM introduced the 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 the number of server address spaces be decreased?
- 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 queueing manager services are getting the benefits of server address space management. Refer to 2.5.4, “The Queuing Manager Services” on page 73 for more information on queueing manager services. Examples of IBM-supplied work managers that use these services are:

- DB2 for stored procedures
- Component Broker

- Domino Go Webserver
- JES2 (WLM-managed initiators in OS/390 R4 and higher)
- JES3 (WLM-managed initiators in OS/390 R8 and higher)

These work managers use WLM services to permit workflow from their network attached address spaces, through WLM, and into server address spaces for execution. The network attached address spaces are also called *queue managers* 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 a new server address space has to be created
  - Reports on goal achievement
  - Monitors the performance of transaction environments, gathers performance information

2.6.1.1 Application Environments

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 scope of the application environment is dictated by the WLM services the work manager uses. The queueing manager services provide system scope, and routing services provide sysplex scope for an application environment. For a description of these services see 2.5.4, “The Queuing Manager Services” on page 73 and 2.5.5, “The Routing Manager Services” on page 74.

You have to define your application environments and assign incoming requests to a specific application environment. For subsystem-related information see 5.2.3.1, “Using WLM-Established Address Spaces” on page 183 and 5.3.5.2, “Define the WLM Application Environments” on page 197.

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 managed by WLM are created to serve work associated with a specific application environment queue. Defining multiple application environments allow you to separate your workload into different address spaces. In addition, within an application environment there is a separate queue for each service class. This means that work run with different goals run in separate address spaces and can be managed independently.

The workload requirements for a given application may determine that multiple server address spaces should be activated. WLM decides to activate a new address space based on the following information:

1. There is available capacity (CPU and storage) in the system and work in a service class is being delayed waiting on the WLM queue.
2. 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 missing goals (receiver).

There is no mechanism provided for customer 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 address spaces. In compatibility mode, the server address spaces must be started and stopped manually (or via automation). JES2's and JES3's use of server address space management only works in goal mode.

2.6.1.2 Application Environment Queues

Various work requests (transactions and jobs) for a given application environment may 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 sub-queues. Each unique combination of application environment and service class defines a single application environment queue, as shown in Figure 16, and each server address space can process requests from only one queue.

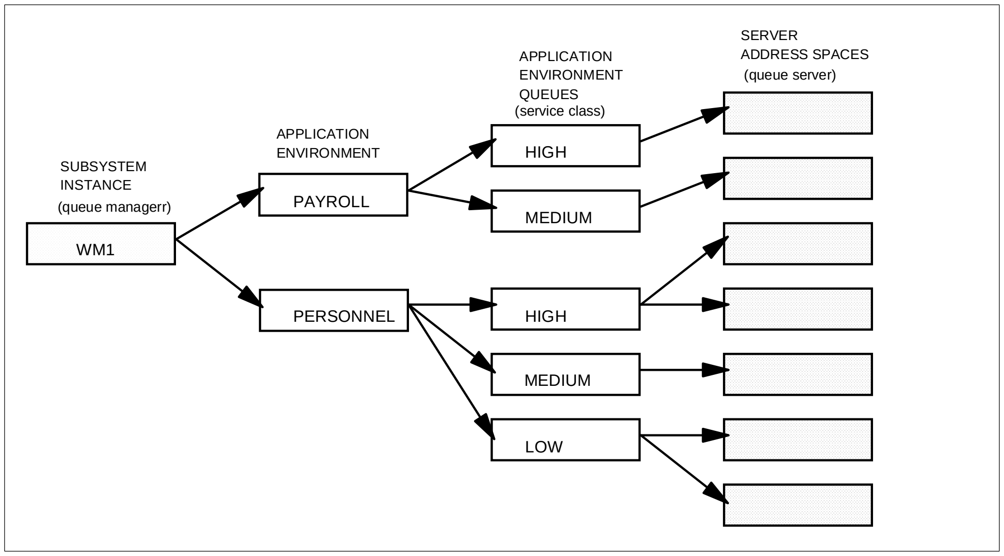


Figure 16. Transaction Flow in an Application Environment

Important

Remember that for every application environment queue, WLM starts at least one address space. Therefore, you should not specify too many service classes, in order to avoid having WLM start too many address spaces.

2.6.1.3 Transaction Flow in an Application Environment

This section gives a short explanation about the transaction flow in a application environment. For a better understanding, refer to Figure 16.

1. A transaction reaches the work manager (WM1), and the work manager checks the request against its definition. If there is no match against a application environment for that request, it is processed in the WM1 address space.
2. If the request matched 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 was 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 tranaction’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.

2.6.2 WLM Batch Initiator Management

The pressures on the batch window have caused a never ending search for solutions. Truly, many companies cannot afford the batch window to finish after a certain hour in the morning.

The advent of WLM in MVS/ESA V5 brought many improvements into the batch workload. Among them is the *possibility* of attributing a response time goal to a batch job (though this is not generally recommended). Using this facility, an installation may determine the job critical path (the sequence of batch jobs that determines the elapsed time of the window) and could apply a challenging response time to each one. Then, by WLM honoring these goals, the installation may gain greater control over the window.

Note: You can only use batch response time goals for service classes with a steady supply of ending jobs — not for setting a response time for individual jobs.

However, such a solution deserves some careful consideration. The response time of a batch job is made of *queue time* plus *execution time*. Before OS/390 2.4, the only way for WLM accomplish a response time goal was to act upon the execution time. This was done through the control of resources as CPU, storage, and I/O during the execution of the job.

Frequently, however, the major reason for not reaching the response time goal was a long queue time. WLM has no control over such time. Because of this, we recommended not to use response time as a preferred type of goal for batch workload.

This batch queue time is caused by two major reasons:

- Resource dependencies with previous ongoing jobs
- Lack of initiators with same class as the queued job

Let us focus on the lack of enough initiators, or lack of initiators of the correct job class. Many installations avoid this situation by overinitiating OS/390, that is, defining a large number of initiators. However, experience shows that this action overwhelms the system, causing overhead and thrashing.

As you can see, there is a trade-off between defining fewer initiators and have a long batch queue time, or defining more initiators and facing some overhead.

The main purpose of batch initiator management is to solve this problem by letting WLM dynamically adjust the number of active initiators. In this way, WLM can manage the time that jobs wait for an initiator based on the goals of the work. For more information on managing batch workloads in goal mode refer to 4.4.2, “Considerations for Migrating Batch Workloads” on page 148

This WLM function is only active in goal mode.

2.6.3 A Brief Job Batch Story of Time

A job starts when submitted to JES. It ends when its last step is finally processed by the terminator. The spool processing time is not included in the job life.

Figure 17 depicts the events throughout a JES2 batch job’s existence.

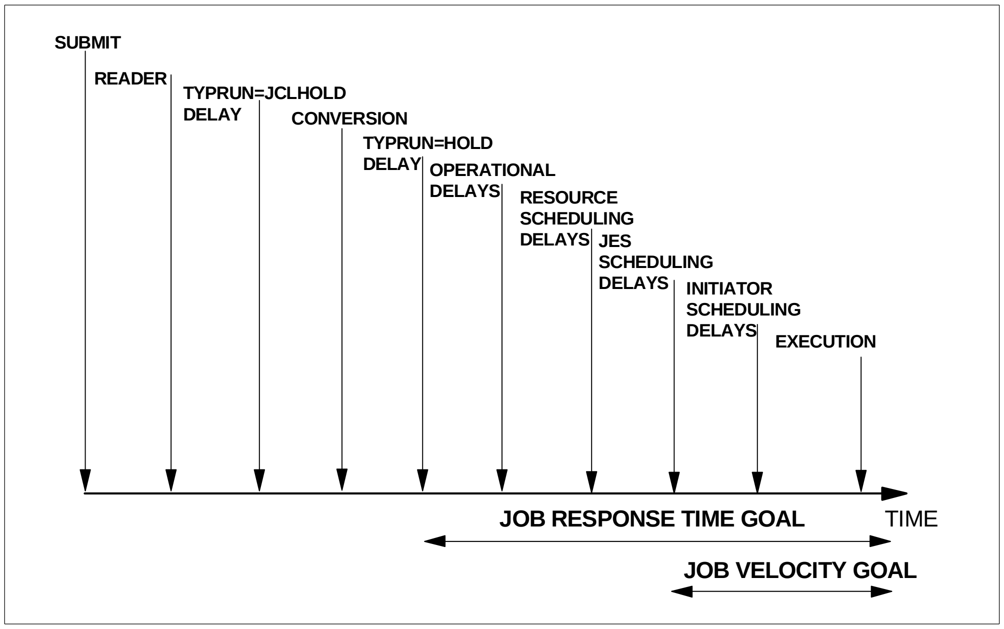


Figure 17. A Batch Job’s Events

The possible delays are:

1. After Reader:
    1. TYPRUN = JCLHOLD delay

       Requests that JES2 hold the job before completing JCL processing. JES2 holds the job until the operator releases it.
2. After Conversion:
    1. TYPRUN = HOLD delay

Requests that the system hold the job before execution until the operator releases it.

2. Operational delay (by operator command: job hold, job class hold). This is shown by the RMF Workload Activity report in the field *Ineligible.* Refer to 2.6.6, “New RMF Fields” on page 84.
3. Resource Affinity Scheduling delay (refer to 2.6.7, “Resource Affinity Scheduling” on page 85). This is shown by the RMF Workload Activity report in the field *R/S Affinity.* Refer to 2.6.6, “New RMF Fields” on page 84.
4. Scheduling delays (class limit, duplicate jobnames). This is shown by the RMF Workload Activity report in the field *Ineligible.* Refer to *2.6.6, “New RMF Fields” on page 84.*
5. Queue delay (waiting for an initiator) - shown by RMF in the field *Queued.*

3. After Initiation:
    1. Delay by data sets (ENQ)
    2. Delay by volumes
4. After starting execution:
    1. Delay by CPU
    2. Delay by IO
    3. Delay by storage
    4. Delay by other reasons

By definition a batch response time includes from 2.2 (operational delays) to the termination. Its value is equal to:

```
Response_time = Queue-time + Service_time
```

Queue_time is measured from 2.2 (operational delay) to 2.5 (queue delay).

Service_time is measured from 3 (initiation) to the end of 4 (execution).

As you can see, among the delays batch initiator management only addresses the queue delay (waiting for an initiator). The other three delays (operational, affinity and scheduling) are included in the response time, but they are out of WLM’s control.

Batch initiator management is achieved by letting WLM control certain designated jobclasses. It is possible that an installation may have a mix of WLM- and JES2-managed jobclasses.

To manage the queue delay for batch jobs, WLM does the following:

- It creates and deletes new initiators for WLM-managed jobclasses.
- It schedules batch work based on WLM service classes, rather than job classes. This allows WLM to treat the batch work differently depending on its goals.

2.6.4 Using Batch Initiator Management

The following is a summary of the changes introduced for batch initiator management in a JES2 environment. Some of these are external changes and they imply an installation action:

- Define WLM-managed jobclasses, with the number of initiators limited by a maximum value, via JES2 jobclass parameters in JES2 parms (or a $T Jobclass command):

  ```
  JOBCLASS=Q, MODE=WLM, XEQCOUNT=MAX=nnnn
  ```

  In this example, jobs from class Q are selected by a WLM-controlled initiator associated with the job's service class.

  JES2 classifies the batch job earlier, at the end of job conversion rather than at job selection. This allows WLM to manage the job's queue delay time according to the service class goal.

- JES2 now maintains two different queue organizations for all jobs awaiting execution (initiator selection) as shown in Figure 18.
  - *All jobs* are queued by job class, priority and the order in which they finish conversion. This is the queue from which JES2-managed initiators select jobs for execution.
  - Jobs awaiting execution in WLM-managed job classes are also queued by their WLM-assigned service class in the order they were made available for execution.

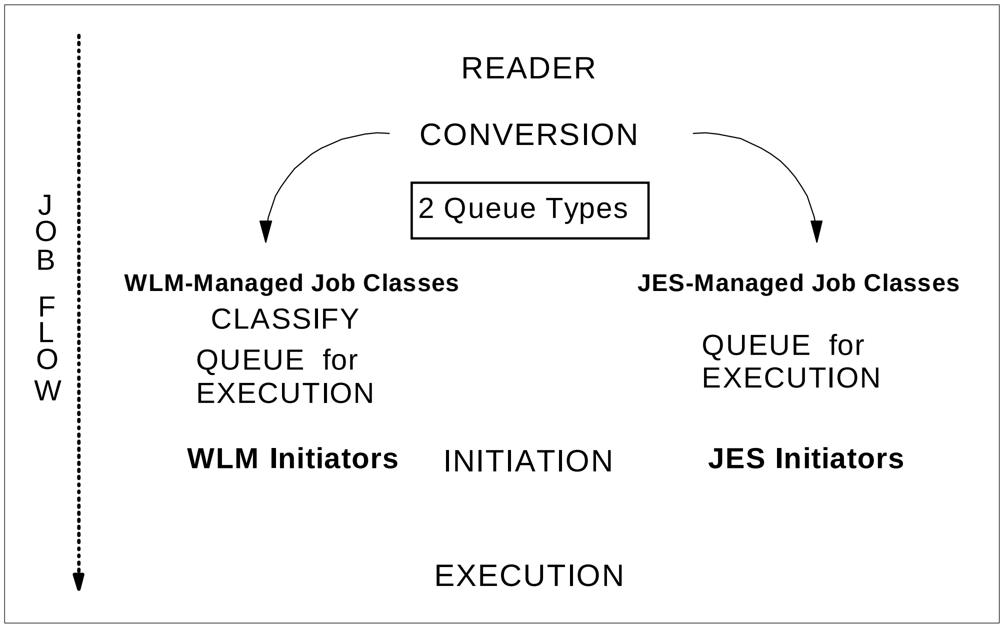


Figure 18. Batch Job Flow Phases

- WLM relies upon additional work queueing delay information provided by JES2.
- The policy adjustment routine triggers a START INIT command when the major delay for a batch service class period is the delay-for-initiator. This starting initiator first serves the service class period suffering these delays.

- WLM-managed initiators can be dynamically reassigned to service class queues that require increased throughput.
- The velocity calculation is changed to include queue delay for jobs in WLM-managed classes. However, this delay only includes the item 2.4 that is the queue delay - waiting for an initiator. As the number of jobs waiting for WLM initiators grows, the velocity of the service class decreases.
- Enhancements in the RMF report showing more detail for job response time with pre-execution job delays. Refer to 2.6.6, “New RMF Fields” on page 84.
- The $P XEQ JES2 operator command halts further scheduling on the system where the command is executed, and causes the related JES2 and WLM initiators to drain. Scheduling can then be resumed by using the $S XEQ command.
- The $S Job JES2 operator command immediately starts a batch job that is in a WLM-managed job class. An initiator is created especially for the job and deleted when the job finishes. All affinities are honored.

2.6.5 Functions Controlled by JES2

Certain functions are still controlled by JES2 even with the activation of batch initiator management. Among them are:

- The management of the traditional JES-managed jobclasses, that is the ones with:

  ```
  JOBCLASS(X) MODE=JES
  ```

- Affinity processing
- JCL processing and the commands for holding/releasing
- The processing of duplicate job names

2.6.6 New RMF Fields

The RMF Workload Activity Report has new fields covering WLM batch initiator management. They only have non-zero values if WLM batch initiator management is applied to the jobs. The fields are:

- **Ineligible:** The average time the job is delayed due to operational delays or JES2 scheduling delays, such as:
  - Job held by operator
  - Job class or job queue held
  - Duplicate jobname serialization
  - Job class execution limits

  Ineligible is included in *total response time*, but not included in *queued time* in the same report and is not inluded in the velocity calculation.
- **Conversion**: The average time the job was delayed due to JCL conversion. Jobs held during conversion (due to affinity, HSM recall, or enqueue contention) contribute only to conversion time, not to ineligible or R/S affinity times. Conversion time is not included in *total response time* and not included in *queued time* in the same report.
- **R/S Affinity:** The average time the job is delayed due to resources required for the job to run. Refer to 2.6.7, “Resource Affinity Scheduling” on page 85.

R/S Affinity is included in *total response time,* but not included in *queued time* in the same report and is not included in the velocity calculation.

Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information about these RMF fields.

2.6.7 Resource Affinity Scheduling

Resource affinity scheduling is exploited by both JES2 and JES3 work managers.

To understand the idea of *resource affinity scheduling*, we need to introduce other concepts such as scheduling environments and resource elements.

A *scheduling environment* is a combination of sysplex systems, database managers, devices, and languages capable of executing an application. In this context, environment should not be confused with the WLM concept of application environment as described in 1.2.9, “Application Environment” on page 15.

A challenge to be resolved in a multisystem configuration is this: *not every OS/390 system is capable of providing all scheduling environments at all times, and this fact should be transparent to the user.* A scheduling environment may be limited to a subset of systems within a sysplex due to:

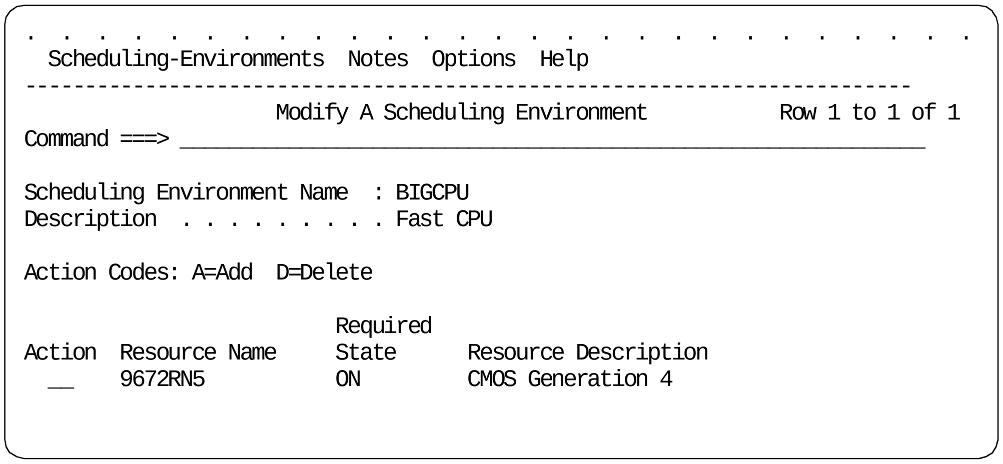
- Software licensing
- Machine resources
- Peripheral device attachments
- Database accessibility
- Existence of other, conflicting application environments
- A specific time frame (during the day or the weekend)

The WLM resource affinity scheduling function extends the capability of workload scheduling components (like JES) by enhancing their scheduling mechanism. The use of this facility is optional and WLM does not need to be in goal mode. Resource affinity scheduling allows an installation to do the following:

- Define *resource elements* in the sysplex. A resource element is a representation of an execution time resource. The installation must identify and name these entities. A resource element can be:
  - A database, peripheral device, machine feature
  - Second shift, cheap cycles, weekend
- Set the state of each resource element (ON, OFF or RESET) independently on each system.
- Define a *scheduling environment*, which is a grouping of some number of resource elements, each element having a specified required state, into a single named entity to be used in making scheduling decisions. The scheduling environment is either available (all the resource elements are in the required state) or unavailable (one or more resource elements are not in the required state).
- Provide information on the state of the resource elements and the availability of the scheduling environments to interested parties (such as JES or

Automation) that make decisions on whether or not the required execution environment is available on an OS/390 instance in the sysplex.

Figure 19 on page 86 shows an example of the panel where a scheduling environment is defined.



```
 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
  Scheduling-Environments  Notes  Options  Help
 --------------------------------------------------------------------------
                     Modify A Scheduling Environment          Row 1 to 1 of 1
 Command ===> ______________________________________________________________

 Scheduling Environment Name  : BIGCPU
 Description  . . . . . . . . . Fast CPU

 Action Codes: A=Add  D=Delete

                          Required
 Action  Resource Name    State      Resource Description
   __    9672RN5          ON         CMOS Generation 4
```

Figure 19. Scheduling Environment Definition Screen

Chapter 3. Implementation of WLM in Goal Mode

This chapter focuses on the systems implementation of WLM goal mode. The primary audience is systems programmers and performance administrators performing the implementation activities.

WLM replaces the primary existing set of SRM controls: the IEAICSxx and IEAIPSxx parmlib members. Processing from an IPS and ICS is called *compatibility mode;* processing towards goals defined in the WLM service definition is called *goal mode*. It is possible to switch individual systems between the two modes of operation with an operator command.

The following sections describe how to perform the transition from compatibility mode to goal mode. The discussion assumes a multisystem sysplex environment, which includes the special case of a single system monoplex. Implementation steps are:

- *Planning*, covered in 3.1, “OS/390 Performance Control Interface” on page 87: Analyze the current performance management parmlib members, understand WLM goal mode implementation changes, and plan the implementation.
- *Preparation*, covered in 3.2, “Preparing the WLM Environment” on page 89: WLM requires a sysplex environment, implementation of shared WLM Couple Data Sets (CDS), access to WLM ISPF application, and operational management processes.
- *Creating the service definition*, covered in 3.3, “Service Definition Implementation” on page 104: Use of the WLM ISPF application, also referred to as the WLM administrative application, to define goal mode service definitions.

3.1 OS/390 Performance Control Interface

Prior to the introduction of WLM, systems performance input was accomplished on each individual system using three key parmlib members:

- IEAICSxx in PARMLIB, Installation Control Specification, associates units of work with performance groups (PGN), which provide the performance characteristics to each address space.
- IEAIPSxx in PARMLIB, Installation Performance Specification, defines each PGN with its performance characteristics that control dispatching, I/O queueing and storage management.
- IEAOPTxx in PARMLIB for parameters that adjust thresholds that influence the SRM algorithms.

Each individual system was tuned for a specific workload by implementing unique IPS, ICS, and OPT parmlib members.

Now, WLM processing in goal mode uses the following:

- In IEAOPTxx in PARMLIB, only a subset of the parameters are now applicable. Basically it is possible to control physical resource thresholds like central and expanded storage, but not values that influence the system behavior such as expanded storage criteria aging, logical swapping, and MPL control.

- The WLM Service Definition is installed and activated into a WLM sysplex couple data set using the WLM ISPF application; 3.3, “Service Definition Implementation” on page 104 describes the ISPF interface to input and manage the service definition.

Performance management with WLM goal mode uses a single service definition for the entire sysplex; plan your service definition to work for all workloads within the sysplex. Using normal change management processes, implement goal mode on a system within the sysplex. When WLM goal mode has stabilized on this system, then identify and implement WLM goal mode on the next system in the sysplex. This is an iterative process until all systems have been converted to WLM goal mode. Whenever adding a system to WLM goal mode, carefully monitor the performance to ensure you are getting the performance characteristics you planned for.

3.1.1 IEAOPTxx Usage

A subset of the parameters in IEAOPTxx continues to be used in goal mode. Values that are not used in goal mode are ignored; no changes are necessary to the existing IEAOPTxx.

The following IEAOPTxx parameters are no longer used in goal mode:

- MPL adjustment constants
  - RCCCPUT: CPU utilization threshold
  - RCCPTRT: Page fault rate threshold
  - RCCUICT: UIC threshold
  - RCCFXTT: Fixed on-line storage threshold
  - RCCFXET: First 16 megabyte central storage threshold
- Logical swapping options
  - LSCTMTE: Think time thresholds
  - LSCTUCT: System think time
  - LSCTFTT: On-line storage fixed thresholds
  - LSCTFET: First 16 megabytes fixed thresholds
- CPU management constants
  - CCCSIGUR: Heavy CPU Mean-time-to-wait threshold
- Maximum swap set size
  - MCCMAXSW: Maximum swap size
- Expanded storage constants
  - ESCTPOC: Page that is to be paged out
  - ESCTSTC: Page that is to be stolen
  - ESCTSWTC: Trimmed page for a swap out
  - ESCTSWWS: Working set page ready for swap out
  - ESCTVIO: Virtual IO page
  - ESCTVF: Virtual fetch page
  - ESCTBDS: Hiperspace page

The following IEAOPTxx 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
- Pageable storage shortage constants
    - MCCFXEPR: First 16 megabytes fixed storage %
    - MCCFXTPR: On-line 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

For additional IEAOPTxx information see *OS/390 MVS Initialization and Tuning Guide,* SC28-1751 and *OS/390 MVS Initialization and Tuning Reference,* SC28-1752.

3.2 Preparing the WLM Environment

To use *goal mode* you must be in a single system sysplex (monoplex) or a multisystem sysplex. We assume prior implementation of required sysplex components which might include shared DASD configuration, CTC connectivity, Sysplex Timer, and a sysplex couple data set. WLM uses a specially formatted WLM couple data set where the service definition will be located. This data set must be accessible by all systems. A WLM ISPF application is used to create the set of goal statements that the service definition is composed of.

Figure 20 on page 90 illustrates the major steps to implement WLM goal mode. The remainder of this section goes into further detail on completing this work.

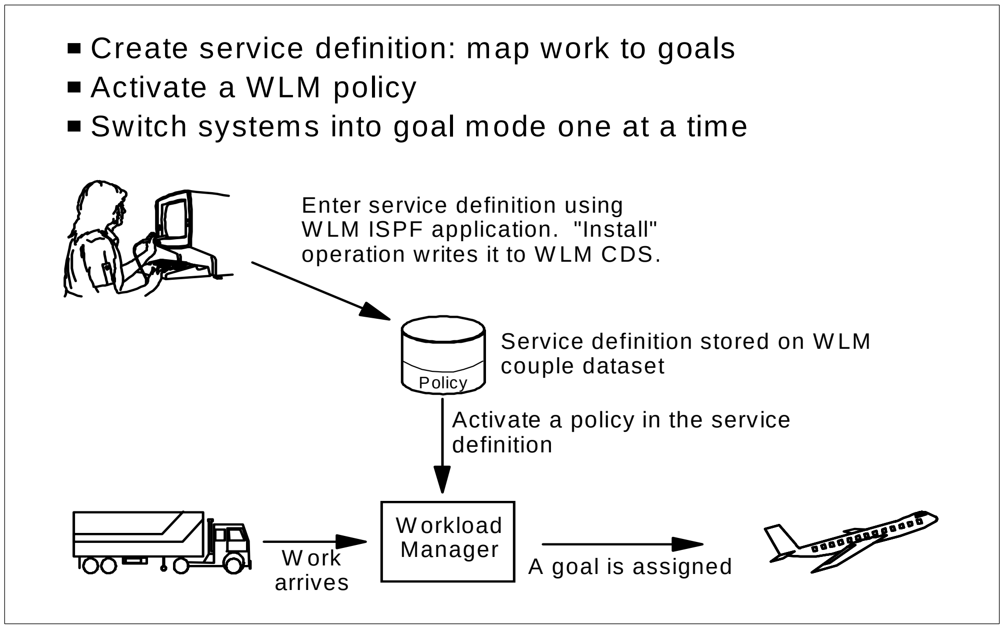

■ Create service definition: map work to goals
■ Activate a WLM policy
■ Switch systems into goal mode one at a time
Enter service definition using WLM ISPF application. "Install" operation writes it to WLM CDS.
Policy
Service definition stored on WLM couple dataset
Activate a policy in the service definition
Workload Manager
Work arrives
A goal is assigned

Figure 20. WLM Goal Mode Implementation

The following are the steps to implement goal mode. You need to prepare the operating systems, then introduce sysplex changes, then the service definition, and finally switch to WLM goal mode implementation. Review your environment, and prepare a plan to address the following activities:

1. Establish an operating environment where WLM goal mode will be implemented. If you are installing a new release of OS/390, do it now.

   If running mixed MVS releases, some additional planning is required. See 3.2.1, “WLM Compatibility and Co-Existence” on page 93 for toleration maintenance considerations, functionality levels, and hints and tips on managing the WLM ISPF application in a mixed release sysplex environment.

2. Be prepared to run MVS in compatibility mode (processing with your existing IPS and ICS) until you complete the steps for migrating to workload management and are comfortable switching to goal mode. Implement service coefficient value changes prior to goal mode, and maintain a comparable IPS/ICS environment as a contingency. See 4.2.6, “Service Definition Coefficients” on page 136 for additional information.

3. Adjust SMF recording. Before you activate a policy and switch your systems into goal mode, you should adjust for SMF goal mode recording changes as follows:

   - Turn off SMF type 99 records. They 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 chargeback based on SMF record type 30 or record type 72 records; you may need to update your accounting package.

For more information about SMF record changes for goal mode, see *OS/390 MVS System Management Facilities (SMF)*, GC28-1783.

4. Adjust your RMF procedures to support the RMF Sysplex Data Server. The RMF Sysplex Data Server is a distributed RMF function that is started on each system of the sysplex. Each copy of the data server communicates with all other copies in the sysplex. RMF uses this sysplex communication method to provide access to distributed RMF measurement data from any point in the sysplex.
5. In order to use WLM goal mode, you must be running in a sysplex with at least an MVS/ESA Version 5 formatted couple data set. You can be running in a single system sysplex (monoplex) or a multisystem sysplex. For more information on implementing sysplex and upgrading the sysplex CDS, see *OS/390 MVS Setting Up a Sysplex*, GC28-1779.
6. Allocate the WLM couple data sets, then make them available for use. See 3.2.3, “Couple Data Set Management” on page 99 for sample couple data set allocation JCL and operations commands.
7. Set up performance objectives. Set up a service definition from the performance objectives (in worksheet format). See Chapter 4, “Migrating to WLM Goal Mode” on page 129 for helpful instructions on planning a service definition. We recommend using the worksheets in Appendix A, “Sample Worksheets” on page 213, since they provide a guideline of what is expected.

   Note: Your initial research/implementation of the service definition will be focused on workload classification, and activated in conjunction with ICS SRVCLASS parameters to capture RMF monitor 1 workload activity report data. For information about the SRVCLASS parameter, refer to step 11 on page 92 and 4.2.1, “Getting RMF Data in WLM Compatibility Mode” on page 132. The data from RMF helps you to set initial goals for the services classes in your service definition. Remember to review peak usage periods when setting your goals. If you use long-term averages to set goals, the goals may be unattainable during peak periods.
8. Set up the WLM ISPF application. You need to consider security access and how your installation will manage service definition resources. 3.2.2, “WLM ISPF Application Implementation” on page 95 provides guidelines for implementation.

   Now you are ready to create the service definition using the WLM ISPF application. In 3.3, “Service Definition Implementation” on page 104 we show you how to use the ISPF application to specify your performance objectives established in step 7.
9. Install the service definition in the WLM couple data set. Go into the ISPF application, specifying the name of the partitioned data set (PDS) containing your service definition. From the Definition Menu, go to **Utilities** on the action bar. Then select the Utilities pull-down option **Install Definition**.
10. Activate a service policy using either the WLM ISPF application or the VARY operator command.

    To activate a service policy from the application, choose the **Utilities** option from the action bar on the definition menu.

    To activate a service policy with the VARY command, specify:

    ```
    VARY WLM,POLICY=xxxx
    ```

where xxxx is the name of a policy defined in the installed service definition.

Once you issue the command, there is an active policy for the sysplex. Systems in compatibility mode are still managing resources according to the existing IEAIPSxx and IEAICSxx parmlib members.

Note: If you have been following these steps, all systems are executing in compatibility mode. If any systems were in goal mode (using the default policy), they would now start managing system resources to meet the goals defined in the newly defined service policy.

11. Refine your service definition with realistic goals based on data from the RMF Monitor 1 Workload Activity report. RMF reporting is based on your performance groups and ICS report groups and SRVCLASS reporting, where the implementation of the service policy in compatibility mode provides actual response times for setting transaction response time goals for subsystems like CICS and IMS.

    Review the overall numbers for performance groups and report performance groups. If the numbers are not reasonable, review your classification rules to see if you can detect any logic errors. When you switch to goal mode, plan on doing a review of your service classes to check classification logic.

    Update and install the new service definition with service class goals. Since no system is running in goal mode, this has no effect on performance, but it prepares you to switch to goal mode.

12. Switch your systems into goal mode.

    1. Switch one system into goal mode. To switch a system into workload management goal mode, specify:

       ```
       MODIFY WLM,MODE=GOAL
       ```

       Monitor the system with RMF or SDSF. Review workload classification by checking the service class assignment. Use RMF Monitor III and the RMF Monitor I Workload Activity report to confirm that goals are being achieved. Be prepared to switch back to compatibility mode, or implement changes in the policy (change service definition, install, activate), depending upon the problem impact.

       To switch into compatibility mode, specify:

       ```
       MODIFY WLM,MODE=COMPAT
       ```

       The IEAIPSxx and IEAICSxx parmlib members that go into effect after a switch to compatibility mode is dependent upon system level:

       - For levels up to and including OS/390 R2, the system uses “internal” IEAIPS and IEAICS values. These are not equivalent to either the last used IPS and ICS values, nor the contents of IEAIPS00 and IEAICS00.

         Issue SET IPS and ICS to ensure that you are using the correct compatibility performance settings, for example:

         SET IPS=02,ICS=03

       - For OS/390 R3 and higher, the system uses the values last used in compatibility mode. If the system was IPLed in goal mode, then messages are issued prompting the operator for the IEAIPSxx and IEAICSxx parmlib members.

Until you remove IPS= from IEASYSxx, the system will IPL in compatibility mode. Use either an automation tool or COMMNDxx to issue the switch to workload management goal mode at each IPL.

2. Switch the next system into goal mode. All systems in the sysplex are already attached to the WLM couple data set, and linked to the active policy. You do not need to activate the policy again. Switch the system into goal mode to start processing towards the goals of the active service policy by specifying:

```
MODIFY WLM,MODE=GOAL
```

3. IPL in goal mode by removing the IPS= keyword from your IEASYSxx parmlib member, and from your IEASYS00 parmlib member. Also remove the ICS= since it is no longer required.

   Perform this final step after you are comfortable with goal mode. By issuing MVS commands, you can switch to compatibility mode. The operator will be prompted to specify the ICS and IPS members, and can respond with:

```
SET IPS=03,ICS=01
```

3.2.1 WLM Compatibility and Co-Existence

Managing mixed MVS versions within a sysplex requires some additional WLM awareness and planning to:

- Install toleration maintenance for WLM co-existence; see 3.2.1.1, “Toleration APARs” .
- Understand the functionality level of your service definition, and how to maintain it without introducing incompatibility; see 3.2.1.2, “Functionality Levels Hints and Tips” on page 94.

3.2.1.1 Toleration APARs

If running mixed releases in a sysplex, Table 6 indicates the required WLM APARs to enable different levels of WLM to co-exist. Also check the PSP bucket (OS390Rx subset BCP) for additional information.

Table 6. Toleration APARs for WLM Compatibility

Highest Release in Sysplex	MVS/SP 5.1 APARs	MVS/SP 5.2 APARs	OS/390 R3 APARs	OS/390 R4/R5 APARs
MVS/ESA 5.2, OS/390 R1/R2	OW08866			
OS/390 R3	OW08866 OW20913	OW20913		
OS390 R4/R5	OW08866 OW20913 OW25831	OW20913 OW25831 OW25542	OW25831 OW25542	
OS/390 R6	OW08866 OW20913 OW25831 OW30930	OW20913 OW25831 OW25542 OW30930	OW25831 OW25542 OW30930	OW30930

Note: OW25542 is included in this list for consistent performance index evaluation in the sysplex; OS/390 2.4 introduced enhancements in this area. See 2.2.1.7, "Performance Index Calculation" on page 33 for more information.

3.2.1.2 Functionality Levels Hints and Tips

Each addition of new WLM externals results in a new *functionality level* for the WLM service definition. If you do not use any of the new functions, then your functionality level does not change, even if you are using the service definition on a new release. Table 7 outlines functionality levels associated with releases and the additional supported function.

Table 7. Service Definition Functionality Levels

Release	Functionality Level	Enhancements
MVS/ESA SP 5.1	LEVEL001	
MVS/ESA SP 5.2 OS/390 R1/R2	LEVEL002	Classification rules using: collection name, correlation information, connection type, package name, plan name Classification groups for: connection type, package name, plan name Classification rule extensions Service definition extensions
OS/390 R3	LEVEL003	Application environment defined Use of classification qualifiers: procedure name, perform, perform group More than 255 report classes Note: Need to use new CDS R3 format
OS/390 R4/R5	LEVEL004	Scheduling environment or scheduling environment resources define Use of Priority classification qualifier Use of SYSTEM and SYSSTC service class names within classification rules Note: Need to use new CDS R4 format
OS/390 R6	LEVEL006	Use of the description field in classification rules of classification groups
OS/390 R6 (or OW33509)	LEVEL007	Use of Process Name as a work classification qualifier

Note: LEVEL005 is RESERVED.

New releases may introduce CDS format changes to support the additional function. At the time of writing the following format levels are in use:

- Pre-R3 format (MVS/ESA 5.1/5.2, OS/390 R1/R2 formatted CDS)
- R3 format (OS/390 R3 formatted CDS)
- R4 and later format (OS/390 R4/R5/R6/R7 formatted CDS)

  Note: R3, R4 and up require you to use a WLM CDS formatted at the correct level, even if you are not using the new functions yet.

You use the WLM ISPF application to communicate the performance objective policy changes to WLM. The WLM ISPF application must be at the same level as the OS/390 system it runs on in order to access the WLM couple data set (using the extract, install, or activate functions). Table 8 on page 95 shows the compatibility matrix based on the WLM ISPF application level. Simple rules to consider are:

- The WLM application can handle any service definition at a functionality level equal to or lower than the one that corresponds to the WLM application release level.
- You must install/activate into a CDS formatted at the same, or higher, level than the WLM application you are using.

Table 8. WLM Function Compatibility

WLM ISPF Application Level	Open/Save PDS or CDS Extract when Service Definition Functionality Level is:					Install/Activate CDS format level		
	1	2	3	4	6	pre-R3	R3	R4 or higher
MVS/ESA SP 5.1	Yes	No	No	No	No	Yes	Yes	Yes
MVS/ESA SP 5.2; OS/390 R1/R2	Yes	Yes	No	No	No	Yes	Yes	Yes
OS/390 R3	Yes	Yes	Yes	No	No	No	Yes	Yes
OS/390 R4/R5	Yes	Yes	Yes	Yes	No	No	No	Yes
OS/390 R6	Yes	Yes	Yes	Yes	Yes	No	No	Yes

Attention

Use the WLM ISPF application on the *lowest level operating system* where you will be implementing WLM goal mode, and also use this system to format the CDS. This way you will not introduce any CDS format, functionality level or WLM ISPF application compatibility issues.

When you migrate the lowest WLM goal mode level system to a higher release, introduce new CDS formats, use the newer level WLM ISPF application, and take advantage of new functionality levels in your service definition.

Note: Toleration maintenance should be installed as required to systems within the sysplex, regardless of the CDS format, ISPF application level used, or WLM mode.

3.2.2 WLM ISPF Application Implementation

The WLM ISPF application, shown in Figure 21 on page 96, accepts service definition parameters from the user. The parameters are stored in ISPF tables which the user can save into an MVS PDS and install into the CDS. Once the service definition is installed, the user can select a policy to activate.

Note: Data set userid.WLM.SAVExx (where userid is the TSO ID running the application and xx is some numeric value such as SAVE01) is allocated by the WLM application for recovery and is deleted by WLM upon exiting the application. Do not use this naming convention for any other purpose.

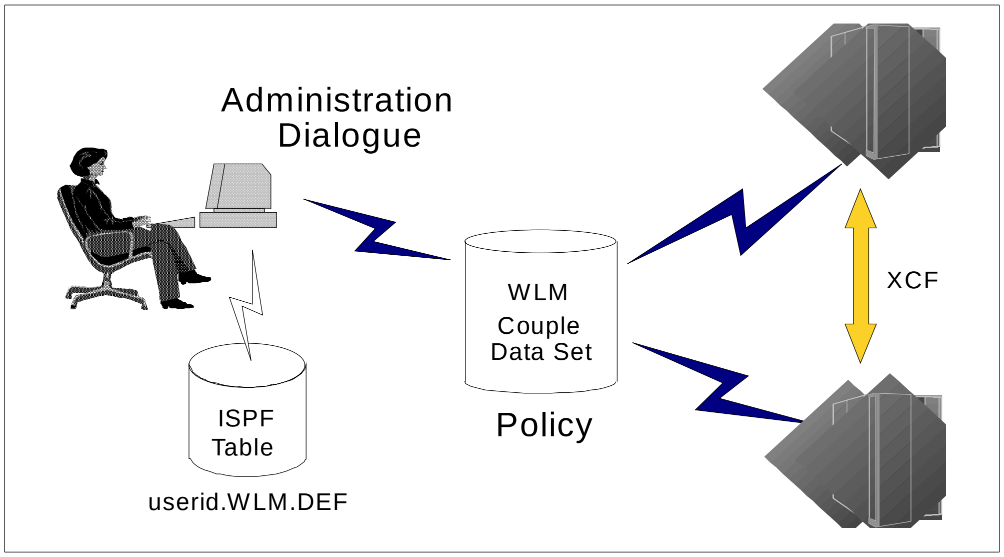


Figure 21. WLM ISPF Application to Create WLM Policy

To successfully manage WLM goal mode you need to safeguard the WLM ISPF application. The application is a vital component in communicating your requirements to WLM. You need to consider the following:

- Security access to the WLM application facilities should be protected using RACF; see 3.2.2.1, "Security Access" on page 96.
- You need usage rules to preserve data integrity, document changes and provide for recovery and backup; see 3.2.2.2, "Process Management Hints and Tips" on page 97.
- Provide the ISPF application as required; see 3.2.2.3, "ISPF Interface Implementation" on page 98.

3.2.2.1 Security Access

Before starting the WLM ISPF application, 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 capacity personnel. With UPDATE access, the user can:

- Perform all the functions available for READ access
- Allocate a WLM couple data set to the sysplex (use SETXCF command)
- Install a service definition to a WLM couple data set
- Activate a service policy

To control access to the application, use RACF to add a profile for the application to the RACF database. Then permit access to the application RACF profile. Do not forget to issue the SETROPTS REFRESH command after the PERMIT command to refresh the RACF profiles in storage.

Example of RDEFINE for the 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 PERMIT for 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.

3.2.2.2 Process Management Hints and Tips

You can store your service definition in either the WLM couple data set or an MVS PDS, or both. The WLM ISPF application is used to:

- Open/Save an ISPF partitioned data set containing a saved service definition
- Extract a Service Definition from the couple data set
- Install/Activate a definition/policy

You should establish operational procedures to:

- Keep a current copy of the production service definition in a RACF-protected PDS.
- Provide a methodology to keep backups, including the current RACF-protected PDS, and ensure data integrity. This is especially important if you have multiple UPDATE authorized users.
- Use the NOTEPAD to identify change history information. Add a comment in the NOTEPAD whenever making service definition changes, and log your install/activate dates.

Important

Sample Change Process

On system PRODQ, to update current policy CURPOL:

1. Enter WLM; Read Saved Definition from YOUR.CURRENT.GOAL.
2. Add line to NOTEPAD with change record, implementation date and time.
3. Implement Service Definition Change.
4. Use pull-down Utilities, then **Install Definition**. You will get a pop-up panel informing you that another service definition was found in the WLM couple data set; you overwrite the current service definition by responding `YES`.
5. Use pull-down File, then **Save** to update PDS YOUR.CURRENT.GOAL.
6. Use pull-down File, then **Save as** to create PDS YOUR.CURRENT.Date.
7. Exit WLM.

Notify operations to activate the policy using the following command:

```
V WLM,POLICY=CURPOL
```

This change management example provides a methodology to ensure that the latest service definition is always updated by using a pre-determined PDS as the single point of change implementation. The current definition is always kept in a specific PDS, which is always used as the starting definition for changes. The application administrator updates, installs, and saves the latest changes into the PDS. Also by using the PDS, multiple users cannot update the policy at the same time (update requires exclusive use of the PDS). The final action is to preserve copies of the service definition for contingency or backup usage.

Note that this example does not actually activate the updated service definition, but expects operations to perform this activity with a `V WLM,POLICY=xxxx` command.

Develop a service definition update methodology that addresses your change management requirements. Consider job responsibilities, data integrity issues (ensure policy changes are always made to the latest service definition), who needs access to service definition information, contingency and recovery procedures.

3.2.2.3 ISPF Interface Implementation

You can start the WLM ISPF application in compatibility mode or goal mode, and enter your service definition.

When you enter the service definition, keep the service definition in an MVS partitioned data set until you are ready to install the service definition into the WLM couple data set. If you are migrating to use a new version of the WLM application, always save the PDS created by the old version of the application for backup purposes.

To start the WLM application, you use the TSO/E REXX exec IWMARIN0. The exec concatenates (via LIBDEF and ALTLIB) the following libraries necessary to run the application:

Table 9. WLM ISPF Application Libraries

Library	Purpose
SYS1.SBLSCLI0	Application REXX code
SYS1.SBLSKEL0	Application skeleton
SYS1.SBLSPNL0	Application panels
SYS1.SBLSTBL0	Application keylists and commands
SYS1.SBLSMSG0	Application messages

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 *OS/390 MVS Planning: Workload Management*, GC28-1761.

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 on how to start the application specifying the WLM exits, see “Appendix A: Customizing the WLM ISPF Application” in *OS/390 MVS Planning: Workload Management*, GC28-1761.

3.2.3 Couple Data Set Management

The following sections describe the initial couple data set (CDS) management functions:

- 3.2.3.1, “Format Couple Data Set” on page 99 describes allocation of the CDS.
- 3.2.3.2, “Make a WLM CDS Available For the First Time” on page 101 provides instructions on adding the WLM CDS resource to the sysplex in parmlib and by operational commands.
- 3.2.3.3, “Make a Newly Formatted Couple Data Set Available” on page 102 discusses how to introduce a new CDS format into the sysplex where a WLM CDS already exists.

For more information on managing sysplex resources, see *OS/390 MVS Setting Up a Sysplex*, GC28-1779.

3.2.3.1 Format Couple Data Set

You can allocate the WLM primary and alternate couple data set from the WLM ISPF application, or by using the following sample JCL. A sample job is available in SYS1.SAMPLIB(IWMFTCDS). If you have a procedure for restricting access to your data sets, you can restrict access to the WLM couple data set just as you can any other data set.

All the following parameters are used for space calculations. You use a series of SETXCF commands to add a new, larger couple data set. Attempts to introduce a smaller CDS will result in XCF error message IXC250. Specify parameters based on your identified needs to XCF, then increase your CDS as required.

To define the CDS you will need update access to the WLM security access profile, see 3.2.2.1, “Security Access” on page 96. The following sample JCL provides a reasonable starting point, and is consistent with the samples used in 3.3, “Service Definition Implementation” on page 104. Review the service definition you developed using Chapter 4, “Migrating to WLM Goal Mode” on page 129 to customize for your needs. Add space entries for APPLENV and SCHENV if you plan on exploiting these facilities.

Remember to copy the JCL into your operational data set used to preserve system resource definition JCL:

```
//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(10)
      ITEM NAME(SRVCLASS) NUMBER(20)
```

Where:

SYSPLEX The name of your sysplex as it appears in your COUPLExx parmlib member.

DSN The name you are calling your WLM couple data set.

VOLSER A volume that you have access to. If you are using DFSMS, you do not need to specify a VOLSER.

TYPE The type of function for which this data set is allocated. For a service definition, the type is WLM.

ITEM NAME(POLICY) NUMBER( )

Specifies that an increment of space large enough to accommodate the specified number of policies be allocated in the WLM couple data set (default=5, minimum=1, maximum=99).

ITEM NAME(WORKLOAD) NUMBER( )

Specifies that an increment of space large enough to accommodate the specified number of workloads be allocated in the WLM couple data set (default=32, minimum=1, maximum=999).

ITEM NAME(SRVCLASS) NUMBER(   )

Specifies that an increment of space large enough to accommodate the specified number of service classes be allocated in the WLM couple data set (default=128, minimum=1, maximum=999).

Notes:

- As of OS/390 R3, WLM allows no more than 100 service classes to be defined in a service definition. The default, however, remains at the pre-OS/390 R3 value of 128. This will set aside as much space as you will ever need for service classes, as well as a little extra for other WLM objects. If you are migrating from a pre-OS/390 R3 system and you previously reserved space for more than 128 service classes, do the following:
  - If you intend to use SETXCF PSWITCH to make the new WLM couple data set active, continue to reserve space for the same number of service classes as you did in the earlier release. SETXCF PSWITCH will not allow a smaller WLM couple data set to be added as an alternate couple data set.
  - If you intend to IPL the sysplex with the new WLM couple data set and install the service definition from a saved ISPF data set, then reserve space for as many service classes as you expect to require.

Additional parameters SVDEFEXT, SVDCREXT, APPLENV, SVAEAEXT, SCHENV, and, SVSEAEXT are not shown in the sample. SVDEFEXT and SVDCREXT were introduced with MVS/ESA 5.2; APPLENV and SVAEAEXT with OS/390 R3; SCHENV and SVSEAEXT with OS/390 R4.

System management vendors can use SVDEFEXT, SVDCREXT, SVAEAEXT, and SVSEAEXT for their own unique information about workload definitions. The respective key words reserve space for extension areas to the WLM service definition, classification rules, application environment, and scheduling environment. The WLM interfaces allow these extensions to accompany the service class definitions, report class definitions, or even classification rules. The amount of extra information is specific to each product that exploits these interfaces; see the product’s documentation to ensure that there is sufficient space available in the WLM couple data set.

3.2.3.2 Make a WLM CDS Available For the First Time

You need to update the sysplex data set with an entry for the WLM couple data set. Issue the SETXCF command to initialize the entry, then update the COUPLExx parmlib member to include the WLM couple data set during subsequent IPLs.

For example, to make a primary WLM couple data set called SYS1.WLMCDS01 residing on volume TEMP01 available to the sysplex, enter the following command:

```
SETXCF COUPLE,TYPE=WLM,PCOUPLE=(SYS1.WLMCDS01,TEMP01)
```

To make an alternate WLM couple data set called SYS1.WLMCDS02 residing on volume TEMP02 available to the sysplex, enter the following command:

```
SETXCF COUPLE,TYPE=WLM,ACOUPLE=(SYS1.WLMCDS02,TEMP02)
```

To make the WLM couple data set available for use in the sysplex during each IPL, you need to update the DATA keyword in the COUPLExx parmlib member, and IPL with that COUPLExx member. As an example:

```
DATA    TYPE(WLM)
    PCOUPLE(SYS1.WLMCDS01,TEMP01)
    ACOUPLE(SYS1.WLMCDS02,TEMP02)
```

Where:

TYPE	The function type, WLM.
PCOUPLE(dataset.name,volume)	The primary WLM couple data set name, and the volume it resides on.
ACOUPLE(dataset.name,volume)	The alternate WLM couple data set name, and the volume it resides on. Always provide an alternate WLM couple data set for contingency.

Specify the modified COUPLExx member on your next IPL.

3.2.3.3 Make a Newly Formatted Couple Data Set Available

This section applies when you are already in goal mode (with existing WLM couple data sets), have just allocated new WLM CDSs, and want to make them available to the sysplex.

There are different reasons why you would want to do this. One reason is to increase the size of the WLM CDSs. Another reason is to migrate your WLM CDS to the new OS/390 format. You may also need to do this if your primary WLM CDS failed and the system automatically switched to the alternate, and now you need to introduce a new alternate.

You must use a series of SETXCF commands to switch from the currently active primary and alternate couple data sets to the new couple data sets. All systems in the sysplex then operate with the newly allocated data set.

If you are making OS/390 R4/R5 formatted WLM couple data sets available to the sysplex, you can continue to use a pre-OS/390 Release 4 WLM application to modify, install and activate your service definition, or you can switch to using the OS/390 Release 5 WLM application.

The following example describes how to make newly allocated CDSs available:

1. Allocate two new couple data sets. 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).

   Note: The new WLM CDS cannot be smaller than the current CDS; an attempt to switch to a smaller CDS will fail with an error message.

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, this causes the contents of the new primary WLM couple data set SYS1.WLMP to be copied to the new alternate SYS1.WLMA.

3.2.4 WLM MVS Operating Commands

The following is a summary of MVS Operating commands related to WLM management. Refer to *OS/390 MVS System Commands*, GC28-1781 for additional information. A general WLM design point is that the VARY commands have a *sysplex-wide scope*, and the MODIFY commands have a *single system scope*.

To set, or reset, the IPS and ICS in use (only valid in compatibility mode):

```
SET IPS=01,ICS=00
DISPLAY DMN
SETDMN
```

Display the WLM CDS in use, set primary and alternate CDS:

```
DISPLAY XCF,COUPLE,TYPE=WLM
SETXCF COUPLE,TYPE=WLM,PCOUPLE=(YOUR.WLM.PRIMARY)
SETXCF COUPLE,TYPE=WLM,ACOUPLE=(YOUR.WLM.ALTERNAT)
SETXCF COUPLE,TYPE=WLM,PSWITCH
```

Display the current policy use, status of a system, or all systems in the sysplex:

```
DISPLAY WLM
DISPLAY WLM,SYSTEMS
DISPLAY WLM,APPLENV=...SCHENV=...RESOURCE=....
```

Change the active WLM policy:

```
VARY WLM,POLICY=xxxx
```

Change between goal and compatibility mode on a system:

```
MODIFY WLM,MODE=GOAL|COMPAT
```

Manage application environments:

```
VARY WLM,APPLENV=xxxx,QUIESCE
VARY WLM,APPLENV=xxxx,RESUME
VARY WLM,APPLENV=xxxx,REFRESH
```

Manage RESOURCE usage on a system; used for affinity scheduling:

```
MODIFY WLM,RESOURCE=xxxx,ON
MODIFY WLM,RESOURCE=xxxx,OFF
MODIFY WLM,RESOURCE=xxxx,RESET
```

Change between goal and compatibility mode; reset RESOURCE usage on a system:

```
MODIFY WLM,MODE=GOAL|COMPAT
MODIFY WLM,RESOURCE=xxxx,ON|OFF|RESET
```

Reset the classification of jobs:

```
RESET jobname,A=asid,SRVCLASS=xxxx
RESET jobname,A=asid,RESUME
RESET jobname,A=asid,QUIESCE
RESET jobname,A=asid,PERFORM=nnn
```

3.3 Service Definition Implementation

Before going into the application you should have already planned your service definition. You should find it helpful to use the worksheets in Appendix A, “Sample Worksheets” on page 213, or use the sample worksheets as a model for your own worksheets. Chapter 4, “Migrating to WLM Goal Mode” on page 129 and Chapter 5, “Considerations for New Workloads” on page 171 guide you through the specific contents for your new service definition.

Your service definition, illustrated in Figure 22, will be composed of:

- One or more service policies
- Classification rules (one set per service definition, which categorize work into service classes, and optional report classes)
- Workloads (JES, TSO, CICS, IMS, and so on), which are arbitrary groups/ collections for reporting purposes
- Service classes (one or more per workload; for each service class there is a service goal and importance)
- Resource groups (optional, associated with service class)
- Application environments (optional)
- Scheduling environments (optional)

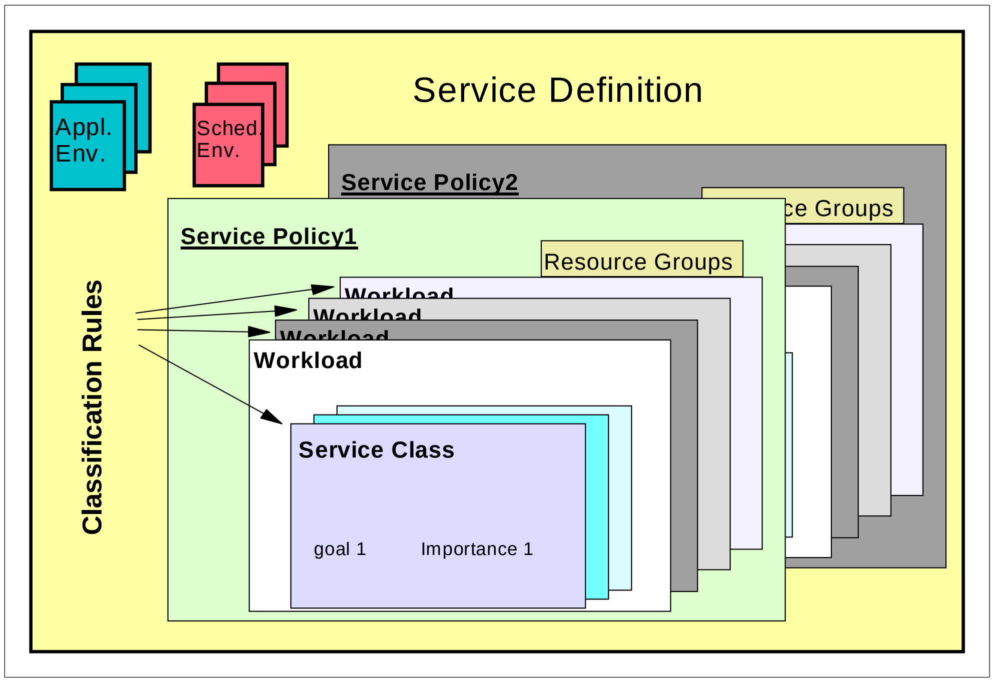


Figure 22. Service Definition Components

Preparing a service definition requires you to:

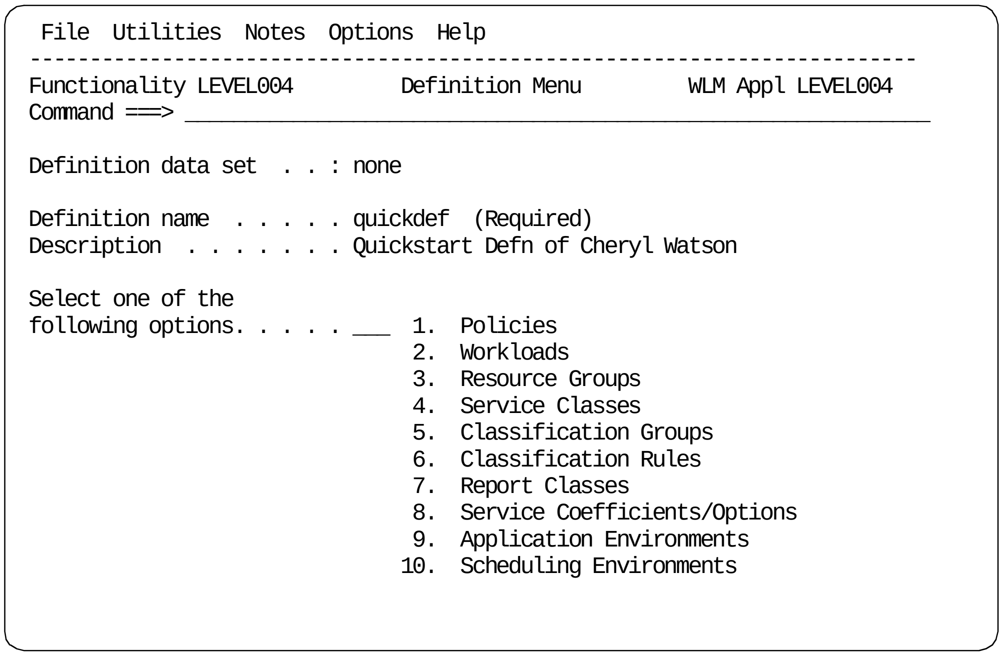
- Understand your workload
- Set service classes for major workloads
- Set classification rules to get work into the correct service classes

Use your ICS/IPS configuration as a model. This may be an opportune time for a careful review of workloads and business needs priorities. It is also highly recommended to plan on *reducing* the number of service class categories compared to the number of performance groups. Less systems programmer fine-tuning is required with WLM goal mode.

In this section we use Cheryl Watson's Quickstart Service Policy, which originally appeared in Cheryl Watson's Tuning Letter, May/June 1995. It appears here with the permission and cooperation of Watson & Walker, Copyright Watson & Walker, Inc. 1995, 1996. A full description, and the latest version of Cheryl Watsons Quickstart Service definition may be found at: `http://www.watsonwalker.com/quickst.html`

3.3.1 WLM ISPF Application Overview

The definition menu is the central place for entering your service definition, as shown in Figure 23.



```
 File  Utilities  Notes  Options  Help
 ---------------------------------------------------------------------------
Functionality LEVEL004        Definition Menu        WLM Appl LEVEL004
Command ===> ______________________________________________________________

Definition data set  . . : none

Definition name  . . . . . quickdef  (Required)
Description  . . . . . . . Quickstart Defn of Cheryl Watson

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 23. WLM ISPF Cockpit Definition Menu

Most panels have a menu bar, action field, status line, scrollable area, function key area, and command line. We assume that you are familiar with ISPF facilities, but if you need extra guidance refer to Chapter 16 in *OS/390 MVS Planning: Workload Management*, GC28-1761. Remember to use on-line help if you need more information while using the application. Many fields support prompting; enter a question mark (?) in the field, and a pop-up menu will appear with the available choices.

When typing data, you can typically choose to either create or copy an existing entry, according to your personal preference.

You do not need to enter your service definition in any order. It is easiest, if you are entering a new definition, to follow the order displayed on the definition menu.

Information usage will sometimes dictate the input order; for example, service class needs to specify workloads that must have been predefined.

After each activity input function, you return to the main definition panel.To enter your service definition, do the following:

1. Access the WLM ISPF application (see 3.3.2, “Specifying Your Service Definition” on page 107, choosing to create a new definition).
2. Input your service definition name and description (see 3.3.3, “Main Definition Menu” on page 108).
3. You can follow the sequence of the options on the main definition menu. The *service coefficients/options* are unique parameters that do not depend on any other service definition information, so they can be input any time. Enter them now by choosing option **8**; see Figure 26 on page 109.
4. Now choose option **1** ***Policies*** (see 3.3.4, “Service Policy Definition” on page 109). Make sure you have meaningful policy names since operations will need to clearly understand what policies are being used, and when.
5. *Workloads* are the building blocks of the service policy. Enter your workloads by choosing option **2** (see 3.3.5, “Workload Specification” on page 110).
6. If your service definition includes *resource groups*, create them by selecting option **3** (see 3.3.6, “Resource Group Specification” on page 111).
7. You are now ready to enter the information which has probably taken the most amount of research time, the *service classes*. Select option **4**, then input your service classes. See 3.3.7, “Working with Service Classes” on page 112 for an example that shows the options and pop-up windows. After building the first service class, you can choose to copy and modify for other definitions (action **2**) or you can use create (action **1**).
8. Use *Classification Groups* to group together work qualifiers to make classification simpler. Use option **5** to enter the groups (see 3.3.8, “Classification Groups” on page 115 for an example).
9. *Classification Rules* define rules to categorize work into service classes, and optionally report classes, based on work qualifiers. A work qualifier is what identifies a work request to the system. You should ensure that every service class has a corresponding rule (see 3.3.9, “Classification Rules” on page 117 for an explanation regarding the usage of option **6** to enter the rules).
10. *Report classes* are optional (3.3.10, “Report Class” on page 120 demonstrates using option **7** to specify a report class).
11. If you have planned *application environments* to support some of the new exploiters, choose option **9** (see 3.3.11, “Application Environment Specification” on page 120 for additional information).
12. Similarly if you need to define your *scheduling environments*, choose option **10** and refer to 3.3.12, “Create a Scheduling Environment” on page 121.
13. Now you have finished inputting your service definition, as outlined in step 8 on page 91. Step 9 on page 91 directs you to install/activate the service definition. (See 3.3.14, “Installing and Activating Your Service Definition” on page 125.)
14. If you have not already saved the service definition into a PDS, you will want to do this when you exit. Pressing **PF3** to exit will prompt you for the data set

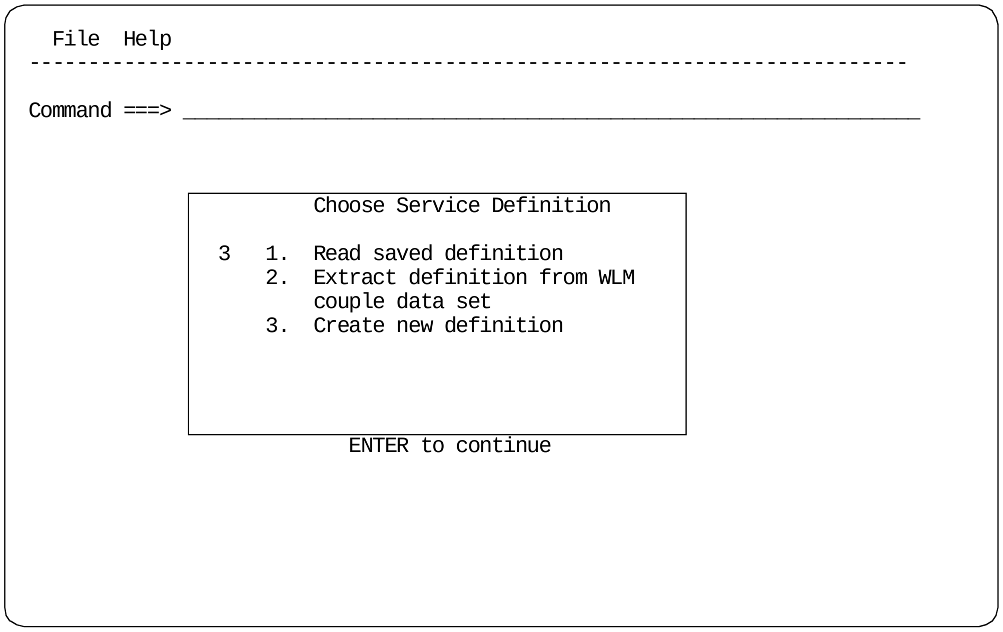
name. (See 3.3.15, “Service Definition File Options” on page 127 for additional information.)

As you complete the service definition activities, remember to follow the change management processes you established in 3.2.2.2, “Process Management Hints and Tips” on page 97.

Congratulations, you have now completed entering you service definition. See step 3 on page 90, for the final steps to implement WLM.

3.3.2 Specifying Your Service Definition

As you enter the WLM application you must specify where you would like WLM to read a prior definition from, or whether you want to create a new definition. The first time you need to create a new service definition, choose option **3** on the Choose Service Definition pop-up menu as illustrated in Figure 24.



```
  File  Help
 --------------------------------------------------------------------------

 Command ===> ______________________________________________________________



                         Choose Service Definition

                3   1.  Read saved definition
                    2.  Extract definition from WLM
                        couple data set
                    3.  Create new definition




                            ENTER to continue
```

Figure 24. Choose Service Definition Panel

If you need to stop updating the service definition, press **PF3**. The PF3 save function saves the tables into a PDS and ends the WLM session. The next time you start the WLM application, choose **Read saved definition** and then specify the PDS data set name. Remember to follow the procedures you implemented in 3.2.2.2, “Process Management Hints and Tips” on page 97 to avoid regressing any changes.

3.3.3 Main Definition Menu

Now is the time to enter the information that you placed in the service definition worksheet, as shown in Table 10.

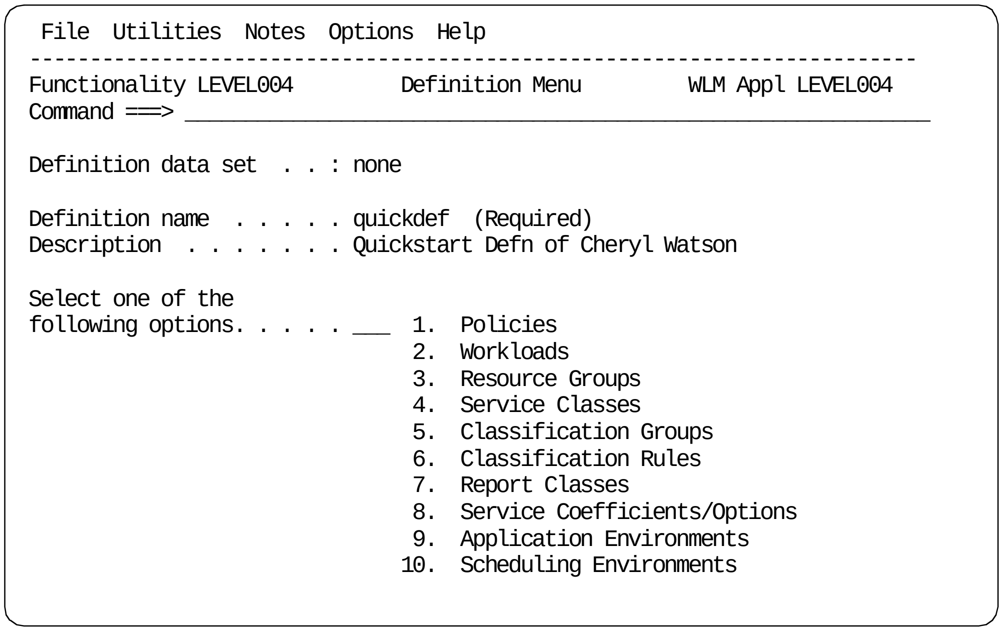
Table 10. QUICKDEF Sample Service Definition

Service Definition Name	Description	Coefficients
QUICKDEF	Quickstart - Cheryl Watson	CPU: 1.0, IOC: 0.1, MSO: 0, SRB: 1.0 IO Priority Management: NO

The definition menu is the central place for entering your service definition. Your service definition name and description are to be entered on the main definition menu panel (see Figure 25).

The definition data set field will be none until you save the service definition into a data set. You specify the data set to save into when exiting the application, or you can issue a save at any time by using the File command bar options; see 3.3.15, "Service Definition File Options" on page 127 for an explanation.

The WLM ISPF application will dynamically allocate the PDS save data set if you specify a data set which does not already exist. Also, the application maintains a copy of PDS names used, and will display this list when you type a question mark (?) prompt into the data set name field.



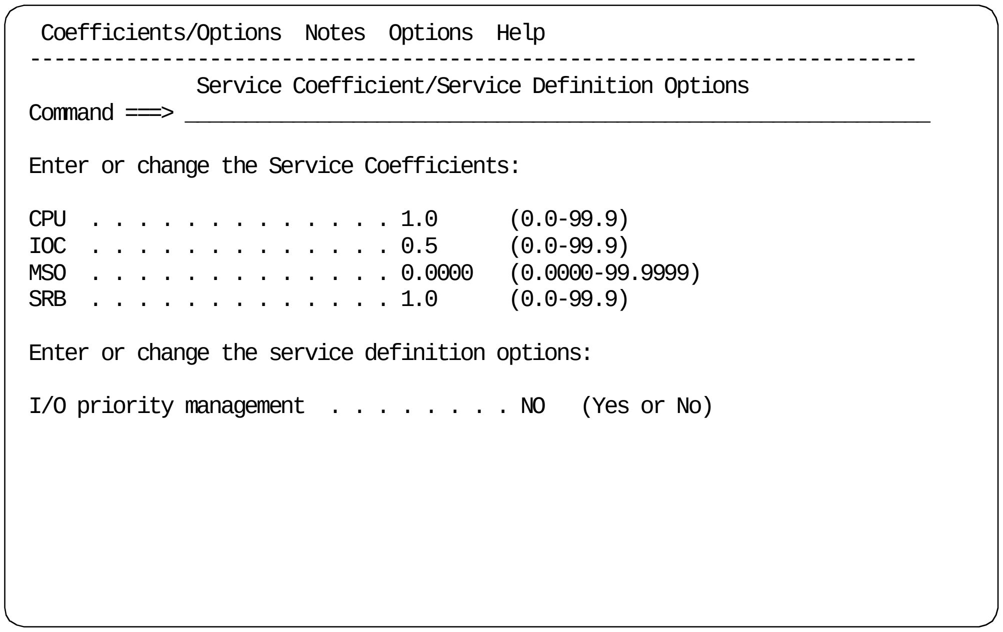
```
 File  Utilities  Notes  Options  Help
--------------------------------------------------------------------------
Functionality LEVEL004        Definition Menu        WLM Appl LEVEL004
Command ===> ______________________________________________________________

Definition data set  . . : none

Definition name  . . . . . quickdef  (Required)
Description  . . . . . . . Quickstart Defn of Cheryl Watson

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 25. Definition Menu Panel

Service coefficients define the weight to be applied to one type of service over another in the calculation of service rates. You can enter new values for the CPU, IOC, MSO, and SRB service coefficients. The WLM application provides the opportunity to change these coefficients. Choosing low numbers will reduce the magnitude of the service units reported. For more information, see 4.2.6, "Service Definition Coefficients" on page 136

There is an additional option on this panel for I/O priority management. The default is no, meaning I/O priorities are to be the same as dispatching priorities. Specifying yes means I/O priorities should be managed separately from dispatching priorities, according to the goals of the work. Note that this option affects the way execution velocities are calculated. With I/O priority management set to yes, the velocity calculation includes I/O using and delay values.

Choose option **8** from the main panel (Figure 25) to bring up the panel shown in Figure 26 on page 109 for you to specify the coefficients/options shown in Table 10.



```
 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)
```

Figure 26. Service Coefficient/Options Panel

3.3.4 Service Policy Definition

The next step is to define your service policy or policies.

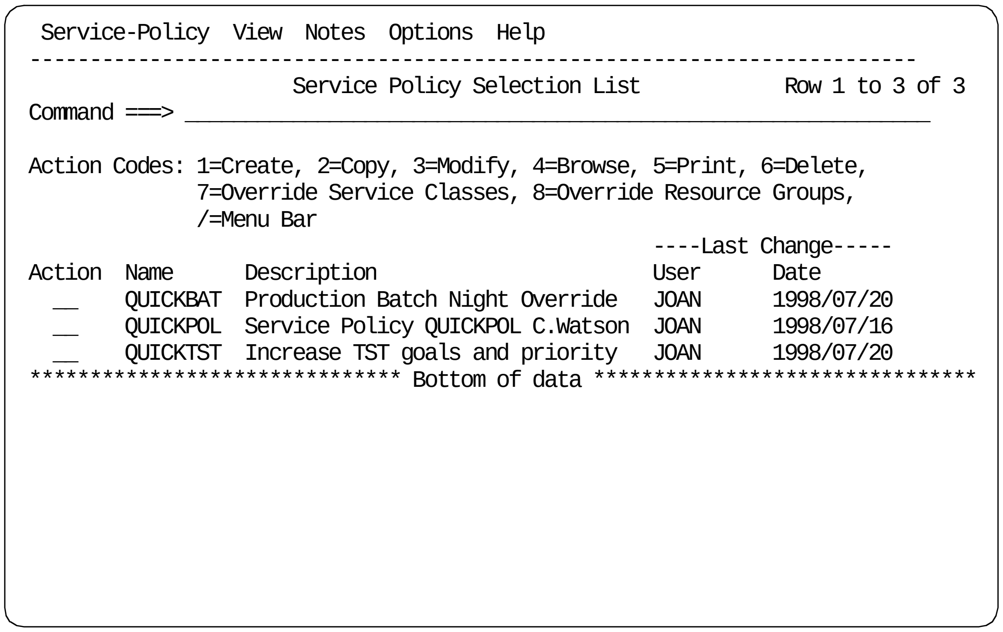
You will need to provide a name and description for each policy, including any policy overrides. The first time you set up a service definition, define a policy name and description. If you do not have a business need to change your goals, you can run with one service policy, without any policy overrides.

You should have outlined your service policy names on a worksheet similar to Table 11. In this example, QUICKPOL is the base policy definition; QUICKBAT and QUICKTST are override policies. See 3.3.13, “Policy Overrides” on page 123 for an example of defining policy override.

Table 11. Sample Service Policy Definitions

Service Policy Name	Description	SRVCLASS or RG Overrides
QUICKPOL	Service Policy QUICKPOL C.Watson	
QUICKBAT	Production Batch Night Override	SRVCLASS: PRDBATHI
QUICKTST	This is a copy from QUICKPOL	SRVCLASS: TSTBATHI, TSTBATMD

Figure 27 on page 110 shows the service policy definitions after completion. Note that the QUICKBAT and QUICKTST were defined several days after QUICKPOL.



```
 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
  __    QUICKBAT  Production Batch Night Override   JOAN      1998/07/20
  __    QUICKPOL  Service Policy QUICKPOL C.Watson  JOAN      1998/07/16
  __    QUICKTST  Increase TST goals and priority   JOAN      1998/07/20
******************************* Bottom of data ********************************
```

Figure 27. Service Policy Selection List Panel

3.3.5 Workload Specification

A workload logically consists of a group of one or more service classes. You associate a workload with a service class in the Service Class panel. Enter your workloads before creating your service classes.

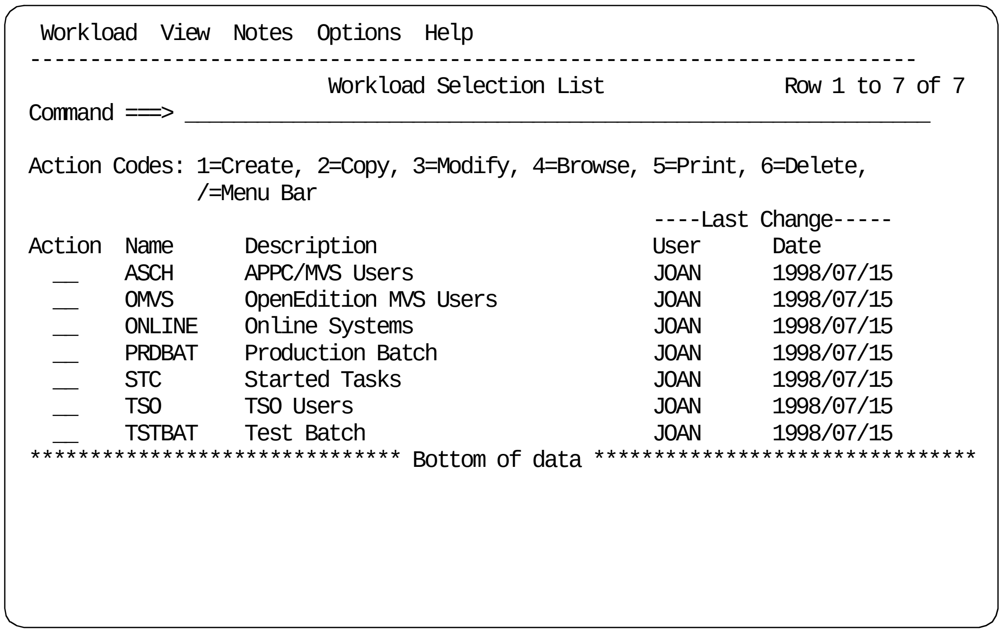
Note: A workload is only used as a grouping of service classes for reporting purposes. It does not affect workload management.

From your worksheets (see Table 12), you need to input your workloads and their descriptions.

Table 12. Sample Workload Classification

Workload Name	Description
ASCH	APPC/MVS Users
OMVS	OpenEdition MVS Users
ONLINE	Online Systems
PRDBAT	Production Batch
STC	Started Tasks
TSO	TSO Users
TSTBAT	Test Batch

Workload class descriptions are entered by using option **2**. Workloads must be defined before they can be used in service class definitions. Figure 28 on page 111 shows the workload selection list panel.



```
 Workload  View  Notes  Options  Help
--------------------------------------------------------------------------
                         Workload Selection List             Row 1 to 7 of 7
Command ===> ______________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar
                                                      ----Last Change-----
Action  Name      Description                         User      Date
  __    ASCH      APPC/MVS Users                      JOAN      1998/07/15
  __    OMVS      OpenEdition MVS Users               JOAN      1998/07/15
  __    ONLINE    Online Systems                      JOAN      1998/07/15
  __    PRDBAT    Production Batch                    JOAN      1998/07/15
  __    STC       Started Tasks                       JOAN      1998/07/15
  __    TSO       TSO Users                           JOAN      1998/07/15
  __    TSTBAT    Test Batch                          JOAN      1998/07/15
******************************* Bottom of data ********************************
```

Figure 28. Workload Selection List Panel

3.3.6 Resource Group Specification

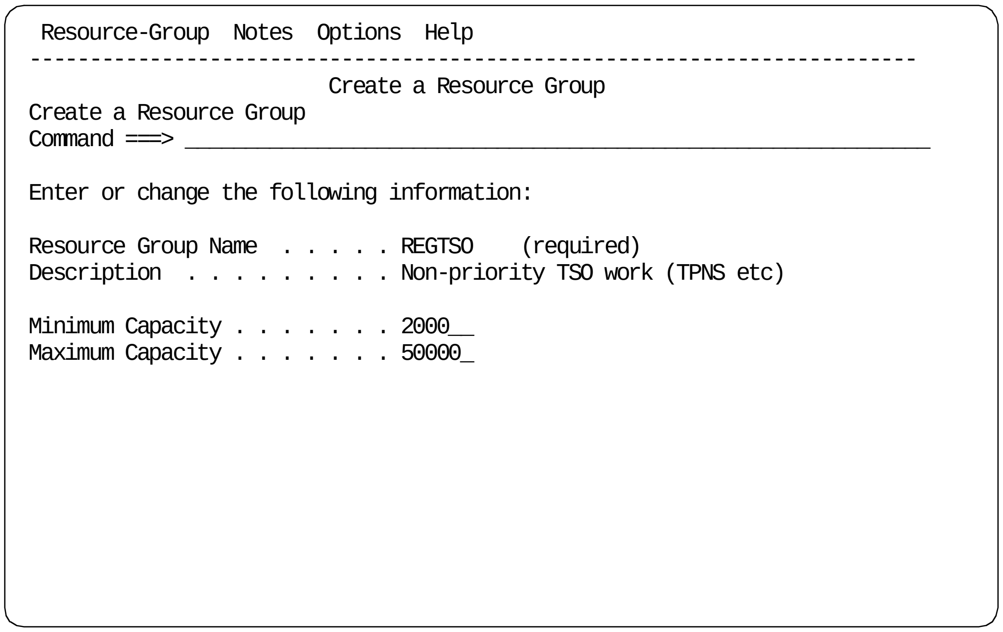
A *resource group* is a minimum or maximum amount of processing capacity. You associate a resource group with a service class in the Service Class panel. A resource group can be associated with multiple service classes. Enter resource groups *before* creating your service classes.

Our QUICKPOL does not have any resource groups defined to it. The following two examples are resource groups defined in the WEB_GOAL service definition referred to in 5.3.3, “Define Your WLM Service Classes for the Scalable Web Server” on page 191. You should have used a worksheet similar to Table 13 if you plan to use resource groups.

Table 13. Sample Resource Group Definition

Resource Group Name (RG)	Description	Min	Max
REGTSO	Non-priority TSO work (TPNS and so on)	2000	50000
RGROUP2	Resource group 2	100	

Enter resource groups using option **3**, which is shown in Figure 29 on page 112.



```
 Resource-Group  Notes  Options  Help
 ---------------------------------------------------------------------------
                          Create a Resource Group
 Create a Resource Group
 Command ===> ______________________________________________________________

 Enter or change the following information:

 Resource Group Name  . . . . . REGTSO    (required)
 Description  . . . . . . . . . Non-priority TSO work (TPNS etc)

 Minimum Capacity . . . . . . . 2000__
 Maximum Capacity . . . . . . . 50000_
```

Figure 29. Create a Resource Group Panel

3.3.7 Working with Service Classes

A *service class* is a group of work with similar performance goals, resource requirements, or business importance. You make the association between service class, workload and resource groups in the Service Class panel. You associate a service class with incoming work in the Classification Rules panel. Make sure you enter service classes before creating classification rules.

After all your preparation and planning, you will have created a worksheet similar to Table 14 on page 113 which describes your service classes.

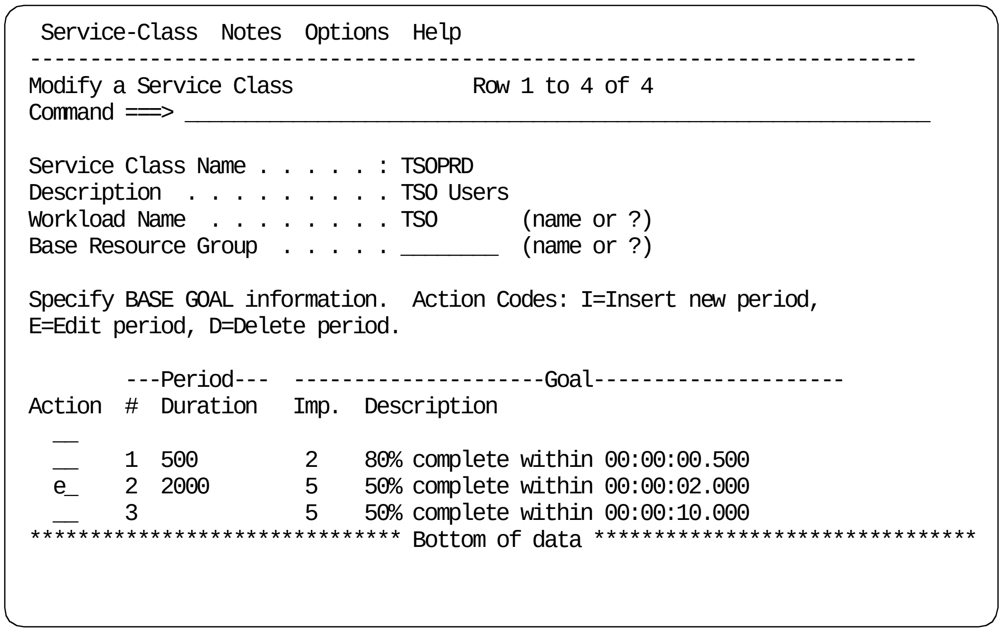
Table 14. Sample Service Class Definition

SRVCLASS	Description	Workload	RG	Per	Dur.	Imp.	Goal
ASCH	APPC/MVS users	ASCH		1	500	2	80% 00:00:00.50
				2		4	Velocity 30
OMVS	OpenEdition MVS users	OMVS		1	500	2	80% 00:00:00.50
				2		4	Velocity 30
ONLPRD	Online production	ONLINE		1		1	Velocity 50
ONLPRDHI	Online production high	ONLINE		1		1	80% 00:00:00.50
ONLPRDLO	Online production low	ONLINE		1		3	50% 00:00:10.00
ONLPRDMD	Online production medium	ONLINE		1		2	80% 00:00:03.00
ONLTST	Online test regions	ONLINE		1			Discretionary
PRDBATHI	Production batch high	PRDBAT		1		2	Velocity 30
PRDBATLO	Production batch low	PRDBAT		1			Discretionary
STCLO	STC low	STC		1			Discretionary
STCMD	STC medium	STC		1		3	Velocity 30
TSOPRD	TSO users	TSO		1	500	2	80% 00:00:00.50
				2	2000	3	80% 00:00:02.00
				3		5	50% 00:10:00.00
TSTBATHI	Test batch high	TSTBAT		1		3	90% 00:10:00.00
TSTBATLO	Test batch low	TSTBAT		1			Discretionary
TSTBATMD	Test batch medium	TSTBAT		1		4	80% 00:30:00.00

Note: Workload specification must be completed prior to using option 3 for service classes definition.

Entering the service classes using main definition panel option **4**, is a relatively minor task (compared to the planning and preparation). Period and duration specification is similar to IEAIPS PGN periods. While you are using the service class input panel, shown in Figure 30 on page 114, a pop-up window will be presented for you to choose one of four types of goal:

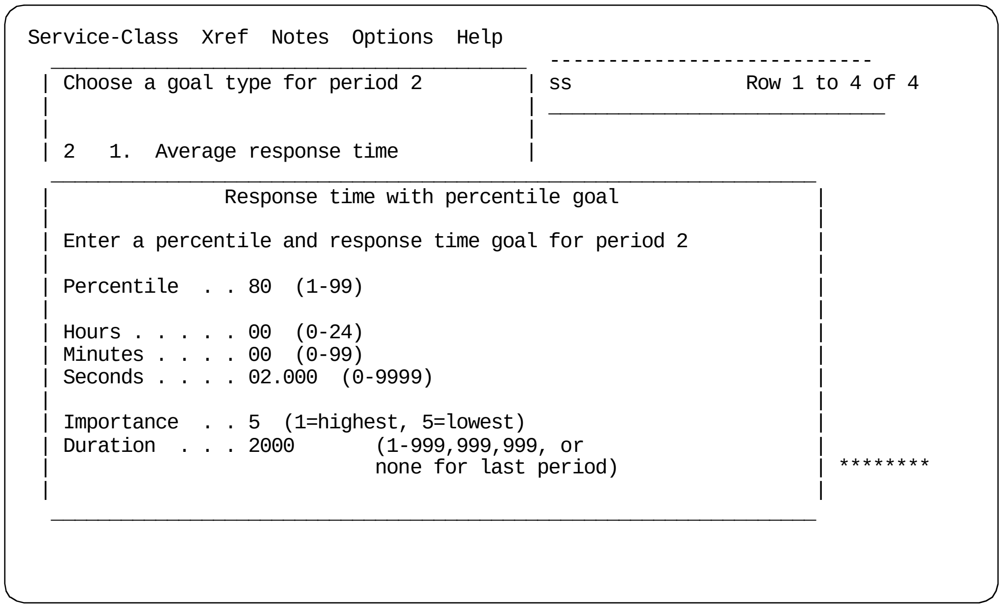
- Average response time
- Response time with percentile
- Execution velocity
- Discretionary



```
 Service-Class  Notes  Options  Help
 --------------------------------------------------------------------------
 Modify a Service Class              Row 1 to 4 of 4
 Command ===> ______________________________________________________________

 Service Class Name . . . . . : TSOPRD
 Description  . . . . . . . . . TSO Users
 Workload Name  . . . . . . . . TSO       (name or ?)
 Base Resource Group  . . . . . ________  (name or ?)

 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 00:00:00.500
   e_    2  2000        5    50% complete within 00:00:02.000
   __    3              5    50% complete within 00:00:10.000
 ***************************** Bottom of data ********************************
```

Figure 30. Create a Service Class Panel

Figure 31 shows you the stacked windows that will be displayed as you either enter or update the goals. In this case a typing error was made inputting the percentile, and needed to be fixed.



```
 Service-Class  Xref  Notes  Options  Help
  _________________________________________  ----------------------------
  | Choose a goal type for period 2         | ss              Row 1 to 4 of 4
  |                                         | _____________________________
  |                                         |
  | 2   1.  Average response time           |
  ___________________________________________________________________
  |              Response time with percentile goal                 |
  |                                                                 |
  | Enter a percentile and response time goal for period 2          |
  |                                                                 |
  | Percentile  . . 80  (1-99)                                      |
  |                                                                 |
  | Hours . . . . . 00  (0-24)                                      |
  | Minutes . . . . 00  (0-99)                                      |
  | Seconds . . . . 02.000  (0-9999)                                |
  |                                                                 |
  | Importance  . . 5  (1=highest, 5=lowest)                        |
  | Duration  . . . 2000       (1-999,999,999, or                   |
  |                            none for last period)                | ********
  |                                                                 |
  ___________________________________________________________________
```

Figure 31. Update a Service Class Pop-Up Panel

When you have completed entering your service classes, you can create any policy overrides that might be required. See 3.3.13, “Policy Overrides” on page 123 for additional information.

3.3.8 Classification Groups

Classification groups are optional. You use groups to simplify classification. You associate a classification group with a service class in the classification rules panel. If you intend to use them, create groups *before* creating classification rules.

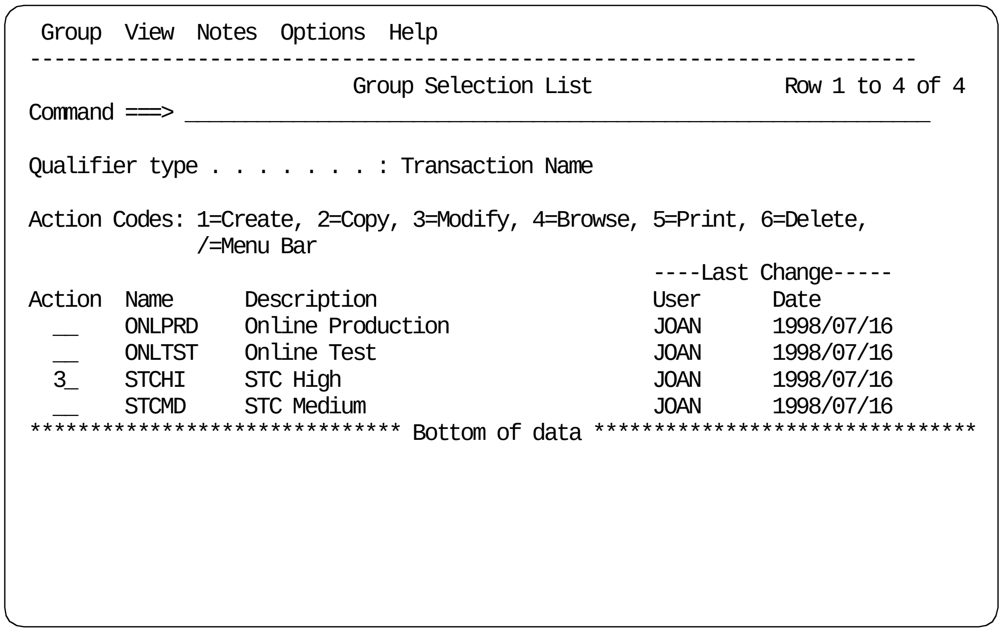
Here, we have chosen to classify as much work as possible into groups; see Table 15. Qualifier groups of more than five members are quicker to check than single instances in the classification rules.

Note: An asterisk (*) represents the use of wildcards.

Table 15. Sample Classification Groups

Type	Group Name	Description	Qualifier	Description
TCG	PRDBATHI	Production batch class group	Q	Payroll Reserved Initiator
TCG	TSTBATHI	Hot test batch group	H	Emergency test batch
TCG	TSTBATLO	Low test batch group	A	IOU department batch
TCG	TSTBATMD	Medium test batch group	M	Medium test batch
TNG	ONLPRD	On-line production regions group	CICSP*	Production CICS
			IMSP*	Production IMS
			DB2P*	Production DB2
			ROSCOE	ROSCOE editor
			WYLBUR	WYLBUR editor
TNG	ONLTST	On-line test regions group	CICST*	Test CICS
			IMST*	Test IMS
			DB2T*	Test DB2
TNG	STCHI	High STCs	NPM RMF JES* APPC ASCH OMVS AOPS DLF IRLM* LLA MIM OMON* OMVSKERN PCAUTH RACF SMS SYSBMAS TRACE TSO VLF VTAM	
TNG	STCMD	Medium STCs	SCHED*	
			SPOOLPGM	
			PRINTMGT	
			OPS_JOBS	

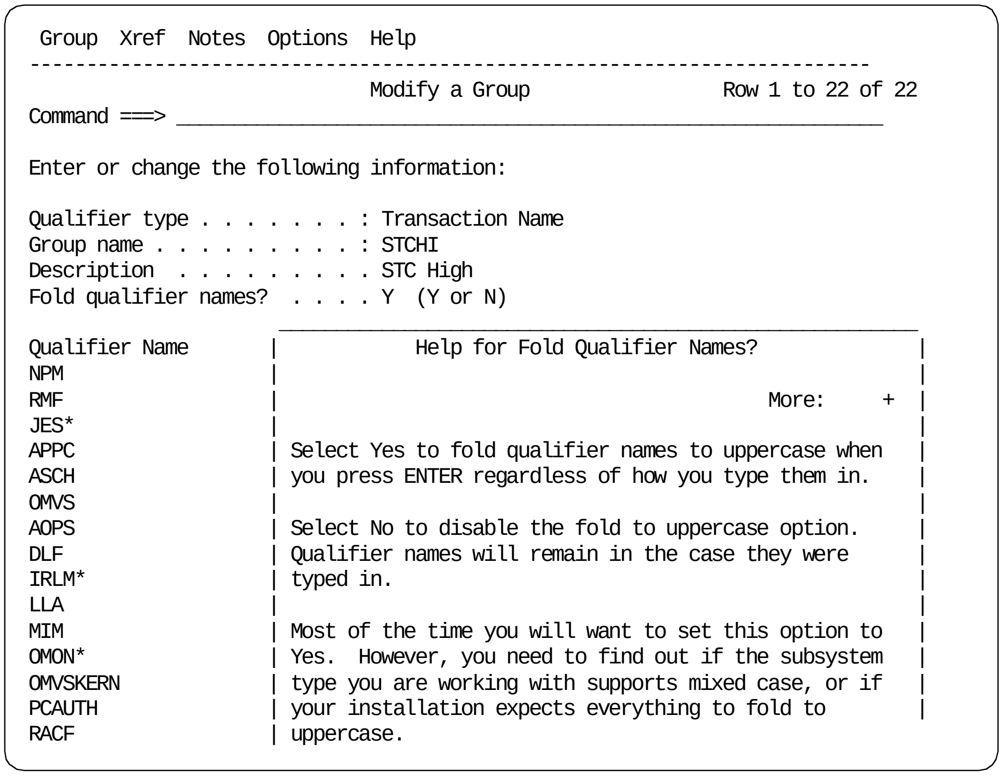
Enter classification groups using option **5**. Figure 32 on page 116 shows you the transaction name classification groups.



```
 Group  View  Notes  Options  Help
--------------------------------------------------------------------------
                           Group Selection List              Row 1 to 4 of 4
Command ===> ______________________________________________________________

Qualifier type . . . . . . . : Transaction Name

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar
                                                    ----Last Change-----
Action  Name      Description                       User      Date
  __    ONLPRD    Online Production                 JOAN      1998/07/16
  __    ONLTST    Online Test                       JOAN      1998/07/16
  3_    STCHI     STC High                          JOAN      1998/07/16
  __    STCMD     STC Medium                        JOAN      1998/07/16
******************************* Bottom of data ********************************
```

Figure 32. Group Selection List Panel

Figure 33 demonstrates the modify group panel. To get information about a field, use the cursor sensitive help function. Place your cursor over the data field, and press **PF1** for help. The fold qualifier name PF1 result is shown in the Figure 33 pop-up. `More: +` indicates there is additional information that can be displayed by scrolling forward (**PF8**).



```
 Group  Xref  Notes  Options  Help
----------------------------------------------------------------------
                           Modify a Group              Row 1 to 22 of 22
Command ===> ____________________________________________________________

Enter or change the following information:

Qualifier type . . . . . . . : Transaction Name
Group name . . . . . . . . . : STCHI
Description  . . . . . . . . . STC High
Fold qualifier names?  . . . . Y  (Y or N)
                     ______________________________________________________
Qualifier Name      |           Help for Fold Qualifier Names?              |
NPM                 |                                                       |
RMF                 |                                          More:     +  |
JES*                |                                                       |
APPC                | Select Yes to fold qualifier names to uppercase when  |
ASCH                | you press ENTER regardless of how you type them in.   |
OMVS                |                                                       |
AOPS                | Select No to disable the fold to uppercase option.    |
DLF                 | Qualifier names will remain in the case they were     |
IRLM*               | typed in.                                             |
LLA                 |                                                       |
MIM                 | Most of the time you will want to set this option to  |
OMON*               | Yes.  However, you need to find out if the subsystem  |
OMVSKERN            | type you are working with supports mixed case, or if  |
PCAUTH              | your installation expects everything to fold to       |
RACF                | uppercase.
```

Figure 33. Modify a Group Panel

3.3.9 Classification Rules

Classification rules assign incoming work to service classes. Before you create your classification rules, you must understand which subsystem's work is represented in each of your service classes.

When you choose the option **Classification Rules**, you go to the Subsystem Type Selection List for Rules. This selection list is primed with all of the IBM-supplied subsystem types. They are reserved names.

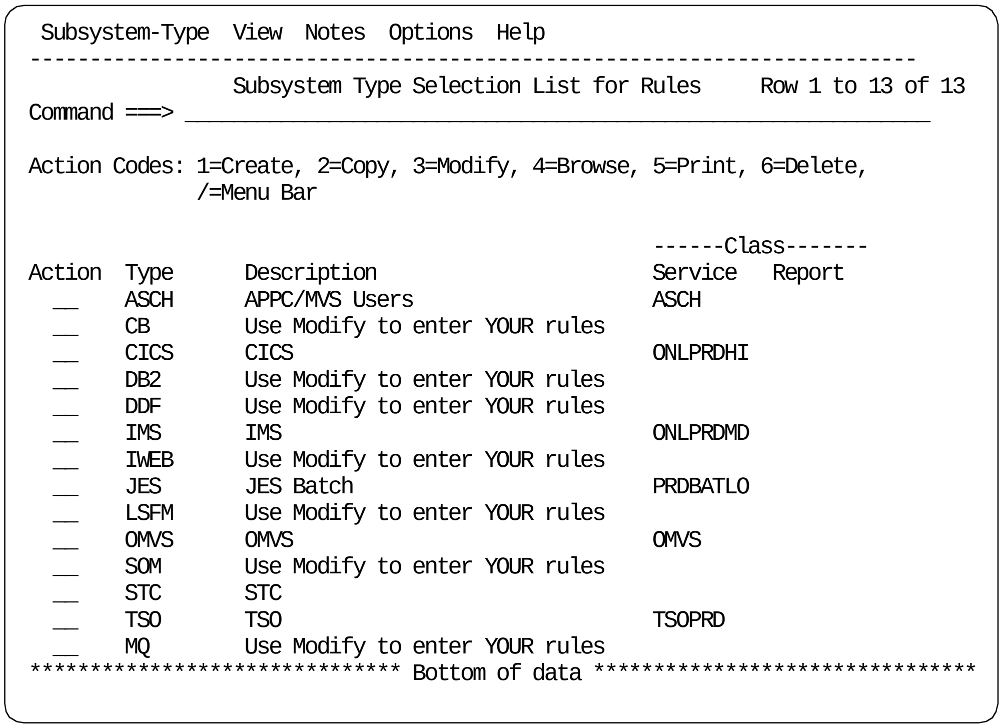
You should have created a worksheet similar to Table 16 on page 118 which describes your classification rules. Remember, there is only *one* set of classification rules in the service definition for a sysplex. They are the same regardless of which service policy is in effect; a policy cannot override classification rules.

Note: An asterisk (*) represents the use of wildcards.

Table 16. Sample Classification Rules Definition

Subsystem	Description	Leve l	Typ e	Name	Start	SRVCLASS	RPTCLASS	Description
ASCH	APPC/MVS			Default		ASCH		All APPC
CB								
CICS	CICS			Default		ONLPRDHI		Prod CICS
DB2								
DDF								
IMS	IMS			Default		ONLPRDMD		Prod IMS
IWEB								
JES	JES Batch			Default		PRDBATLO	PRDBAT	
		1	TCG	PRDBATHI		PRDBATHI	PRDBAT	
		1	TCG	TSTBATHI		TSTBATHI		
		1	TCG	TSTBATM D		TSTBATMD		
		1	TCG	TSTBATLO		TSTBATLO		
LSFM								
OMVS	OpenEdition MVS			Default		OMVS		All OMVS
SOM								
STC	Started Tasks			Default		SYSSTC		
		1	TNG	STCHI		SYSSTC		
		1	TNG	STCMD		STCMD		
		1	TNG	ONLPRD		ONLPRD		
		1	TNG	ONLTST		ONLTST		
		1	TN	*		STCLO		
TSO	TSO			Default		TSOPRD		
RYO	Roll Your Own 1			Default		ONLPRDLO		
		1	AI	DEPT12*	5	ONLPRDLO		
		2	PR	PRNAME		ONLPRDMD		
		3	NET	NETN*		ONLPRDHI		
		2	PR	PRNAM2		ONLPRDHI		

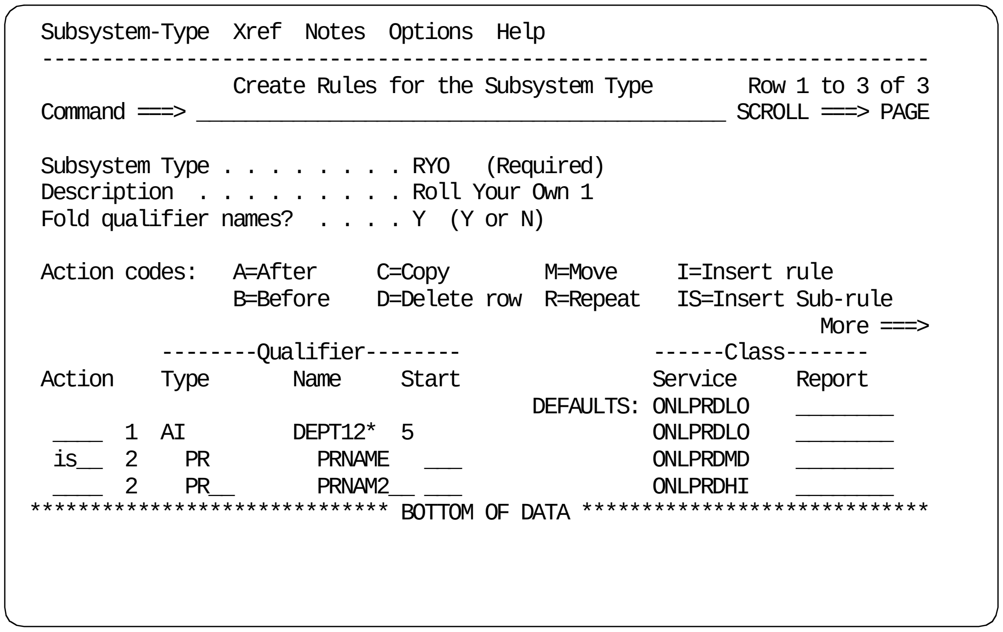
Figure 34 on page 119 shows the subsystem type selection list of rules panel. This is the main panel for defining classification rules. Do not delete any of the entries provided by IBM, even subsystems which you do not currently use, because you may to need to use them in the future.



```
 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
  __    ASCH      APPC/MVS Users                    ASCH
  __    CB        Use Modify to enter YOUR rules
  __    CICS      CICS                              ONLPRDHI
  __    DB2       Use Modify to enter YOUR rules
  __    DDF       Use Modify to enter YOUR rules
  __    IMS       IMS                               ONLPRDMD
  __    IWEB      Use Modify to enter YOUR rules
  __    JES       JES Batch                         PRDBATLO
  __    LSFM      Use Modify to enter YOUR rules
  __    OMVS      OMVS                              OMVS
  __    SOM       Use Modify to enter YOUR rules
  __    STC       STC
  __    TSO       TSO                               TSOPRD
  __    MQ        Use Modify to enter YOUR rules
******************************* Bottom of data ********************************
```

Figure 34. Subsystem Type Selection List of Rules Panel

Figure 34 shows that you have completed entering the IBM reserved subsystem workloads, and need to create RYO. Figure 35 shows the entry for RYO. Note the `More ===>` indication; this tells you that you can scroll right to enter descriptions.



```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Create Rules for the Subsystem Type       Row 1 to 3 of 3
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . . . . . . . . RYO   (Required)
 Description  . . . . . . . . . Roll Your Own 1
 Fold qualifier names?  . . . . Y  (Y or N)

 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: ONLPRDLO    ________
  ____  1  AI         DEPT12*  5                   ONLPRDLO    ________
  is__  2    PR         PRNAME   ___               ONLPRDMD    ________
  ____  2    PR__       PRNAM2__ ___               ONLPRDHI    ________
****************************** BOTTOM OF DATA ******************************
```

Figure 35. Classification Rules for Roll Your Own (RYO)

3.3.10 Report Class

A *report class* is a group of work for which you want reporting data. You do not have to define report classes before assigning them to work in classification rules. You can create them from within the classification rules menu.

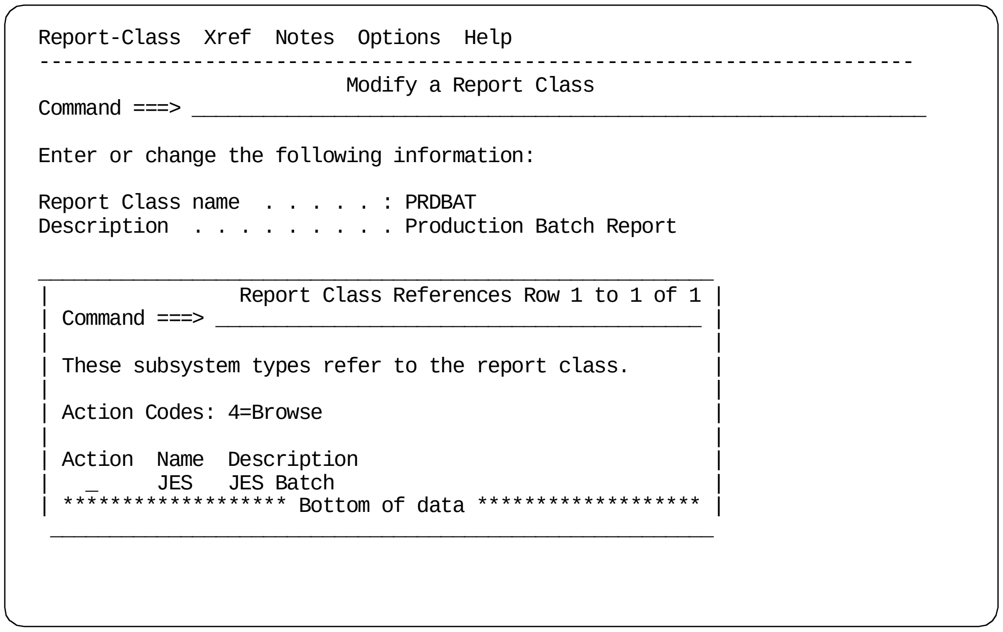
RMF will automatically produce data which enables you to analyze SRVCLASS performance so we do not need to provide a RPTCLASS for each SRVCLASS. Here, as shown in Table 17, we have only one report class.

Table 17. Sample Report Class Definition

Report Class Name	Description
PRDBAT	Production Batch Report

Report class descriptions can be specified using option **7**. Also specifying the report class in classification rules will prompt for the description.

Figure 36 shows you the class modification panel. Use the **Xref** on the command bar to find which classification rules use PRDBAT. The results from the Xref are displayed in the Figure 36 pop-up window.



```
Report-Class  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                          Modify a Report Class
Command ===> ______________________________________________________________

Enter or change the following information:

Report Class name  . . . . . : PRDBAT
Description  . . . . . . . . . Production Batch Report

_________________________________________________________
|                Report Class References Row 1 to 1 of 1 |
| Command ===> _________________________________________ |
|                                                        |
| These subsystem types refer to the report class.       |
|                                                        |
| Action Codes: 4=Browse                                 |
|                                                        |
| Action  Name  Description                              |
|   _     JES   JES Batch                                |
| ****************** Bottom of data ******************* |
 ________________________________________________________
```

Figure 36. Create a Report Class Panel

3.3.11 Application Environment Specification

An *application environment* is a group of application functions invoked by request and executed in server address spaces. You can have workload management start and stop these server address spaces automatically, or do this manually or through automation. You define the application environment name, an optional procedure name for starting the server address spaces, and any start parameters needed for the start procedure.

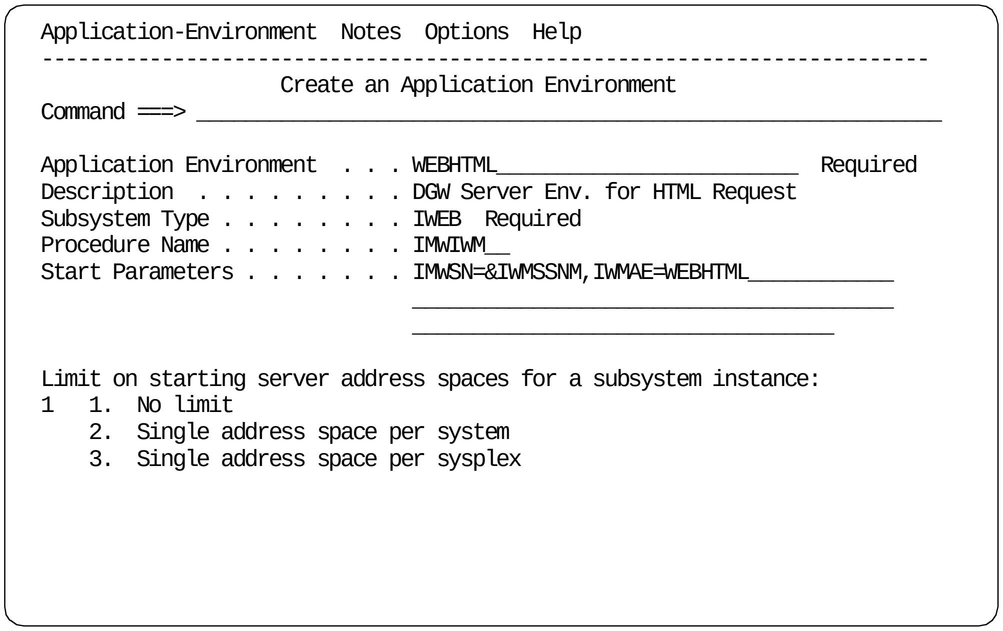
The example in Table 18 on page 121 is from Chapter 5, “Considerations for New Workloads” on page 171. For more information about application environments;

see 2.6.1.1, “Application Environments” on page 78 and 5.3.5, “WLM Definition for Web Server Address Space Management” on page 195.

Table 18. Sample Application Environment Definition

Env. Name	Description	Subsystem	Procedure	Lim.
WLMENV2	Large stored procedure env.	DB2	DBC1WLM2	1
	Starting parms: DB2SSN=DBC1,NUMTCB=2,APPLENV=WLMENV2			
WEBHTML	DGW server env. for HTML requests	IWEB	IMWIWM	1
	Starting parms: IMWSN=&IWMSSNM,IWMAE=WEBHTML			

Application environment specifications can be specified using option **9**. Figure 37 shows the scalable server entry.



```
Application-Environment  Notes  Options  Help
--------------------------------------------------------------------------
                    Create an Application Environment
Command ===> ______________________________________________________________

Application Environment  . . . WEBHTML_________________________  Required
Description  . . . . . . . . . DGW Server Env. for HTML Request
Subsystem Type . . . . . . . . IWEB  Required
Procedure Name . . . . . . . . IMWIWM__
Start Parameters . . . . . . . IMWSN=&IWMSSNM,IWMAE=WEBHTML____________
                               ________________________________________
                               ___________________________________


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 37. Create an Application Environment Panel

3.3.12 Create a Scheduling Environment

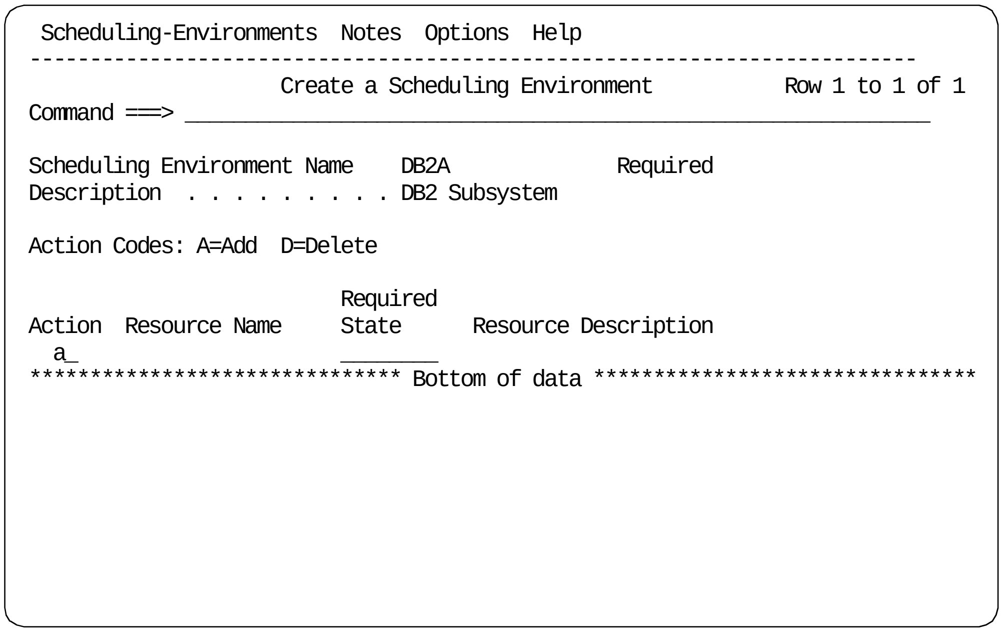
A *scheduling environment* is a list of resource names along with their required states. By associating incoming work with a scheduling environment, you ensure that work is assigned to a system only if that system satisfies all of the requirements. You define the scheduling environment, listing all of the resource names and required states that are contained within. You also define the resource names themselves. Currently, JES2, JES3 and the IBM Intelligent Data Miner can use scheduling environments.

Table 19 shows examples of scheduling environment definitions.

Table 19. Sample Scheduling Environment Definition

Scheduling Env.	Description	Resource	Resource Desc.
DB2A	DB2 subsystem	9672RN2	CMOS Generation 2
PRIMETIME	Peak business hours	IDONTKNOW	Just a label

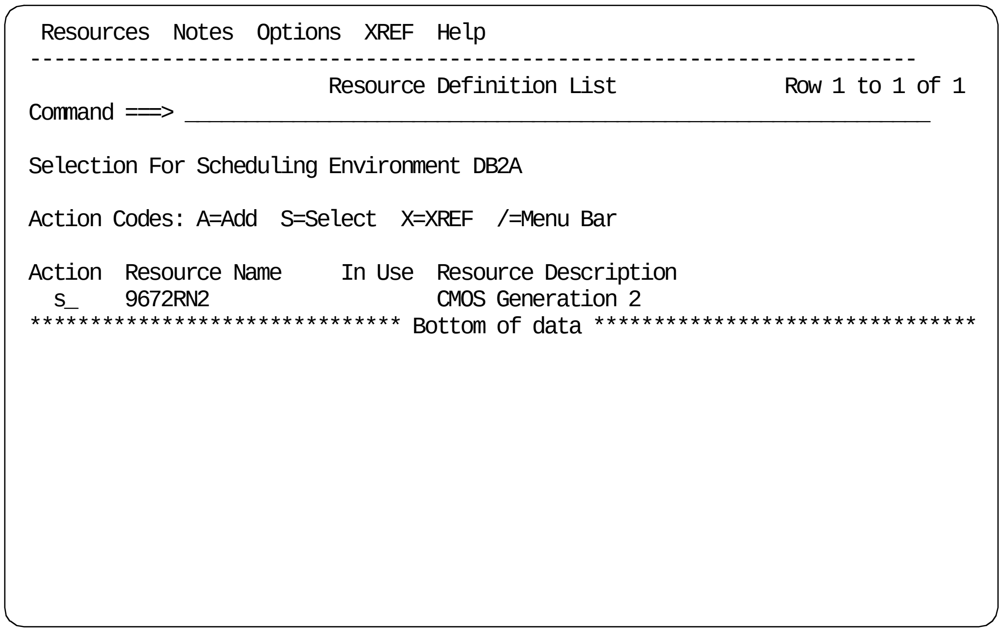
Scheduling environment specifications can be specified using option **10**. Figure 38 shows you the panel to create a scheduling environment definition. Keep specifying A to add until you get a chance to select the resource group. For additional assistance, use online HELP or refer to Chapter 16, section “Creating a New Resource” in *OS/390 MVS Planning: Workload Management*, GC28-1761.



```
 Scheduling-Environments  Notes  Options  Help
 --------------------------------------------------------------------------
                     Create a Scheduling Environment          Row 1 to 1 of 1
Command ===> ______________________________________________________________

Scheduling Environment Name    DB2A             Required
Description  . . . . . . . . . DB2 Subsystem

Action Codes: A=Add  D=Delete

                          Required
Action  Resource Name     State      Resource Description
  a_                      ________
****************************** Bottom of data ********************************
```

Figure 38. Create a Scheduling Environment Panel

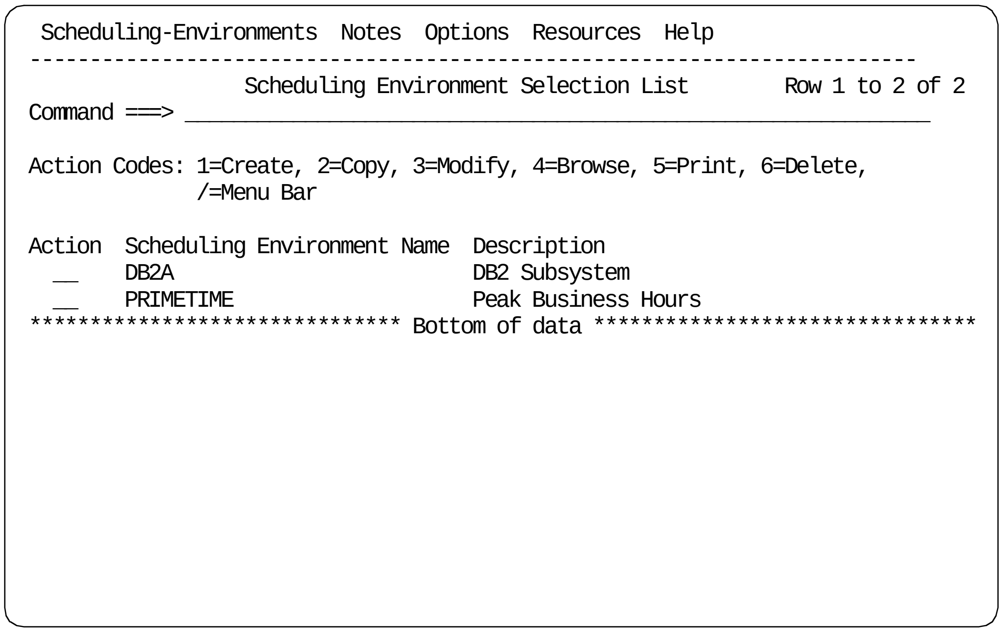
The next panel (Figure 39) shows the resources you have defined. When you select the resource, Figure 38 will be re-displayed and you need to specify the Required State.



```
 Resources  Notes  Options  XREF  Help
 --------------------------------------------------------------------------
                         Resource Definition List            Row 1 to 1 of 1
Command ===> ______________________________________________________________

Selection For Scheduling Environment DB2A

Action Codes: A=Add  S=Select  X=XREF  /=Menu Bar

Action  Resource Name     In Use  Resource Description
  s_    9672RN2                   CMOS Generation 2
****************************** Bottom of data ********************************
```

Figure 39. Scheduling Environment Resource Definition List

Later, if you need to delete or add resources, select **Resources** on the menu bar as shown in Figure 40.



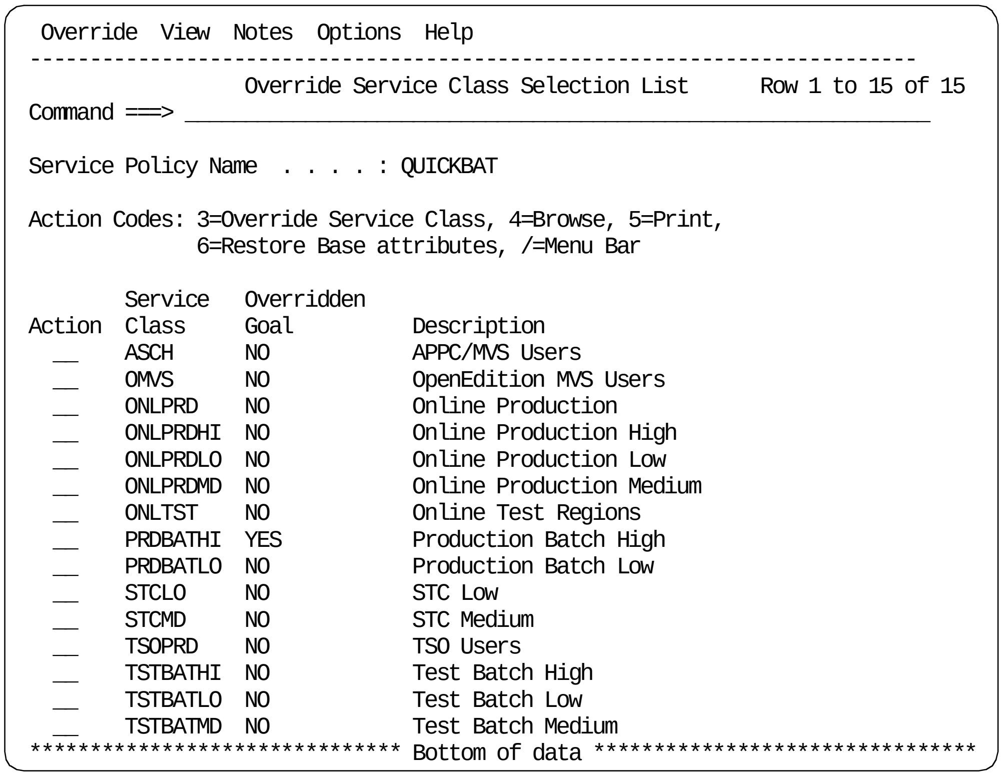
```
 Scheduling-Environments  Notes  Options  Resources  Help
--------------------------------------------------------------------------
                  Scheduling Environment Selection List       Row 1 to 2 of 2
Command ===> ______________________________________________________________

Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete,
              /=Menu Bar

Action  Scheduling Environment Name  Description
  __    DB2A                         DB2 Subsystem
  __    PRIMETIME                    Peak Business Hours
******************************* Bottom of data ********************************
```

Figure 40. Scheduling Environment Selection List

3.3.13 Policy Overrides

You use a policy override only if you have a business need to change a goal for a certain time, such as for the weekend or for nighttime. If you were using multiple IPS/ICS members, changed with the SET command based on business needs, then you may need comparable service policy overrides. Alternatively, you may find with the dynamic management of goal mode that one policy with one set of goals and realistic importance values is sufficient. Less important work will automatically be assigned more resource as more important work tapers off. You can define your policy overrides once you have defined your service classes.

To define an override policy, choose option **1 Policies** from the main definition panel, then use action **7** on the Service Policy Selection List Panel (Figure 27 on page 110) to override service classes. Action **8** can be used to override resource groups. Figure 41 on page 124 shows action 7 after the overrides for QUICKBAT have been entered.



```
 Override  View  Notes  Options  Help
--------------------------------------------------------------------------
                 Override Service Class Selection List      Row 1 to 15 of 15
Command ===> ______________________________________________________________

Service Policy Name  . . . . : QUICKBAT

Action Codes: 3=Override Service Class, 4=Browse, 5=Print,
              6=Restore Base attributes, /=Menu Bar

        Service   Overridden
Action  Class     Goal          Description
  __    ASCH      NO            APPC/MVS Users
  __    OMVS      NO            OpenEdition MVS Users
  __    ONLPRD    NO            Online Production
  __    ONLPRDHI  NO            Online Production High
  __    ONLPRDLO  NO            Online Production Low
  __    ONLPRDMD  NO            Online Production Medium
  __    ONLTST    NO            Online Test Regions
  __    PRDBATHI  YES           Production Batch High
  __    PRDBATLO  NO            Production Batch Low
  __    STCLO     NO            STC Low
  __    STCMD     NO            STC Medium
  __    TSOPRD    NO            TSO Users
  __    TSTBATHI  NO            Test Batch High
  __    TSTBATLO  NO            Test Batch Low
  __    TSTBATMD  NO            Test Batch Medium
******************************* Bottom of data ********************************
```

Figure 41. Override Service Class Selection List Panel

Figure 42 on page 125 demonstrates how the system uses the overrides when a new policy is introduced. Essentially all the parameters from the base policy are effective, unless an override is specified.

Within the service class or resource group override definition you can override any parameter associated with the original definition, for example RG min and/or max, SRVCLASS RG, number of Periods, Durations, Importance and/or goal. You cannot override classification rules, scheduling environments or application environments. Also, you cannot add or delete service classes in an override. Changes to these areas require a re-install of the service definition and a reactiviation of a service policy.

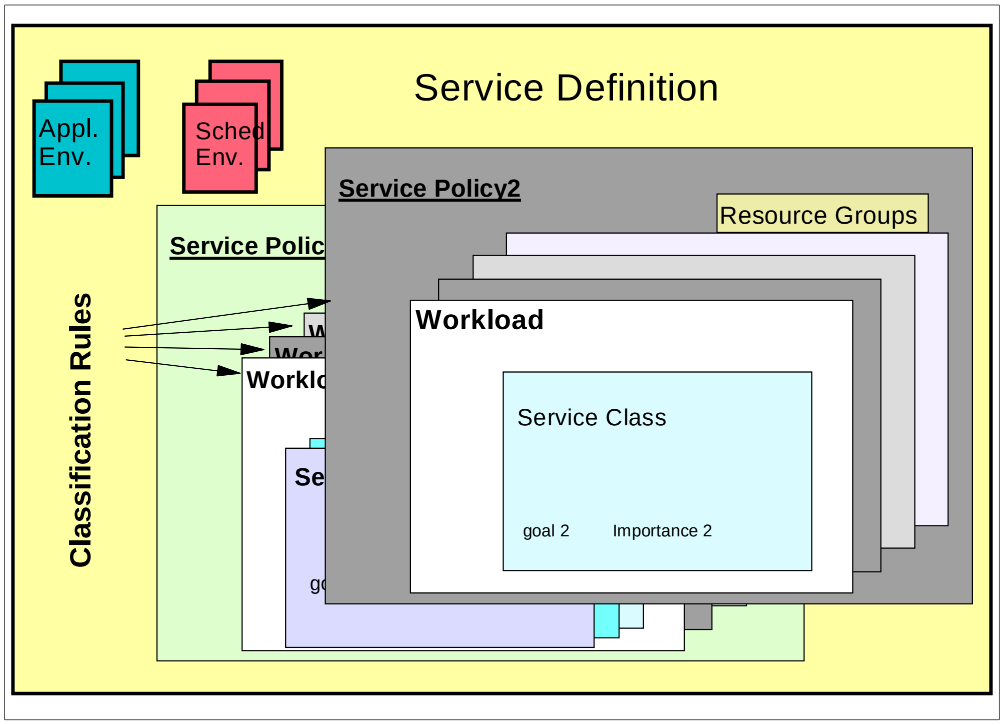
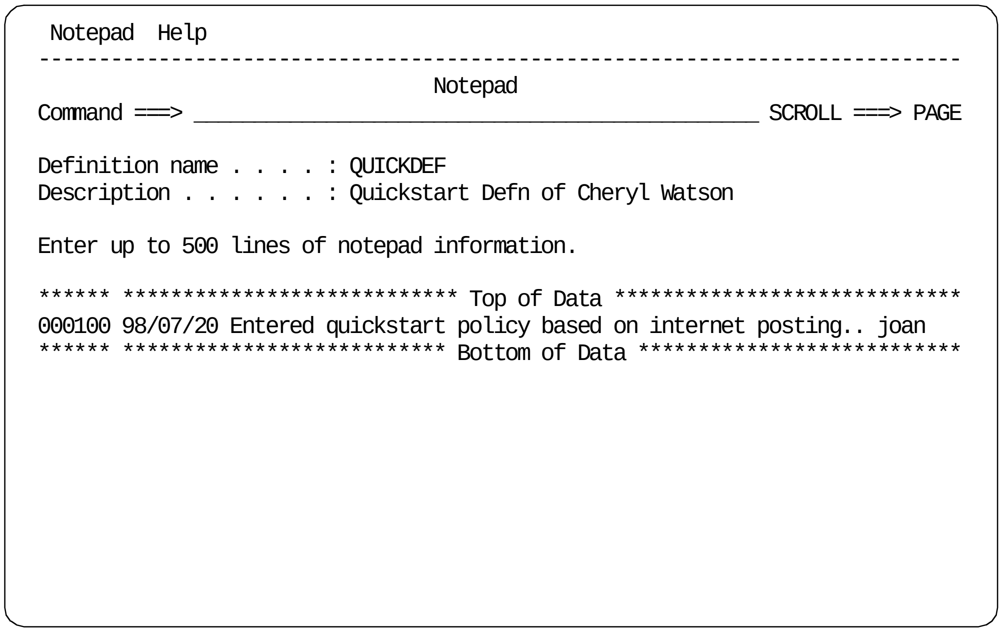


Figure 42. Service Definition Components with Override Policy

3.3.14 Installing and Activating Your Service Definition

Figure 43 shows you the Notepad panel that is displayed after selecting the Notepad option from the menu bar. Enter a description of the changes made.



```
 Notepad  Help
 ------------------------------------------------------------------------------
                                 Notepad
Command ===> ________________________________________________ SCROLL ===> PAGE

Definition name . . . . : QUICKDEF
Description . . . . . . : Quickstart Defn of Cheryl Watson

Enter up to 500 lines of notepad information.

****** **************************** Top of Data *****************************
000100 98/07/20 Entered quickstart policy based on internet posting.. joan
****** *************************** Bottom of Data ***************************
```

Figure 43. Notepad Panel

Figure 44 on page 127 shows the command bar Utilities options. They are as follows:

Install definition
Use this option to install the service definition onto the WLM couple data set. Installing the service definition makes any changes available for policy activation.

Extract definition
Use this option to extract the service definition previously installed on the WLM couple data set.

Activate service policy
Use this option to activate a policy. When you select this option, the application displays a list of the service policies defined in the service definition currently installed on the WLM couple data set. You activate the service policy by selecting it from the list. Policy activation makes the policy go into effect for all systems in the sysplex.

Note: If you have just made changes to a service definition, make sure you *install* it and activate a policy to have the changes take effect.

Allocate couple data set
Use this option to allocate both your primary and alternate WLM couple data sets. This option is for users who are doing one of the following:

- Allocating a WLM couple data set for the first time
- Migrating to an OS/390 R3 or later release where the current WLM couple data set was allocated with a pre-OS/390 R3 release. In this case you should run the IWMARSZ utility to determine the size of the current WLM couple data set.

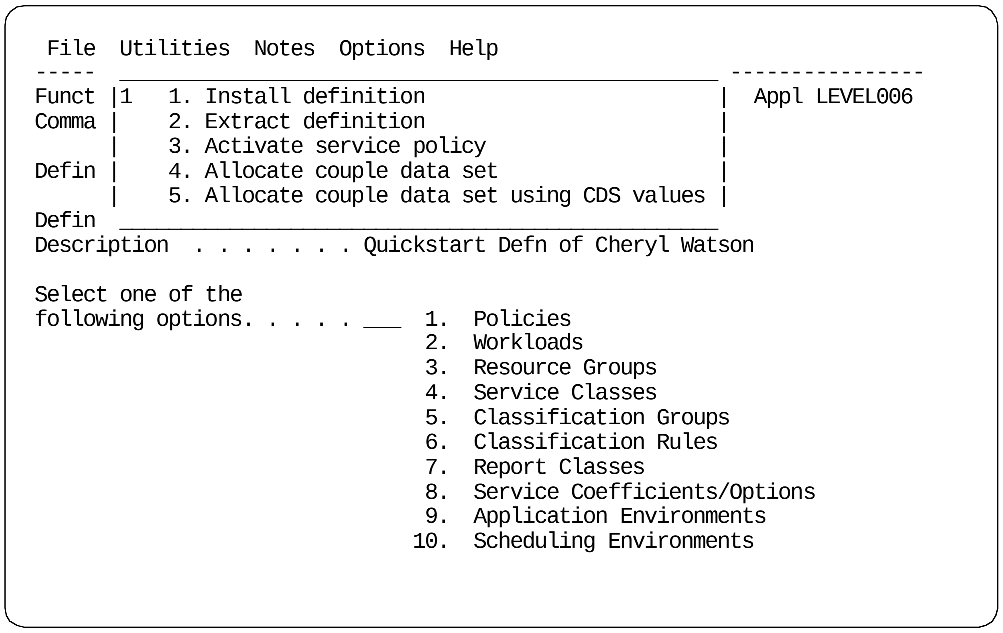
See 3.2.3, “Couple Data Set Management” on page 99 for the sample JCL that is created, as well as for instructions on allocating the WLM CDS for sysplex usage.

Allocate couple data set using CDS values

Use this option to allocate both your primary and alternate WLM couple data sets based on your existing WLM couple data set size. Use this option if your current WLM couple data set was allocated using OS/390 R3 or higher. The application displays the current size values on the panel.

To make the WLM couple data set available for use in the sysplex, you must update your COUPLExx parmlib member and issue the SETXCF command; see 3.2.3, “Couple Data Set Management” on page 99.

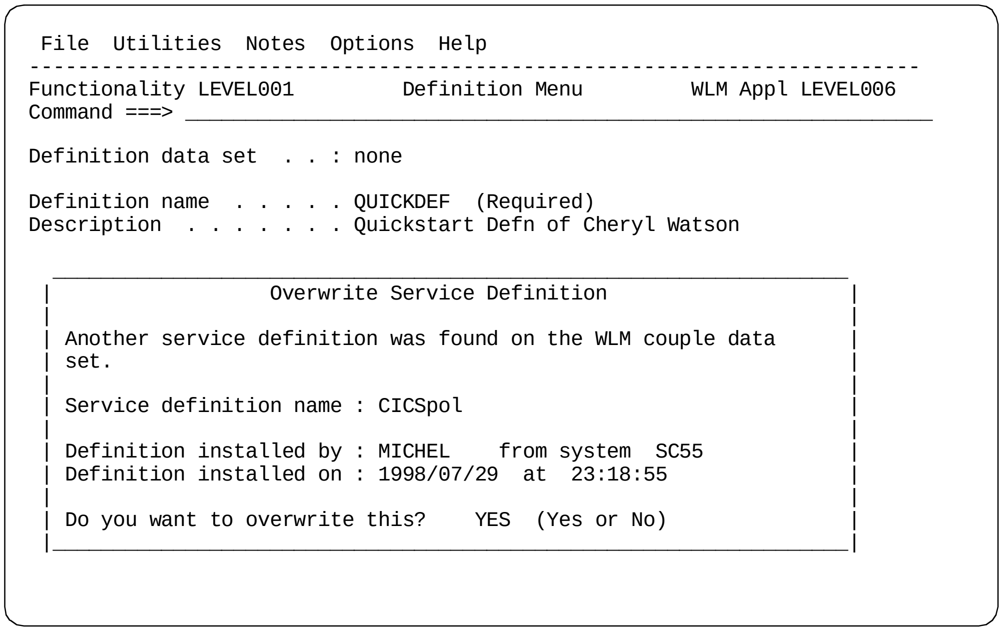
Select option **1**, as shown in Figure 44 on page 127.



```
 File  Utilities  Notes  Options  Help
-----  _________________________________________________ ----------------
Funct |1   1. Install definition                        |  Appl LEVEL006
Comma |    2. Extract definition                        |
      |    3. Activate service policy                   |
Defin |    4. Allocate couple data set                  |
      |    5. Allocate couple data set using CDS values |
Defin  _________________________________________________
Description  . . . . . . . Quickstart Defn of Cheryl Watson

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 44. Install Definition Panel

You may then get another window pop-up, requesting your confirmation to install the service definition, as shown in Figure 45. You will need to make a decision on whether you can overwrite the CDS service definition; specify Yes to overwrite.



```
 File  Utilities  Notes  Options  Help
--------------------------------------------------------------------------
Functionality LEVEL001        Definition Menu        WLM Appl LEVEL006
Command ===> ______________________________________________________________

Definition data set  . . : none

Definition name  . . . . . QUICKDEF  (Required)
Description  . . . . . . . Quickstart Defn of Cheryl Watson

  __________________________________________________________________
 |                  Overwrite Service Definition                    |
 |                                                                  |
 | Another service definition was found on the WLM couple data      |
 | set.                                                             |
 |                                                                  |
 | Service definition name : CICSpol                                |
 |                                                                  |
 | Definition installed by : MICHEL    from system  SC55            |
 | Definition installed on : 1998/07/29  at  23:18:55               |
 |                                                                  |
 | Do you want to overwrite this?    YES  (Yes or No)               |
 |__________________________________________________________________|
```

Figure 45. Overwrite Service Definition

3.3.15 Service Definition File Options

Figure 46 on page 128 shows the command bar File options. They are as follows:

New Use New to define a new service definition.

Open Use Open to read a previously defined service definition. The Read saved definition panel is displayed, where you can specify the data set name.

Save Use Save to save the currently displayed service definition.

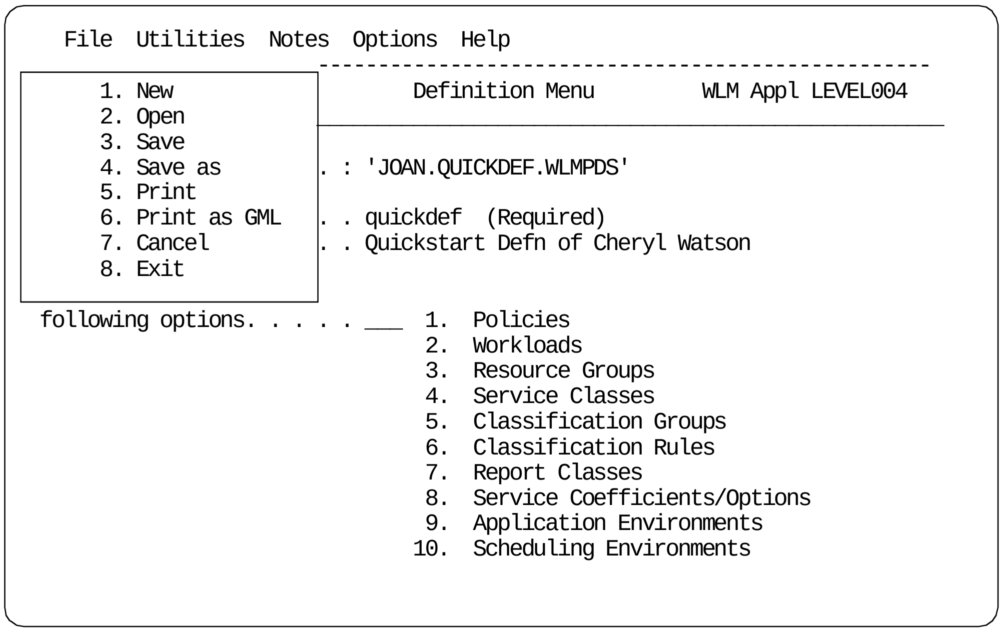
Save as Use Save as to save the currently displayed service definition in a PDS. The Save to... panel is displayed where you can specify the data set name. You do not need to previously allocate the data set. If the data set does not exist, the application displays the Create Data Set?, panel where you can continue with the PDS create.

Print Use Print to print the complete service definition to the ISPF list data set. Use the Options menu bar option to process the ISPF list data set. This option requires no formatting step.

Print as GML Use Print as GML for a more readable, tabular display of service definition objects and values. This option creates a source data set with GML starter set tags imbedded. The data set must be allocated as variable blocked with a logical record length of 255. This data set can then be formatted with the SCRIPT/VS processor.

Cancel Use Cancel to cancel any actions performed. Cancel is the same as using the cancel PF key.

Exit Use Exit to exit from the definition menu and the application. Exit is the same as using the exit PF key.



```
   File  Utilities  Notes  Options  Help
                           ---------------------------------------------------
  ------------------------
       1. New             |        Definition Menu        WLM Appl LEVEL004
       2. Open            |_____________________________________________________
       3. Save            |
       4. Save as         |. : 'JOAN.QUICKDEF.WLMPDS'
       5. Print           |
       6. Print as GML    |. . quickdef  (Required)
       7. Cancel          |. . Quickstart Defn of Cheryl Watson
       8. Exit            |
  ------------------------
  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 46. Service Definition File Options

Chapter 4. Migrating to WLM Goal Mode

In this chapter, we cover the task of migrating performance definitions from WLM compatibility mode to WLM goal mode. Typically, in compatibility mode the IEAIPSxx PARMLIB member (IPS) and IEAICSxx PARMLIB member (ICS) are regularly tuned towards the resources and the workloads. The reason for migrating to goal mode is that you do not need to be involved with further tuning efforts. However, if the IPS and ICS are not tuned, that is not a impediment to the migration to goal mode.

This chapter assumes that you are familiar with how WLM works. If you are not, refer to Chapter 2, “How WLM/SRM Works” on page 21.

Our final objective is to guide you through the creation of a service definition based on your existing IPS and ICS. This is a good time to remind you about the importance of having a Service Level Agreement (SLA) for your business, your IT department, and this migration. Refer to Chapter 1, “Introduction” on page 1, if you are not familiar with this concept. If you are, but your company did not define a SLA, this might be the right time to work backwards and use your future service definition as the initial skeleton for your future SLA. If you have a SLA, you will see that the task of migration is much easier.

There are other two aspects in the migration that we want to stress:

- Goal mode is not replacing your job. Goal mode allows you to accomplish your performance management activity better. The implementation of goal mode makes you closer to your business needs, with additional report data, more control and more recognition from your peers.
- You do not need to IPL the system in order to migrate to goal mode or to switch back to compatibility mode. A pair of commands enables you to easily go back and forth. So, if something exhibits incorrect performance characteristics as a result of your service definition, return to compatibility mode, learn by the mistake, correct it and switch back to the exciting realm of goals, service classes, and performance indexes (PIs).

We also explain both generic migration recommendations that apply to all workloads, and more specific ones regarding unique workloads. The unique workloads described here are the traditional workloads such as batch, TSO, STC, CICS, IMS,APPC and DDF. For new workloads, including DB2 stored procedures, WEB server based work, DB2 parallel queries and so on refer to Chapter 5, “Considerations for New Workloads” on page 171.

4.1 Workload Classification Considerations

Refer to 1.2.3, “Introduction to Workload Classification” on page 7 to get basic information about IEAICSxx (ICS) and WLM classification rules.

OS/390 is able to manage many workloads with distinct business importance and processing characteristics. The installation must categorize to OS/390 the arriving transactions in order to achieve this task. The categories are picked by the OS/390 system programmer directly from the SLA or an equivalent document where you have informally recorded your service level objectives.

In WLM compatibility mode, the categories are contained in the ICS construct. In certain cases, such as associating a PGN to an enclave, there is also use of the WLM classification rules (even in compatibility mode). The technical reason for the need for classification rules is that in the ICS you may define the same subsystem names as classification rules. However the ICS is short in work qualifiers (only ACCTINFO, TRXCLASS, TRXNAME, and USERID) when compared with the 16 possibilities of the classification rules.

For goal mode, you only use classification rules. *Classification rules* are the rules WLM uses to associate a transaction's external properties, which are also called *work qualifiers* (as LUname, User ID), to a transaction goal. These goals are located in service class constructs. Classification rules and service classes are described within a service definition. Refer to Figure 4 on page 8 for a pictorial view of the classification rules.

A *service definition* is the set of all performance policies and other related objects defined by the installation. There is just one service definition for an entire sysplex. It is stored in a WLM Couple dataset and accessed by all OS/390s in sysplex. For more information about service definitions, see 1.2.4, "Service Definition and Service Policies" on page 9.

The following is a list of work qualifiers which can be used in classification rules:

- Subsystem type: IMS, JES2/JES3, TSO/E, STC, ASCH, CICS, OMVS, DB2 (parallel queries), DDF, IWEB, SOM, LSFM (LAN service), MQ, CB, RYO
- Accounting Information.
- LU Name/Netid.
- Subsystem Instance: For CICS, it is the VTAM applid. For IMS, it is the IMS system name. For JES, it is JES2 or JES3.
- Transaction Class/Job Class.
- Transaction Name/Job Name. For ASCH, it is the jobname on the JOB card in the APPC/MVS transaction program (TP) profile. For CICS, it is Tranid. For IMS, it is the CODE parameter in IMS Transact macro. For OMVS, it is the Jobname in the JCL card defining the transaction program.
- Userid (for DDF, it is the DDF server thread's primary AUTHID, after inbound translation).
- Correlation Information: The DB2 correlation ID of the DDF server thread.
- Collection Name: The DB2 collection name of the first SQL package accessed by the DRDA requester in the unit of work.
- Connection Type: The DB2 connection type of the DDF server thread. This contains the value DIST, indicating the thread is a distributed server thread.
- Package Name: The name of the first DB2 package accessed by the DRDA requester in the unit of work.
- Plan Name: The DB2 plan name associated with the DDF server thread.

A group of classification rules is called a Classification Family. It consists of a group of nested rules where the order of the nesting determines the hierarchy of the classification rules. Figure 47 on page 131 shows an example of classification rules for CICS.

```
* 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
```

Figure 47. Classification Rules For CICS

In the example, a PAYR transaction coming from Paris is going to be associated with service class CICSC and Report Class ABC.

A PAYR transaction coming from London is going to run in CICSB service class as indicated by the default service class.

Figure 48 shows the possible work classifiers for each subsystem.

	ASCH	CB	CICS	DB2	DDF	IMS	IWEB	JES	LSFM	OMVS	SOM	STC	TSO
Accounting Information	●			●	●			●		●		●	●
Collection Name		●		●	●						●		
Connection Type [2]				●	●								
Correlation Information				●	●								
LU Name [2]			●	●	●	●							
Netid [2]				●	●	●							
Package Name [2]				●	●								
Perform [2]				●				●				●	●
Plan Name [2]				●	●								
Priority				●				●[1]					
Procedure Name				●	●								
Subsystem Instance [2]	●		●	●	●	●	●	●	●				
Subsystem Parameter		●		●			●				●	●	
Transaction Class/Job Class [2]	●			●		●	●	●					
Transaction Name/Job Name [2]	●		●	●		●	●	●	●	●		●	
Userid [2]	●	●	●	●	●	●	●	●		●	●	●	●

Figure 48. Classification Rules: Qualifiers Supported by Each IBM Subsystem Type

Notes:

1. Applies only to JES2.
2. Can be used with classification groups.

4.2 General Recommendations for Migration

This section describes a general methodology for using the IEAICSxx (ICS) and IEAIPSxx (IPS) members of PARMLIB to build a service definition (other words, to migrate from WLM compatibility mode to WLM goal mode).

These recommendations are valid for all types of workloads.

4.2.1 Getting RMF Data in WLM Compatibility Mode

For some specific subsystems like CICS, IMS, and DDF, you need to have a WLM service policy active to get RMF data in WLM compatibility mode. Therefore, we recommend you do your service definition in two steps:

1. Construct the Service Definition to be used in WLM compatibility mode. In this step, do not worry about the value of the goals you use. They will not be used by WLM for managing the system performance.

   Update the ICS, associating subsystems and PGNs (for DDF) and RPGNs (for DDF, CICS and IMS) to the service classes you defined using the SRVCLASS keyword. Make any changes to the IPS, as needed. Activate the changes in ICS and IPS, using the SET command.

   Activate your policy. Your system is still running in compatibility mode, which means it is still managed by the ICS/IPS. The goals you gave are not being used and you get RMF data/reports to help you define realistic goal values when running in WLM goal mode.

2. Based in the RMF Workload Activity reports, set the correct goal values.

For setting the goal values, use RMF data from peak periods. This helps you avoid setting goals that are too aggressive.

4.2.2 RMF Workload Activity Report in Compatibility Mode

After getting the Workload Activity Report as explained in 4.2.1, “Getting RMF Data in WLM Compatibility Mode” on page 132, you need to understand the major fields whose values can be used when migrating to goal mode. In this section, we describe the most important ones and how they can be used, independently of the particular workload being migrated. To see an example of these fields, refer to Figure 49 on page 133.

```
                                        W O R K L O A D   A C T I V I T Y
                                                                                                    PAGE    1
         OS/390               SYSTEM ID SYS1           DATE 01/15/1998        INTERVAL 20.20.060      MODE=COMPAT
         REL. 02.06.00        RPT VERSION 2.6.0        TIME 10.30.00          IPS = IEAIPS60

       OPT = IEAOPT60   I/O PRTY MGMT = YES REPORT BY PERFORMANCE GROUP    SERVICE DEFINITION COEFFICIENTS     SU/SEC= 619.4
       ICS = IEAICS60                                     PERIOD           IOC = 1.0   CPU = 1.0   SRB = 1.0   MSO = 1.0000

PGN  PGP DMN TSG  TRANSACTIONS   ----TRANS. TIME-----  --DASD I/O--  ---SERVICE---  -SERVICE RATES- PAGE-IN RATES  ----STORAGE----
                                      HHH.MM.SS.TTT

0001 1   001 **   AVG    33.00   ACTUAL        10.616  SSCHRT   0.0  IOC    18067   ABSRPTN  1,318  SINGLE   0.00  AVERAGE  1,237
                  MPL    33.00   EXECUT         8.316  RESP    15.2  CPU    358.5K  TRX SERV 1,318  BLOCK    0.00
                  ENDED      3   QUEUED         2.300  CONN     4.8  MSO    52636K  TCB      578.8  SHARED   0.00  TOTAL   40,841
                  END/S   0.00   R/S AFF            0  DISC     4.5  SRB    78612   SRB      126.9  HSP      0.00  CENTRAL 40,841
                  #SWAPS     6   INELIG             0  Q+PEND   5.9  TOT    53091K  RCT        0.0  HSP MISS 0.00  EXPAND    0.00
                                 CONV           8.388  IOSQ     0.0  /SEC  43,515   IIT        3.7  EXP SNGL 0.00  SHARED  4325.6
                                 STDDEV         3.388                               HST        0.0  EXP BLK  0.00
                                                                                    APPL%     58.1  EXP SHR  0.00
                                                                                    EX VEL%   24.1

0002 1   001 **   AVG     2.15   ACTUAL         1.761  SSCHRT   0.0  IOC    5,760   ABSRPTN  1,280  SINGLE   0.05  AVERAGE 360.43
                  MPL     1.48   EXECUT         1.755  RESP    28.4  CPU    168.5K  TRX SERV   881  BLOCK    0.00
                  ENDED  1,797   QUEUED             6  CONN    13.5  MSO     2139K  TCB      271.9  SHARED   0.00  TOTAL   534.26
                  END/S   1.47   R/S AFF            0  DISC     1.4  SRB    2,534   SRB        4.0  HSP      0.00  CENTRAL 534.26
                  #SWAPS 1,850   INELIG             0  Q+PEND  13.5  TOT     2316K  RCT        7.6  HSP MISS 0.00  EXPAND    0.00
                                 CONV              16  IOSQ     0.0  /SEC   1,898   IIT        1.2  EXP SNGL 0.00  SHARED    0.00
                                 STDDEV        11.140                               HST        0.0  EXP BLK  0.00
                                                                                    APPL%     23.3  EXP SHR  0.00
```

Figure 49. Workload Activity Report in Compatibility Mode

SYSTEM ID: SMF identifier of the system being reported.

INTERVAL mm.ss.ttt: The length of the measurement interval, where mm is the minutes, ss is seconds, and ttt is thousandths of seconds.

You use this field to calculate, for a PGN or RPGN, the minimum or maximum unweighted service units per second in a resource group as follows:

$$\text{Resource Rate} = \frac{\text{TCB} \times \text{SU/sec}}{\text{INTERVAL}}$$

MODE=COMPAT: The WLM operating mode.

IPS and **ICS**: PARMLIB member names being used during the interval.

I/O PRTY MGMT: Indicates if I/O Priority Management is being used (YES/NO). If YES, I/O delays and I/O using counters have been included in the calculation of execution velocity.

You use this field to understand how execution velocity is being calculated. It applies if you have already decided, as recommended, to use I/O Priority Management when in goal mode.

SU/SEC: The number of SRM processor service units per CPU second. This value depends on the CPU speed.

You use this field as a guideline to help calculate minimum or maximum unweighted service units per second, in a resource group.

SERVICE DEFINITION COEFFICIENTS: Values defined in the IPS for the IOC, CPU, MSO, and SRB service definition coefficients.

You use these values to convert new duration values when changing coefficients values, as follows:

$$\text{New Service value} = \frac{\text{service} \times \text{New coef}}{\text{Old coef}}$$

PGN: The performance group number being reported.

PGP: The performance group period being reported.

SERVICE:

- IOC: Total amount of input/output service consumed in the interval.
- CPU: Total amount of task and preemptible-class SRB processor service consumed in the interval.
- MSO: Total amount of main storage occupancy service consumed in the interval.
- SRB: Total amount of non-preemptible SRB service consumed.

Those values are used to convert new duration values when changing service definition coefficient values.

ENDED: The number of transactions that ended during the interval.

You use this field for calculating duration.

END/S: The number of transactions that ended per second.

You use this field for throughput verification.

ACTUAL: The average response time (HH.MM.SS.ttt) required to complete the job.

You use this field value to choose response time goal values.

STDDEV or **STANDARD DEVIATION** or **SD**: A measure of variability of the data in the sample. The higher the standard deviation, the more spread out it looks on a graph. The standard deviation is expressed in the same format as the ACTUAL.

This field gives you some idea as to whether to choose a response time with percentile goal. The higher the standard deviation, the more seriously you should consider using percentile goals.

TCB: Task control block time used in seconds.

You use this field to calculate, for a PGN or RPGN, the minimum or maximum unweighted service units per second in a resource group.

EX VEL %: Execution velocity.

You use this field value to help set an execution velocity goal.

4.2.3 Service Level Agreement

If your installation has a service level agreement (SLA), then you can use this document as a good starting point for the definition of goals. However, the use of additional data from IEAIPSxx and IEAICSxx is also required.

When using SLA, there are things you must remember:

- When the objectives are described in terms of throughput (External Throughput Rate - ETR), you have problems translating this to a response time goal. The ETR formula for online work is the following:

  ETR = number_of_users / (Tt + Rt),

  where:

  Tt = Thinking time

  Rt = Response time that includes network time

  As an example, suppose you have an ETR of 40 transactions per second, thinking time of 5 seconds, and 240 users. Response time would be 1 second, including network time.

  Your problem in deriving the response time goal is that you usually do not know the following:

  - The number of users
  - The value of think time
  - The value of the network time

  So, in the case where ETRs are specified in the SLA, it is better to get the response time goal from RMF compatibility mode reports or some other performance monitor. Refer to 4.2.2, “RMF Workload Activity Report in Compatibility Mode” on page 132 for additional information. However, if the ETR applies to a batch workload, refer to 4.4.2, “Considerations for Migrating Batch Workloads” on page 148. In that section there is a method of introducing an ETR-like goal for batch.

- If the SLA describes a response time, it may include network time. A WLM response time goal does not include network time. In this case you should talk to the network people to get information on the network response time and then subtract it from the SLA value.

3. The SLA may define the worst case for response or turnaround time, that is, the limit that the user can accept. In this case, it would be better to use a response time goal with a percentile to guarantee this worst case. Also, the SLA might define a worst case response time that has always been beaten. In this case, it is better to set goals that will match the current performance.

4.2.4 Recommendations When Using More Than One IPS/ICS

In 4.3, “Service Definition, Step by Step” on page 144, we assume you use one IPS and ICS. If you use more than one IPS, like one for daytime and another for nighttime, we suggest you choose the IPS of the most important shift to make your service definition. When the service definition is ready, you define the overrides needed for the other shifts. The items you can override in a service definition are service classes and resource groups.

If you use more than one ICS for classify your workload (which is not recommended), we recommend that you merge them into just one set of classification rules for simplicity and better documentation. In WLM goal mode, to define another set of classification rules you must define and install another complete service definition.

4.2.5 Using Service Definitions

Refer to 1.2.4, “Service Definition and Service Policies” on page 9, to get the basics about service definition.

Keep your service definition as simple as possible. Use the minimum number of policies, workloads, resource groups, service classes, service class periods, classification rules and report classes that you can get away with.

A reasonable number of service classes is around 20 to 25:

- Use maybe 5 to 7 for STCs
- Use 5 to 7 for different batch categories, including IMS MPRs, archive and BMPs
- Use the rest for interactive work being run by other subsystems

Except for SYSTEM and SYSSTC service classes, WLM decides how to allocate system resources based on sampling, so as more work goes into a service class, there are more entities contributing samples and better history data for WLM.

Define a simple set of workloads and service classes and only use multiple period service classes when you have specific requirements, because WLM management is more responsive. WLM is much more effective with a simple definition having a few service class periods than with a definition having many dozens of service class periods. Also, a simple service definition is easier to maintain and saves time and space when processing data and creating reports.

4.2.6 Service Definition Coefficients

Service units are a measure of resources productively consumed by an address space or enclave. They are calculated based on the task time (CPU), SRB time (SRB), I/O, and storage (MSO) service consumed. Installations can assign weight to one kind of service over another. This weight is called a *service definition coefficient* (SDC).

With the exception of the MSO piece, the concept of service units is designed to have repeatability, that is, same transaction consuming the same amount of resources shows the same service units independently of the processor used.

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 a transaction running in the service class period has consumed the amount of service specified by the duration, WLM moves it to the next period. Because goal and importance are associated with the service class period, the work is then managed to the goal and importance of the new period.

Discarding workloads defined as SUBSYS equal to CICS or IMS because period switching is not supported, and also SYSTEM or SYSSTC, there is a recommendation to use periods for service classes associated with workloads such as CB, IWEB, TSO, DB2 and possibly JES. All trivial transactions (80%) of total should have high priority and must finish in the first period - adjust your duration for that. These transactions are light consumers and by ending quickly, they release resources that can then be used by other workloads.

Another way of saying the same thing is: putting 80% of the transactions in the first service class period will make 80% of the users happy, and with low resource consumption.

You can continue using the same coefficients as in compatibility mode. That may be wise if you have billing tools based on service units and if those tools cannot be changed. Also, if you keep them the same, then you can directly watch RMF data in compatibility and check your durations properly, when in goal mode.

However, remember that your coefficients can be inflated. Because of the increase in processor speed and capability, the total workload can consume much greater amounts of service. As a result, service unit consumption numbers can be very high. These high numbers may cause problems, if they reach the point where they wrap in the SMF fields. In such a case, you may see abnormally small transaction service units consumed, and the current period work may be restarted in the first period. If so, and, if you plan to use WLM, it is probably a good time for you to rethink your SDC coefficients. It is possible to make them smaller and still maintain the same relationship between the previous coefficient values. Consider changing your service definition coefficients to the recommended values shown in Table 20:

Table 20. Coefficient Definition Recommendations

Parameter	Default			Description	Recommended
	System	IPS	Goal		
CPU	1.0	10	10.0	CPU service multiplier	1.0
IOC	1.0	5	5.0	IO service multiplier	0.5
MSO	1.0	3	0.0	Storage service multiplier	0.0
SRB	0.0	10	10.0	SRB service multiplier	1.0

If you change these coefficients, do not forget to adjust the duration (DUR) when defining service class periods. Remember to also make the change in the IPS, to keep the service units consistent, because during the migration process you may have to return to WLM compatibility mode while fixing the service definition and then return to goal mode.

We present an example of deriving a new set of duration values for a TSO workload. However, this method applies to any workload where you have more than one period and you have changed the SDC coefficients.

When data was collected, these service coefficient values were defined in IPS:

```
IOC=5 CPU=10 SRB=10 MSO=3
```

and these periods for TSO:

```
PGN=2,(DMN=20,DP=F70,DUR=1K)
      (DMN=20,DP=F50,DUR=2K)
      (DMN=21,DP=M3)
```

From the RMF Workload Activity report, obtained for peak periods for the TSO workload, one obtains the number of ended transactions (ENDED field), the average response time (TRANS. TIME column, ACTUAL field) and its standard deviation (SD field). Specific service units consumption is obtained from IO, CPU, MSO, and SRB fields. TOT presents the total service units consumption. Assume the following RMF Workload Activity report for TSO performance group periods contains:

```
1st Period ENDED=320; ACTUAL=0.020; SD=0.018
IOC=4,325    CPU=51237    MSO=371.4K    SRB=6,258    TOT=433.2K

2nd Period ENDED=56;  ACTUAL=0.158; SD=0.109
IOC=6,355    CPU=48158    MSO=359.8K    SRB=1,897    TOT=416.2K

3rd Period ENDED=42;  ACTUAL=0.635; SD=0.514
IOC=24320    CPU=155.4K    MSO=917K     SRB=5,453    TOT=1102K

all periods ENDED=418;
IOC=35,000    CPU=252.5K    MSO=1848.2K     SRB=13,608    TOT=1951.4K
```

Also assume the installation plans to work with the recommended service definition coefficients, as follows:

```
CPU=1, SRB=1, IOC=0.5, MSO=0
```

For service classes duration periods, the system must be informed of the weighted service units, according to the new service definition coefficients; see 4.2, “General Recommendations for Migration” on page 132. The report presents service units weighted by the old coefficients. So, converting the values extracted from the report, we get the following:

```
Old weighted service units = Service units x old service definition coefficient
New weighted service units = (Old w.s.u /old s.d.c.) x New s.d.c
```

The first period consumption includes the consumption of those transactions ended in the first period and the transactions that moved to other periods. During the second period, only transactions ended during the second and the third periods consumed service.

```
1st period consumption = ((4325 x 0,5/5) + (51237 x 1/10) + (6258 x 1 /10))/418
= 15
2nd period consumption = (635,5 + 4815,8 + 189,7)/98 = 58
```

Some transactions, in the first period, finished without consuming all the service they were permitted. Just putting the new average as the duration will not result in the same proportion. Therefore, make the duration slightly higher than the new average and then adjust it if necessary to ensure that 80% of transactions finish in the first period.

These performance groups can be transformed into three service class periods, with heavy transactions having a discretionary goal:

SRVCLASS	Description	Workload	RG	Per	Dur.	Imp.	Goal
TSOPRD	TSO Users	TSO		1	20	2	80% 00:00:00.02
				2	70	3	70% 00:00:00.10
				3			discretionary

Another possibility is transform them into only two service class periods, with heavy transactions having a discretionary goal:

SRVCLASS	Description	Workload	RG	Per	Dur.	Imp.	Goal
TSOPRD	TSO Users	TSO		1	70	2	80% 00:00:00.5
				2		3	discretionary

4.2.7 Using Service Classes

Refer to 1.2.6, "Service Class" on page 11, to get the basics about service classes.

In an online environment other than CICS or IMS, you can use multiple service class periods when the service class has transactions with uneven resource consumption, that is, heavy and light (trivial) transactions. We recommend that you favor the light ones, because they are either important to the business or for performance reasons you want to complete them quickly.

Do not mix transactions and address spaces in the same service class, for example SUBSYS=CICS and SUBSYS=STC. The sampling data, plots and projections that SRM assembles for each service class period can become very distorted if work of such diverse structures is combined in the same service class.

Unfortunately, installations do not always have total control over new incoming work. So keep your eye on service class SYSOTHER (the default service class for non-STC address spaces where no classification rules exist for the subsystem type). This service class has a discretionary goal (low priority). If work is running in this service class, it means that this work was not classified by the classification rules. It is better to assign this work to an adequate service class, even with a discretionary goal.

4.2.8 Using Response Time Goals

Refer to 1.2.6.1, "Performance Goals" on page 11, for the detailed definition of this type of goal.

The work assigned to a response time goal must have sufficient completions. As a rule of thumb, expect at least 10 completions in twenty minutes for a response time goal to be effective.

4.2.9 Using Discretionary Goals

Refer to 1.2.6.1, "Performance Goals" on page 11, for the detailed definition of this type of goal.

Assign a discretionary goal to work that has low business importance and no particular goal. The discretionary work competes for processor access after:

- No non-discretionary work needs CPU (pre-OS/390 R6).
- All non-discretionary work is achieving its goals (OS/390 R6 and higher)

Refer to Chapter 2.2.1.10, "WLM/SRM Discretionary Goal Management" on page 40, for additional information on how WLM manages discretionary goals.

4.2.10 Using Execution Velocity Goals

Refer to1.2.6.1, “Performance Goals” on page 11, for the detailed definition of execution velocity type of goal.

The default for I/O priority management is *no;* in this case WLM calculates velocity by the following formula:

$$\text{velocity} = \frac{\text{using CPU}}{\text{using CPU} + \text{delayed by CPU} + \text{delayed by storage}} \times 100$$

If you change I/O priority management to *yes* (by using the WLM application option 8), WLM dynamically sets I/O priorities based on goals and I/O activity, and includes the I/O information when calculating execution velocity as follows:

$$\text{velocity} = \frac{\text{using CPU} + \text{using I/O}}{\text{using CPU}+\text{using I/O} + \text{delayed by CPU} + \text{delayed by I/O} + \text{delayed by storage}} \times 100$$

As a result, you might see some changes in your velocity values due to I/O priority management. Even when I/O priority management is turned off, the DASD I/O USING samples and DASD I/O DELAY samples are reported in RMF. This allows you to calculate the new velocity values when migrating to I/O priority management. IBM recommends turning on I/O priority management when all systems in the sysplex have been migrated to goal mode at OS/390 R3 or higher. Since OS/390 R4, RMF has enhanced the Workload Activity report to include the I/O samples in the Velocity Migration field. See *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information on this field.

If you choose to use WLM to manage batch initiators, then delay by initiator in the denominator is also included in the velocity. Refer to 2.6.2, “WLM Batch Initiator Management” on page 80 to get more information.

For setting velocity goals, you can get the velocity of the PGN from the EX VEL% field in the RMF Workload Activity report in compatibility mode. If you plan to use I/O priority management in goal mode, you may declare this in your policy. In this case, RMF calculates EX VEL% including the I/O samples and the RMF Workload Activity report headings display *I/O PRTY MGMT = YES.* Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950.

In WLM goal mode, you have three fields from which to get velocity information in the RMF Workload Activity report:

- The EX VEL% field displays the velocity being achieved.
- I/O PRTY shows what the velocity would be with I/O priority.
- INIT MGMT shows the velocity if running with WLM-managed initiators.

This information helps you adjust the velocity goals when you decide to use either I/O priority management or batch initiator management.

Table 21 on page 141 shows, for each field, what is included in the execution velocity calculation if you use I/O priority management, and/or WLM-managed batch initiators, or neither.

I/O Priority / WLM Init Management	VEL %	I/O PRTY	INIT MGMT
NO/NO	CPU+storage	CPU+storage+I/O	CPU+storage+init
YES/NO	CPU+storage+I/O	CPU+storage+I/O	CPU+Storage+I/O+init
NO/YES	CPU+storage+ Init	CPU+storage+I/O +Init	CPU+storage+init
YES/YES	CPU+storage+I/O +Init	CPU+storage+I/O +Init	CPU+storage+I/O +Init

Table 21. Velocity Calculation, I/O Priority and Initiator Management

In the following generic (that is, not applied to a specific workload) items, we describe some important aspects of the execution velocity goal:

- It is not recommended to use a execution velocity goal above 90%. Such a numeric value is very difficult to achieve due to *reduced preemption* and the fact that all the SYSTEM and SYSSTC address spaces have a higher dispatching priority than any velocity-type goal. Reduced preemption means 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.
- The execution velocity goal is dependent on the CPU speed. So if you install another generation of a CMOS machine, we recommend you review your velocity goals.
- The execution velocity of multitasking address spaces is sensitive to the number of CPUs in use by OS/390. At equal CPU utilizations, systems with different numbers of CPUs running heavy multitasking workloads, like DB2, have different CPU queue lengths:

```
DB2 Achieved Velocity (9021 - 1 CPU) = 13.3%
DB2 Achieved Velocity (9021 - 5 CPUs) = 74.9%
```

- Service class periods with a large number of address spaces or enclaves (compared with the number of CPUs) should not receive a high value of execution velocity goal because it is unachievable. For example, a velocity goal of 50 is impossible to obtain for a set of 100 CICS AOR address spaces in a OS/390 image with 10 CPUs.
- Certain system address spaces, such as VTAM and IRLM, spend the majority of time in the *unknown* state and the *idle* state. Because of this, there may 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 like VTAM with low figures.
- Due to the variability of velocities, micromanagement of many different service class periods with only slightly different velocity goals is not practical. Many sites find it useful to think in terms of high/medium/low velocities, at least 10 percentage points apart, rather than concentrate on the specific values.
- The execution velocity formula does not include the unknown state. Refer to 1.2.5, “Sampling Unit of Work States” on page 10, to understand the meaning of unknown state. This state includes the delays not tracked by SRM such as locks or enqueues. Contrast this with response time goals, where the elapsed time that corresponds to unknown samples is included in the response time.

Unknown delays do affect SRM’s decision for response time goals, but the same delays do not affect velocity goals. In a resource-constrained system, service class periods with large unknown times that are managed using velocity goals exhibit PIs that are more variable than other goal types since the work is effectively being managed on a subset of its total delays. In this case we recommend that you use response time goals if possible instead of execution velocity goals.

- When a service class period has a velocity goal, the amount of history that SRM needs is determined by the number of address spaces active in that period. If only a single address space is in the period, SRM needs several minutes to collect enough samples. With more address spaces, the history could easily reflect recent seconds and therefore be more effective. So, when choosing velocity goals, avoid having only one address space in that service class.

4.2.11 Importance Keyword

The *importance* keyword does not correspond to the control of the dispatching priority. WLM uses goal importance when there is insufficient capacity for all work in the system to meet its goals. WLM attempts to satisfy all goals of importance one before going after the goals of importance two and so on. Either way, WLM uses importance to determine which remaining work can donate resource needed by the work with higher importance. You assign an importance to a service class period, which indicates how important it is that this goal be met relative to other goals. There are five levels of importance: lowest (5), low, medium, high, and highest (1).

For example, it might be very important to an installation that long-running jobs continue to execute occasionally throughout the day. Velocity may be 5%, but the importance is 1. WLM may find that a low dispatching priority can satisfy the goal.

Refer to 1.2.7, “Importance of a Goal” on page 13, to get more information.

4.2.12 Using Resource Groups

As we see in 1.2.8, “Resource Group (RG)” on page 13 with WLM resource groups you may limit (by capping) the CPU capacity or guarantee (protection) some minimum CPU capacity to a workload on a service class basis.

With resource groups, you tell WLM how much CPU capacity you want to cap or to protect through unweighted CPU service units rate (service units not multiplied yet by the SDC coefficient). Because of that, you should not associate service classes from CICS and IMS (SUBSYS = CICS or IMS) with resource groups. As you know, WLM does not measure service units for these types of service classes. Therefore, the only way to use resource groups with CICS and IMS is by associating their address spaces (regions) to a service class in the STC subsystem.

All the service classes associated with enclaves may use the resource group facilities, so you may cap or protect, for example, a set of specific DDF, IWEB, or SOM transactions.

RMF displays data about the results of capping and protection (actual consumed, service classes associated with the RG) in the Workload Activity report under the label RESOURCE GROUPS. Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950, to get more information about these fields.

4.2.12.1 Resource Group Capping

Resource group capping is a new concept in goal mode, and it does not exist in compatibility mode. You could ask: why bother talk about this subject in a migration chapter? The answer is that the lack of capping in OS/390 forced many installations to move workloads to a different MVS logical partition to exploit the capping function available in LPAR. Maybe now, with WLM resource group, it is time to migrate/consolidate such applications back to a single OS/390. Let us see how to implement such a migration.

It is important to note that the resource group capping algorithms have precedence over WLM goals. This means that it is possible that the service class goals will not be achieved if resource groups are incorrectly defined. In general, WLM obeys the maximum value of resource consumption and does not care about the goal and importance.

Traditionally, OS/390 did not have the capping function. This meant that even a low-dispatching priority address space could get all the CPU it wanted, so long as higher dispatching priority address spaces had all dispatchable units in wait.

Capping in OS/390 terms means to limit the CPU capacity of a set of address spaces, even if spare capacity is available. This technique can be applied in a Service Bureau where each company, as a customer, should not consume more processor power than was contracted. This also applies to a situation where your installation wants to reserve some CPU capacity for future growth by keeping users at a lower service level. There are IT organizations which use capping for running applications that are famous for getting into CPU loops. With capping, the damage is contained.

Usually you would want to cap a workload on a percentage basis (20% for example) of CPU capacity installed in one OS/390 image. The problem is how to determine the total numerical value of such capacity. To derive this figure, take a look at the SRM constant (CPU service units per second) of your OS/390 image and multiply it by the number of logical CPs (if in LPAR mode) or by the number of physical CPs (if in basic mode). Now you have the total unweighted CPU service units per second that your OS/390 image can ideally deliver. If you apply 20% to this figure, you have the maximum capacity in the resource group. If you are in a sysplex, the CPU service rate is a summation of the consumption in all OS/390 images.

Resource group capping cannot be used as a replacement for the old IPS response time objective (RTO). This parameter was used to induce some swap-in delay when a TSO transaction starts; capping induces CPU delay when a specific workload presents a demand higher than a defined maximum.

The RMF Workload Activity report shows a capping delay figure indicating for how long the address spaces of the capped service class were made non-dispatchable. This data may be used to explain why the goal was not reached.

4.2.12.2 Resource Group Protection

Protection in OS/390 terms means to guarantee a minimum CPU capacity for a set of address spaces (service classes), if this capacity is required. This technique can be applied if you want to ensure a certain throughput for a workload. If you want to guarantee a certain percentage of CPU capacity to a workload, you should proceed as described for capping, but specifying a *minimum* capacity instead of *maximum* capacity.

The minimum CPU capacity is delivered in the situation where the goals are not being achieved by the work in the resource group, so be sure you set the goal sufficiently aggressive to allow the minimum to be honored.

Associating a discretionary goal with resource group minimum is a way of implementing a sort of throughput goal in WLM goal mode. Refer to 4.4.2, “Considerations for Migrating Batch Workloads” on page 148, to understand how you can implement such a goal.

4.2.13 Report Classes

Do not set a report class equal to a service class. RMF provides reports by service class already. Report classes are available to permit additional reporting data within a service class or across service class. For example, you can have a report class combining two service classes, or a report class for information on a single transaction. In report classes, information is not double-counted. For example, if you have a report class for all CICS transactions, and a second report class for CICS transactions related to a specific application, the statistics for the CICS transactions related to the specific application do not appear in the report for all CICS transactions.

4.3 Service Definition, Step by Step

To define a policy, you need RMF data or performance monitor reports from compatibility mode. The RMF Workload Activity report provides information to understand what is running in each performance group. For example, RMF Version 5 and higher, in the Workload Activity report there is the velocity and average response time achieved by PGNs and domains. Remember that domains do not need to be migrated to goal mode because they vanish and are not replaced by any WLM construct. Refer to 4.2.2, “RMF Workload Activity Report in Compatibility Mode” on page 132 for a description of the most important fields.

To begin making your service definition, list your current IPS/ICS and follow these instructions in the order they are presented:

1. Compile the service definition on paper before entering any information into the WLM application. Appendix A, “Sample Worksheets” on page 213, presents a set of worksheets that we suggest you use.
2. Define your Service Definition Coefficients. Appendix A.1, “Service Definition” on page 213, presents a worksheet you can use.
3. Rank the workloads in *descending priority* order. That is a good opportunity to review and correct the priorities in your installation. Appendix A.3, “Ranking Workloads” on page 214, presents the worksheet to be used.
4. Separate the workload into groups of workload types having the same performance needs, resource requirements and business importance to the installation. This is the basis for constructing the service classes. For WLM, a *workload* means a collection of service classes to be reported as a unit. You

can arrange workloads by subsystem, by major application (test, batch, office) or by line of business (ATM, inventory, department, and so on). The simplest way is to separate them by subsystem. For each subsystem, follow the specific instructions. Use the worksheet presented in Appendix A.4, “Workload Classification” on page 215.

5. For each group you created in the last item, create one (or more) service class. If necessary, define more than one period for the service class. In the subsequent sections there are specific recommendations for each type of subsystem. Refer to them while creating service classes.

   For each service class period, choose the goal type, its importance, and goal numeric value. Do not set unrealistic goals. If a goal is too aggressive, two consequences can result:

   - WLM spends time trying to help the work associated with the goal before determining that nothing more can be done.
   - Stealing the resources it needs from other work is not a good trade-off based on the other work’s goal and importance.

   Assign the importance to the goals of different service class periods with the agreement of the installation’s organizations and functions.

   Review your service classes and avoid having too many. There should not be a one-to-one relation with PGNs. Examine your results to see if you can eliminate or merge some service classes. Keep your service policy as simple as possible.

   Appendix A.5, “Service Class Definition” on page 216, presents a worksheet that can help you.

6. If you need to reserve a minimum capacity to a workload or limit its consumption, define a resource group (this step is optional). Use the worksheet presented in Appendix A.9, “Resource Group Definition” on page 226.

7. Define your classification rules. These will ensure that your workload will be be assigned to the appropriate service class, and therefore have the correct goal. For more information on classification rules, refer to 4.1, “Workload Classification Considerations” on page 129

8. At this point, your service definition is ready to be implemented. After following the implementation steps listed in Chapter 3, “Implementation of WLM in Goal Mode” on page 87, your system is ready to operate in WLM Goal Mode, by use of a MODIFY command.

   When you change to goal mode, you can track your definitions using the SDSF DA option. If something is wrong, correct it, save, re-install and activate your policy.

   Note: This does not require you to switch back to compatibility mode.

   You can use RMF Monitor III to see if the goals that you chose are being achieved and to examine the delays that service classes are suffering. Refer to *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950 for more information on these reports.

   You can return your system to compatibility mode at any time by using the MODIFY command:

   ```
   F WLM,MODE=COMPAT
   ```

9. To adjust the goals you defined in your policy, you must use the RMF Workload Activity report. The report in WLM goal mode is different from the compatibility mode version, and only exists as a postprocessor report. To produce this report, you specify the following in the SYSIN file to the ERBRMFPP program:

   ```
   SYSRPTS(WLMGL(SCPER))
   ```

   This option will generate a report for all service class periods. This report contains detailed information about subsystem delays, response time goals versus actuals, general execution delays, and response time distribution. Verify if the goals you gave are too aggressive or too conservative and adjust them accordingly.

   To obtain other levels of the data in the report, you can use other options as follows:

   SCLASS: This provides a summary of data for each service requested. In the example, it was requested to provide a report for each service class defined:

   ```
   SYSRPTS(WLMGL(SCLASS))
   ```

   WGROUP: This provides a summary on a workload basis. The following example requests a report for the workloads CICS and TSO:

   ```
   SYSRPTS(WLMGL(WGROUP(CICS,TSO)))
   ```

   POLICY: This option provides a summary of data for the active service policy:

   ```
   SYSRPTS(WLMGL(POLICY))
   ```

   RCLASS: This provides a report for the report classes defined in a service policy.

   To analyze performance problems, we suggest you use *OS/390 Resource Measurement Facility Performance Management Guide,* SC28-1951. In that publication:

   - The chapter “Diagnosing a problem: the first steps” helps you recognize a performance problem. It indicates the RMF monitors options, RMF reports, and additional tools you can use.
   - The chapters “Analyzing Processor Activity”, “Analyzing Processor Storage Activity” and “Analyzing I/O Activity” helps you identify specific problems, and which are the RMF Monitor I and Monitor II indicators. Also it gives you guidelines for improvement.
   - The chapter “Analyzing Sysplex Activity” guides you through the monitoring sysplex activity. It helps you to understand the new CICS and IMS Workload Activity report in goal mode, and to analyze coupling facility activities.

4.4 Workload-Specific Considerations

In this section, we present a list of workload types whose migration aspects are discussed. The section includes the traditional workloads such as batch, TSO, STC, IMS, CICS, DDF, APPC and roll-your-own (RYO). The new workloads arriving on the OS/390 scene, such as those mentioned in the following list, are covered in Chapter 5, “Considerations for New Workloads” on page 171:

- Component Broker Client Object (CB)
- SOM Client Object Class Binding (SOM)

- UNIX System Services Forked Children Address Spaces (OMVS)
- LAN Server for MVS (LSFM)
- DB2 Complex Parallel Queries (DB2)
- Distributed DB2 (DDF)
- DB2 SQL Stored Procedure DDF)
- Domino Go Web Server (IWEB)

4.4.1 Considerations for Migrating TSO Workloads

TSO is an interactive workload. There are users at the terminals waiting for a quick response when they press Enter.

Each TSO user has his/her own address space in which only one transaction is being executed at a time. Each command issued from the foreground is accounted as a transaction. An exception for this rule is the use by the installation of the option CNTCLIST=YES, in IEAOPTxx. Using CNTCLIST=YES causes SRM to account for each command in the CLIST as a new transaction.

For TSO transactions, we recommend you use response time goals. The transactions are short and quick with enough completions to allow SRM to collect a reasonable statistical sample set to base decisions on.

If you assign a velocity goal, it causes the SRM to control the swap protect time on a individual address space basis rather than on a period-wide basis. In cases when transactions do take a while to complete, the SRM looks at the address spaces with a velocity goal to decide what expanded storage access to give to their demand paging, VIO paging, and hiperspace paging. This might involve monitoring address spaces for working set management control even though the work might still end quickly. Therefore, using velocity goals is less efficient than running with response time goals. Additionally, response time goals are less sensitive to upgrades of the processor power or capacity. For velocity goals, you would need to revisit the velocity, following such an upgrade, to ensure goals were being met.

Using response time with percentile is generally better than average response time because the worst response times tend to distort the average.

Your installation probably uses performance group periods for TSO. If so, you obtain the duration of each period from the IPS. However, if you changed your service definition coefficients, you need to convert the duration values and 4.2.6, “Service Definition Coefficients” on page 136 provides an example of how to do that.

If your installation does not have performance group periods, you may consider implementing them now.

You can use multiple periods; we propose two or three. You can “punish” those transactions that use more than the installation feels is a reasonable amount of service for a TSO user. You do this by setting the last period of a service class to a discretionary goal. SRM may swap the discretionary address spaces out for long periods of time, if needed. Since OS/390 V2.6, WLM gives better treatment to discretionary goals. Refer to 2.2.1.10, “WLM/SRM Discretionary Goal Management” on page 40, for more information on this enhancement.

The values you enter can be refined as you gain experience. When working in WLM goal mode, you can adjust them, based on Workload Activity report data for the TSOPRD service class periods.

You may choose to create a special TSO service class with more aggressive goals and with an importance level higher than the other TSO users. It can be used by support in case of system problems, such as hang situations.

For classification rules, you extract information from the ICS. For TSO, you can use the qualifiers listed in Figure 48 on page 131.

4.4.2 Considerations for Migrating Batch Workloads

In WLM goal mode, a batch transaction is a batch job, even for a job having JCL PERFORM specified at the step level. It begins when the job is submitted (that is, when the JES reader processes the job), and completes when the initiator finishes the job. It means that it includes the time queued by JES waiting for an initiator but does not include output processing. So, a response time goal must include the sum of queue and execution time. The type of goal to be chosen depends on what type of batch work is being processed.

A response time is best for short, homogeneous batch jobs with a steady flow of completions within a service class. The response time goal may be average or percentile in type. Percentile goals are often preferable because they are not affected by a few very long response times.

A response time goal should be used for batch only when sufficient initiators exist for the associated job class such that most jobs do not experience lengthy queue time and where there is a steady flow of completions. Observe that WLM (before OS/390 2.4) does not control the job queue time delay. A batch response time goal is less effective if one of the following is true:

- The goal is long.
- The jobs stays in the JES queue for a long time.
- The service class period has less than 10 job completions in 20 minutes.

A velocity goal is more appropriate for jobs or job classes that remain held for a long time. When those jobs are released, the installation wants them to run quickly. Due to the long hold time, a response time goal is inappropriate. But a velocity goal will tell SRM how to run the work once it has been released.

A velocity goal is also appropriate for long-running jobs, jobs with an infrequent number of transaction completions, and for any IMS or CICS regions that an installation runs as batch jobs. The velocity goal specified for CICS or IMS regions running as batch jobs (SUBSYS=JES) applies only to the time when the region is starting when no transaction response time goals have been specified for IMS or CICS (which is similar to the situation before CICS V4).

Be aware of changes in variables that can alter the velocity, such as processor capacity, number of CPs, I/O priority and WLM batch initiator management. For more information, refer to 4.2.10, “Using Execution Velocity Goals” on page 140.

With WLM batch initiator management, introduced in OS/390 V2.4, an installation can let WLM decide the current number of batch initiators. When this function is active there can be advantages in declaring response time goals for batch jobs. If the response time is not being achieved because of delays in JES2 queues, caused by the lack of initiators, then WLM automatically starts another initiator. Refer to “WLM Batch Initiator Management” on page 80, for more information. The queue time is also included in the velocity calculation, therefore it is still quite correct to continue to use velocity goals.

However, when lots of jobs are released and go to the JES2 queue in a very short time, thereby creating long JES2 queues, the WLM Batch Initiator reaction to start new initiators may be slow. If this situation is very common for certain job classes, we recommend you continue to use JES2 initiators for those jubilates instead of WLM initiators.

If you have a service class having a throughput (service units per second) requirement, a discretionary goal with a resource group having a minimum value defined is a good option; refer to 2.2.2.4, “Select a Receiver Candidate” on page 45 for additional information. To obtain this minimum value do the following:

1. Isolate these jobs in a service class or report class.
2. From an RMF Workload Active Report collected in a peak period for that PGN, calculate the minimum unweighted service rate as follows:

$$\text{Minimum Service Rate} = \frac{\text{TCB} \times \text{SU/sec}}{\text{Interval}}$$

   The TCB field is on the SERVICE RATES column and the SRM constant SU/SEC is a field of the same report. The interval of the report is in seconds.

On the other hand, if your SLA states a throughput value as jobs/second (ETR), the mechanics are slightly different:

1. Isolate these jobs in a service class or report class.
2. From an RMF Workload Active Report collected in a peak period for that PGN, calculate the minimum unweighted service rate as follows:

$$\text{Minimum Service Rate} = \frac{\text{TCB} \times \text{SU/sec}}{\text{Interval} \times \text{Ended\_ Jobs}} \times \text{ETR}$$

A discretionary goal can be set for other work that the installation does not consider important to its business. Refer to 2.2.1.10, “WLM/SRM Discretionary Goal Management” on page 40, for more information.

In compatibility mode, installations generally use performance group periods for batch, with dispatching priority descending in each period. In a WLM policy, the first period might contain a response time goal. Succeeding periods can have longer response time goals, possibly with decreased importance, with either a velocity or a discretionary goal for the last period.

In WLM goal mode, you do not need to create a special service class to force a job to be swapped out. You do that by using the RESET operator command, with the QUIESCE option:

```
E jobname,QUIESCE
```

This command swaps out swappable work. For non-swappable work, it leaves the job swapped in, but always at the end of the dispatcher queue.

You can create a special service class with an aggressive goal, with no classification rules assigned to it. It can be useful when a job “must run now”. The operator can use the RESET operator command to assign running work to that special service class, as follows:

```
E jobname,SRVCLASS=classname
```

As this service class is only for emergency situations, it is unlikely to have enough completions for historical data, so a velocity goal is most appropriate.

Five service classes for batch is a good number. Seven is acceptable if your installation has a complex batch environment. You can group the jobs by importance and create service classes for high, medium, and low importance.

For creating classification rules, you extract information from the ICS. For batch, you can use the qualifiers for the JES subsystem listed in Figure 48 on page 131.

The process of using an IPS/ICS and RMF Workload Activity report is analogous to that used at “Considerations for Migrating TSO Workloads” on page 147, with some additional points:

- You need reports collected in peak periods for the service classes you created.
- If your installation is using PERFORM specified on the JCL of individual job steps, supported with optional control performance group (OPGN) specifications in the ICS, the RMF Workload Activity report in compatibility mode counts each step with PERFORM as one transaction. In WLM goal mode, however, the entire job is one transaction.
- From reports obtained from RMF V4.3, the velocity of PGNs is not obtained in the same way as with RMF V5 and later. In this case, you must estimate the velocity. You can do that by giving a higher velocity goal for the most important service class period, around 30%, and less for the others. When running in WLM goal mode you can adjust the velocities, once RMF obtains the velocity data.

4.4.3 Considerations for Migrating STC Workloads

Many types of work within an OS/390 system today run, or can be run, as a started task. For example, CICS systems can be run 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 kind of workload: SYSTEM and SYSSTC. Refer to 4.2.7, “Using Service Classes” on page 139, for more information about them. These two service classes are assigned by default to system functions.

The way STC address spaces are assigned a default service class is dependent upon attributes specified on the ASCRE (create address space) macro and whether the task is a system task or privileged. Table 22 shows the default assignments of service class for started task address spaces.

Table 22. STC Service Class Assignments

ASCRE Attribute	Privileged or System Task		Neither Privileged nor System Task	

HIPRI	Goal Mode:
	SC = 'SYSTEM'
	DP = x 'FF'
		Compatibility Mode:
		PGN = 0
		DP = x 'FF'
			Goal Mode:
			SC = 'SYSTEM'
			DP = x 'FF'
				Compatibility Mode:
				PGN = default for STC
				DP = x 'FF'

NONURG	Goal Mode:
	SC = 'SYSSTC'
	DP = x 'FE'
		Compatibility Mode:
		PGN = 0
		DP = PVLDP value
			Goal Mode:
			SC = default for STC
			DP = managed to SC goal
				Compatibility Mode:
				PGN = default for STC
				DP = value from PGN

where:

DP = dispatching priority

PGN = performance group number

SC = service class name

STC = started task subsystem type

4.4.3.1 SYSTEM Service Class

System tasks are those given the privileged and/or system task attribute in the IBM-supplied program properties table or in the SCHEDxx PARMLIB member; see the SCHEDxx in *OS/390 MVS Initialization and Tuning Reference,* SC28-1752. As can be seen in Table 22 on page 151, it is also possible for a non-system task or a non-privileged task to be classified 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 such system tasks are as follows: MASTER, GRS, DUMPSRV, SMF, CATALOG, RASP, XCFAS, CONSOLE, IOSAS, SMXC, ANTMAIN, JESXCF, ALLOCAS, IXGLOGR, WLM plus others.

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. For an example, see Figure 50 on page 153.

If you want to control system address spaces, you can define a service class for them setting up a classification rule in the STC subsystem type. If you do that, you must be very sure of what you are doing. The system address spaces that you change lose the high dispatching priority attribute and run at the dispatching priority of the assigned service class period. The high dispatching priority attribute can be restored by one of the following methods:

- By using the RESET operator command:

```
E stcname,SRVCLASS=SYSTEM
```

- By changing the classification rules to explicitly classify the started task to SYSTEM and activate the policy.

Note: These methods only work for systems at OS390 V2 R4 and above.

MASTER, INIT and WLM always run in the SYSTEM service class and cannot be reassigned via the service definition or via the RESET command.

4.4.3.2 SYSSTC Service Class

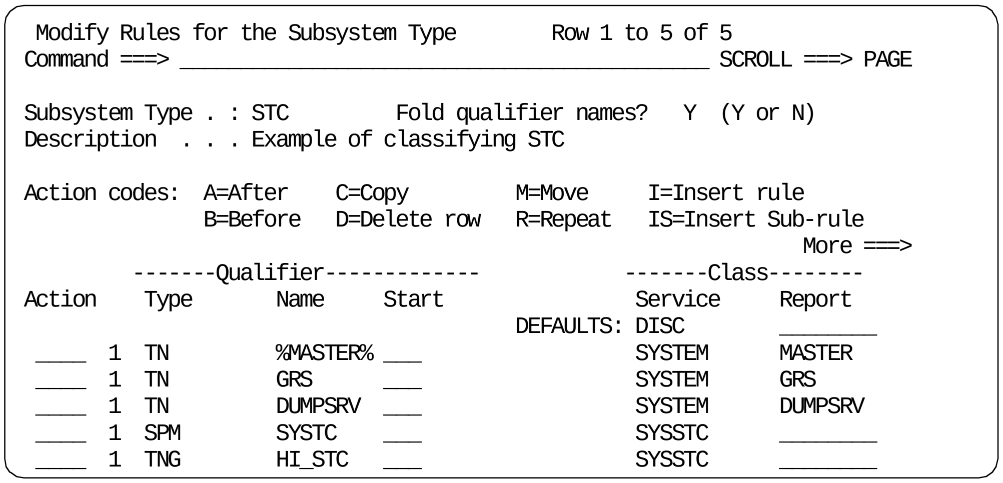
SYSSTC is the default service class for started tasks not in the SYSTEM service class, unless you provide another default in SUBSYS=STC.

The SYSSTC service class has DP=254, IOP=254 (APAR OW19265 changed this from DP=253, IOP=253) and no specific storage protection because they are cross-memory. 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 (VTAM, JES, DB2) should run. If you want to define a service class for some of these address spaces, a high importance and a high velocity are recommended.

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 will not cause any performance issues. We recommend that you assign high-importance tasks here such as VTAM, JESx, VLF, LLA, IRLM, TSO, RMF/CMF, 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. See examples in 3.3.8, “Classification Groups” on page 115, and 3.3.9, “Classification Rules” on page 117. These sections describe how STC classification works in OS/390 V2 R4 or later. For systems with an OS/390 level earlier than OS/390 R4, refer to “Using a Sysplex with some Pre-OS/390 R4 Systems”, in *OS/390 MVS Planning: Workload Management*, GC28-1761 for guidance on STC classification.

A CPU-intensive started task is not appropriate for SYSSTC, since the task could use a large amount of processor cycles. However, if your processor is lightly loaded, or in a 6-way, 8-way, or 10-way MP, SYSSTC might be appropriate, because that one task may not affect the ability of the remaining processors to manage the important work.

Figure 50 provides an example of classification rules for started tasks.



```
 Modify Rules for the Subsystem Type       Row 1 to 5 of 5
Command ===> ________________________________________ SCROLL ===> PAGE

Subsystem Type . : STC        Fold qualifier names?   Y  (Y or N)
Description  . . . Example of classifying STC

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: DISC        ________
 ____  1  TN         %MASTER% ___                 SYSTEM      MASTER
 ____  1  TN         GRS      ___                 SYSTEM      GRS
 ____  1  TN         DUMPSRV  ___                 SYSTEM      DUMPSRV
 ____  1  SPM        SYSTC    ___                 SYSSTC      ________
 ____  1  TNG        HI_STC   ___                 SYSSTC      ________
```

Figure 50. Classification Rules for STC

In Figure 50, DISC is the default service class. Separate reporting is used for Master, GRS and DUMPSRV. When you explicitly declare system address spaces and leave the service class blank, the address space goes to the SYSSTC service class, even if it is not the default service class. In other words, a system address space will always have SYSTEM or SYSSTC service class assigned. Non-system STCs that are not classified go to the default service class (which is DISC, in the example).

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 as a maximum. You could increase the velocity goal value, according to the importance of the group. You can obtain the velocity based on the installation's compatibility mode RMF reports by assigning a Report PGN in IEAICSxx using the same classification group structure.

Usually started tasks are long-running address spaces. But if you have short-running STCs, with at least 10 completions in 20 minutes, you can create a service class for them, assigning response time goal with percentile.

For a STC service class of low importance, you can assign a discretionary goal. Since OS/390 V2.6, a discretionary goal is managed in a different way; refer to 2.2.1.10, “WLM/SRM Discretionary Goal Management” on page 40, for more information.

Since the OLTP regions (CICS, IMS,DB2,and so on) are long-running batch jobs or started tasks, they should be assigned a velocity goal. When setting a velocity goal for OLTP regions, it is best to set a goal that will enable the regions to run sufficiently so the transactions they serve will meet the service level objectives.The manner in which WLM manages these regions depends on whether their releases exploit the WLM services provided since MVS SP 5.1.

Whether these regions exploit them or not, when they start up, they are classified as batch jobs or started tasks and are assigned a service class. Once classified, like all other address spaces, these regions are then managed by the MVS

Workload Manager to meet their specified velocity goals. During this time period they are regarded by the Workload Manager as “non-servers”.

Once an OLTP region from an exploiting product starts processing transactions, WLM manages the region according to the goals of the transactions it serves. During this time period the region is regarded by the Workload Manager as a “server”. You must specify a goal for these regions to ensure timely initialization during start-up and proper treatment when the region is not processing any transactions for a long period of time.

OLTP regions that are not exploiting the MVS workload management services must be given a goal which ensures that the transactions they are processing achieve their response time objectives.

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 sense in giving 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 the possibility of 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 extract information from your ICS. You can use the following qualifiers:

- Accounting Information (AI)
- Perform (PF)
- Perform Group (PFG)
- Priority (PRI)
- Subsystem Parameter (SPM)
- Transaction Name (TN)
- Transaction Name Group (TNG)
- Userid (UI) and
- Userid Group (UIG)

After migrating to goal mode, do not worry when you see some address spaces like IRLM and VTAM with low execution velocity. This can happen when idle time is high since 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.

4.4.4 Considerations for Migrating CICS Workloads

Before attempting to set a goal for CICS transactions, an installation must first determine if the level of the CICS processing these transactions exploits the workload management services. These services allow SRM to be aware of transaction starts and completions, as well as the response time being achieved. All of those are necessary to compare to a goal, and to influence a decision to change resource allocation. The OS/390 Workload Manager is only able to manage these transaction and CICS regions processing these transactions if they exploit specific OS/390 WLM services provided since MVS/ESA SP 5.1. CICS/ESA V4.1 and higher exploit WLM services. Earlier levels of CICS subsystems can of course be run, but that is accomplished with a velocity goal for the address spaces, rather than a goal for the interactive work, or transactions running in the regions.

The CICS function for OS/390 WLM causes negligible impact on CICS virtual storage. A great benefit of using WLM is that you no longer have to continually monitor and tune CICS to achieve the desired response time. Also, WLM produces performance reports that you can use to establish reasonable performance goals and for capacity planning.

Note: Watch out for increased CPU usage by the WLM address space due to a high CICS MAXTASK setting. For CICS 4.1 and later releases, WLM collects delay samples for each performance block. Because the number of performance blocks created is based on the MAXTASK value (a value of 100 means 100 performance blocks created per region), a MAXTASK value that is too high can cause a large sampling overhead when a CICS workload is switched to goal mode. If MAXTASK has been set to an arbitrarily high value, it should be reduced to a true “high water mark” value, plus a buffer.

For many years, CICS has had the ability to route transactions from one CICS region to another. CICS provides two mechanisms to do this:

- Multiple region operation (MRO) to route transactions between CICS regions on the same OS/390 image. Since CICS V4, MRO is able to connect CICS regions in different OS/390 images within a sysplex, using XCF as service provider.
- Inter-system communication (ISC) to route transactions between regions on multiple OS/390 images. ISC uses a VTAM connection.

Today, a CICS system with only a single CICS address space is rare. For capacity and availability reasons, the majority of installations using CICS have already implemented MRO, splitting off specific functions into a separate CICS regions, for example:

- Terminal-Owning regions (TOR) for terminal functions. A TOR routes the incoming transactions to the best candidate application owning region (AOR) within the AOR clone set. In this way, a TOR can help with workload balancing. To make this balancing more effective, an installation can write its own dynamic distribution routine or use CICSPlex/SM.
- Application-Owning Regions (AOR), where the actual application code runs. Typically AORs are defined in clone sets, running a number of applications.
- File-Owning Regions (FOR) to give multiple CICS address spaces concurrent update access to VSAM and BDAM files and CICS tables. In a Parallel Sysplex environment, with CICS TS 1.1 and later and DFSMS 1.3, you can choose to use VSAM Record-Level Sharing (RLS) to access VSAM files instead of having an FOR. VSAM RLS improves availability and performance due to a true data sharing implementation.

Figure 51 on page 156 presents an MRO configuration in a sysplex environment, using VSAM RLS.

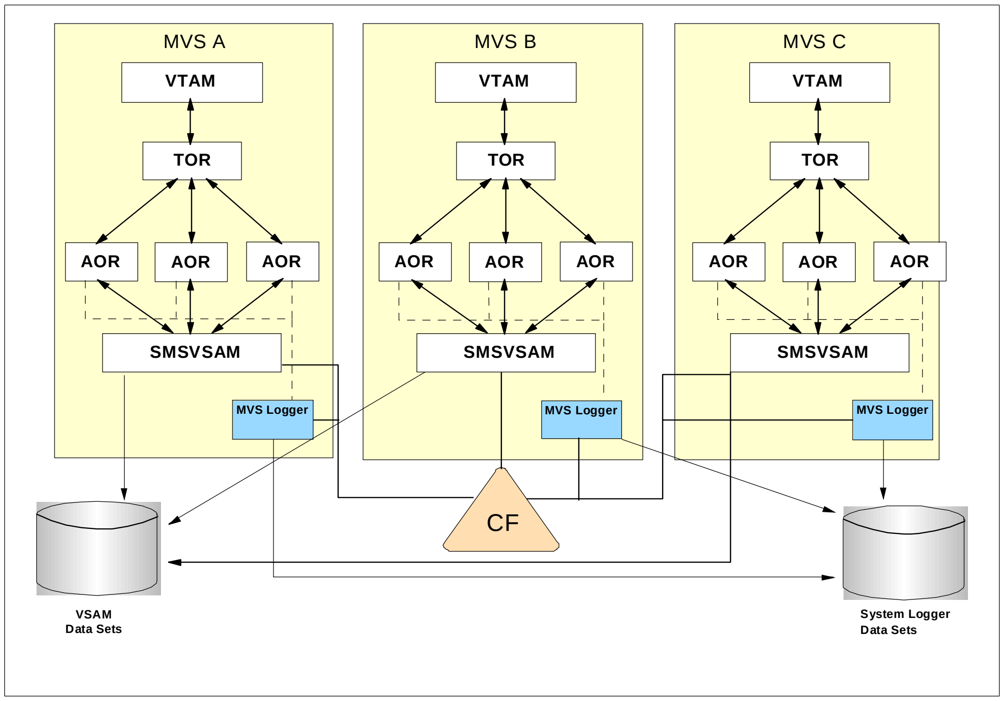


Figure 51. CICS in MRO Configuration with VSAM RLS

Installations that do not work with CICS in an MRO configuration can migrate to WLM goal mode as well. You can find very good information on how to migrate to an MRO configuration in *OS/390 Parallel Sysplex Application Migration*, GC28-1863.

To migrate to WLM goal mode, you separate your CICS transactions into groups that share common characteristics that make them meaningful for your installation to manage or monitor as a group. In WLM terminology, you group CICS transactions by similar importance and goals (for example, all CICS work, or all CICS order entry work, or all CICS development work).

You can create service classes for groups of transactions with similar performance goal and business importance. However, do not mix the following in the same service class, or AOR 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 one with an average response time of 0.5 seconds.
- Transactions with different levels of business importance.

In an OS/390 CICS environment, it is the address space that receives resources and has priority. A CICS transaction runs with the resources and priority of the CICS region. So, it is recommended to have more AOR regions than service classes on each OS/390 image to make the WLM management task easier. This decreases the chance of having transactions with different goals being executed in the same address space. If it does happen, the transactions with less aggressive goals get a "free ride", while WLM tries to honor the more aggressive and important goals.

For service classes associated with CICS transactions (SUBSYS=CICS), you *cannot* define the following:

- Velocity goals. This is 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 being delayed.
- Multiple periods are not supported for transactions in the CICS subsystem environments because service units are accumulated to the address space, not the individual transactions. Therefore, WLM cannot track a duration for those transactions.
- Discretionary goals. Since you cannot define periods, there is no point in classifying transactions to have this goal type.

You can define an average response time goal or a response time with percentile. If your installation uses CICSPlex SM, you may want to use average response time goals rather than percentile goals to take advantage of goal-based trasnaction routing within CICSPlex/SM. Percentile goals are, however, still preferred for any workload that can have a few unusually long transactions distorting the average response time. With percentile goals, CICSPlex/SM uses the "shortest queue" algorithm for transaction routing. For more information, see "Using CICSPlex System Manager" on page 160 and *OS/390 MVS Planning: Workload Management*, GC28-1761.

A CICS transaction begins when the initial CICS region (TOR) receives a message from VTAM and ends when its processing finishes, leaving the CICS address space (TOR) to deliver the result to VTAM. If the transaction is routed to other CICS regions via MRO, the time spent processing by those regions is accounted to the original transaction. When the transaction is routed via VTAM intersystem communication (ISC) links, the perspective from a WLM viewpoint is that they behave differently than when they are connected via MRO. With ISC, both the TOR and the AOR are receiving a request from VTAM, so each believes it is starting and ending a given transaction. So for a given user request routed from the TOR via ISC to an AOR, there would be two completed transactions. The following is an example of a transaction running in MultiRegion Operation (MRO) without ISC:

1. The transaction begins in a Terminal-Owning Region (TOR).
2. It is routed via MRO to an Application-Owning region (AOR).
3. It can be routed via MRO to other CICS regions like File-Owning Region (FOR), Queue-Owning Region (QOR), other resource-owning region, or other product region as DB2 or DL/I.
4. The transaction returns to the TOR from the AOR.

5. The TOR returns the result to VTAM.

This can be seen graphically in Figure 52 on page 158.

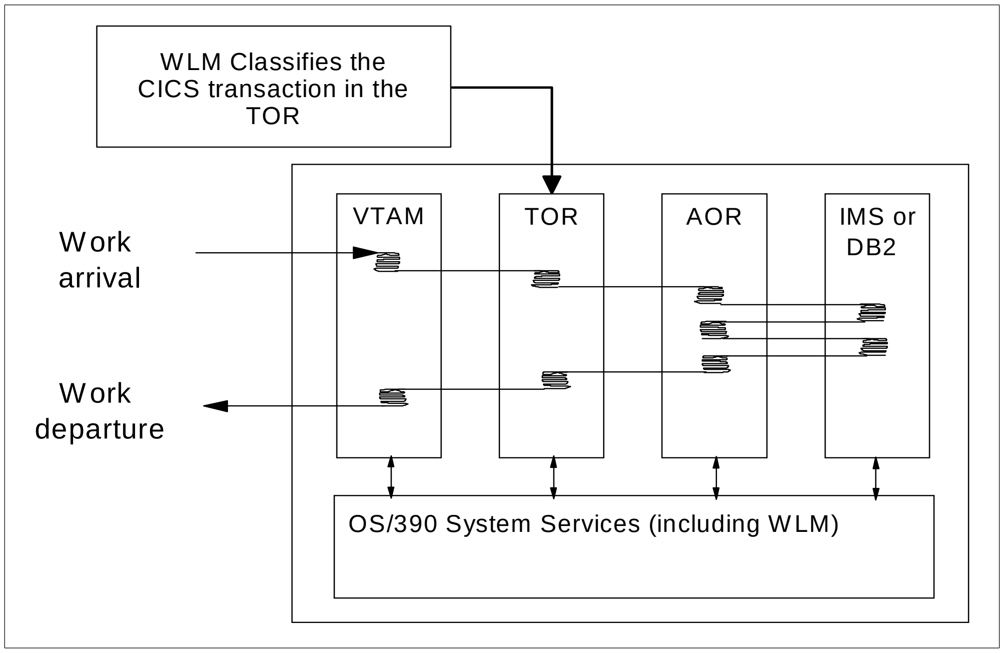

WLM Classifies the CICS transaction in the TOR
VTAM
TOR
AOR
IMS or DB2
Work arrival
Work departure
OS/390 System Services (including WLM)

Figure 52. CICS MRO Transaction Workflow

The cycle TOR-to-TOR is counted as one transaction and the time TOR-to-TOR is the response time of the transaction. Conversational transactions are an exception, because they are counted as one no matter how many TOR-to-TOR cycles occur. This is because CICS classifies a conversational transaction to WLM in the first cycle and delivers the transaction response time to WLM when the last cycle finishes.

Classically, conversational CICS transactions imply several interactions with the user. Response time for conversational transactions includes user think time as well. This think time cannot be managed by WLM, but it affects WLM decisions when it compares actuals versus goals. Response time goals with percentile can help since they permit a response time with large amounts of idle time (think time, in this case) to be ignored. In such cases, for service classes composed of conversational transactions, give a percentile value that will exclude the conversational transactions.

A response time is composed of service time plus queue time. For CICS transactions running in MRO, the queue time is composed as follows:

$$\text{Queue Time} = \text{TOR Queue Time} + \text{TOR Execution Time} + \text{AOR Queue Time}$$

As you can see, the TOR execution time is also considered queue time.

4.4.4.1 Gathering Information in Compatibility Mode

Before you set goals for CICS transactions, you can determine CICS current response times by running CICS in compatibility mode with an arbitrary goal. For this purpose, use the SRVCLASS parameter, provided by MVS 5.1 or higher in the ICS:

```
SUBSYS=CICS,
SRVCLASS=CICSHI,RPGN=100
SRVCLASS=CICSMED,RPGN=101
SRVCLASS=CICSLOW,RPGN=102
```

This parameter associates a service class with a report performance group, when running in compatibility mode. You can assign only RPGN, not PGN. You would then:

1. Define a service policy with a default service class (or classes) for your CICS transactions, and specify an arbitrary response time goal. Because you are still running in compatibility mode, this goal will not be used.
2. Define classification rules for the service class (or classes).
3. Install the service definition.
4. Activate the service policy in compatibility mode. Your system will still run in compatibility mode, but with an active policy. The policy will only be used to classify the CICS transactions into a service class.

The average response time for transactions within the service classes is reported under the report performance group in the RMF Monitor I Workload Activity Report. This information helps you set realistic goals for running your CICS transactions when you switch to goal mode. To start with, consider setting average response time goals, based on this compatability mode data. Once you have gathered response time distribution information in goal mode, consider changing to using a percentile-based goal. The data produced by the RMF reports is organized by service class and contains reasons for any delays that affect the response time for the service class (for example, because of the actions of a resource manager or an I/O subsystem). From this information, you may be able to determine any configuration changes to improve performance. Refer to “Understanding RMF workload manager data”, in *CICS Performance Guide*, SC33-1183.

In an MRO configuration, transactions are usually associated with a terminal. In such cases, work requests originate in TORs and are classified in TORs. Therefore, you could consider defining classification rules for the TOR only. You would use the TOR name as the subsystem instance name in the classification rules.

Note: You cannot classify MRO transactions by the AOR that processes them.

For transactions not associated with terminals, you must classify them according to the region where they begin. For CICS environments not in an MRO configuration, you can classify according to CICS regions to which the transactions are related.

You can set up a hierarchy of classification rules. When CICS receives a transaction, WLM searches the classification rules for a matching qualifier and its service class or report class. Because a piece of work can have more than one work qualifier associated with it, it may match more than one classification rule. Therefore, the order in which you specify the classification rules determines which service classes are assigned. The first match of a transaction to a classification rule is used to assign the service class and report class. You have to define a default service class for CICS. Figure 5 on page 9 shows an example of classification rules for CICS transactions. We recommend you keep classification rules as simple as possible. For classification rules, Figure 48 on page 131 shows the qualifiers you can use.

Velocity is the goal type recommended for a CICS region. If you do not classify your CICS transactions (that is, you have no classification rules for the CICS subsystem type), the transactions run with the goal of the CICS region. However, transactions have better storage protection when being managed with a response time goal than with execution velocity.

Not classifying CICS transactions, that is, using just a velocity goal for the CICS region, can be adequate for environments where the majority of applications are conversational. WLM does not include the idle time in the velocity calculation. As a result, the think time does not affect WLM’s actions.

If you choose not to classify your CICS transactions, then put all AORs and TORs in the same service class. Small differences in velocity goals make WLM work unnecessarily to micro-manage the velocity. Remember that service class periods with a large number of address spaces (compared with the number of CPUs) should not receive a high value of execution velocity goal because it will be unachievable. For example, a velocity goal of 50 is not possible for a set of 100 busy CICS AOR address spaces in a OS/390 image with 10 CPUs.

4.4.4.2 Using CICSPlex System Manager

CICSPlex/SM is a system management tool that enables you to manage multiple CICS systems as if they were a single system. It provides a dynamic transaction routing (DTR) program that can route eligible transactions from a TOR region to a suitable AOR region selected at the time the transaction is initiated.

CICSPlex/SM does not always have a free choice of which AOR to route a transaction to. There are often constraints that you need to define when routing transactions. CICSPlex/SM honors the constraints when making decisions about which AOR to route a transaction to. This is called *workload separation*. Your CICS workload can have constraints, as follows:

- Intertransaction affinity is a relationship, of a specified duration, between transactions that requires them to be processed by the same AOR. For example, assume that transaction TRN1 was routed to AOR1. While TRN1 was running it issued an EXEC CICS GETMAIN SHARED command and placed some data into the main storage area returned from the GETMAIN command. TRN2 (which runs later) must have access to the same data in the same storage area; therefore, TRN2 must also be routed to AOR1.

  Another example is when you have a pseudoconversation made up of separate transactions, and each transaction passes data to the next transaction in the sequence using a temporary storage queue. All transactions must be processed by the same AOR and this affinity lasts for the duration of the pseudoconversation. You could avoid the affinity in this example by defining a temporary storage-owning region (TSOR). You would define the temporary storage queue in the temporary storage table (TST) in each AOR, thus causing all requests that access that queue to be function shipped to the TSOR. In a Parallel Sysplex, with CICS TS 1.1 and higher, temporary storage queues can be placed in coupling facility structures.

- Transaction-system affinity is not an affinity between the transactions themselves, but between a transaction and a particular CICS region, where the transaction interrogates or changes the properties of that CICS region. A typical example of transaction-system affinity is a transaction using the INQUIRE or SET commands.
- Installation-specific reasons for routing transactions to a specific AOR or group of AORs. Examples of this could be:
  - The installation has not fully implemented the FOR concept, that is, some files are not moved to the FOR but, for performance reasons, belong to an AOR. This means that, to avoid or reduce the incidence of function shipping requests, transactions with a large number of accesses to the files must run in the AOR where those files are defined.
  - Some exits are only activated in some of the AORs.
  - Some application programs or program packages are only installed in some of the AORs.

You can use the IBM CICS Transaction Affinities Utility MVS/ESA (program number 5696-582) to assist you in finding intertransaction affinities in your applications.

CICSPlex/SM provides a DTR program that executes in TORs and supports the following:

- Workload separation, which is the routing of particular transactions to a particular group of AORs based on any combination of user ID, terminal ID, and transaction name.
- Workload balancing, which is the routing of transactions from a TOR among a group of AORs according to the availability and activity levels of those AORs.

CICSPlex/SM works with one of two algorithms (queue and goal) to choose which AOR processes any particular transaction:

- The queue algorithm causes CICSPlex/SM to select the AOR that has the shortest queue of transactions waiting to be processed, relative to the maximum number of tasks permitted in the AOR.
- With the goal algorithm, CICSPlex/SM selects the AOR that is the most likely to enable the transaction to meet response-time goals set for it using the WLM. The goal algorithm is only available in WLM goal mode and TOR regions of CICS/ESA 4.1 or higher, when using average response time goals. If you use a response time goal with a percentile, transaction routing reverts to using the shortest queue algorithm.

Both algorithms also select the AOR that is:

- Least affected by conditions such as short-on-storage, SYSDUMP, and TRANDUMP.
- Least likely to cause the transaction to abend. This factor may be taken into account for CICS/ESA 4.1 (or later) AORs only.

- Connected to its TOR via MRO/XCF instead of a VTAM-connected AOR, when all other considerations before are equal. This is because communication MRO/XCF has a lower overhead than ISC and throughput is higher.

The only CICSPlex/SM definition change required to make full use of the WLM goal mode if you are using average response time goals is to amend the workload specification to use the goal algorithm instead of the queue algorithm. CICSPlex/SM automatically switches to the queue algorithm if the system switches from WLM goal mode to WLM compatibility mode.

In WLM goal mode, when CICSPlex/SM runs with the queue algorithm, CICSPlex/SM routes the transaction to the AOR with the shortest queue independently of what is specified in the WLM policy. WLM will try to honor the goals you gave to CICS transactions but CICSPlex/SM, with the queue algorithm, will try to maximize transactions throughput and standardize response time. Therefore, we recommend you run CICSPlex/SM with the goal algorithm when running in WLM goal mode. (However, for many customers, the need to run with percentile goals outweighs the value of the goal algorithm.)

When CICSPlex/SM runs with the goal algorithm, CICSPlex/SM calls WLM to get the goals associated with a transaction. CICSPlex/SM gets the response time goal, but not the importance. CICSPlex/SM does not call WLM for every transaction, but builds a look-aside table. When a transaction is initiated, CICSPlex/SM first checks to see if the goals for that transaction are already in the table, and only if CICSPlex/SM does not find it there does it call the WLM. However, CICSPlex/SM refreshes its service class goals tables following WLM policy changes. If all your transactions are in one service class, the goal algorithm works the same as the queue algorithm.

With CICSPlex/SM, you must use the average response time goal for your CICS transactions. CICSPlex/SM does not handle percentile goals. If you do assign response time goal with percentile to a service class consisting of CICS transactions, CICSPlex/SM reverts to using the shortest queue algorithm.

CICSPlex/SM is of particular benefit in those enterprises that are running CICS in a Parallel Sysplex, because CICSPlex SM can route transactions throughout the sysplex. As an example, consider the start of a transaction controlled by CICSPlex/SM, using the goal algorithm, in the configuration shown in Figure 53 on page 163:

1. The transaction TRNX arrives in the TOR on system MVS A.
2. The TOR uses IWMCLSFY, a WLM service, to associate the incoming work request with a service class. The service caller provides the work request qualifiers, such as user ID and transaction name. WLM then checks the classification rules and provides a service class token representing the service class and report class (if any).
3. The TOR passes control to the DTR program of CICSPlex/SM that uses the IWMWQRY WLM service to obtain the service class goal. CICSPlex/SM saves the service class goal in a table for subsequent transactions.
4. With the goal, CICSPlex/SM decides the best AOR to execute the transaction. As an example, suppose that the AOR selected belongs to MVS B.

5. The TOR in MVS A sends TRNX to the selected AOR in MVS B, using MRO/XCF.
6. TRNX executes in the selected AOR in MVS B.
7. The transaction control returns to the TOR in MVS A.
8. The transaction ends. The TOR in MVS A delivers the transaction response to the VTAM of MVS A.
9. The TOR in MVS A returns the response time to WLM.

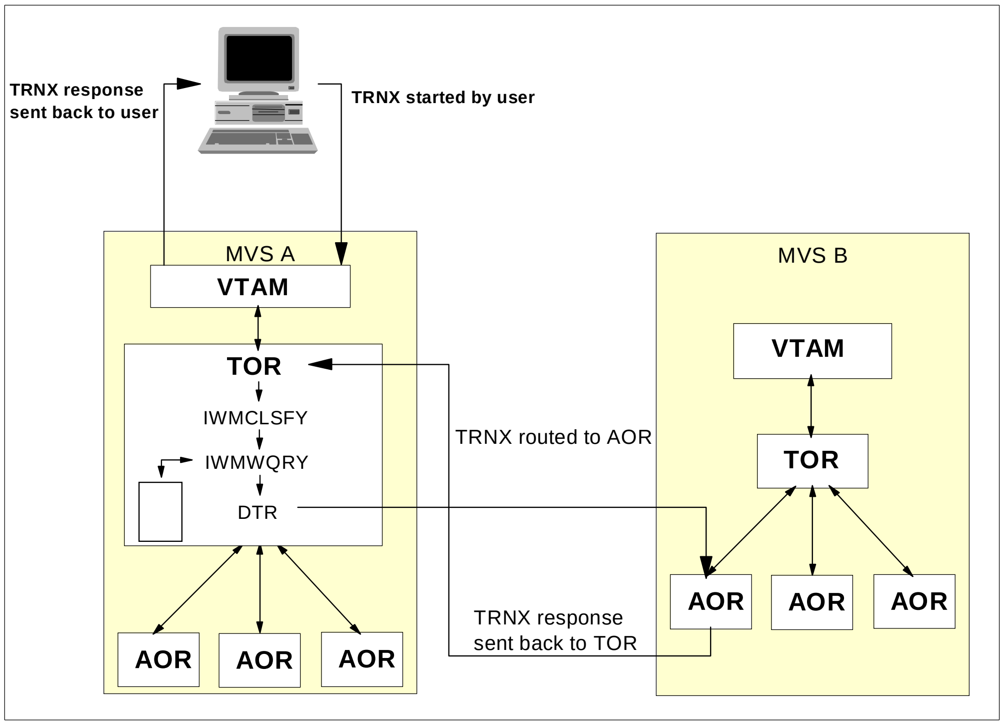


Figure 53. Routing a CICS Transaction

4.4.4.3 CICS and VTAM Generic Resource

In a sysplex you can implement the generic resources function of VTAM, which gives you added availability as well as the ability to balance the session workload. If you implemented three TORs, for example, and let them register to the same generic resource group (named CICS, for instance), VTAM will distribute the incoming session requests among all three TORs, based on installation-defined criteria. Should one of the TORs become unavailable for some reason, then the users can still log on to CICS, where VTAM now chooses between the two remaining TORs.

The multinode persistent sessions (MNPS) feature provided with VTAM V4R4 gives you even greater availability. Even with the single-node persistence provided by CICS/ESA V4R1, VTAM freezes the sessions of a failing CICS region. Therefore, if the CICS region restarts successfully, the sessions are ready to be used again and your users will only experience a short delay in response time. They will not have to log on again to continue to work with CICS.

Moreover, if a large number of sessions need to be restarted, this can place a considerable load on the network, causing additional delay. While single-node persistence requires the failing CICS region to be restarted in place (that is, under the same OS/390 and VTAM), MNPS lets you restart the region on a different OS/390 and VTAM in a different processor; the VTAM on this OS/390 image will take over the frozen CICS sessions and the users will continue to work.

MNPS makes use of CICS's existing persistent session function, with extra capability added. This extra capability is provided by the PTF for PQ01573 (for CICS V4) and PQ01878 (for CICS TS). It increases the CICS time-out, during which it waits to receive details of suspended sessions, and contains other functions designed to make CICS aware of the recovery situation.

4.4.5 Considerations for Migrating IMS Workloads

Before attempting to set a goal for IMS transactions, an installation must first determine if the product level of the IMS region processing these transactions exploits the WLM services. Starting with version 5.1, IMS provides information about IMS units of work to WLM.

IMS supports WLM by establishing the WLM monitoring environments using a performance block (PB) token and uses the report services of WLM for monitoring IMS units of work (represented by messages) to balance workload. IMS uses the same WLM interfaces as CICS for providing transaction information to WLM.

The WLM change state performance block (PB) service is used to show the current state of the transaction. The PB service codes are interpreted as follows for IMS:

- Active: The transaction is executing an application program.
- Free: This is not reported by WLM.
- Idle: The transaction is waiting for work.
- Waiting - I/O: IMS is waiting on I/O.
- Waiting - Lock: IMS is waiting on a lock request.

To migrate to WLM goal mode, use the recommendations for CICS running in MRO, with no CICSPlex/SM, as presented in 4.4.4, "Considerations for Migrating CICS Workloads" on page 154.

To help you set initial goals for IMS service classes, you get data from RMF Workload Activity Reports in compatibility mode by setting an RPGN for the IMS subsystem in the ICS. You must set one RPGN for each service class you defined, as follows:

```
SUBSYS=IMS,
SRVCLASS=IMSHI,RPGN=103
SRVCLASS=IMSMED,RPGN=104
SRVCLASS=IMSLOW,RPGN=105
```

After activating your policy, in compatibility mode, you get, from the RMF Workload Activity report, the average response time for each service class (associated to a RPGN). Based on the standard deviation, you can estimate a response time with a percentile goal value. If you are not sure about this type of goal, you can choose the average response time value and later, in goal mode, use the RMF Workload Activity report in goal mode with response time distribution information to determine a percentile to specify.

For classification rules, see the qualifiers supported by the IMS subsystem in Figure 48 on page 131.

There have been a number of APARs against IMS 5.1 to correct problems with the response times being reported to WLM. Make sure you keep current on service.

4.4.6 Considerations for Migrating DDF Workloads

The distributed data facility (DDF) is an optional feature of DB2 that allows a DB2 application to access data at other DB2s and at remote relational database systems that support IBM's Distributed Relational Database Architecture (DRDA). In addition, DDF allows applications running in a remote application requester environment that supports DRDA to access data in DB2 subsystems.

In the past, all DDF requests ran with the priority of the DDF address space. There was no way to prioritize among requests according to their business value. The consequence of this was that the priority of the DDF address space was set so that the processing requirements of the requests most valuable to the business are satisfied. Also, all accounting information was related to the DDF address space.

Figure 54 presents one transaction running in a DDF environment prior to the implementation of enclaves:

- DDF schedules a local SRB, which is a non preemptible dispatchable unit.
- The local SRB runs in cross-memory mode in the DB2 address space.
- The DDF address space is the unit of accounting and priority.

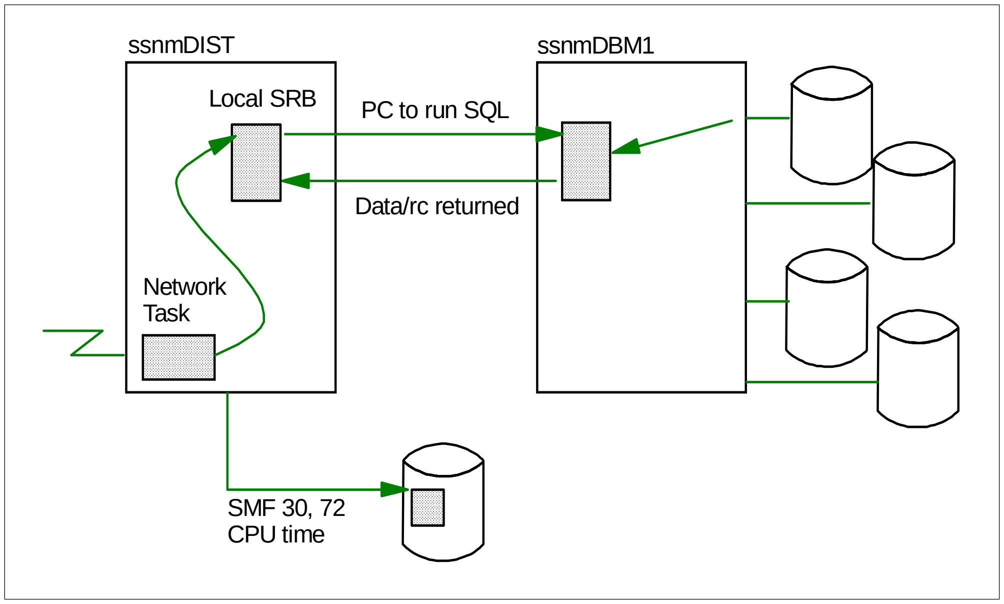


Figure 54. A Transaction Running in a DDF Environment Prior to Enclaves

To address this problem, in DB2 V4.1 and up, DDF uses preemptible dispatchable units, called enclave SRBs. They were introduced with MVS/ESA V5.2.0 to replace local SRBs. Each DDF transaction is associated with an enclave, which permits individual transactions to be managed with their own priority.

Figure 55 on page 166 shows how the DDF transactions are managed with enclaves:

- A request arrives from the network into the DDF address space.
- DDF creates an enclave.
- With a policy active, WLM associates the enclave with a service class.
- DDF schedules the SRB into the enclave.
- When the dispatcher finds the enclave SRB, it is dispatched with its own unit of priority and accounting.

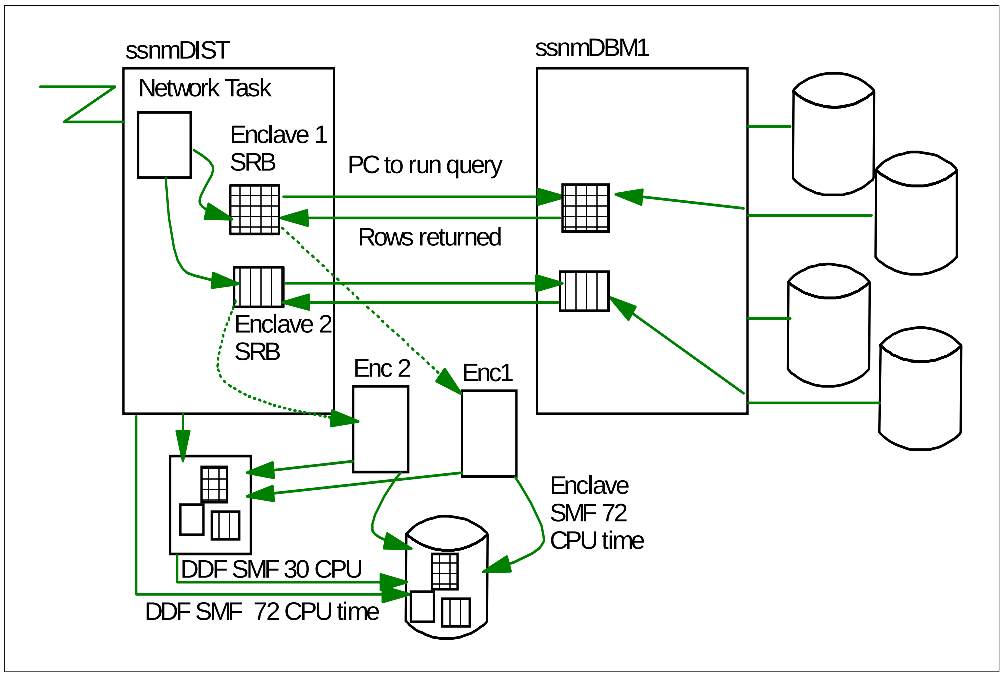


Figure 55. Managing DDF Transactions with Enclaves

4.4.6.1 Compatibility Mode Considerations

In WLM compatibility mode, with no active service policy, enclaves have the DDF address space priority, but you still have the benefit of the task not being starved for CPU time as was the case before with non-preemptible SRBs. For more information about enclaves, see 2.3.4, “Enclaves” on page 62.

To manage DDF transactions in WLM compatibility mode, you must do the following:

1. Define a policy with service classes and classification rules for subsystem type DDF. Table 48 on page 131 shows the qualifiers you can use to classify your DDF enclaves to WLM.

Also classify the DDF transactions to a default service class, because enclaves with no service class are associated to the PGN of the DDF address space.

The goals are not important. While your system is in compatibility mode, the goals are not used, only the classification is performed.

2. Install the policy in the WLM CDS.
3. Update ICS and, in the DDF subsystem entry, associate PGNs to the service classes you defined. Suppose you defined two service classes: DDFPRD, with two service class periods, and DDFHI. Then define them in ICS as follows:

```
SUBSYS=DDF,
    SRVCLASS=DDFHI,PGN=18,RPGN=100
    SRVCLASS=DDFPRD,PGN=18,RPGN=101
```

   Do not mix other workloads in the same PGN or RPGN. Only one RPGN is allowed per service class.

   All work requests in the service class associated with the PGNs are processed according to the PGNs' controls. Transactions in service classes not associated to PGNs are assigned to the PGN of the DDF address space.
4. In the IPS, define the PGNs and give them dispatching priority, as in the following example:

```
PGN=18,(DP=F41,DUR=10K),
       (DP=M6)
```

   Time slicing, domains and storage isolation controls are ignored for enclaves.
5. Use the SET command to activate your changes in the ICS and IPS.
6. Activate the policy you defined to WLM. You will still be in compatibility mode. But now the DDF enclaves are classified according to the classification rules you defined, and have the priorities you specified in the IPS. The RMF Workload Activity Report shows, by performance group, the number of transactions, velocity, and average response time for each service class. This data will help you define goals to your DDF enclaves for running in WLM goal mode.

   If you want to manage you DDF workload with the priority of the DDF address space, simply do not define PGNs in the ICS and IPS. However, you still need RPGNs to get the RMF reports to help you set the service class goals. For example:

```
SUBSYS=DDF,
    SRVCLASS=DDFHI,RPGN=100
    SRVCLASS=DDFPRD,RPGN=101
```

4.4.6.2 Goal Mode Considerations

Before migrating to goal mode, you need to execute the previous steps to get the data for setting your goals. You should define a default service class for DDF enclaves. In WLM goal mode, DDF enclaves not otherwise classified go to the SYSOTHER service class, which has a discretionary goal.

Note: The WLM application requires you to supply a default service class. The only way for a DDF transaction to default to SYSOTHER is if you have not entered *any rules at all* for DDF.

Assign goals and importance to service classes to be associated with enclaves according to how important they are to your business.

Before setting your goals, refer to Figure 56 on page 168 in order to understand how a DDF request is treated according to the DB2 options used. Note the following options:

- With THREADS=INACTIVE and RELEASE(COMMIT), DDF creates one enclave per active interval, and response time *does not* include think time. In this case, you can use a response time goal and use multiple service class periods.
- With THREADS=ACTIVE or RELEASE(DEALLOCATE), DDF creates one enclave for the life of the thread, and response time *does* include think time. In this case, you should not use response time goals or multiple periods. Instead, the use of a velocity goal is more appropriate.

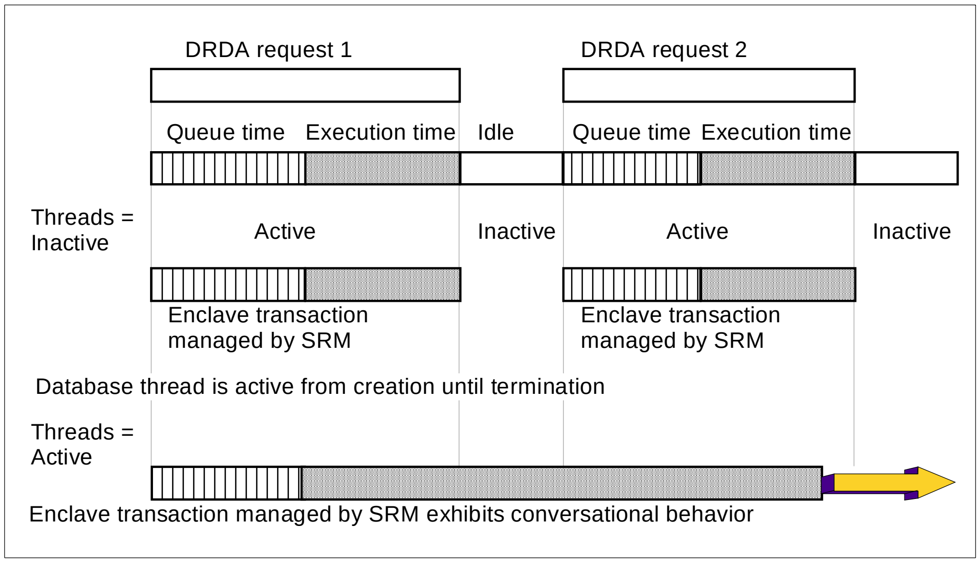

DRDA request 1
DRDA request 2
Queue time
Execution time
Idle
Queue time
Execution time
Threads = Inactive
Active
Inactive
Active
Inactive
Enclave transaction managed by SRM
Enclave transaction managed by SRM
Database thread is active from creation until termination
Threads = Active
Enclave transaction managed by SRM exhibits conversational behavior

Figure 56. Enclave Transaction According to DB2 Options

4.4.7 Considerations for Migrating APPC Workloads

WLM recognizes APPC work that has been scheduled by the APPC scheduler (ASCH). If your installation uses a different APPC scheduler, consult that product's documentation for information on classifying its transactions. Figure 57 presents APPC communications using an ASCH scheduler:

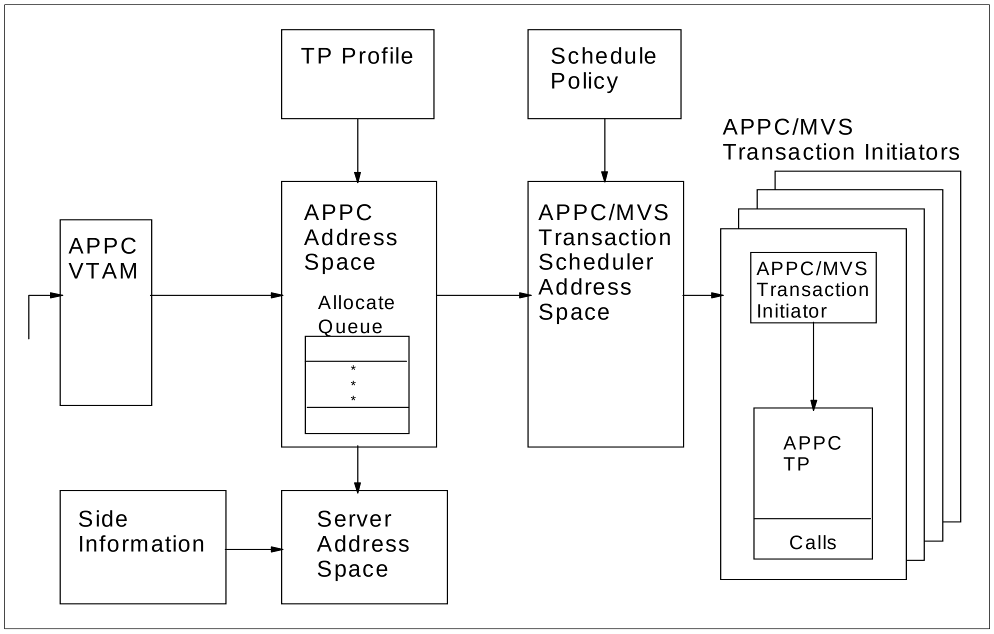


Figure 57. APPC Environment Using ASCH Scheduler

We recommend you assign the ASCH and APPC address spaces to the SYSSTC service class. This ensures that the scheduler can quickly process requests for new APPC transaction programs.

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. (This is 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.

The type of goal to be chosen depends on the type of APPC Transaction Program (TP):

- If an installation is sure that all TPs will process a single request and have at least 10 completions in 20 minutes, then the service class for these transactions could 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 may stay connected to the network for an indeterminate time period. In these cases, a velocity goal should be used.

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.

For classification rules, you can use the qualifiers presented in Figure 48 on page 131.

For setting goals for TP transactions, you obtain information in WLM compatibility mode from the RMF Workload Activity Report. For the subsystem ASCH, you should define in IEAICSxx the same classification you use in the classification rules. Define an RPGN for each service class you created. For example:

```
SUBSYS=ASCH,PGN=5
ACCTINFO=D944,RPGN=41
TRXNAME=MAIL,RPGN=42
USERID=SPECIAL1,RPGN=43
TRXCLASS=J,RPGN=44
```

From an RMF Workload Activity report, which represents peak periods for the ASCH workload, you obtain the number of ended transactions, the average response time, and velocity. This information will help you in setting the service class goals.

After migrating to goal mode, you can adjust your goals by using the RMF Workload Activity Report for the APPC service classes you defined.

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 APPLn instances. If all WLMs in the sysplex that have applications in this generic resources group are running in goal mode, WLM checks its service policy and checks the loading on each of the APPLn address spaces. It then chooses the instance that it calculates as being the best able to handle a new session while meeting the service goals. WLM tells VTAM which APPLn it has chosen.

4.4.8 Considerations for Migrating Non-IBM Products

Consult the product's documentation to see if the product exploits WLM services. If it does exploit WLM, follow the instructions. Remember that for those products that generate transactions in their address spaces, WLM only recognizes these transactions if the products inform WLM about them.

If the product does not exploit WLM services, you can probably classify it as running under JES or STC subsystems. Then classify the workload according to the subsystem and the characteristics of the workload the product generates. Typically a velocity goal is most suitable.

4.4.9 Considerations for Migrating Roll-Your-Own (RYO) Workloads

If your installation has a homemade subsystem, it can be modified to use WLM services. Refer to 2.5, “The Subsystem Work Manager Service” on page 68 to get information about WLM services.

For each WLM service you need, there is a set of assembler macros or C language interfaces you use to invoke it. *OS/390 MVS Programming: Workload Management Services*, GC28-1773, provides instructions and examples of how to invoke these services.

If you do not intend to change your RYO applications, you can still migrate to goal mode. You can probably classify them as running under JES or STC subsystems, like other non-IBM software as described in 4.4.8, “Considerations for Migrating Non-IBM Products” on page 170. Then you classify the workload according to the characteristics of the workload the application generates.

Chapter 5. Considerations for New Workloads

In Chapter 4, “Migrating to WLM Goal Mode” on page 129 we described migration considerations for the traditional workloads in an OS/390 environment, like TSO, batch, CICS, and IMS. But OS/390 has become more and more open in the last few years, transforming from a mainframe to an enterprise server. Many new server functions are available under OS/390. The WLM support for these new server functions allows these applications to benefit from the full strength of performance management that OS/390 can provide.

In this chapter we provide guidelines for exploiting the WLM functions for the client/server, Web-based, and UNIX System Services workloads.

5.1 DB2-Based Workloads

While DB2 is not a new workload, there are new 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 WLM in order to improve performance management.

5.1.1 DB2 Sysplex Query Parallelism

In this section we describe how sysplex query parallelism exploits the new function of WLM.

5.1.1.1 What is Query Parallelism?

Query parallelism is a way of reducing the elapsed time for long-running queries which 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 may have query functions such as joins on multiple tables, column and scalar functions, and grouping and ordering of data.

Query parallelism was implemented in stages, as shown in Figure 58 on page 172. DB2 Version 3 introduced query I/O parallelism, which enables a much greater I/O throughput for I/O-intensive queries.

Query CP parallelism in DB2 V4 extended the benefits of parallel processing to CP-intensive queries. Query CP parallelism is designed to make full use of the total processing power in a central processing complex (CPC) and take advantage of the increased I/O bandwidth from partitioned table spaces.

DB2 V5 provides additional types of queries that can use parallelism, and it also introduces sysplex query parallelism, which extends parallel processing to allow a single query to use all 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.

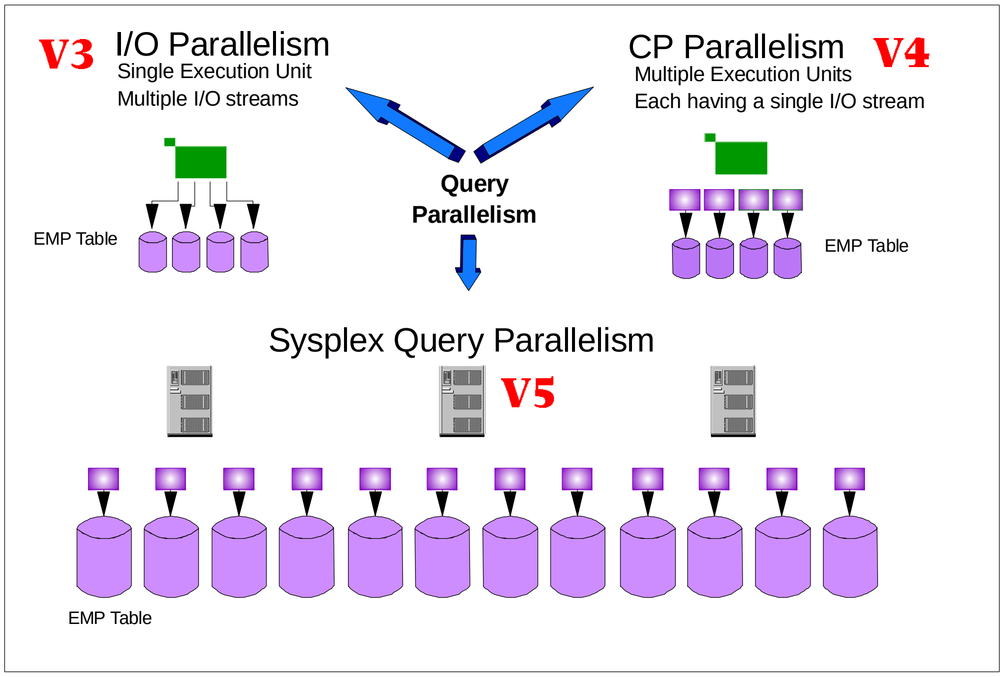


Figure 58. The Evolution of Parallelism in DB2

5.1.1.2 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 V5 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 is running DB2 Version 5 and at least MVS/ESA Version 5 Release 2. For this DB2 member, the COORDINATOR parameter must be 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 run time, the query runs within a single DB2.

To be considered as candidates for assisting the parallelism coordinator, the other members must run DB2 Version 5 and at least MVS/ESA Version 5 Release 2. YES must be 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 not relevant.

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 59 shows all members of a data sharing group participating in processing a single query.

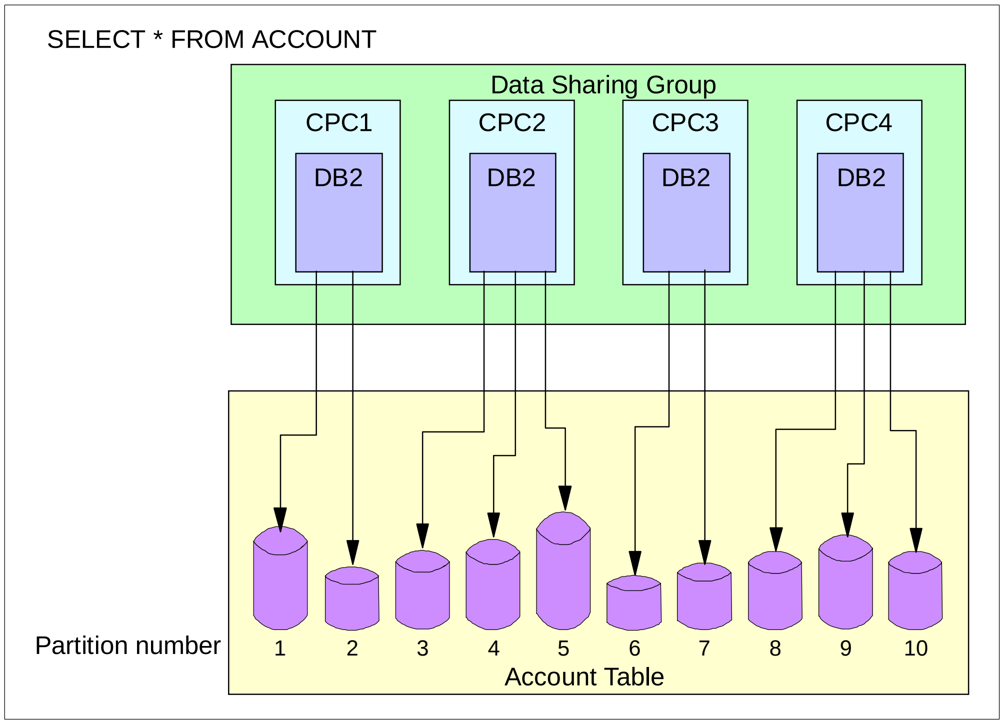


Figure 59. Parallel Query Processing

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 for OS/390 Version 5 Data Sharing: Planning and Administration*, SC26-8961.

5.1.2 WLM Considerations for Sysplex Query Parallelism

It is important to define how you want MVS to handle the work for sysplex query parallelism.

Group-wide goals for workload management should be defined for both work that originates on a particular DB2 and 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 in the past. However, you must also classify work that runs on the assistant DB2. If you do not do this, the following things can occur:

- If you are running in compatibility mode, the part of the query that runs on the assistant has the same priority as the DB2 database services address space (DBM1). This is probably not what you want.
- If you are running in goal mode, 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.

As you can see, is it not necessary to be in WLM goal mode in order to use sysplex query parallelism. But only DB2 in conjunction with MVS WLM goal mode can assign query requests on a very granular level using the WLM classification rules in goal mode. So you can assign priority to your query at a user level, plan level or package level.

Additionally, only in WLM goal mode can you define a goal for your query. You can define response time, velocity or discretionary goals. In compatibility mode you define a static assignment of resources to your query. Period aging can be defined in both modes to help facilitate the flow of short-running and long-running work in the system. In goal mode, consider the use of resource groups in order to limit the amount of resources 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 will fall through multiple performance periods and get fewer resources, the longer they run. This will automatically control 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.

5.1.3 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).

To set workload management goals for parallel queries on a DB2 *assistant* subsystem, you have to do the following things, depending on the WLM mode.

We explain the definitions necessary for both compatibility mode and goal mode in the following sections.

5.1.3.1 DB2 Sysplex Query Parallelism in WLM Compatibility Mode

In WLM compatibility mode, you classify and attribute a performance group number (PGN) to the query running in the coordinator DB2 and to the queries running in the assistant DB2s. There are two ways for doing this:

1. In the IEAICSxx member, you associate a PGN with the query running in the coordinator address space. This PGN is associated under the subsystem type where the query originated, and you do not mention any PGN for the assistants. In this case the enclaves associated with the split-up queries run with the original PGN of the assistant DB2 address space.
2. In the IEAICSxx, you associate a PGN with the query running in the assistant address space. First, you define in SUBSYS=DB2 a service class associated with a PGN using the SRVCLASS parameter for the assistant address space. In the IEAIPSxx, you give the priorities of the split-up query enclave via PGN keywords namely DP and IODP. WLM uses the classification rules in the active policy to establish a service class that matches the external properties of the arriving query. You need an active WLM policy for this even in compatibility mode.

   DB2 creates enclave SRBs for sysplex queries, and enclaves are always classified to a service class via the active service policy, even in compatibility mode.The service class must be the same as indicated in IEAICSxx. See the following example for the assistant DB2:

   ```
   IEAICSxx:
   SUBSYS=DB2,
   SRVCLASS=DSHIGH,PGN=15

   IEAIPSxx:
   PGN=15,(DP=F92,DUR=2K)
   ```

   For more information about enclaves, see 2.3.4, “Enclaves” on page 62.

5.1.3.2 DB2 Sysplex Query Parallelism in WLM Goal Mode

When migrating to WLM goal mode, besides 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 migration are as follows:

- Create classification rules and service class for the coordinator query.

  The pieces of the query which 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 Chapter 4, “Migrating to WLM Goal Mode” on page 129 for guidance in how to convert from WLM compatibility mode into WLM goal mode for such workload types.

- Create classification rules for the assistant enclave query.

  If in WLM compatibility mode you did not define a specific PGN for the split-up enclave queries running in the assistant DB2 address spaces, then you do not have to migrate an ICS definition to classification rules. In this case you define classification rules under the DB2 subsystem type, using any of the 16 possible work qualifiers (see Figure 48 on page 131 for the complete list), 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.

If in WLM compatibility mode you already defined a specific PGN for the split-up enclave queries running in the assistant DB2 address spaces, then you already have a classification rule and a chosen service class. Review these to ensure they meet your goal mode requirements.

- Create service classes for the assistant enclave query.

  Define one or more service class representing the goal for your split-up enclave queries. If you are in OS/390 R.3 or higher, you can use the same service classes as for the coordinator. If you in an earlier release of OS/390, you have to define different service classes for the assistant. The goal can be the same but the names have to be different. This is because WLM in these releases cannot manage enclave work and address space work with the same service class name.

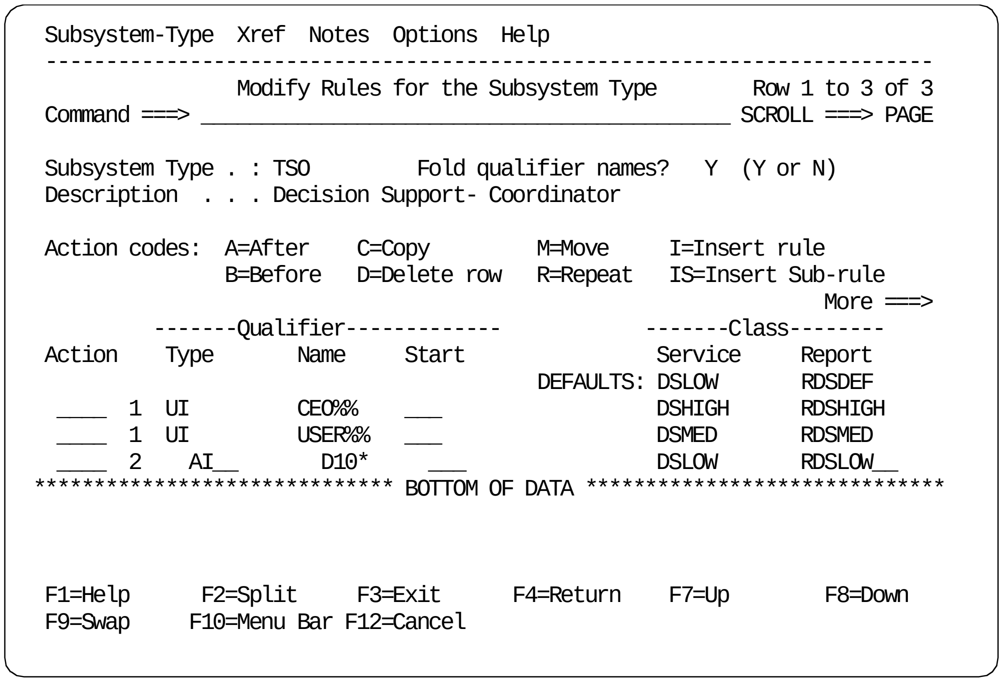
Defining your Classification Rules

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.

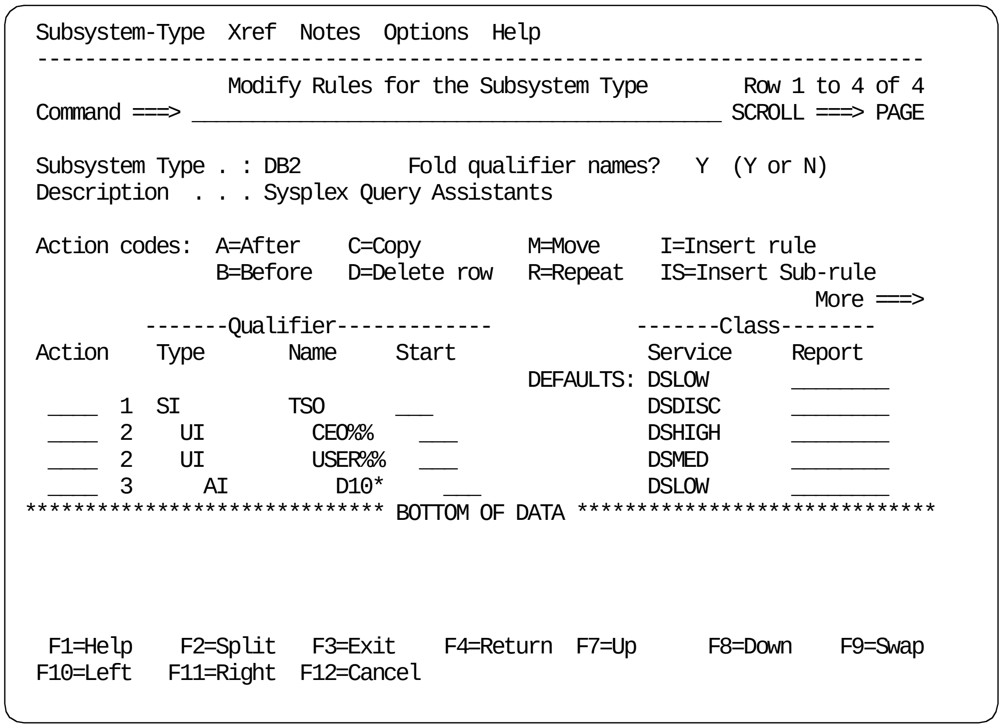
Figure 60 on page 177 and Figure 61 on page 178 show examples of how to define your WLM classification definitions for your sysplex query parallelism workload. One definition is for the coordinator, where the query originates locally from TSO, and the other definition is for the assistants, where the distributed parts of the query run within the data sharing group.



```
 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__
****************************** BOTTOM OF DATA ******************************



 F1=Help      F2=Split     F3=Exit      F4=Return    F7=Up        F8=Down
 F9=Swap     F10=Menu Bar F12=Cancel
```

Figure 60. WLM Classification Rules for Coordinator

Figure 60 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 USERxx 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.



```
  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       ________
 ***************************** BOTTOM OF DATA ******************************




   F1=Help    F2=Split   F3=Exit    F4=Return  F7=Up      F8=Down    F9=Swap
  F10=Left   F11=Right  F12=Cancel
```

Figure 61. WLM Classification Rules for the Assistant DB2 Subsystems

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 Figure 61 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.

If your system is not running at OS/390 R.3 or a subsequent release, you cannot use the same names for your service classes in the classification rules for the coordinator and assistants.

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 Figure 48 on page 131, and *OS/390 MVS Programming: Workload Management Services*, GC28-1773.

Defining your 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 performance 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 10 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 in compatibility mode. 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.

Attention

Be aware that the response time for a query could vary widely depending on the degree of parallelism. This may 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 you specify so that period switch occurs after the desired number of service units have been consumed.

Use a discretionary goal if all your other workload is more important than your queries.

5.2 DB2 Stored Procedures

In this section we explain a new DB2 function — stored procedures, and how it exploits WLM to manage the performance of individual client requests. Stored procedures is a special feature of DB2 and provide substantial improvements for distributed data access.

5.2.1 What are Stored Procedures

The distributed environment brought to modern data processing a series of problems. 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 on the DB2 server. They can be invoked by a client application. This function enables a new kind of client/server application having 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 62 shows an example of how DB2 stored procedures are invoked.

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 (dsnxSPAS).

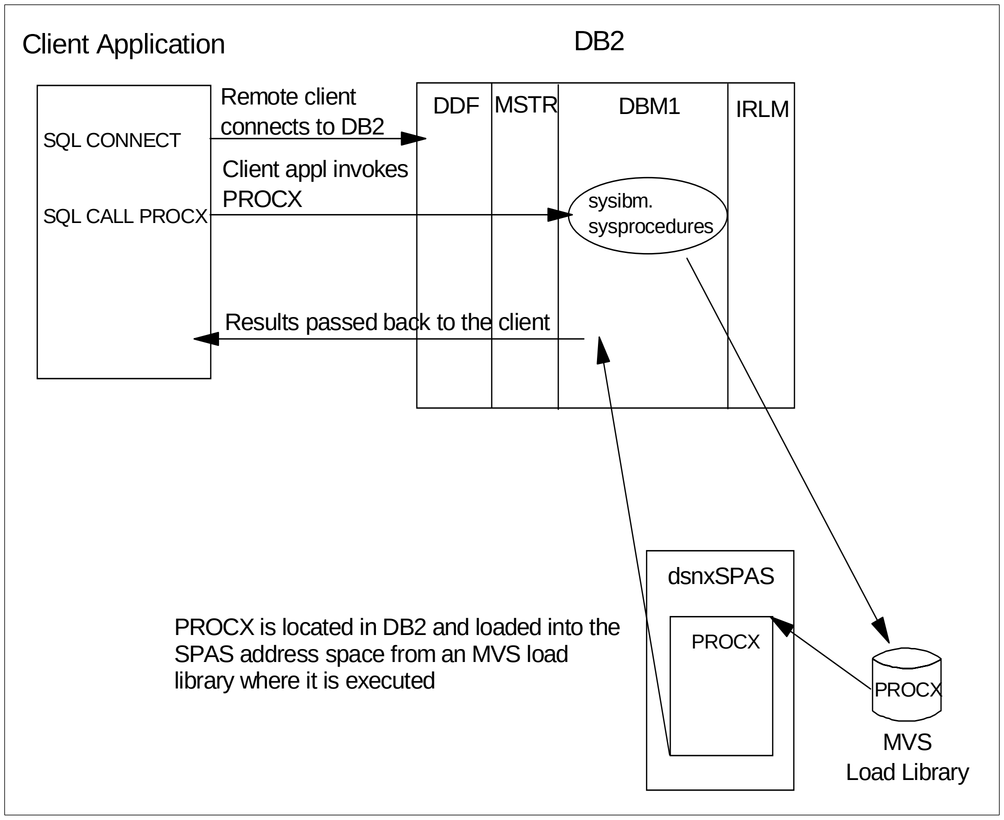

Client Application
DB2
SQL CONNECT
SQL CALL PROCX
Remote client connects to DB2
Client appl invokes PROCX
DDF
MSTR
DBM1
IRLM
sysibm. sysprocedures
Results passed back to the client
dsnxSPAS
PROCX
PROCX is located in DB2 and loaded into the SPAS address space from an MVS load library where it is executed
PROCX
MVS Load Library

Figure 62. Example of Stored Procedures

In DB2 V4.1, you have 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. We do not recommend specifying a large number because of storage constraints. For further guidelines, refer to *DB2 for MVS/ESA V4 Installation Guide,* SC26-3456.

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.

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. For further discussion of DB2-established and WLM-established stored procedure address spaces see *DB2 for OS/390 Version 5 Performance Topics*, SG24-2213.

5.2.2 WLM Considerations for DB2 Stored Procedures

WLM is involved with DB2 stored procedures in three ways:

- It enforces goals (if in WLM goal mode) or assigns priority values (if in WLM compatibility mode) to the DB2 stored procedure enclaves. These enclaves also support TCBs.
- In goal mode, it controls the number of DB2 stored procedure address spaces automatically, based on your service class goals. WLM starts additional address spaces when the queue time of a stored procedure justifies it; the work is missing its defined goal. This function is called *server address space management.*

  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 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.

5.2.2.1 Considerations For Enforcing WLM Goals

The following statements apply to both WLM goal and WLM compatibility modes:

- When you access a stored procedure from a local client application (TSO, CICS, IMS, and so on), you do not need to define a specific performance requirement. All such calls inherit the performance requirements from the calling address space. This is 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.
- Only when you access the stored procedure remotely via the distributed data facility (DDF) address space do you need to define specific performance requirements for the independent enclave under the DDF subsystem type.

5.2.2.2 DB2 Stored Procedures in Compatibility Mode

While staying in compatibility mode you have the ability to use some of the WLM functions.

1. You can classify your stored procedure requests coming from the DDF address space and give these requests different resource access
2. You can use the capability of spreading the stored procedures over multiple address spaces.

In WLM compatibility mode, you may classify and attribute a PGN to the enclave transaction accessing the stored procedure via DDF. There are two ways of doing this:

- You do not mention any PGN for the DDF enclave associated with the stored procedure. In this case the enclave runs with the original PGN of the DB2 stored procedure address space.
- In the IEAICSxx member, you define a service class in SUBSYS=DDF associated with a PGN using the SRVCLASS parameter. In the IEAIPSxx member, you specify the priorities of the enclave through the PGN keywords such as dispatching priority, I/O priority and duration. WLM uses the classification rules in the active policy to establish a service class that matches the external properties of the enclave. You need an active service class with classification rules for the DDF subsystem. The service class must be the same as indicated in IEAICSxx. See the following example:

```
ICS member:
SUBSYS=DDF,
SRVCLASS=DSHIGH,PGN=15

IEAIPSxx member:
PGN=15,(DP=F92,DUR=2K)
```

You need an active WLM service policy to take advantage of WLM's support of multiple server address spaces for DB2 stored procedures in compatibility mode. The service definition has to have at least one application environment definition for stored procedures. Furthermore you need to assign the application environment to the stored procedures. For more information, see also 5.2.3.1, "Using WLM-Established Address Spaces" on page 183.

WLM does not automatically manage the number DB2 stored procedure address spaces in compatibility mode. The management of the server address spaces for each of the DB2 application environments must be done manually. The installation must determine how many address spaces to start for each application environment and must manually terminate address spaces no longer required in order to save system resources.

For information about application environments, see 2.6.1.1, "Application Environments" on page 78.

5.2.3 WLM Definition for DB2 Stored Procedures

This section describes the WLM definition using DB2 stored procedures in goal or compatibility mode.

5.2.3.1 Using WLM-Established Address Spaces

If you decide to use WLM-established stored procedures address spaces, you must define a WLM application environment. WLM uses these definitions for its server address space management.

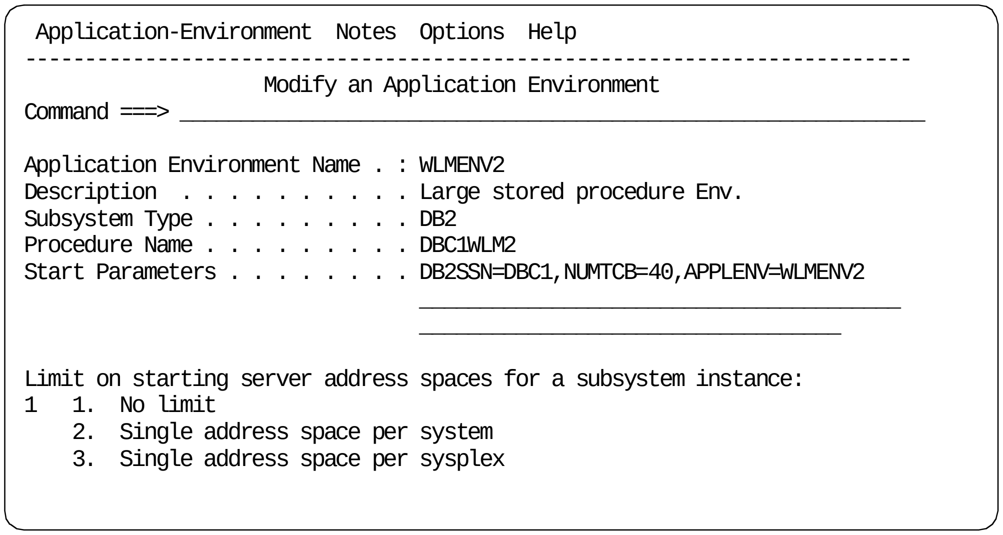
There are other tasks that must be completed before a stored procedure can run in a WLM-established stored procedures address space. One of these tasks is to assign the stored procedures to an application environment. You do this by updating the WLM_ENV column of the SYSIBM.SYSPROCEDURES table in your DB2 subsystem. The following statement could be used, 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 for OS/390 V5 Administration Guide*, SC26-8957.

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.

Figure 63 shows an example of how to define your WLM application environment.



```
 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
```

Figure 63. Application Environment Definition for DB2 Stored Procedures

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 which 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. Since 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.

To prevent creating unnecessary address spaces, create only a relatively small number of WLM application environments and service classes.

WLM routes work to stored procedures 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 5.2.3.2, “Define Your Classification Rules” on page 185. For information about the address space manager function of WLM see 2.6.1, “Server Address Space Management” on page 77.

Figure 64 shows how DB2 stored procedures uses the server address space management function of WLM:

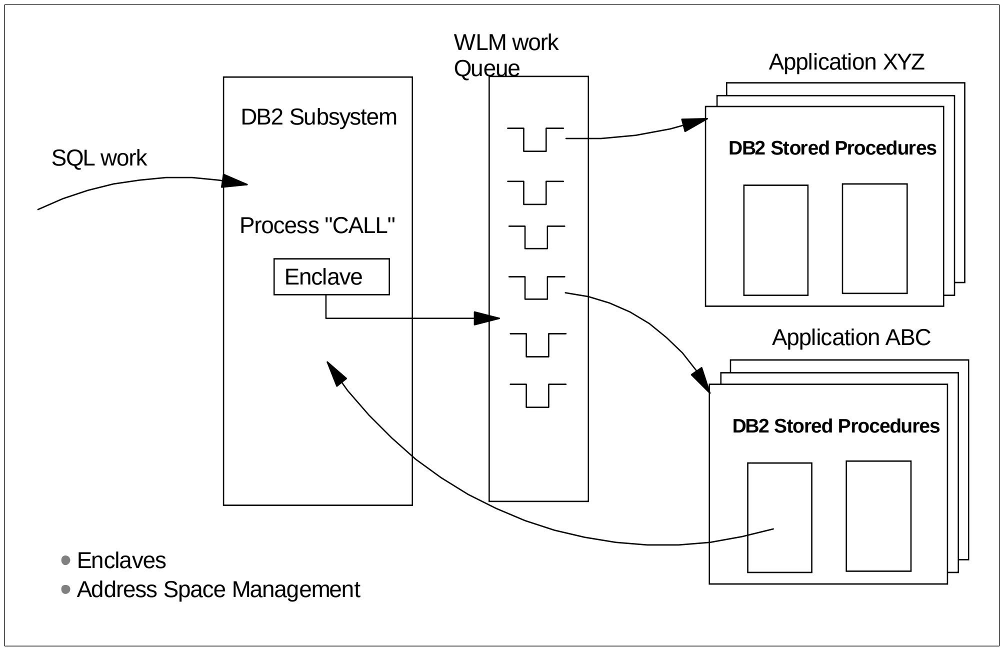


Figure 64. DB2 Stored Procedures and WLM

The steps are as follows:

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 may 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 which 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 (DSNX9STP). 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 will join 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.

5.2.3.2 Define Your Classification Rules

You can access a stored procedure from a local client application (TSO, CICS, IMS) or through remote services provided by the DB2 DDF facility.

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.

You have to define classification rules for the incoming DDF work.

Attention

In goal mode, 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 may not be what you want.

If in WLM compatibility mode you already defined a specific PGN for the enclaves running in the stored procedure DB2 address spaces, then you already have classification rules and an associated service class.

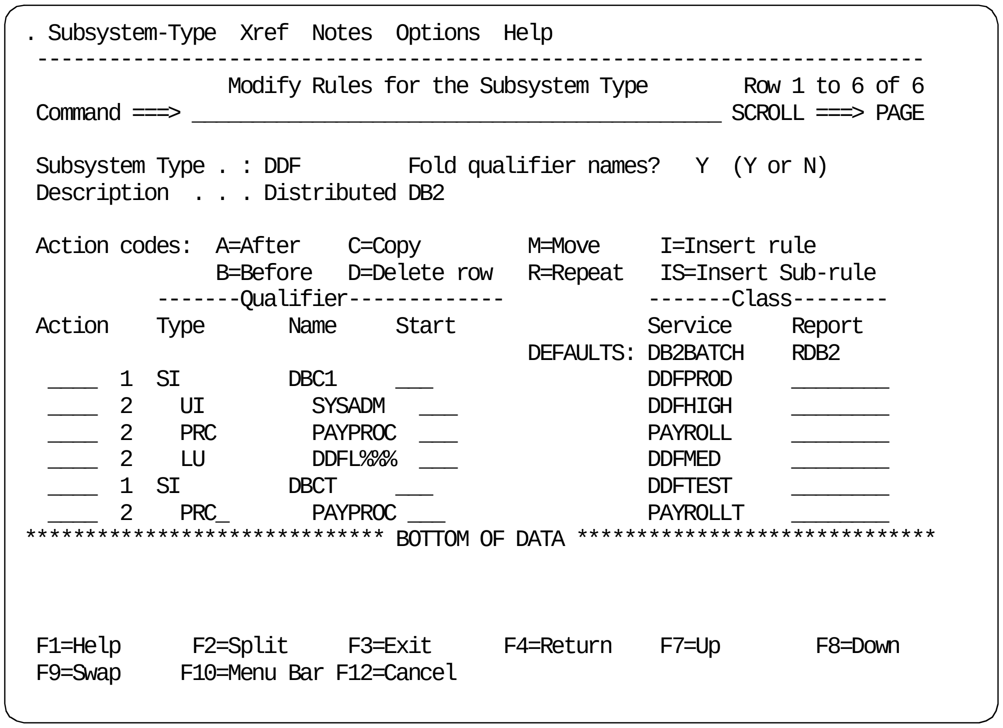
If in WLM compatibility mode you did not define a specific PGN for the enclave stored procedures, then you did not have an ICS from which to define classification rules. In this case you have to define classification rules for SUBSYS=DDF using the possible work qualifiers (there are 11 applicable to 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 for OS/390 Version 5 Administration Guide*, SC26-8957.

Figure 65 shows an example of the classification of DDF threads.



```
. 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    ________
****************************** BOTTOM OF DATA ******************************



 F1=Help      F2=Split     F3=Exit      F4=Return    F7=Up        F8=Down
 F9=Swap     F10=Menu Bar F12=Cancel
```

Figure 65. Classification Rules for Stored Procedures Invoked From Remote Clients

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 in the test system DBCT it gets the service class PAYROLLT. All other users in the production system DBCT have the service class DDFTEST.

5.2.3.3 Defining Your Service Classes

Attention

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 % if you have fewer than 10 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 in compatibility mode. 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.

5.2.3.4 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 determines how quickly DB2 is able to perform operations associated with managing the distributed DB2 workload, such as adding new users or removing users that have terminated their connections.

5.3 Internet Connection Secure Server/Domino Go Web Server

The Domino Go Webserver for OS/390 is the http server for OS/390. You can use it in different configuration modes as required by your business requirements. There are stand-alone server, multiple server and scalable server modes.

In this section we explain the modes of operation and the impact of WLM for Domino Go Webserver.

5.3.1 Domino Go Webserver Modes of Operation

The stand-alone server mode means that only a single instance of a server runs on one system. Both multiple server mode and scalable server mode enable you to run with more than one server on a single system. However, the two modes are very different.

5.3.1.1 Stand-Alone Server

A stand-alone server is simply an independent server running in a system. Each stand-alone server runs independently of any other server and must have its own unique TCP/IP port to use for listening.

In this mode WLM is not used to classify and control incoming individual requests. WLM handles the stand-alone server as an address space. You can let the started task default in the SYSSTC class, or you can specify your own classification rule for your Domino Go Webserver STC with a velocity goal depending on your business requirement.

5.3.1.2 Scalable Server Subsystem

In this mode the tasks of the Domino Go Webserver are processed by two types of address space, a queue manager and queue servers. The queue manager address space receives the incoming requests and, depending on the definition, it puts the request in a WLM queue or processes the request itself. The queue server address spaces can be managed by WLM and are responsible for processing requests from specific WLM queues.

You can divide your work into various work queues known as *application environments*. This method can be used to group work with similar data access requirements, and can also be used to isolate work requests from each other. To do this, the scalable Web server uses the address space manager function of WLM. For more information about the WLM server address space management see 2.6.1, “Server Address Space Management” on page 77.

We describe this mode in more detail in 5.3.2, “WLM Considerations for the Scalable Web Server Subsystem” on page 188.

5.3.1.3 Multiple Servers

Multiple server mode refers to a setup where more than one server runs on a single system. The servers can be multiple stand-alone servers, multiple scalable server subsystems, or a combination of both. Each server must run independently of any other server, have its own unique port for listening, and a separate configuration file.

From a WLM perspective there is no difference between a stand-alone server or the WLM exploitation by a scalable server to the multiple server mode, so we will not explain this mode in more detail. For more information about this mode see *Domino Go Webserver Release 5.0 for OS/390: Webmaster’s Guide*, SC31-8691.

5.3.2 WLM Considerations for the Scalable Web Server Subsystem

A scalable server in Domino Go Webserver is a Web server that is set up to use WLM to distribute work requests among a set of WLM-managed address spaces.

The Domino Go Webserver uses the address space management function, described in 2.6.1, “Server Address Space Management” on page 77, in order to manage the address spaces and distribute the requests.

By using WLM functions Domino Go Webserver can separate work into queues (application environments), assign specific performance objectives (goals) for them, and let WLM allocate the necessary resources for accomplishing these goals. In goal mode, WLM is able to start new server address spaces to reach the goals.

With a scalable server, incoming requests are received by a queue manager, placed on a WLM-managed queue, and processed by a queue server. (Note that there are also situations where the request is processed solely by the queue manager.)

Both the queue manager and the queue server are Domino Go Webserver address spaces, but with different characteristics:

- The queue manager is started manually or by automation, and the queue servers can be started by WLM on demand in goal mode.
- The queue manager processes all requests not matched against a specific application environment, while the queue server only processes requests matching a specific WLM application environment and service class.
- Only the queue manager can serve as a proxy cache, because the cache index is stored on the queue manager and is not accessible from the application servers.

Attention

Before Domino Go Webserver Release 5.0, all Secure Socket Layer (SSL) requests are handled by the queue manager. If most of your requests are SSL requests, you would not see much advantage by using the scalable server configuration.

In Release 5.0 this is changed, and SSL requests can now be distributed to one or more queue server as well.

Queue manager and queue server address spaces have to run in the same OS/390 image. The queue manager passes sockets to the queue server, and there is no way to do this to another OS/390 without re-engineering the TCP/IP stack. However, the queue server can run plug-ins that access sysplex-wide resource managers like CICS, IMS or DB2. FastCGI processes can be configured as “remote” and the queue server will pass work to them via TCP/IP.

Figure 66 on page 190 shows how a client request is processed in a scalable server environment.

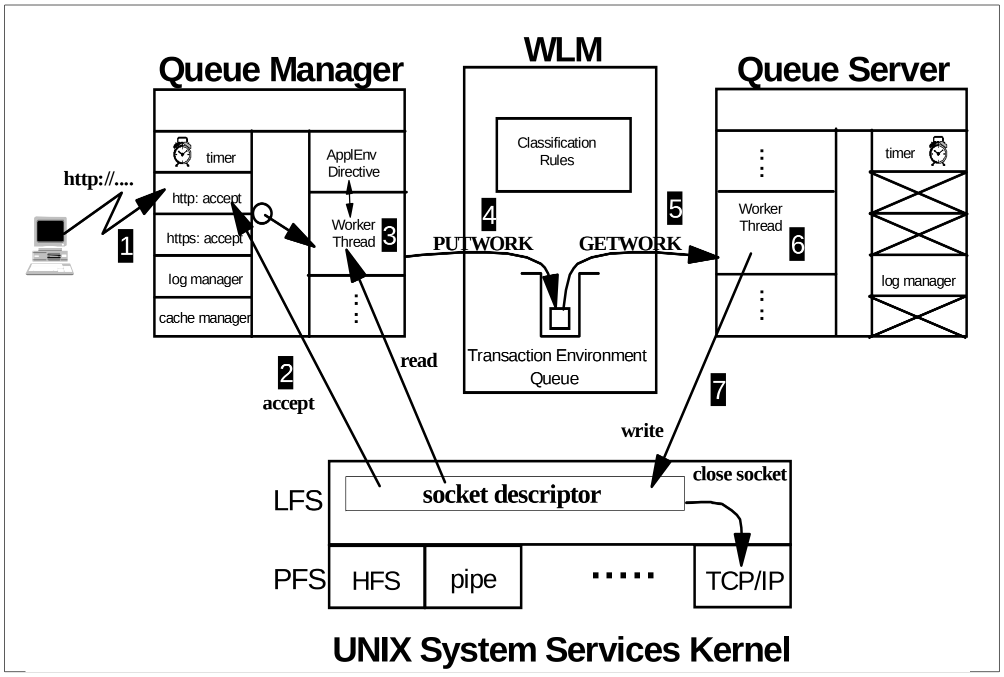


Figure 66. Processing Client Requests in a Scalable Server Subsystem

When the client connects to the Domino Go Webserver, the request is processed as follows, from a WLM perspective:

1. The client connects to the queue manager and sends a request.
2. An accepting thread receives a new socket descriptor for that request from the Logical File System. The accepting thread passes the connection to a worker thread for processing, or queues it internally if all worker threads are busy.
3. The worker thread starts reading the HTTP request package from the socket. The requested resource is matched against ApplEnv directives in the configuration file of the Domino Go Webserver. If no ApplEnv directives match, the request is processed by this worker thread.
4. If the worker thread determines that this request matches an ApplEnv directive and should processed by a queue server, it issues a PUTWORK request to the UNIX service kernel. As part of this PUTWORK request, an enclave is created for the HTTP request and the work is classified. The classification is done based on the WLM classification rules and the transaction class defined for that request in the ApplEnv directive. The transaction class, if defined, is used under the IWEB subsystem type as the TC (transaction class) work qualifier in the classification rule. After work classification, the HTTP request has an assigned service class.

   At the end of the PUTWORK request WLM queues the HTTP request to the appropriate application environment. The combination of service class and application environment builds an application environment queue. Every unique combination of application environment and service class has its own WLM queue.

   For more information about application environments, see 2.6.1.1, “Application Environments” on page 78.

5. A worker thread in the queue server responsible for that specific application environment queue issues a GETWORK request to WLM to get the request from the queue. For each queue, at least one queue server address space has to be available. If there is no server address space available WLM starts one. The queue server can be managed by WLM or started manually. For more information see 5.3.5.3, “Using WLM Started Address Spaces” on page 198.
6. The work thread on the queue server processes the request. The consumed service units during request execution are charged to the service class of the enclave representing the request.
7. After execution, the response is sent back to the client and the enclave is deleted. If the request was part of a persistent connection, this thread returns to step 3, possibly transferring the connection to another application environment. This causes multiple requests for a persistent connection to be executed under the same WLM enclave. For more information about persistent connection see 5.3.5.5, “Considerations for Using Persistent Connections” on page 199.

The scalable server brings a number of advantages over a single address space design, including:

- Scalability; a single address space is limited to the number of tasks that it can run concurrently and the amount of virtual storage available to it. Allowing work to be spread over multiple address spaces allows significantly more work to be accepted and processed.
- Availability; you can set up your server to separate different types of work into different queue server address spaces. In particular, you can separate ICAPI programs from simple GET request processing from proxy processing, as well as separating test programs from production programs. Failure in one address space will not result in an outage of the Web server. The queue manager remains available while a queue server address space is restarted automatically by WLM.

5.3.3 Define Your WLM Service Classes for the Scalable Web Server

You have to define your service goals using the WLM administration application.

You can define response time, velocity and discretionary goals for Domino Go Webserver requests as required by your business requirements.

You can also establish performance periods for your Web server requests.

By establishing multiple performance periods, you can cause the Web server request’s performance objectives to change, based upon the request’s service consumption. So, a long-running request can be given a lower importance and not cause short-running requests to miss goals.

The definition of your service classes depends on your business requirements: it is not a one-time process to reach your requirements. Instead it is an ongoing process of gathering your performance data, analyzing it, and defining new performance goals as necessary.

For a starter, you can use the sample WLM service policy. This policy is shipped with your Domino Go Webserver in the file IMW.V4R6M1.SIMWTBL1.

You can read this sample policy by using option **1** in the Choose Service Definition panel of the WLM administration application as shown in Figure 67:

```
Choose Service Definition

 Select one of the following options.
 1_  1.  Read saved definition
     2.  Extract definition from WLM
         couple data set
     3.  Create new definition

  F1=Help       F2=Split      F5=KeysHelp
  F9=Swap      F12=Cancel
```

Figure 67. Read saved WLM Definitions

There is a default service class for the Web server requests called FAST in this policy. This class has the following goals assigned:

```
 * Service Class FAST - fast class
   Class assigned to resource group RGROUP2

   Created by user IBMUSER on 1994/03/10 at 13:55:24
   Base last updated by user IBMUSER on 1994/03/10 at 13:55:24

   Base goal:

     #  Duration   Imp  Goal description
     -  ---------  -    ----------------------------------------
     1  2000       1    Average response time of 00:00:00.100
     2             4    80% complete within 00:00:10.000
```

Figure 68. Service Class FAST from the Default Service Definition

The FAST service class has two performance periods. The duration of the first one is 2000 service units and has a goal of an average response time 0.1 seconds. The second (and last) one has an average percentage response time goal of 10 seconds. Because this is the last period, it has no duration. The first period has an importance of 1, which tells WLM that it is very important to reach this goal.

If you have a problem with the response time of requests served by this class, you can:

- Make the duration for the first period longer. There is a rule that 80% of the transaction should finish in the first period. Use RMF reports to check your values.
- Consider using percentile goals if a few long transactions are skewing the average response time.
- Decrease the response time for the second period.
- Add a new (third) performance period.

You should define service classes for Web requests that have the same performance requirements. But take care not to define too many service classes.

In a scalable server subsystem using the address space server function, WLM starts at least one server address space for every unique combination of application environment and service class which could in turn result in storage shortages, or unnecessary load on the system.

So keep it simple and do not define too many service classes.

5.3.4 Define Your WLM Classification Rules for Scalable Web Server

You categorize your workload into service classes using classification rules. The categorization is based on work qualifiers which identify a work request to the system. You define your classification rules using the WLM administration application.

For Domino Go Webserver, you define your classification rules under the IWEB subsystem.

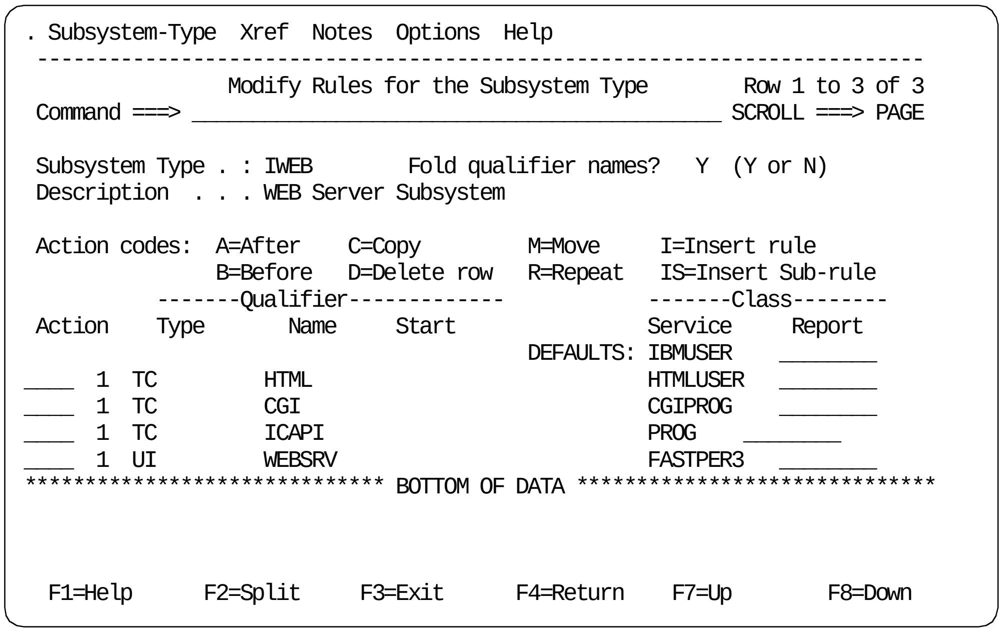
The work qualifiers you can use to classify your work are subsystem-dependent. For the IWEB subsystem, you can use the following work qualifiers:

- Userid - The Web server's userid (not the original requestor's userid). It is implemented this way because the requestor's userid is not available to the Web server at the time the transaction is classified. This probably has limited usefulness, since the userid is the same for most transactions. The default userid for the Web server is WEBSRV.
- Subsystem Instance - This is the subsystem name used in the application environment definition, since this is unique for each instance of the Web server.
- Subsystem Parameter - The actual format is:

0-7	Subsystem name (same as what is planned for Subsystem Instance)
8	blank
9-23	Source IP address
24	blank
25-39	Target IP address
40	blank
41-46	Target port

- Transaction Class - This is probably the most useful qualifier because of its flexibility. Transaction class is the arbitrary class name you specify in the APPLENV directive in the Web server configuration file. You can use the filtering function in the Web server to assign transactions to transaction classes based on the requested URL. Then in turn, the transaction classes can be assigned unique service classes via the WLM policy using the transaction class qualifier.
- Transaction Name - The method name, for example, GET, HEAD, POST, PUT and DELETE.

Figure 69 on page 194 shows an example of defining a classification rule for the Domino Go Webserver.



```
. Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                Modify Rules for the Subsystem Type       Row 1 to 3 of 3
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : IWEB       Fold qualifier names?   Y  (Y or N)
 Description  . . . WEB Server Subsystem

 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: IBMUSER    ________
____  1  TC         HTML                            HTMLUSER   ________
____  1  TC         CGI                             CGIPROG    ________
____  1  TC         ICAPI                           PROG    ________
____  1  UI         WEBSRV                          FASTPER3   ________
****************************** BOTTOM OF DATA ******************************



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

Figure 69. WLM Classification Rule for Scalable Web server

Because a piece of work can have more than one work qualifier associated with it, it may match more than one classification rule. Therefore, the order in which you define the classification rules determines which service class is assigned.

Important

Special Note for the Fast Response Cache Accelerator

In OS/390 Release 7, there is a new function called the fast response cache accelerator for high performance handling of cached static Web pages. You must classify this work as described in the following section. Otherwise, it will be assigned the default service class for IWEB work.

The transactions handled by the cache accelerator are all joined to a single, long-lived enclave. This enclave should be assigned a unique transaction class (which is assigned via a Web directive). This transaction class should then be assigned to a service class with a single period and a velocity goal in the service policy under the IWEB subsystem type. Neither response time goals nor multiple periods are appropriate for this work, as WLM is not aware of the individual cache accelerator requests. (Because each individual transaction is so trivial, it would cost more resource to manage them than to just process them.)

In RMF reports, you will see zero ended transactions for the cache accelerator service class (assuming you have no other work running in this service class), but you will see some amount of accumulated service for this single enclave.

For more information about defining classification rules see 3.3.9, “Classification Rules” on page 117, and *OS/390 MVS Planning: Workload Management*, GC28-1761.

5.3.5 WLM Definition for Web Server Address Space Management

The scalable server subsystem is set up to use WLM to distribute work among a set of WLM-managed Domino Go Webserver address spaces. To do this, Domino Go Webserver uses the address space manager function and queue manager services of WLM. These services were introduced in OS/390 Release 3.

There are two steps to set up the scalable server subsystem from a WLM point of view.

- Define the configuration file ApplEnv in your Domino Go Webserver environment.
- Define the WLM application environments in accordance with the ApplEnv directives.

We describe these steps in more detail in the following sections.

5.3.5.1 Define the WLM Application Environment (ApplEnv) Directives

When defining your WLM ApplEnv directives, you build a link between incoming Web requests and your WLM application environment definitions.

You define the ApplEnv directives in your Domino Go Webserver configuration file HTTPD.CONF.

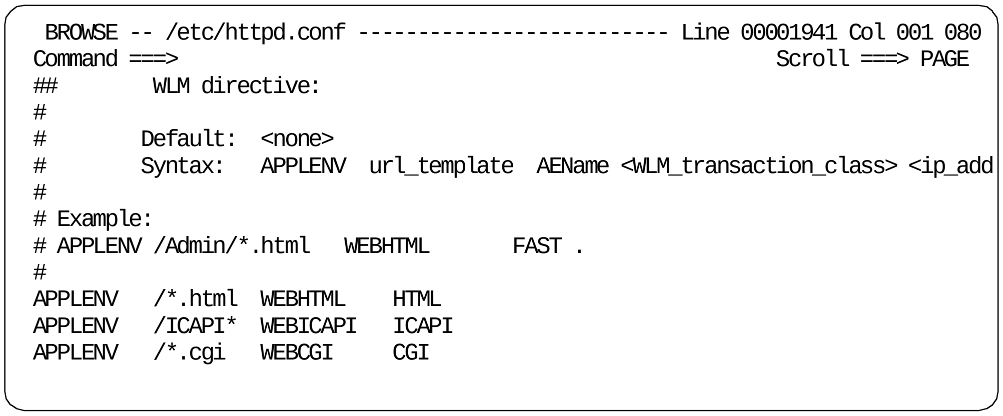
The ApplEnv directives tell the queue manager to put “matched” requests to a WLM application environment queue and give WLM information it can use to assign this request to a service class. This is done by specifying the application environment and transaction class for processing URL requests matching a given template. The format of the directive is:

```
ApplEnv url_template AEName [WLM_transaction_class [ip_addr_template]]
```

ApplEnv	This specifies a unique application environment directive and, from a WLM perspective, a unique transaction queue managed by WLM.
url_template	This is a template for requests that you want your server to match to this application environment. You can use wild cards in this template.
AEName	This specifies the WLM application environment name this request belongs to. Any application environment names that are referenced by using this directive *must* also be defined to WLM.
WLM_transaction_class	This is optional. You can assign a transaction class to the matching requests and WLM can use it as a work qualifier in its classification rules to assign service classes to incoming work. Any transaction classes that are referenced using this directive *should* also be defined in the classification rules of WLM. You must define them under the IWEB subsystem using the transaction class (TC) qualifier. For more information about classification rules, also see 5.3.4, “Define Your WLM Classification Rules for Scalable Web Server” on page 193.

ip_addr_template If you are using multiple IP addresses, use this parameter to specify to which IP address this directive applies.

Figure 70 shows an example of defining WLM directives in your scalable Web server subsystem configuration file:



```
 BROWSE -- /etc/httpd.conf ------------------------- Line 00001941 Col 001 080
Command ===>                                                  Scroll ===> PAGE
##       WLM directive:
#
#        Default:  <none>
#        Syntax:   APPLENV  url_template  AEName <WLM_transaction_class> <ip_add
#
# Example:
# APPLENV /Admin/*.html   WEBHTML       FAST .
#
APPLENV   /*.html  WEBHTML    HTML
APPLENV   /ICAPI*  WEBICAPI   ICAPI
APPLENV   /*.cgi   WEBCGI     CGI
```

Figure 70. Application Environment Directives Examples

There are three application environment directives defined in this example: one for each of all HTML, ICAPI, and CGI requests. We assign a unique application environment and a different transaction class to these requests. Both are used in the WLM administration application to assign a service class and put the requests in the appropriate WLM queue.

It would be sufficient to use just one application environment name and different transaction classes for the three application environment directives in order to separate the work requests. However, Domino Go Webserver Release 5.0 provides a way to tailor your application environment using application environment configuration (ApplEnvConf) directives.

In these directives you can tailor your application environments to support various workloads and conserve resources. You do it by defining the following directives inside an ApplEnvConf directive:

- PluginInclude
- PluginExclude
- PluginDefault Include/Exclude
- CacheLocalFile
- ServerInit
- MaxActiveThreads

These directives used inside an ApplEnvConf directive only apply for the specific application environment. For more information about ApplEnvConf directive see *Domino Go Webserver Release 5.0 for OS/390: Webmaster's Guide*, SC31-8691.

In order to tailor your queue server address spaces, you have to define multiple application environments, one application environment for every request type.

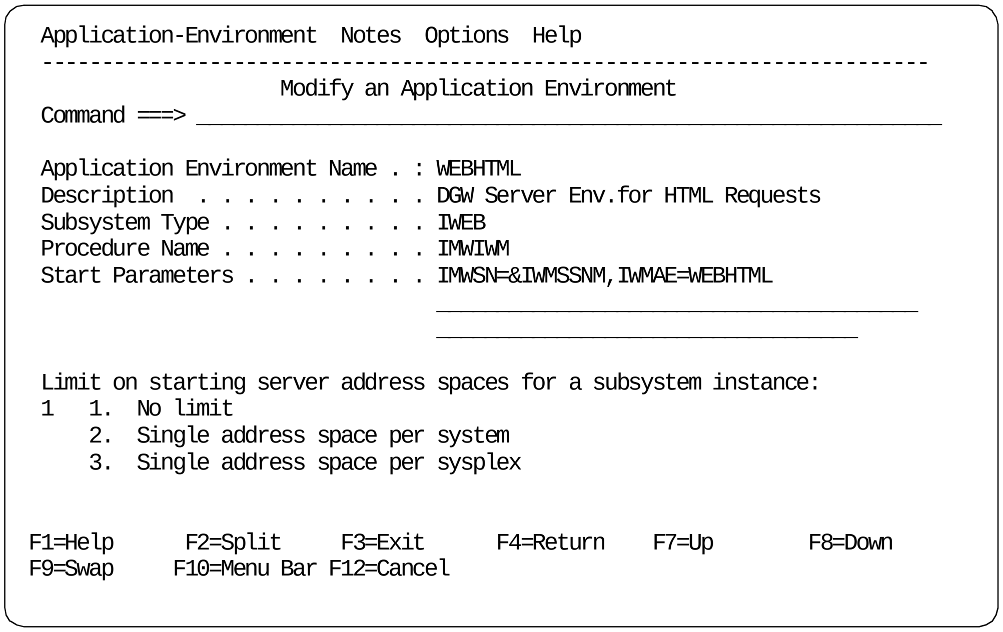
5.3.5.2 Define the WLM Application Environments

You define the application environments in the WLM service definition using the WLM ISPF application.

An *application environment* is a group of application functions requested by a client. The workload supported by a given work manager may be partitioned into multiple application environments. The application environment is used to bind the incoming work requests together with a set of work manager address spaces. For more information, also see 2.6.1, “Server Address Space Management” on page 77.

You have to define all application environments in the WLM service definition you use in your Domino Go Webserver ApplEnv directive.

Figure 71 on page 197 shows the definition panel for an HTML application environment:



```
 Application-Environment  Notes  Options  Help
 --------------------------------------------------------------------------
                     Modify an Application Environment
 Command ===> ______________________________________________________________

 Application Environment Name . : WEBHTML
 Description  . . . . . . . . . . DGW Server Env.for HTML Requests
 Subsystem Type . . . . . . . . . IWEB
 Procedure Name . . . . . . . . . IMWIWM
 Start Parameters . . . . . . . . IMWSN=&IWMSSNM,IWMAE=WEBHTML
                                  ________________________________________
                                  ___________________________________


 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


F1=Help      F2=Split     F3=Exit      F4=Return    F7=Up        F8=Down
F9=Swap     F10=Menu Bar F12=Cancel
```

Figure 71. WLM Application Environment Definition for Domino Go Webserver

WLM uses the application environment definition for managing the workload running in server address spaces. WLM uses it for starting server address spaces and to queue requests to the right server queue. For more information, see 2.6.1, “Server Address Space Management” on page 77.

The Procedure Name is used to start the queue server address spaces for that application environment automatically by WLM. If you omit a procedure name, WLM does not start queue server address spaces; you have to do this manually or by automation.

WLM needs the start parameters in order to start the queue server address spaces. You can define these parameters in the procedure itself or put them in the WLM definition panel.

One of the start parameters is the application environment name to which this address space has to connect. If you define it in the start procedure, you need one procedure for every application environment. Therefore we recommend you define it in the WLM application environment definition panel.

Another start parameter is the subsystem name of the scalable server subsystem. This name is set by the queue manager address space. You can use the &IWMSSNM symbol and WLM will substitute the subsystem instance name provided to WLM by the queue manager. Using the symbol guarantees you that a server address space will not fail because of a wrong subsystem name defined in the procedure or the WLM panel.

Defining the start parameters in the WLM application environment definition panel allows you to have only one start procedure for many application environments and many scalable server subsystems.

In the application environment panel, you can also limit the number of address spaces WLM is allowed to start automatically. The scalable Web server has a single system scope, which means that the queue servers have to run in the same system as the queue manager. Because of that, you can only define option 1 or 2.

5.3.5.3 Using WLM Started Address Spaces

If you define a procedure name in the application environment, WLM will manage the server address spaces automatically. WLM starts and stops the address spaces as required to meet the goals for the workload. If a work request for an application environment arrives and no address space is active to serve that request, WLM starts one.

Domino Go Webserver takes several seconds or even minutes to initialize, depending on what is in the configuration file and how many static files are cached in memory. The start time is seen as latency on the first request. If you cannot accept this delay, you have to pre-start at least one queue server for each of your application environments.

5.3.5.4 Using the Scalable Web Server in Compatibility Mode

You can run your scalable Web server in compatibility mode and use the capability provided by WLM application environments. The only differences from a scalable Web server running in a goal mode are that you have to start all your queue servers manually or by automation, and you assign your Web server requests to performance groups rather than service classes.

WLM cannot manage the server address space while running in compatibility mode; as a result, the queue server address spaces must be pre-started before the first request matching an application environment arrives. Otherwise your Web server will not respond to these requests. On the START command, you have to define the subsystem name of your scalable subsystem (defined by the queue manager) and the application environment that a queue sever shall belong to. An example of the START command is as follows:

```
S IMWIWM,JOBNAME=WEBHTML,IMWSN=IMWE,IMWAE=WEBHTML
```

Using the scalable Domino Go Webserver in compatibility mode, you have to make almost all the same definitions as you would for goal mode. You have to do the following:

- Define an ApplEnv directive in your Domino Go Webserver configuration file HTTPD.CONF. For information how to do this, see 5.3.5.1, “Define the WLM Application Environment (ApplEnv) Directives” on page 195.
- Define an WLM service definition and activate it. The WLM service definition has to contain:
    - At least all application environment definitions used in the ApplEnv directive. For information how to do this, see 5.3.5.2, “Define the WLM Application Environments” on page 197.
    - Classification rules in order to classify your Web server requests. During classification, each request is assigned a service class. The scalable Domino Go Webserver creates enclaves for every request, and an enclave is always classified using WLM classification rules.

      If you do not have classification rules for the IWEB subsystem defined or even no active WLM service definition in compatibility mode, all Web server requests run with the priority of the Domino Go Webserver address space. For information about defining classification rules, see 5.3.4, “Define Your WLM Classification Rules for Scalable Web Server” on page 193.

Because WLM does not use goals in compatibility mode, it does not matter what goals you assign the IWEB service classes. The service class is simply used to map the Web request to a performance group. This mapping is done in the IEAICSxx parmlib member through the SRVCLASS parameter under the IWEB subsystem type. The following is an example of IEAICSxx and IEAIPSxx members:

```
IEAICSxx:
SUBSYS=IWEB,
SRVCLASS=HTMLUSER,PGN=15

IEAIPSxx:
PGN=15,(DP=F92,DUR=2K).
```

5.3.5.5 Considerations for Using Persistent Connections

The idea behind WLM is that one logical piece of work (for instance, a request for a home page) should be treated consistently in terms of performance, and that you can define different performance requirements (goals) for different requests.

If you use an HTTP/1.0-style of Web request processing, logical user requests cannot be grouped. A request for an HTML page by a user may actually result in a number of separate requests from the browser to the server: one for the HTML page itself, and perhaps five other requests for graphic images embedded in the page. Every request contains one CONNECT, one GET, and one CLOSE SOCKET command, and the connection between client and server is terminated after each request. This causes a lot of overhead.

Each of these requests is considered separate and is managed separately from a WLM point of view, because every request is managed in its own enclave.

The HTTP/1.1 protocol introduces *persistent connections*, which reduce the overhead of the HTTP/1.0 connection. Instead of creating and tearing down the TCP/IP infrastructure for every request, persistent connections allow multiple GET requests during one connection. If the client and the server support the

HTTP/1.1 protocol and agree to use a persistent connection for a logical request (such as the HTML page and the five graphic files), it is possible to get a view of the performance of the logical request. However, you need to be careful that you do not get a misleading view. Figure 72 on page 200 illustrates a possible flow of a user request for a home page:

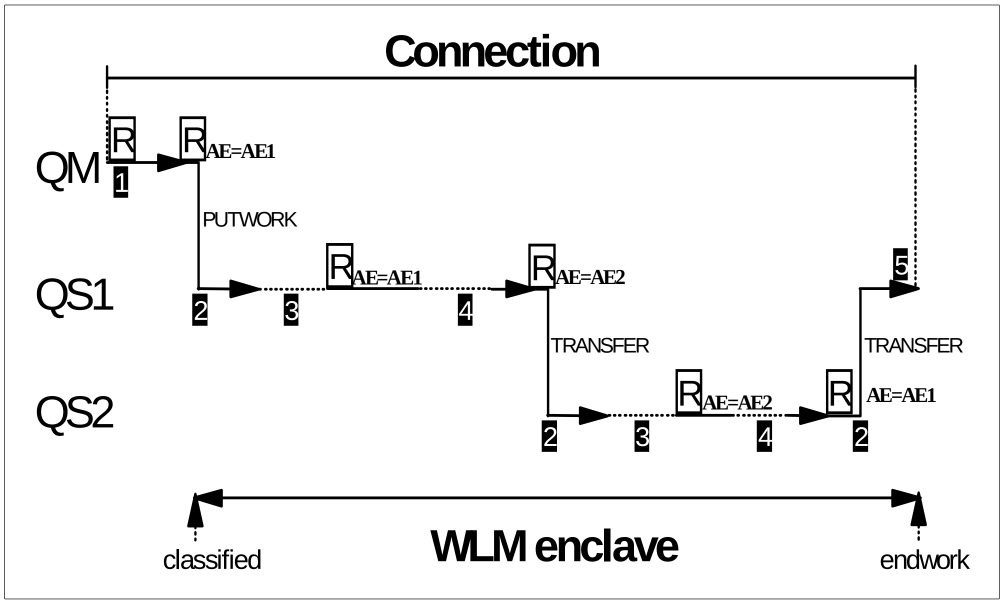


Figure 72. Request Processing on a Persistent Connection

Figure 72 shows a queue manager and two queue servers that together process a user request for a home page, which is made up of six separate requests, indicated by "R." The six requests are made over one persistent connection. The requests are processed as follows:

1. The first request comes from the client and is processed by the queue manager because the request type does not match an ApplEnv directive. This request is therefore not classified, and is not known to WLM (_**1**_).
2. The second request on the connection is matched with an ApplEnv directive (application environment AE1) and is classified. A WLM enclave is built for all related work. The request is put on the queue of queue server 1 (QS1) with a PUTWORK request. Queue server 1 does GETWORK to get the work from the queue and process it (_**2**_). Once the request processing is complete (_**3**_), the queue server has to wait because a persistent connection is in effect. Although this time is included in the enclave, it is not processing time, it is wait time. If your client is attached to a slow network, it might give you the impression the enclave is not meeting performance goals since wait time cannot be distinguished from processing time.
3. The third request is for the same application environment as the second, so it is processing in the same queue server. If this request required some significant processing, such as a database lookup, the time required to perform that function would be reflected in the enclave (_**4**_).
4. The fourth request is for a different application environment (AE2). Queue servers can place work on WLM queues just as queue managers can, so in this case QS1 issues a TRANSFER to place the new request on the queue for

QS2 (_**2**_). The request is processed by QS2, and on completion, wait time considerations apply, as discussed earlier (_**3**_).

5. Request five is processed in QS2 as before.
6. Request six is for AE1, so is transferred from QS2 to QS1. Once the request is completed and the persistent connection closed (_**5**_), the enclave is terminated.

This example shows that WLM starts to classify the workload of Domino Go Webserver according to the first client request matching an application environment, and spans all remaining requests on the same connection. We can draw the following conclusions from this discussion:

- Not all requests making up a persistent connection will be reflected in the WLM enclave. The enclave starts with the first request that matches an application environment.
- The WLM enclave, once established, will contain “wait” time (time the server was waiting for a request to arrive on the persistent connection) that cannot be distinguished from processing time. This means that you may not be able to use response time goals for these requests.
- Request processing that causes work to be done by other subsystems (such as DB2 or CICS) will cause the inclusion of that processing in the enclave.

Each of these considerations should be taken into account when setting service class goals and reviewing WLM enclave data for persistent connections.

The grouping of multiple requests in an enclave and the inclusion of wait time makes it difficult to establish reasonable response time goals if you are using persistent sessions. Moreover, the RMF reporting of these transactions is at worst misleading and at best difficult to understand.

Therefore, we recommend disabling persistent sessions if you plan to use response time goals and/or if the RMF data is important to you. This will increase the CPU cost caused by the establishment of additional sockets, but will make the performance management of Web transactions much easier.

The grouping of multiple requests in a persistent session is planned to be provided in a future release of OS/390.

5.4 OS/390 UNIX System Services Function and Application

The term UNIX System Services stands for the UNIX component of OS/390 formerly called Open Edition. UNIX System Services 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 the UNIX System Services address spaces.
- To propagate an enclave across a fork, spawn or pthread_create.

5.4.1 UNIX Services Fork and Spawn Function Calls

Before OS/390 Version 2.4, the OS/390 UNIX System Services used APPC/MVS services to create an address space for fork and spawn function calls.This was an asynchronous process. The requesting task waited for the creation of the address space. Having an inadequate number of APPC initiators could considerably increase this wait time.

In OS/390 Version 2.4 and subsequent releases, WLM is used to create and delete the UNIX Services fork and spawn address spaces. The CPU cost of creating a child process via fork or spawn function calls is now considerably decreased.

In OS/390 Version 2.4, WLM creates server address spaces on demand, and when the servers are idle, 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. A request for a WLM fork initiator is synchronous if the free pool of initiators is not empty, as shown in Figure 73. 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.

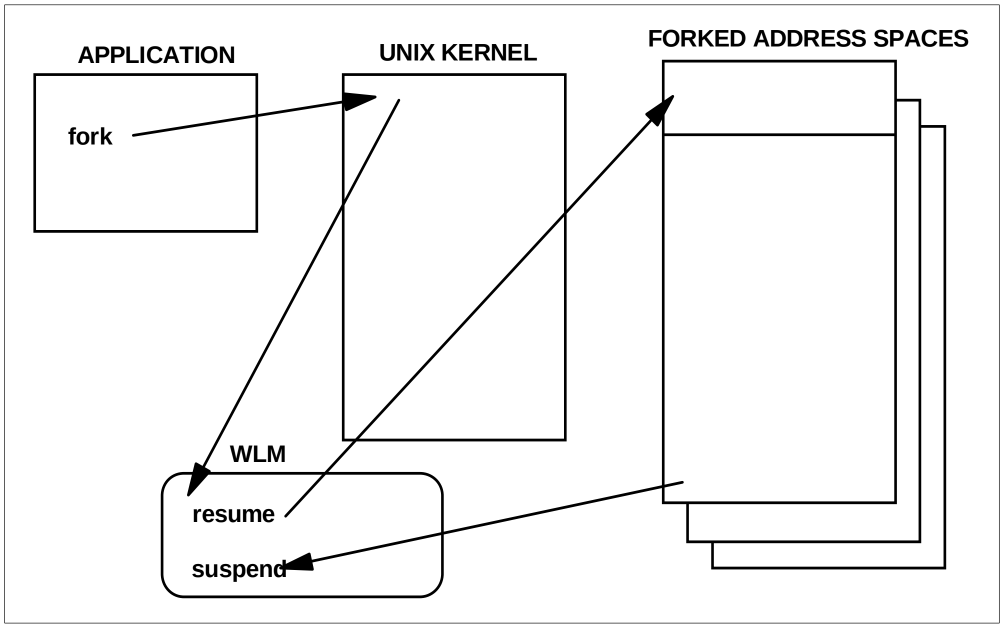


Figure 73. WLM Fork and Spawn Synchronous Processing

If the fork initiator pool is empty, WLM returns a non-zero return code to the UNIX Services, creates an ECB, and the application goes into a wait state, as shown in Figure 74 on page 203. A request for an additional initiator is queued to WLM and then control is returned to the requesting process. The UNIX Services kernel then places the requestor into an ECB wait, pending availability of a WLM fork initiator.

WLM creates a new fork initiator using the SYS1.PROCLIB(BPXAS) procedure, adds it to the pool, and posts the ECB.

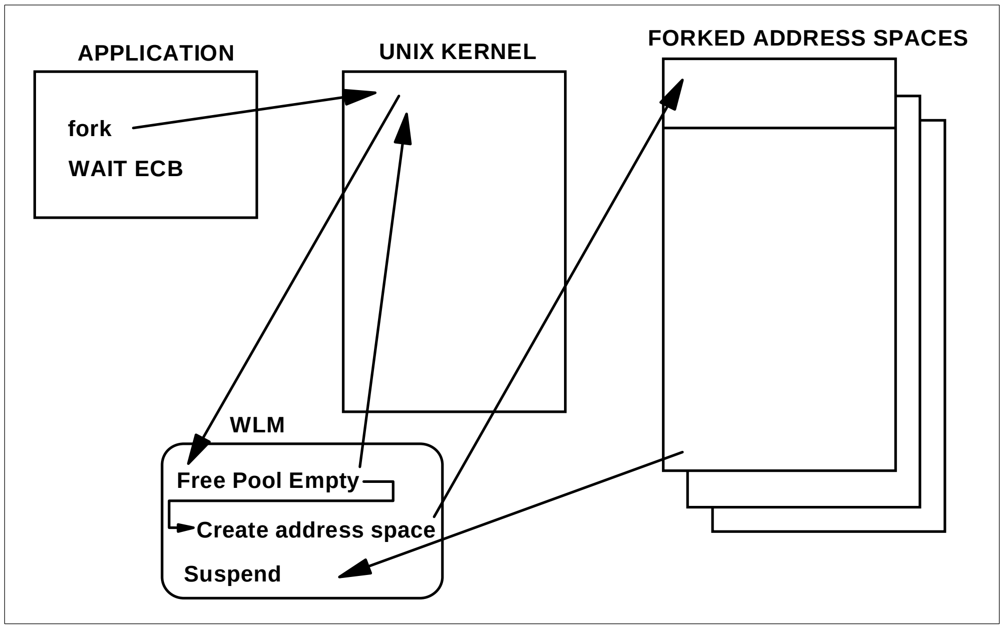


Figure 74. WLM Fork and Spawn Asynchronous Process

You do not need any tuning or definition in order to use WLM for the fork and spawn function calls. WLM services are used for creating the address spaces in either compatibility mode or goal mode. Within OS/390 UNIX System Services, it is possible to set the maximum number of forked children address spaces in the BPXPRMxx member of SYS1.PARMLIB.

If you use UNIX System Services for OS/390 Version 2.4 or higher, you can delete the APPC transaction profile formerly needed 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 model address spaces are overwritten when an actual fork or spawn is processed. BPXAS initiators are started by WLM only on demand. However, during system initialization, the running of initialization routines may cause the creation of some BPXAS initiators (around 10), which become idle after initialization. Any idle BPXAS initiators (address spaces in the WLM free address space pool) will time out after 30 minutes of idle time.

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. The command and its output are shown in Figure 75 on page 204.

```
D A,BPXAS
 IEE115I 16.20.17 1999.019 ACTIVITY 830
  JOBS     M/S    TS USERS    SYSAS    INITS   ACTIVE/MAX VTAM     OAS
 00010    00035    00004      00029    00055    00004/00030       00020
  BPXAS    BPXAS    IEFPROC  OWT  IO  A=0065   PER=NO   SMC=000
                                      PGN=N/A  DMN=N/A  AFF=NONE
                                      CT=000.045S  ET=00167.51
                                      WUID=STC21480 USERID=OMVSKERN
                                      WKL=OMVS     SCL=OMVS     P=2
                                      RGP=N/A      SRVR=NO  QSC=NO
                                      ADDR SPACE ASTE=314D5940
  BPXAS    BPXAS    IEFPROC  OWT  IO  A=0020   PER=NO   SMC=000
                                      PGN=N/A  DMN=N/A  AFF=NONE
                                      CT=000.015S  ET=00167.51
                                      WUID=STC21482 USERID=STC
                                      WKL=OMVS     SCL=OMVS     P=2
                                      RGP=N/A      SRVR=NO  QSC=NO
                                      ADDR SPACE ASTE=314D4800
  BPXAS    BPXAS    IEFPROC  OWT  IO  A=0067   PER=NO   SMC=000
                                      PGN=N/A  DMN=N/A  AFF=NONE
                                      CT=000.016S  ET=00167.51
                                      WUID=STC21485 USERID=FTPDOE
                                      WKL=OMVS     SCL=OMVS     P=2
                                      RGP=N/A      SRVR=NO  QSC=NO
                                      ADDR SPACE ASTE=314D59C0
```

Figure 75. Displaying the WLM Created Fork and Spawn Address Spaces

This command shows you all active or idle (but still in the WLM pool) address spaces 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. The assigned user ID for these idle address spaces is the USERID of the task that last used this address space.

5.4.1.1 Fork, Spawn and pthread_create Enclave Propagation

When an application issues a fork or spawn, it is desirable for the performance characteristics of the application to be propagated to the forked or spawned process. This is achieved through the use of enclaves. Figure 76 on page 205 shows a diagram illustrating the propagation of the enclave to the new process.

Note: This function is not related to the WLM function that maintains the pool of available fork and spawn address spaces.

In Figure 76 on page 205, there is some code (called here setup code) which runs in the forked (or spawned) address space before the application code receives control. The service consumed by this setup code is *not* charged to the enclave that the application is running in. The service consumed by the setup code is charged under the OMVS subsystem according to the classification rules you set up.

Be aware that this setup code counts as a transaction in the service class it is associated with (under the OMVS subsystem), as does the original application transaction (under the subsystem of the original transaction, for example IWEB). This results in a count of two transactions for one business unit of work. Note, however, that the service units consumed are not double-counted. Those used by the application are charged to the enclave, and those consumed by the setup code are charged to the OMVS subsystem.

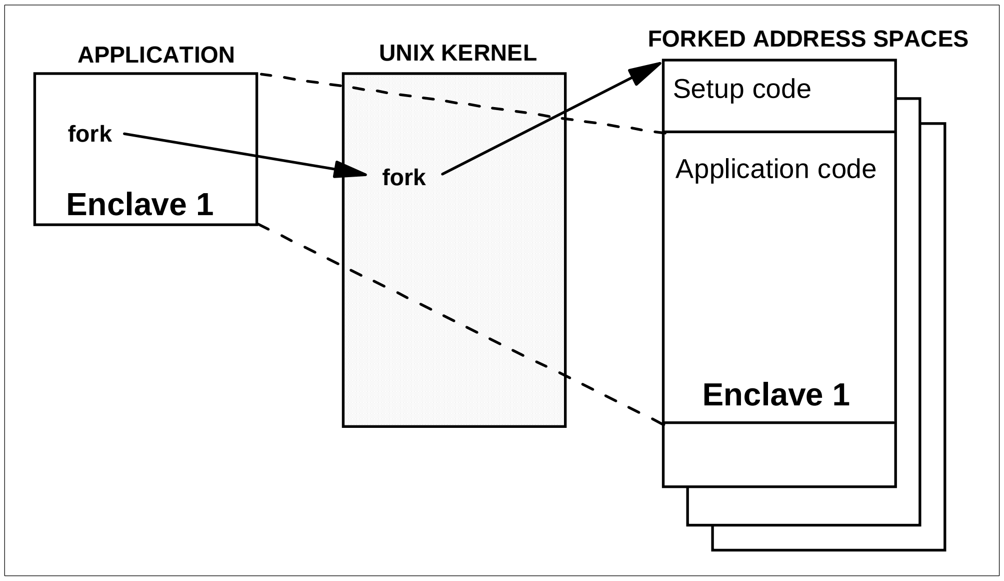


Figure 76. Forked Child Enclave Propagation

Note: For pthread_create, WLM has no involvement in creating the new thread. The new thread will, however, be joined to the enclave of the caller of pthread_create.

5.4.2 Controlling the Performance of your UNIX Services Work

In an OS/390 environment, you are able to control the performance of your UNIX work using functions provided by the UNIX services and WLM. Some considerations and recommendations for controlling your performance are covered in the next sections.

5.4.2.1 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 OS/390 UNIX System Services and have an interface to WLM to provide the system control and monitoring support customers are accustomed to.

However, it is not recommended that customers enable *nice()*, *setpriority()*, and *chpriority()* support. Instead, we recommend exclusive use of normal WLM controls. For information about enabling these services, see *OS/390 UNIX System Services Planning*, SC28-1890.

5.4.2.2 Performance Control Using SRM and WLM

You can control the performance of your UNIX workload running your installation in WLM compatibility or goal mode. Running in compatibility mode, you have to customize your IEAIPSxx and IEAICSxx parmlib members. In WLM goal mode, you have to customize your WLM service definition.

5.4.3 UNIX System Services Work in WLM Compatibility Mode

In order to define a default performance group for forked or spawned address spaces, you have to add a SUBSYS=OMVS section in your current IEAICSxx parmlib member.

If you do not, the system puts all UNIX forked processes in the system default performance group and this may result in a wait condition at startup.

The SUBSYS=OMVS section assigns performance groups to forked address spaces. These performance group assignments do not apply to dubbed address spaces, such as batch programs that issue UNIX callable services. If a TSO/E, batch, or started task address space uses UNIX services, it does not change subsystem type; that is, it does not use SUBSYS=OMVS.

You can assign specific resource controls to your forked and spawned address spaces by specifying different PGNs to different work requests. You can use the ACCTINFO and USERID as work qualifiers to distinguish the requests.

At a minimum, you should specify a USERID=OMVSKERN statement to specify a performance group for the UNIX System Services startup processes, those which are forked by the kernel or by the initialization process BPXOINIT.

The following is an IEAICSxx member example for SUBSYS=OMVS:

```
SUBSYS=OMVS,PGN=5            /* OpenMVS forked children , Default PGN*/
USERID=OMVSKERN,PGN=40       /* OpenMVS startup processes */
USERID=SUPER1,PGN=50         /* Special for user super1 */
ACCTINFO=D001(1),PGN=60      /* Special for acct D001 */
```

You also have to update the IEAIPSxx parmlib member by specifying performance attributes for the performance groups added to IEAICSxx.

For more information about UNIX System Service work in WLM compatibility mode, see *OS/390 UNIX System Services Planning*, SC28-1890.

5.4.4 Prioritizing Your UNIX System Services Work in WLM Goal Mode

Installations running in WLM goal mode can take the following steps to customize their WLM service definition for the UNIX System Services workload:

1. Define a workload for UNIX System Services work that runs as forked or spawned children. (You could choose to combine this workload with an existing one.)
2. Define service classes, with goals, for your UNIX System Services work.
3. Define classification rules to assign service classes to your work.

These are the same steps as for all other workloads. Note again, that these steps have nothing to do with enabling address space creation for fork and spawn functions; there is nothing required to enable that function. See also 5.4.1, “UNIX Services Fork and Spawn Function Calls” on page 201 for more information.

Defining a workload for UNIX System Services work in your service definition is simply a process of having a named collection for all your work running under UNIX System Services control for reporting reasons. We do not explain anything about this step here; for more information about defining workloads, see 3.3.5, “Workload Specification” on page 110.

5.4.4.1 Define Your Service Classes

Defining service classes is required in order to reflect your service level agreements or objectives for a specific set of workloads in your WLM definition. In this section we give some general recommendations for defining your service classes.

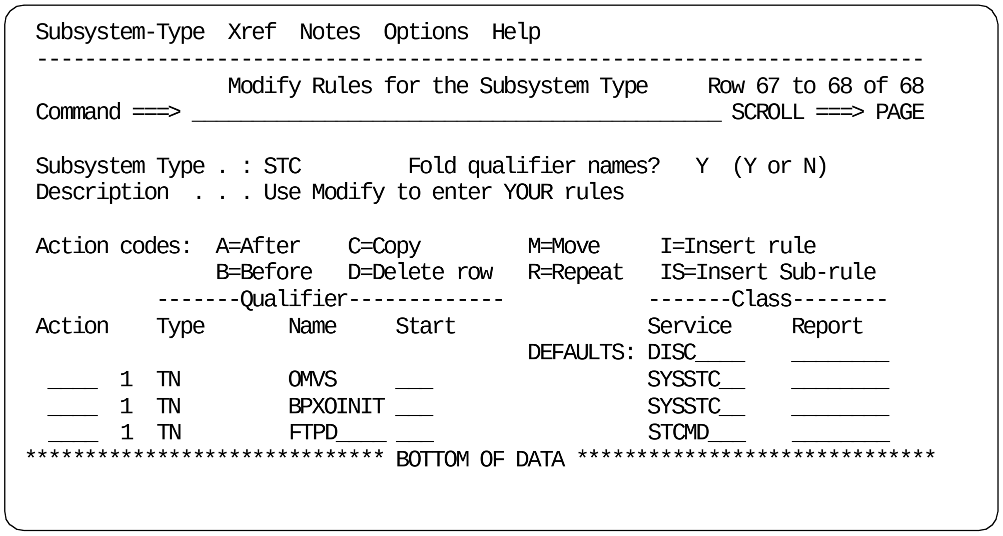
The UNIX System Services work can be divided into two groups:

- UNIX System Services kernel and daemon processes
- UNIX System Services forked processes

The UNIX System Services kernel (OMVS started task) and the initialization process (BPXOINIT started task) should be treated as high-priority started tasks. Our recommendation is to classify both into the system started class service class SYSSTC which has a very high priority, instead of defining your own service class.

Note: By default, both OMVS and BPXOINIT will be classified into the SYSTEM service class.

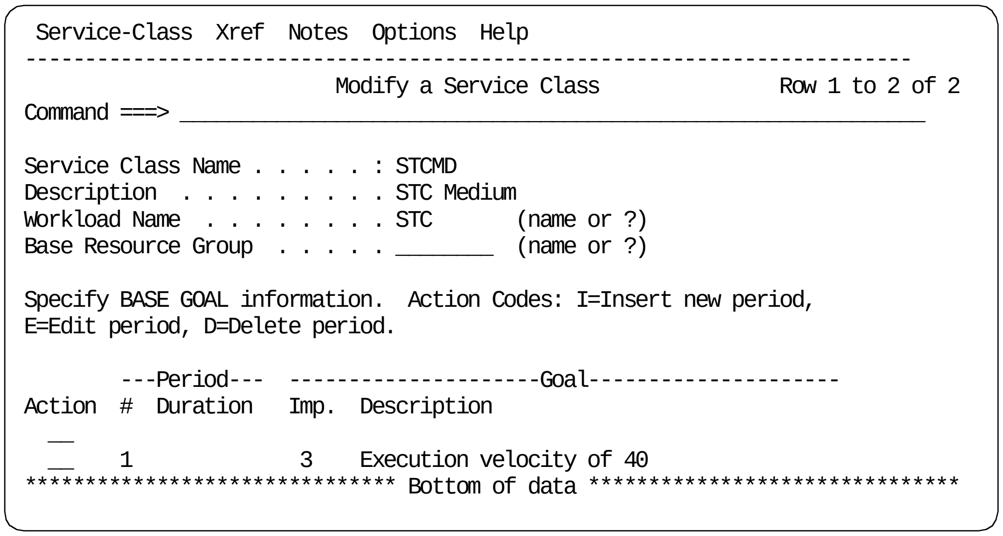
For an example of the classification rules for the UNIX System Services started tasks, refer to Figure 77.



```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type     Row 67 to 68 of 68
 Command ===> ____________________________________________ SCROLL ===> PAGE

 Subsystem Type . : STC         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
           -------Qualifier-------------            -------Class--------
 Action    Type       Name     Start                Service     Report
                                           DEFAULTS: DISC____    ________
  ____  1  TN         OMVS     ___                  SYSSTC__    ________
  ____  1  TN         BPXOINIT ___                  SYSSTC__    ________
  ____  1  TN         FTPD____ ___                  STCMD___    ________
****************************** BOTTOM OF DATA ******************************
```

Figure 77. UNIX Systems Services STC Classification Rules

UNIX System Services daemons, for example FTPD, as shown in Figure 77, are long-lived processes that run to perform continuous or periodic system-wide functions. These daemons should be treated the same as other started tasks of similar importance. For an example of the service class definition for STCMD, refer to Figure 78 on page 208. If any of your daemons are very critical to your business, consider classifying them into the SYSSTC service class.



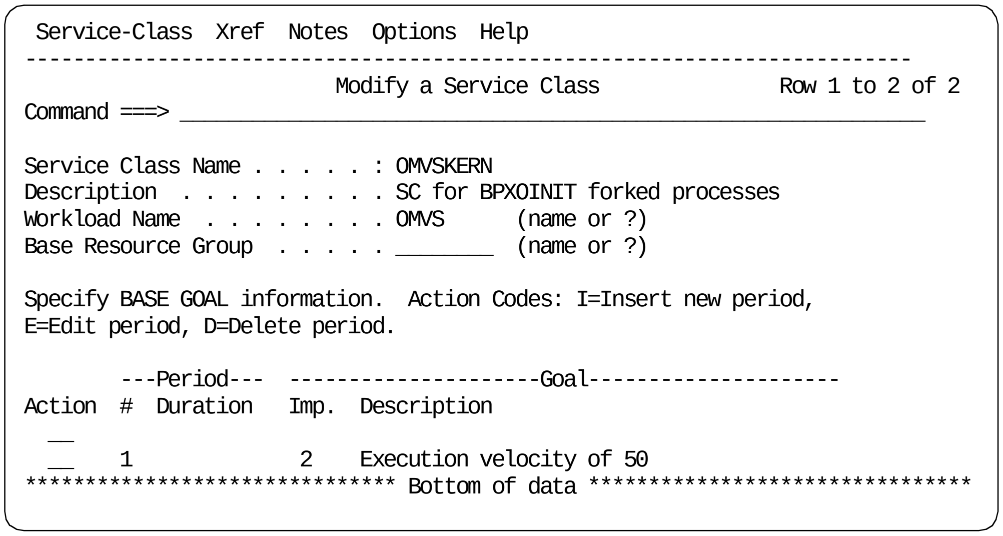
```
 Service-Class  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                          Modify a Service Class             Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : STCMD
Description  . . . . . . . . . STC Medium
Workload Name  . . . . . . . . STC       (name or ?)
Base Resource Group  . . . . . ________  (name or ?)

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

        ---Period---  ---------------------Goal--------------------
Action  #  Duration   Imp.  Description
  __
  __    1             3    Execution velocity of 40
****************************** Bottom of data *******************************
```

Figure 78. Service Class Definition for STCMD

The appropriate goal for your daemon processes is the velocity goal. A good starting point is a velocity of 40.

Some of the UNIX System Services work runs within a BPXAS initiator. These are forked or spawned children. This kind of work falls under the subsystem type OMVS. However, the resource requirement can be very different, depending on the type of forked process.

Some UNIX System Services processes are forked by the UNIX System Services initialization process BPXOINIT. These processes run under the user ID of OMVSKERN and they should be given a velocity goal that is higher than that of other forked children. You may need to create a service class for this, or use another service class that has an appropriate velocity, such as a service class used for STCs. An example of the definition for the service class OMVSKERN can be seen in Figure 79.



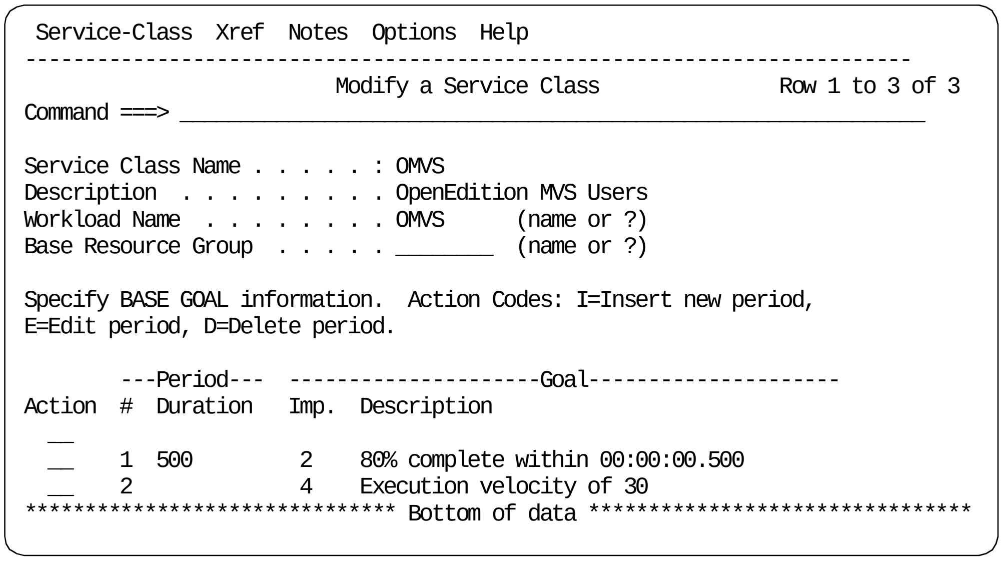
```
 Service-Class  Xref  Notes  Options  Help
--------------------------------------------------------------------------
                          Modify a Service Class             Row 1 to 2 of 2
Command ===> ______________________________________________________________

Service Class Name . . . . . : OMVSKERN
Description  . . . . . . . . . SC for BPXOINIT forked processes
Workload Name  . . . . . . . . OMVS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)

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 50
****************************** Bottom of data ********************************
```

Figure 79. OMVSKERN Sample Service Class Definition

Another kind of UNIX System Services work is the more user transaction-oriented work. Many, but not all, UNIX System Services transactions use few resources. (which is similar to simple TSO transactions). Therefore, a service class with multiple periods is appropriate for that kind of UNIX System Services work.This allows you to assign different goals for long-running background processes than for short-running shell commands.

The first period should have a response time goal. The duration of this period will need to be determined iteratively, such that a predetermined percentage of transactions are considered trivial and complete in this period. Use the RMF Workload Activity report, or an equivalent tool, to get information about your transaction response time distribution and service consumption.

The last period should contain a velocity goal that fits the installation's objective for long-running and/or CPU-intensive forked children. A low velocity goal should be given to this period, with an importance to compare this goal to others in the system. For an example of a service class definition for UNIX System Services work, refer to Figure 80.



```
 Service-Class  Xref  Notes  Options  Help
---------------------------------------------------------------------------
                          Modify a Service Class            Row 1 to 3 of 3
Command ===> ______________________________________________________________

Service Class Name . . . . . : OMVS
Description  . . . . . . . . . OpenEdition MVS Users
Workload Name  . . . . . . . . OMVS      (name or ?)
Base Resource Group  . . . . . ________  (name or ?)

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 00:00:00.500
  __    2              4    Execution velocity of 30
****************************** Bottom of data ********************************
```

Figure 80. Sample OMVS Service Class Defintion

You could choose to insert a third period with a response time goal between the two periods previously discussed and shown in Figure 80. This would be valuable if your installation has a very significant number of UNIX System Services forked children and you want more distinction between trivial and complex transactions.

5.4.4.2 Define Your Classification Rules

You assign a specific service class to your workload using a work qualifier in your classification rules. The first level qualifier is the subsystem type. Classification rules are grouped by subsystem type.

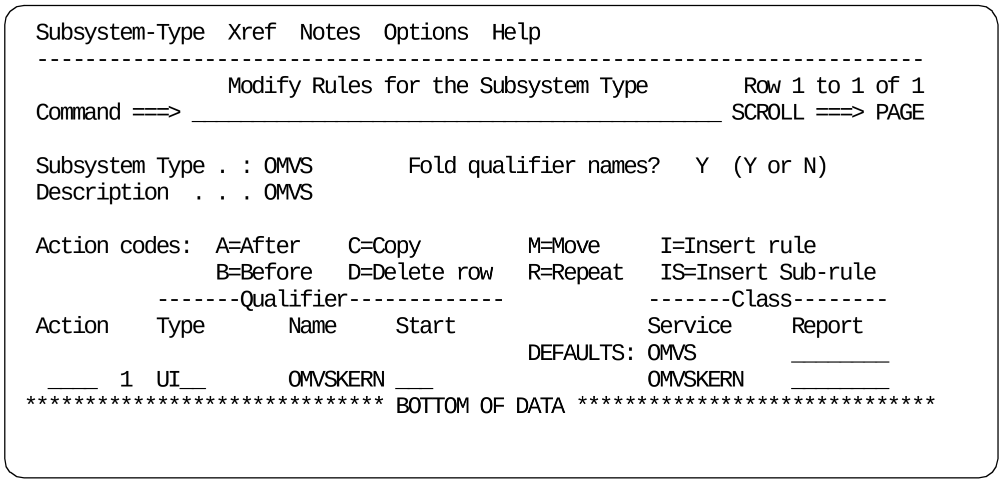
The OMVS subsystem type includes work requests processed in UNIX System Services forked address spaces. It does not apply to dubbed address spaces, such as batch programs that issue UNIX System Services callable services. If a TSO/E, batch, or started task uses UNIX System Services services, it does not change subsystem type and does not use subsystem type OMVS.

As already mentioned, remember to make sure that the kernel (transaction name OMVS), the initialization process BPXOINIT (transaction name BPXOINIT), and the DFSMS buffer manager SYSBMAS (transaction name SYSBMAS) are all put into a high priority started task service class. Our recommendation is to put these started procedures in the default service class for started tasks SYSSTC, as shown in Figure 77 on page 207 for OMVS and BPXOINIT.

Define your classification rules for the forked address spaces under the OMVS subsystem type. You can use the following work qualifiers for classifying your work in a forked address space:

- Account Information (AI)
- Transaction Name (TN)
- Userid (UI)

An example of classification rules for the OMVS subsystem can be seen in Figure 81, where the processes running under user ID OMVSKERN are classified to a service class OMVSKERN. The default service class for all other OMVS subsystem work is OMVS.



```
 Subsystem-Type  Xref  Notes  Options  Help
 --------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type       Row 1 to 1 of 1
 Command ===> ____________________________________________ SCROLL ===> PAGE

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

 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: OMVS        ________
  ____  1  UI__       OMVSKERN ___                 OMVSKERN    ________
****************************** BOTTOM OF DATA ******************************
```

Figure 81. Classification Rules for OMVS Subsystem Work

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 (1to 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, server and daemons in UNIX System Services address spaces can easily be assigned different performance attributes than other 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).

For more information about UNIX System Services accounting, transaction/job names or user IDs, see *OS/390 UNIX System Services Planning*, SC28-1890.

Appendix A. Sample Worksheets

While planning your service definition you will collect information from various sources into *worksheets*. These working documents may be interim steps to organize work, or they may be the final service definition parameters that will be input using the WLM ISPF application.

The following are blank worksheets for your usage. In 3.3, “Service Definition Implementation” on page 104, this format of worksheets is used to describe using the WLM ISPF application to input a service definition. See Chapter 4, “Migrating to WLM Goal Mode” on page 129 for detailed background information on planning your service definition.

Attention

Worksheet Implementation

Review the sample worksheets provided, and then consider nputting these worksheets, as tables, into a PC word processor or spreadsheet package. Using a PC tool for the worksheets has the advantages of:

- You define exactly the number of rows required (the sample worksheet may be too small).
- You eliminate handwriting legibility problems.
- You are able to sort entries (this helps prioritization and organization).
- You are able to share data by placing the worksheet file on a LAN.
- You can easily communicate the service definition to others by sending them a copy of the worksheet.

A.1 Service Definition

There is only one service definition for any given sysplex. Within the service definition there is a base policy as well as override policies for special circumstances. If you need to change your classification rules, then you may need to create a new service definition.

Choose a service definition name that is meaningful for the environment. For example, in a two-sysplex environment with PRODPLEX and TESTPLEX, you could implement service definitions PROD1 and TEST1.

You need to establish coefficients and options across the sysplex. If possible, convert to the recommended coefficients (for more information, refer to 4.2.6, “Service Definition Coefficients” on page 136). Also remember to factor service definition coefficient changes into service class duration calculations as required. Changing the coefficients also impacts accounting procedures, including capacity planning.

The service definition name and description are specified on the main definition menu panel; coefficients/options are specified using Option 8. Fill in the table, where:

- Service Definition Name is 8 characters.
- Description is an optional 32-character description.

- CPU is a numeric value between 0.0 and 99.9.
- IOC is a numeric value between 0.0 and 99.9.
- MSO is a numeric value between 0.000 and 99.999.
- SRB is a numeric value between 0.0 and 99.9.
- IO Priority Management is Yes or No (default is No).

Table 23. Worksheet: Service Definition

Service Definition Name	Description	Coefficients
		CPU: IOC: MSO: SRB: IO Priority Management:

A.2 Service Policy Definitions

The first policy specified is your base policy. Decide if other policies are also required and any resource(s) that are different from the base. In 4.2.4, "Recommendations When Using More Than One IPS/ICS" on page 135, we discuss situations when you might have more than one service policy in your service definition.

Operations will need to know policy names, so make them meaningful.

The service policies are managed using option 1. Fill in the table, where:

- Service Policy Name is 8 characters, unique within the service definition.
- Description is an optional 32-character description.
- Identify the SRVCLASS or RG overrides that will be included in this policy (when you describe SRVCLASS/RG, identify the override values).

Table 24. Worksheet: Service Policy Definitions

Service Policy Name	Description	SRVCLASS or RG Overrides

A.3 Ranking Workloads

This is an interim worksheet to inventory your current workloads and rank them in the *descending* order of priority.

In your ICS, Subsystem will be one of:

- ASCH
- CICS
- DDF
- IMS
- JES2
- OMVS
- STC

- TSO

In addition, you may also have the following workloads:

- CB
- DB2
- IWEB
- LSFM
- SOM
- Non-IBM
- Roll Your Own

Use this worksheet to develop a list of workloads, service classes, and classification groups. This inventory list should include all systems in the sysplex, and putting this information into a PC tool should be helpful in analyzing the workloads to derive the service policy. Depending upon your level of familiarity with the system, you may want to collect additional data such as PGN, total service, % of total service, number of transactions, response time, dispatching priority, position, and also denote WLM-assigned service classes.

Table 25. Worksheet: Ranking Workloads

Subsystem	ICS Qualifier	Description	Priority

A.4 Workload Classification

For WLM, a *workload* is a named collection of work to be reported as a unit. You can arrange workloads by subsystem (CICS, IMS), by major application (production, batch, office) or by line of business (ATM, inventory, department). Logically, a workload is a collection of service classes.

Typically, reporting tools will print reports in alphabetical order based on workload name. When you assign a workload name, use the naming convention to influence the reporting order; for example, to get all production workloads reported together, use a naming convention of PRDxxxx.

The workloads are managed using option 2. Workloads must be defined before they can be used in service class definitions, where:

- Workload Name is 8 characters, unique within the service definition.
- Description is an optional 32-character description.

Table 26. Worksheet: Workload Classification

Workload Name	Description

A.5 Service Class Definition

When you rank your business workloads in A.3, “Ranking Workloads” on page 214, you should see groups of work requiring similar management. This will be the basis for service classes (see 4.2.7, “Using Service Classes” on page 139).

You can have up to 100 service classes. Assign the minimum number of service classes required to describe your business workloads. A service class is a named group of work within a workload with similar performance characteristics; refer to the following sections for more detailed explanations:

- Performance Goals (see 4.2.8, “Using Response Time Goals” on page 139, 4.2.9, “Using Discretionary Goals” on page 139, and 4.2.10, “Using Execution Velocity Goals” on page 140)
- Resource requirements (do not define resource groups unless really required; see 4.2.12, “Using Resource Groups” on page 142)
- Business Importance to the installation (see 4.2.11, “Importance Keyword” on page 142)

WLM manages a service class period as a single entity when allocating resources to meet performance goals. A service class can be associated with only one workload.

The initial activation of the service definition does not require accurate service goals since it will only be used in compatibility mode. While in compatibility mode, collect and analyze RMF reports to set reasonable performance goals. All the systems in the sysplex should be reviewed, with particular attention to performance objectives based on peak time usage.

Enter Service Classes using option 4. Workload specification must be input first using option 3, where:

- SRVCLASS is the 8-character service class name, unique within the service.
- Description is an optional 32-character description.
- Workload is required, and must have been previously defined in A.4, “Workload Classification” on page 215.
- RG is an optional association with this service class, and only one RG can be assigned to a service class (typically, this is not required).
- Per is the Performance Period, which consists of a performance goal, importance, and duration for a service class. Up to eight performance periods can be defined to manage work that has changing performance requirements as work consumes more and more resources.

  Check and refine the setting based on compatibility mode report analysis.
- Dur is the Duration which specifies the length of the period in service units. Remember to factor in any changes to service coefficient constants. (Do not use this with CICS or IMS.)

  Check and refine the setting based on compatibility mode report analysis.
- Imp is the relative importance of the service class goal, expressed by number from 1 to 5, where 1 is the highest importance. Remember, this is the importance of *meeting your goal*, not the business importance of the work.
- Goal is a precise definition of the your performance objective. Take an initial guess at what would be an appropriate goal.

  C*heck and refine the setting, based on compatibility mode report analysis.*

  Your goals can be one of:

  1. Average response time
  2. Response time with percentile
  3. Execution velocity
  4. Discretionary

Analyze each workload individually, then consolidate the service policy into one location similar to Table 27 on page 218 for an overall performance management review. Table 28 on page 219 provides a summary of where to find more information when migrating your workloads to WLM.

Table 27. Worksheet: Service Class Definition

SRVCLASS	Description	Workload	RG	Per	Dur.	Imp.	Goal

Table 28. Workload Considerations Overview

Type	Description	Reference
ASCH	Advanced Program to Program Communication (APPC) transaction programs	4.4.7, “Considerations for Migrating APPC Workloads” on page 168
CB	Component Broker client object method requests	No information at this time
CICS	CICS V4 transactions	4.4.4, “Considerations for Migrating CICS Workloads” on page 154
DB2	DB2 Sysplex Query Parallelism	5.1, “DB2-Based Workloads” on page 171
DB2	DB2 Stored Procedures	5.2, “DB2 Stored Procedures” on page 179
DDF	DB2 Distributed Data Facility (DDF V4 and above)	4.4.6, “Considerations for Migrating DDF Workloads” on page 165
IMS	IMS V5 and above messages	4.4.5, “Considerations for Migrating IMS Workloads” on page 164
IWEB	World Wide Web request being serviced by Internet Connection Server (ICS)	5.3, “Internet Connection Secure Server/Domino Go Web Server” on page 187
JES	Jobs initiated by JES2 or JES3	4.4.2, “Considerations for Migrating Batch Workloads” on page 148
LSFM	Work from LAN server for MVS	No information at this time
OMVS	OS/390 UNIX System Services forked children address spaces	5.4, “OS/390 UNIX System Services Function and Application” on page 201
SOM	SOM client object class binding requests	No information at this time
STC	START, MOUNT, and system component address spaces	4.4.3, “Considerations for Migrating STC Workloads” on page 150
TSO	Commands issued from foreground TSO sessions	4.4.1, “Considerations for Migrating TSO Workloads” on page 147
non-IBM	Workloads executing non-IBM product code	4.4.8, “Considerations for Migrating Non-IBM Products” on page 170
RYO	Roll Your Own	4.4.9, “Considerations for Migrating Roll-Your-Own (RYO) Workloads” on page 170

A.6 Classification Groups Definition

If you do not have standard naming conventions that allow masking or wildcarding, a classification group effectively collects the work together. A *group* is a collection of the same work qualifiers.

Qualifier groups of more than five members are quicker to check than single instances in the classification rules. If you have, for example, a long list of CICS or IMS transaction names that you want to group in a service class or report class, consider setting up a group.

Classification Groups are managed using option 5. Fill in Table 29, where:

- Group Type is one of:
  1. Connection Type Groups (CTG)
  2. LU Name Groups (LUG)
  3. Net ID Groups (NETG)
  4. Package Name Groups (PKG)
  5. Plan Name Groups (PNG)
  6. Subsystem Instance Groups (SIG)
  7. Transaction Class Groups (TCG)
  8. Transaction Name Groups (TNG)
  9. Userid Groups (UIG)
  10. Perform Groups (PFG)
- Group Name is 8 characters, unique within the service definition.
- Description is an optional 32 character description of the group.
- Qualifier identifies the data to match.
- Description allows an optional 32 character description of the qualifier.

Table 29. Worksheet: Classification Groups

Group Type	Group Name	Description	Qualifier	Description

A.7 Classification Rules Definition

Classification rules define how to categorize work into service classes, and optional report classes, based on *work qualifiers*. There is only one set of classification rules in the service definition for the entire sysplex. It cannot be overridden. Define your classification rules *after* you have defined service classes, and ensure that each service class has a corresponding rule.

Keep your classification rules as simple as possible. Use defaults to cover most cases, and list the exceptions. Remember that the rules are order-sensitive, and the first matched rule applies. The Start field applies only to the accounting information and subsystem parameter qualifiers.

For performance reasons, use Classification Groups if you have five or more entries, see A.6, “Classification Groups Definition” on page 219 for more details.

Use wildcards, masks, or equations where possible:

*****	This is a wildcard notation to indicate multiple character replacement in a character string. This wildcard can be the last character in a string, or can be used by itself. If an asterisk is specified in a position other than the last, it is treated as a character. For example, CI*S requires an exact match to CI*S. An * by itself indicates a match to all characters.
%	This is a masking notation to replace a single character within a qualifier. It allows any character to match the position in the rule. You can mask multiple consecutive characters for multiple character replacement. A mask at the end of a character string matches either a null value or a single character.
<>=	This is a relational operator which represents a type of comparison. You can use a relational operator for the priority work qualifier only. The relational operators are < (less than), <= (less than or equal), > (greater than), >= (greater than or equal), and <> (not equal). The relational operator must precede the value, for example:

```
-------Qualifier-------         -------Class--------
Type    Name     Start          Service     Report
                                BATALL__    ________
PRI     >10_____ ___            BATSPEC_    ________
```

The following is a brief overview of the valid work qualifiers for each IBM subsystem type:

- ASCH Qualifier Types (* indicates Group classification option):

AI	Accounting Information - passed on job statement
SI*	Subsystem Instance - VTAM applid for SI
TC*	Transaction Class - job class used for work selection
TN*	Transaction Name - jobname on JCL JOB card in APPC/MVS transaction program (TP) profile
UI*	Userid - userid of user requesting APPC/MVS services

- CB Qualifier Types (* indicates Group classification option):

CN	Collection Name - logical server group name
SPM	Subsystem Parameter - 246 byte string
UI*	Userid - userid of user requesting CB services

- CICS Qualifier Types (* indicates Group classification option):

LU*	LU Name, the 8-byte NETNAME of the principal facility of the transaction instance
SI*	Subsystem Instance - VTAM applid for SI
TN*	Transaction Name - parameter on many CICS commands, commonly referred to as a CICS transaction identifier (tranid)
UI*	Userid - userid specified at logon

- DB2 Qualifier Types (* indicates Group classification option):

AI	Accounting Information - query originator AI
CI	Correlation Information - correlation ID associated with originator
CN	Collection Name - associated with query originator
CT*	Connection Type - associated with query originator

LU*	LU Name - originator of the query
NET*	Net ID - originator of the query
PF*	Perform - originator of the query
PK*	Package Name - originator of the query
PN*	Plan Name - originator of the query
PR	Procedure Name - originator of the query
PRI	Priority - originator of the query
SI*	Subsystem Instance - subsystem type of originator of query (TSO for TSO/E, JES for batch, DDF for DDF)
SPM	Subsystem Parameter - SPM from originator of the query
TC*	Transaction Class - originator of the query TC
TN*	Transaction Name - originator of the query TN (or jobname)
UI*	Userid - userid of the originator of the query

- DDF Qualifier Types (* indicates Group classification option):

AI	Accounting Information - DDF server thread DB2 accounting
CI	Correlation Information - DB2 CI of DDF server thread
CN	Collection Name - DB2 CN of first SQL package accessed by DRDA requestor
CT*	Connection Type - DB2 CT of first DDF server thread
LU*	LU Name - VTAM LU of system issuing SQL request
NET*	Net ID - VTAM netid of system issuing SQL request
PK*	Package Name - first DB2 package accessed by DRDA requestor
PN*	Plan Name - DB2 plan name associated with DB2 server thread
PR	Procedure Name - blank or, if first SQL statement is CALL, then unqualified name of DB2 stored procedure
SI*	Subsystem Instance - DB2 server's MVS subsystem name
UI*	Userid - DDF server thread's primary AUTHID

- IMS Qualifier Types (* indicates Group classification option):

LU*	LU Name, the 8 byte device LU name
NET*	Net ID - only available for LU 6.2 device
SI*	Subsystem Instance - IMS subsystem name from IMSID in IMS DFSMPR procedure, a 1- to 4-character unique identifier
TC*	Transaction Class - CLASS keyword on PGMTYPE= parameter in the APPLCNT macro
TN*	Transaction Name - CODE= parameter on the IMS TRANSACT macro
UI*	Userid - userid specified at LOGON time

- IWEB Qualifier Types (* indicates Group classification option):

SI*	Subsystem Instance - Internet Connection Server instance
SPM	Subsystem Parameter, see 5.3.4, “Define Your WLM Classification Rules for Scalable Web Server” on page 193 for details
TC*	Transaction Class, see 5.3.4, “Define Your WLM Classification Rules for Scalable Web Server” on page 193 for details
TN*	Transaction Name - method name, such as GET, POST or DELETE
UI*	Userid - userid of Web server address space

- JES Qualifier Types (* indicates Group classification option):

AI	Accounting Information - JOB statement AI

PF*	Perform - PERFORM keyword on JCL JOB statement
PRI	Priority - value between 0 and 15 associated with batch job priority, available with OS/390 R4 JES2 (otherwise no match)
SI*	Subsystem Instance - JES2 or JES3 subsystem name from IEFSSNxx
TC*	Transaction Class - job class used for work selection
TN*	Transaction Name - jobname of the JES-managed job
UI*	Userid - userid on the JOB card on the RACF USER keyword

- LSFM Qualifier Types (* indicates Group classification option):

SI*	Subsystem Instance - procname of address space in which LAN Server for MVS is running
TN*	Transaction Name - one of LSFMMMTX (multimedia), LSFMFITX (file), LSFMAMTX (administration), LSFMCMTX (communication) transactions

- OMVS Qualifier Types (* indicates Group classification option):

AI	Accounting Information - inherited from parent process
TN*	Transaction Name - jobname on the OS/390 UNIX System Services address space
UI*	Userid - RACF userid associated with the address space, inherited from the parent process or assigned by a daemon

- SOM Qualifier Types (* indicates Group classification option):

CN	Collection Name - logical server name defined using REGIMPL
SPM	Subsystem Parameter the 246-byte string which has class name and method name
UI*	Userid - userid of user requesting the SOM service

- STC Qualifier Types (* indicates Group classification option):

AI	Accounting Information - JOB statement AI
PF*	Perform - either specified with START, or, on JCL JOB statement
SPM	Subsystem Parameter - indicates the system provided service class name that will be assigned
TN*	Transaction Name -name from START, name from MOUNT, system address space name, or the name on the jobcard
UI*	Userid - userid assigned to the started task by RACF

- TSO Qualifier Types (* indicates Group classification option):

AI	Accounting Information -TSO/E AI
PF*	Perform - specified on logon panel
UI*	Userid - userid specified at LOGON time

When the subsystem receives a work request, the system searches the classification rules for a matching qualifier and its service class or report class. There is no prescribed order for searching the work qualifiers; instead you define it. Your order can be different for each subsystem.

The classification rules are managed using option 6. Any classification groups you plan on using must be defined *before* you define classification rules. Fill in Table 30 on page 225, where:

- Subsystem is one of the IBM subsystem types previously listed, or your own.
- Description is an optional 32-character description.

- Level indicates match level nesting. If you get a match at level 1, then check level 2, because work is assigned the service class of the first match and highest sub-level match. You cannot nest most qualifier types within themselves. The first level you need to define is a DEFAULT SRVCLASS in each subsystem type you intend to use, except STC.
- Type is the qualifier type you want to check for a match.
- Name is 8 characters of data to match. Use wildcard, masks, or comparison logic whenever possible. Specify the * wildcard to match strings of less than 8 characters.
- Start is used for work qualifiers longer than 8 characters. You use the start position to designate the start position of the character string to match. Accounting Information, correlation information, connection type, procedure name, and subsystem parameter are greater than 8 characters.
- SRVCLASS is the service class name to assign to the workload, which will define the performance objectives for that transaction.
- RPTCLASS is the report class to assign. Do not assign report classes unless you want report information gathered in a manner different than SRVCLASS. SRVCLASS reporting will be done automatically.
- Description allows another 32-character description of the rule that has been defined.

When you have completed the classification rules, you should have designated all of your service classes.

Table 30. *Worksheet: Classification Rules Definition*

Subsystem	Description	Level	Type	Name	Start	SRVCLASS	RPTCLASS	Description

A.8 Report Class Definition

Optionally, classification rules can assign incoming work to a report class. WLM provides data for reporting on all of the service definition terms for a service class period and workload basis. Report classes are for additional reporting within a service class or across service classes; see 4.2.13, “Report Classes” on page 144.

The data made available for report classes includes:

- Number of transactions completed
- Average response times
- Resource usage data
- State samples

You can assign up to 999 report classes, with a maximum of one report class per work request or transaction. Report Class Descriptions can be specified using option 7. Specifying the report class in classification rules will prompt for the Description. Fill in Table 31, where:

- Report Class Name is 8 characters, unique within the service definition.
- Description is an optional 32-character description.

Table 31. Worksheet: Report Class Definition

Report Class Name	Description

A.9 Resource Group Definition

The resource groups are managed using option 3. Resource Groups must be defined before they can be used in a service class definitions. Since a resource group is new in WLM, you probably should not need to define resource groups; see 4.2.12, “Using Resource Groups” on page 142 for additional information.

The same resource group may be used by several service classes in A.5, “Service Class Definition” on page 216. You cannot assign a resource group to service classes representing transaction-oriented work, such as IMS or CICS. However, you can assign a resource group to their regions, where:

- Resource Group Name is 8 characters, unique within the service definition.
- Description is an optional 32-character description.
- Min is the minimum unweighted CPU service units per second.

- Max is the Maximum unweighted CPU service units per second.

Table 32. Worksheet: Resource Group Definition

Resource Group Name (RG)	Description	Min	Max

A.10 Application Environment Definition

An *application environment* is a group of application functions requested by a client that execute in server address spaces. WLM can dynamically manage the number of address spaces to meet the performance goals of the work requests. Each application environment should represent a named group of server functions that require access to the same libraries.

The following conditions are necessary for application environment usage:

- The work manager subsystem must have implemented the WLM services that make use of application environments; today, this includes:
  - DB2 (subsystem type DB2) for stored procedure requests
  - SOMobjects (subsystem type SOM) for SOM client object class binding requests
  - Component Broker (subsystem CB) for Component Broker object method requests
  - Internet Connection Server (subsystem type IWEB) for Hyper Text Transfer Protocol (HTTP) requests
- One or more application environments must be defined in the service definition.
- The subsystem's work requests must be associated with the appropriate application environment. See the appropriate section referred to in Table 28 on page 219.

The application environments are managed using option 9. If you have workloads that fit the criteria listed, then fill in Table 33 on page 228, where:

- Env. Name is the 1- to 32-character name of the application environment. You must use this name when specifying to the subsystem how to map work. The operator will also use this to perform actions on the application environment. The name cannot start with SYS.
- Description is an optional 32-character description.
- Subsystem is the subsystem type associated with the application environment.
- Procedure is an optional name of the JCL procedure for starting the server for the application environment requests. If you specify a procedure, WLM manages the servers. If you do not specify a procedure, you will need to start the servers either manually or with automation.
- Lim is the required specification of limits on creating server address spaces for a subsystem instance. Reasons for limiting include serialization, testing, or a server implementation restriction. Choices are:
  1. No limit (valid for DB2, IWEB, CB; default for DB2, IWEB, CB)
  2. Single address space per system (valid for DB2, IWEB, CB)
  3. Single address space per sysplex (required value for SOM)

- Start Parameters are optional parameters required for the JCL procedure defined in Procedure. Specify parameters that you would use with an MVS START of the server address space. WLM substitutes the subsystem instance name provided to WLM when the subsystem connected, for symbol &IWMSSNM.

Table 33. Worksheet: Application Environment Definition

Env. Name	Description	Subsystem	Procedure	Lim.

A.11 Scheduling Environment Definition

A *scheduling environment* is a list of resource names, along with their required states, that allows scheduling of work in an asymmetric sysplex. If an OS/390 image satisfies all the scheduling environment requirements associated with a unit of work, then that work can be assigned to that image. If any of the resource requirements are not satisfied, the unit of work cannot be assigned to that MVS image.

Scheduling environments and resource names reside in the service definition and apply across the entire sysplex. You can use scheduling environments in both compatibility and goal modes.

Each element in a scheduling environment consists of the name of a resource, and a required state which is either ON or OFF. Each resource represents the potential availability of a resource on an MVS system. The resource can represent an actual physical entity such as a database, or it can be an intangible quality such as a time of day. You can have up to 999 unique scheduling environments in a service definition.

Scheduling Environments are managed using option 10. Fill in Table 34 on page 229, where:

- Scheduling Env is 1- to 16-character name, unique within the service definition. Special characters are allowed in the name. Names cannot start with SYS_.
- Description is an optional 32-character description.
- Resource is 1- to 16-character name, with the same naming convention rules as the scheduling environment name. Up to 999 resources can be defined, and there can be more than one resource name listed in a scheduling environment
- State indicates whether the required state is ON or OFF. If set to ON, then the resource name must be set to ON for work to be assigned to that system. If set

to OFF, then the resource name must be set to OFF for work to be assigned to that system.

Table 34. Worksheet: Scheduling Environment Definition

Scheduling Env.	Description	Resource	Resource Desc.	State

Note: Resources are put into RESET state when the system is IPLed or when a policy is activated that defines a resource name that did not exist in the previous active policy. A RESET condition prevents matching both ON and OFF required states.

Do not attempt to use the F WLM,RESOURCE= command in the COMMNDxx parmlib member because this member is processed too early during system initialization for the command to be successful. Instead, use an automation product such as System Automation for OS/390.

How do you use the scheduling environment? Specify the SCHENV=RUN_AFTER_8PM parameter on the JES2 or JES3 JOB card JCL statement, or have an exit or your job scheduling package add the scheduling environment to the jobs, as required. The job is now associated with the requirement that the scheduling environment named RUN_AFTER_8PM be satisfied. If the scheduling environment name specified is not defined in the active WLM policy, the job will fail with a JCL error during conversion.

Appendix B. Special Notices

This publication is intended to help Systems Programmers and Performance Specialists to implement OS/390 Workload Manager in *goal mode*. The information in this publication is not intended as the specification of any programming interfaces that are provided by OS/390. See the PUBLICATIONS section of the IBM Programming Announcement for OS/390 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

ACF/VTAM	CICS
CICS/ESA	CICSPlex

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 235.

- *System/390 MVS/ESA Version 5 Workload Manager Performance Studies*, SG24-4352
- *DB2 for OS/390 Version 5 Performance Topics*, SG24-2213
- *OS/390 Release 3 Implementation*, SG24-2067

C.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs:

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbook	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

C.3 Other Publications

These publications are also relevant as further information sources:

- *CICS Performance Guide*, SC33-1183
- *DB2 for OS/390 Version 5 Administration Guide*, SC26-8957
- *DB2 for OS/390 Version 5 Data Sharing: Planning and Administration*, SC26-8961
- *DB2 for MVS/ESA V4 Installation Guide,* SC26-3456
- *Domino Go Webserver Release 5.0 for OS/390: Webmaster’s Guide*, SC31-8691
- *OS/390 MVS Initialization and Tuning Reference,* SC28-1752
- *OS/390 MVS Planning: Workload Management*, GC28-1761
- *OS/390 MVS Programming: Workload Management Services*, GC28-1773
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779
- *OS/390 MVS System Commands*, GC28-1781

- *OS/390 MVS System Management Facilities (SMF)*, GC28-1783
- *OS/390 Parallel Sysplex Application Migration*, GC28-1863
- *OS/390 Resource Measurement Facility Performance Management Guide,* SC28-1951
- *OS/390 Resource Measurement Facility Report Analysis*, SC28-1950
- *OS/390: Resource Measurement Facility User's Guide*, SC28-1949
- *OS/390 UNIX System Services Planning,* SC28-1890

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this redbooks site.

  Redpieces are redbooks in progress; not all redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders via e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at `http://www.redbooks.ibm.com/` and for IBM employees at `http://w3.itso.ibm.com/`.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may also view redbook. residency, and workshop announcements at `http://inews.ibm.com/`.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

active service policy. The service policy that determines workload management processing if the system is running in goal mode. See goal mode.

application environment. A group of application functions requested by a client that execute in server address spaces.

application owning region (AOR). In a CICSPlex configuration, a CICS region devoted to running applications.

automatic control. One of two distinct methods of managing application environments in goal mode. Under automatic control, the name of the start-up JCL procedure has been defined for an application environment, giving workload management the ability to automatically start server address spaces. Contrast with manual control.

CICSplex. A configuration of interconnected CICS systems in which each system is dedicated to one of the main elements of the overall workload. See also application owning region, file owning region, and terminal owning region.

classification rules. The rules workload management and subsystems use to assign a service class and, optionally, a report class to a work request. A classification rule consists of one or more of work qualifiers such as subsystem type, subsystem instance, userid, accounting information, transaction name, transaction class, source LU, netid, and LU name.

compatibility mode. A mode of processing in which the IEAIPSxx and IEAICSxx parmlib members determine system resource management. See also goal mode.

control region. The main storage region that contains the subsystem work manager or subsystem resource manager control program.

couple data set. A data set created through the XCF couple data set format utility. The data set is shared by MVS systems in a sysplex. There are several types of couple data sets for different purposes. See also XCF couple data set.

CPU service units. A measure of the task control block (TCB) execution time multiplied by an SRM constant which is CPU model-dependent. See also unweighted CPU service units per second, and service unit.

delay monitoring services. The workload management services that monitor the delays encountered by a work request.

distributed data facility (DDF). An optional feature that allows a DB2 application to access data at other DB2s and at remote relational database systems that support IBM's Distributed Relational Database Architecture (DRDA).

duration. The length of a service class performance period in service units.

enclave. A transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as a unit.

execution velocity. A service goal naming the rate at which you expect work to be processed for a given service class or a measure of the acceptable processor and storage delays while work is running.

goal mode. A mode of processing where the active service policy determines system resource management. See also compatibility mode.

importance level. The degree of importance of a service goal relative to other service class goals, in five levels: lowest, low, medium, high, highest.

installation. A particular computing system, including the work it does and the people who manage it, operate it, apply it to problems, service it, and use the results it produces.

installed service definition. The service definition residing in the WLM couple data set for WLM.

I/O service units. A measure of individual data set I/O activity and JES spool reads and writes for all data sets associated with an address space.

logical unit (LU). In VTAM, the source and recipient of data transmissions. Data is transmitted from one logical unit (LU) to another LU. For example, a terminal can be an LU, or a CICS or IMS system can be an LU.

LU. Logical unit.

LU name. The second level of the source LU name after the "." for fully qualified names.

LU 6.2 session. A session that is initiated by VTAM programs on behalf of a logical unit (LU) 6.2 application program, or a session initiated by a remote LU in which the application program specifies that the VTAM programs are to control the session by using the APPCCMD macro instruction.

manual control. One of two distinct methods of managing application environments in goal mode. Under manual control, the name of the start-up JCL procedure has not been defined for an application environment. The installation must therefore manually start server address spaces when needed. Contrast with automatic control.

masking. Using a % for a single character replacement in classification rules. See also wild carding.

MVS image. The system id of any MVS system included in the sysplex as it appears in the SYS1.PARMLIB member.

performance administration. The process of defining and adjusting workload management goals and resource groups based on installation business objectives.

performance block. A piece of storage containing workload management's record of execution delay information about work requests.

performance management. The process workload management uses to decide how to match resources to work according to performance goals and processing capacity.

performance period. A service goal and importance level assigned to a service class for a specific duration. You define performance periods for work that has changing performance requirements as work consumes resources.

policy. See service policy.

relational database management system (RDBMS). A relational database manager that supports SAA.

report class. A group of work for which reporting information is collected separately. For example, you can have a report class for information combining two different service classes, or a report class for information on a single transaction.

resource group. An amount of processing capacity across one or more MVS images, assigned to one or more service classes.

resource. When used as part of a scheduling environment, a resource is an abstract element that can represent an actual physical entity (such as a peripheral device), or an intangible quality (such as a certain time of day). A resource is listed in a scheduling environment along with a required state of ON or OFF. If the corresponding resource state on a given system matches the required state, then the requirement is satisfied for that resource.

scheduling environment. A list of resource names along with their required states. If an MVS image satisfies all of the requirements in the scheduling environment associated with a given unit of work, then that unit of work can be assigned to that MVS image. If any of the requirements are not satisfied, then that unit of work cannot be assigned to that MVS image.

server address space. Any address space that does work on behalf of a transaction manager or a resource manager. For example, a server address space could be a CICS AOR, or an IMS control region.

service administration application. The on-line ISPF application used by the service administrator to specify the workload management service definition.

service class. A group of work which has the same performance goals, resource requirements, or business importance. For workload management, you assign a service goal and optionally a resource group to a service class.

service coefficient. A value that specifies which type of resource consumption should be emphasized in the calculation of service rate. The types of resource consumption are CPU, IOC, MSO, and SRB.

service definition. A definition of the workloads and classification rules in an installation. The definition includes workloads, service classes, systems, resource groups, service policies, and classification rules.

service level administrator. The user role introduced by workload management whose main task is to make sure overall installation operation is consistent with performance goals and objectives.

service level agreement (SLA). A written agreement of the information systems (I/S) service to be provided to the users of a computing installation.

service policy. A named set of performance goals workload management uses as a guideline to match resources to work. See also active service policy.

service request block (SRB) service units. A measure of the SRB execution time for both local and global SRBs, multiplied by an SRM constant which is CPU model-dependent.

service unit. The amount of service consumed by a work request as calculated by service definition coefficients and CPU, SRB, I/O, and storage service units.

single-system sysplex. A sysplex in which only one MVS system is initialized as part of the sysplex. In a single-system sysplex, XCF provides XCF services on the system, but does not provide signalling services between MVS systems. See also multi-system sysplex, XCF-local mode, and monoplex.

source LU. A fully qualified two level name separated by a ".", where the first level is the network id and the second is the LU name, or merely a single LU name. See also LU name.

storage service units. A measure of the central storage page frames multiplied by 1/50 of the CPU service units. The 1/50 is a scaling factor designed to bring the storage service component in line with the CPU component.

subsystem instance. 1) For application environments, a unique combination of subsystem type (as specified in the service definition for an application environment) and subsystem name (as specified by the work manager subsystem when it connects to workload management). 2) For classification, a work qualifier used to distinguish multiple instances of a subsystem.

subsystem work manager. An address space defined in the SYS1.PARMLIB member as SUBSYS=nnn.

terminal owning region (TOR). A CICS region devoted to managing the terminal network.

unweighted CPU service units per second. The unweighted service units per second of task or SRB execution time. This measure is CPU-model dependent, but is independent of the values of the service coefficients.

velocity. A service goal naming the rate at which you expect work to be processed for a given service class, or a measure of the acceptable processor and storage delays while work is running.

wild carding. The use of an asterisk (*) as a multiple character replacement in classification rules. See also masking.

WLM couple data set. A type of couple data set that is created through the XCF couple data set format utility for the WLM function. The data set contains the service definition information.

workload. A group of work to be tracked, managed and reported as a unit. Also, a group of service classes.

workload management mode. The mode in which workload management manages system resources on an MVS image. The mode can be either compatibility mode, or goal mode.

work qualifier. An attribute of incoming work. Work qualifiers include: subsystem type, subsystem instance, userid, accounting information, transaction name, transaction class, source LU, netid, and LU name.

work request. A piece of work, such as a request for service, a batch job, an APPC, CICS, or IMS transaction, a TSO LOGON, or a TSO command.

Index

ITSO Redbook Evaluation

OS/390 Workload Manager Implementation and Exploitation
SG24-5326-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ Customer   _ Business Partner   _ Solution Developer   _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other redbooks would you like to see published?

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

SG24-5326-00

Printed in the U.S.A.

OS/390 Workload Manager Implementation and Exploitation

SG24-5326-00